RAD, Rules, and Compatibility: What's Coming in Kuali Rice 2.0

Eric Westfall - Indiana University

JASIG 2011

For those who don't know…

- Kuali Rice consists of multiple sub-projects which provide:
  - Middleware Services
  - Application Development Framework
- These different pieces are integrated into a cohesive software stack
- This provides a common "platform" for Enterprise application development and integration

Kuali Rice Vision

- Support the needs of the Kuali Application projects
  - Kuali Financial System (KFS)
  - Kuali Coeus (KC)
  - Kuali Student (KS)
  - Kuali Open Library Environment (OLE)
  - Kuali People Management for the Enterprise (KPME)
- Support the creation of non-Kuali projects
  - Local projects at an institution or organization

Kuali Rice Components

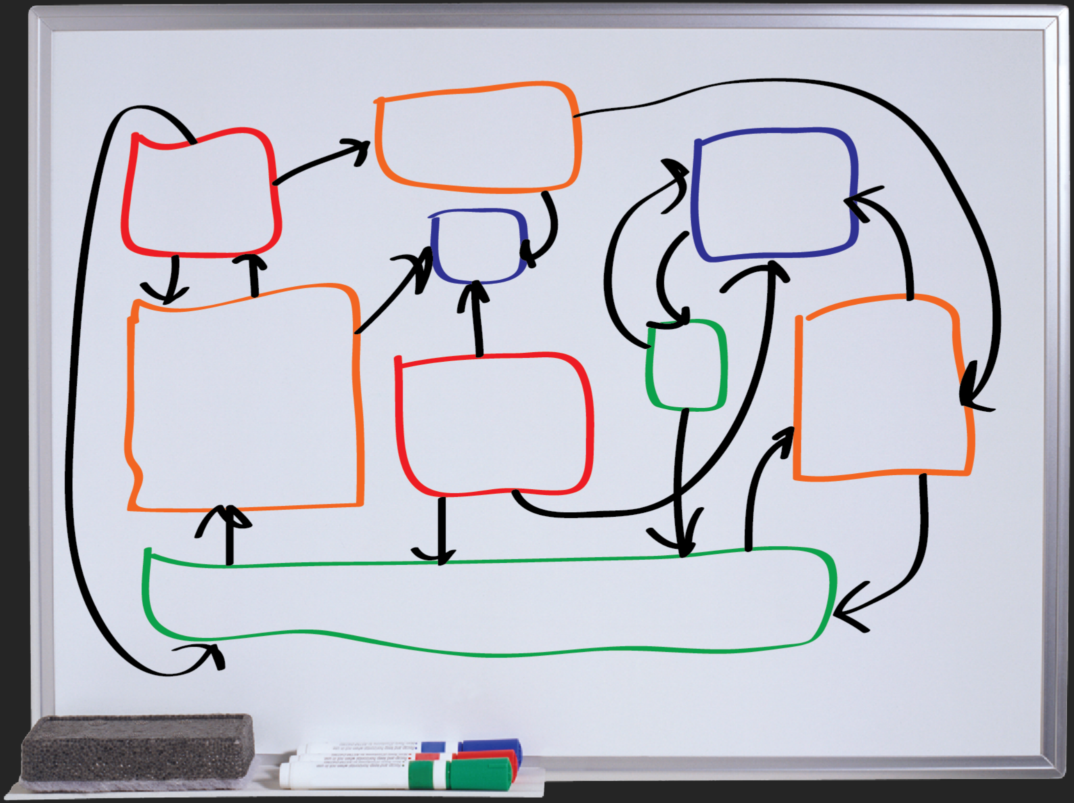
- Most recent release of Kuali Rice is 1.0.3.1
- Version 1.0.x
  - KSB - Kuali Service Bus
  - KIM - Kuali Identity Management
  - KEW - Kuali Enterprise Workflow
  - KEN - Kuali Enterprise Notification
  - KNS - Kuali Nervous System
- Version 2.0.x – (Q3 2011)
  - KRMS – Kuali Rule Management System
  - KRAD – Kuali Rapid Application Development

Kuali Rice 2.0 Deliverables

- Modularity
  - To more loosely decouple different components
- Version Compatibility
  - Provide deployment and upgrade flexibility
- Kuali Rule Management System (KRMS)
  - Business Rule Management
  - Requirements for Kuali Coeus and Kuali Student
- Kuali Rapid Application Development (KRAD)
  - “Modernize” the KNS
  - Requirements for Kuali Student user experience

Modularity

- Kuali Rice has a lot of different pieces



- But in the past, components have not always been well organized!

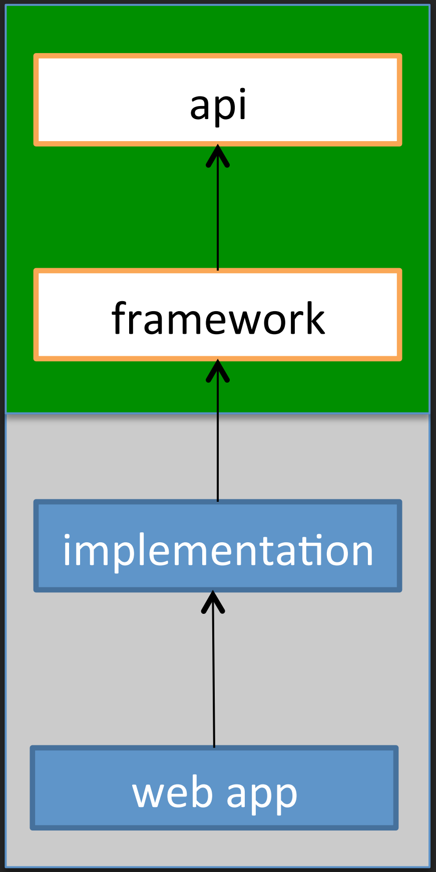
Modularity – In the Past

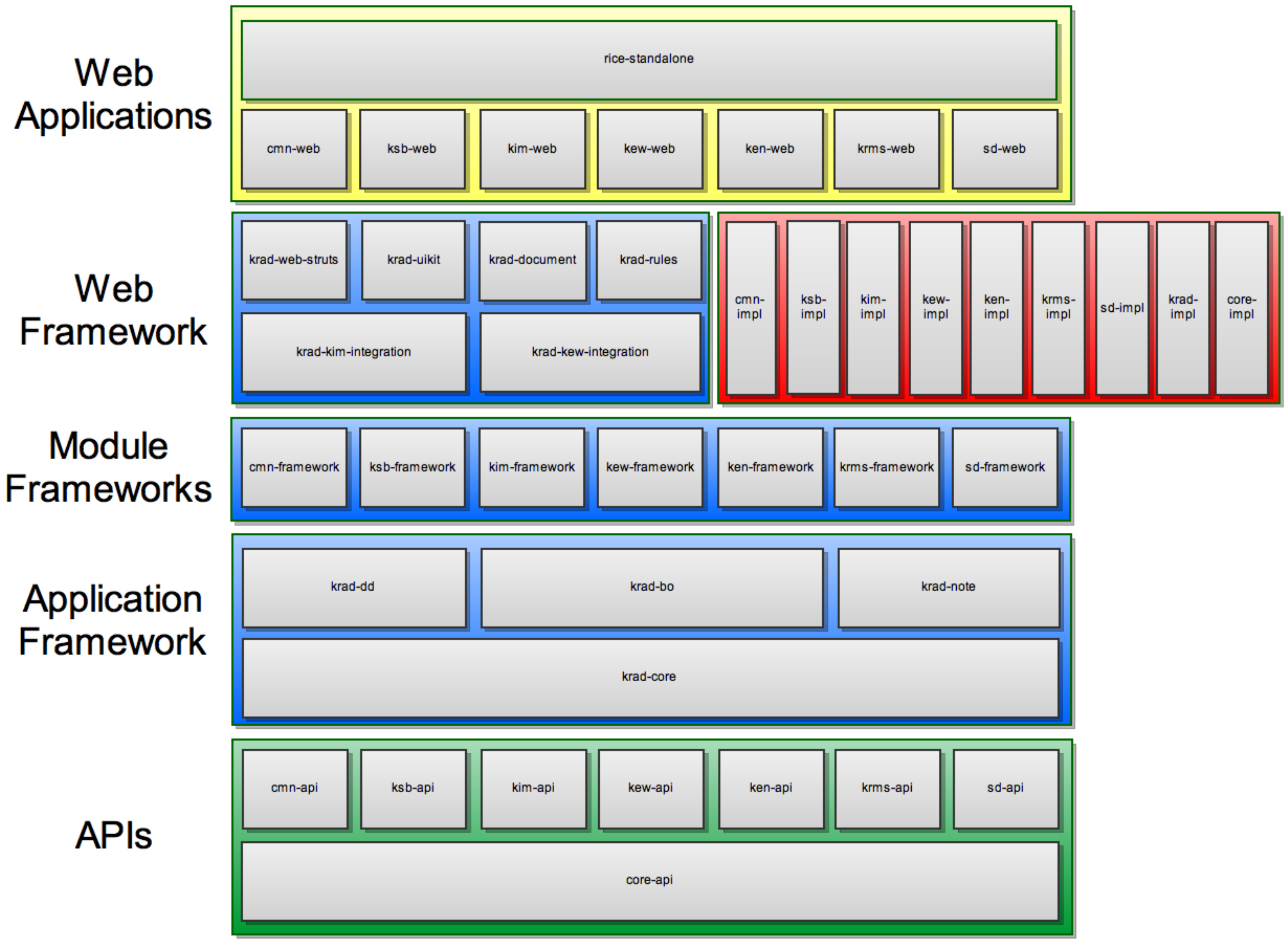
- Level of coupling between components has not been given as much attention as it deserves
- Code base overly “monolithic”
- Difficult for a client of the software to understand which apis and services they should be using (as opposed to “internal” ones)

Modularity – Making it Better

- In Rice 2.0:
  - Working to separate out different conceptual modules of Rice into multiple-maven modules with our Maven-based build
  - Maven can enforce dependencies (both internal and external) and help with documenting them
  - Updating package names such that it should be clear which classes constitute "apis" and which are for internal use only

Typical Module Breakdown





Acronyms

cmn	- Common Services
ksb	- Kuali Service Bus
kim	- Kuali Identity Management
kew	- Kuali Enterprise Workflow
ken	- Kuali Enterprise Notification
krms	- Kuali Rule Management System
sd	- Shared Data Services
krad	- Kuali Rapid Application Development
core	- Kuali Rice Core

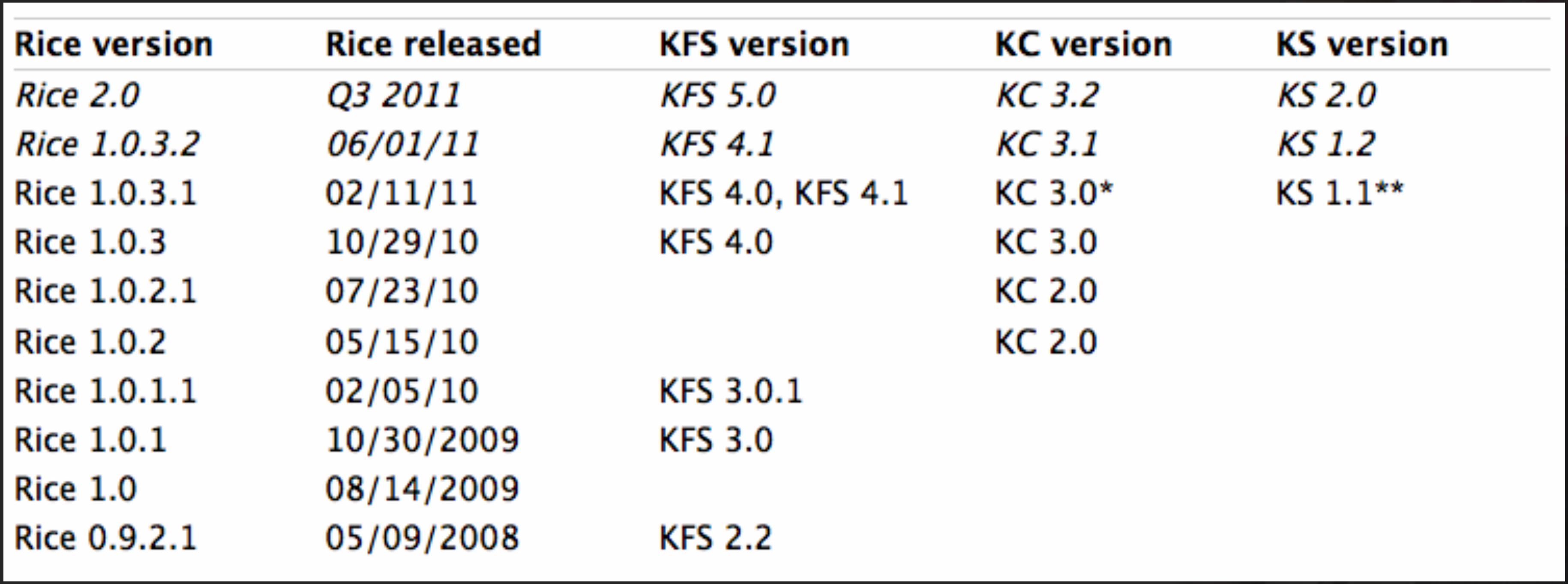
Why Do This?

- Decrease the complexity of rice
- Isolate external dependencies
- Reduce coupling in rice
- Allow modules of rice to be developed and tested in isolation
- Improve the quality of the rice codebase
- Make it explicit what code client apps can use
  - which helps rice make guarantees on releases
  - make client upgrades easier
- Provide more deployment and integration flexibility
- Version Compatibility

Version Compatibility

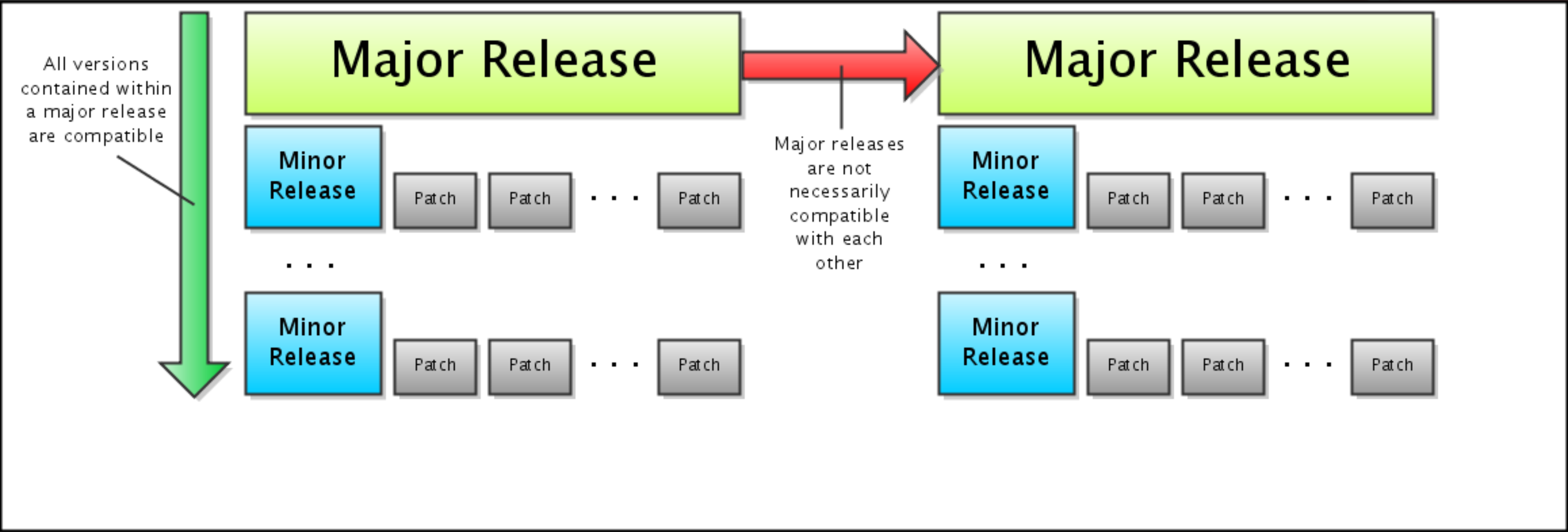
- Concerned primarily with client-server communication with Kuali Rice services
  - We call this "middleware" compatibility
- Applications written against different versions of the Rice middleware services need to be able to interoperate
- Should be able to upgrade middleware services without requiring every client to upgrade at the same time

Current Situation



Rice version	Rice released	KFS version	KC version	KS version
Rice 2.0	*Q3 2011*	*KFS 5.0*	*KC 3.2*	*KS 2.0*
Rice 1.0.3.2	*06/01/11*	*KFS 4.1*	*KC 3.1*	*KS 1.2*
Rice 1.0.3.1	02/11/11	KFS 4.0, KFS 4.1	KC 3.0*	KS 1.1**
Rice 1.0.3	10/29/10	KFS 4.0	KC 3.0	
Rice 1.0.2.1	07/23/10		KC 2.0	
Rice 1.0.2	05/15/10		KC 2.0	
Rice 1.0.1.1	02/05/10	KFS 3.0.1		
Rice 1.0.1	10/30/2009	KFS 3.0		
Rice 1.0	08/14/2009			
Rice 0.9.2.1	05/09/2008	KFS 2.2		

Desired Situation



Lifespan Summary:

Patch Release – as needed
Minor Release – every 6 months
Major Release – every 2-3 years

Existing Compatibility Challenges

- Service contracts not always well defined
- Difficult for clients to know which code constitutes apis that they should be using
- Using Java Serialization over HTTP in many places
- Direct connections from client applications into the Kuali Rice database
- Project moving quickly the last few years, lots of change
- Verifying and enforcing compatibility

Path to Compatibility

- Reduce amount of direct database integration with Rice database from clients
  - Can't be totally eliminated for 2.0
- Move public services apis to "api" modules
- Create package structures that reflect modularity
- Use SOAP for service integration
- Design message formats to allow for extensibility and compatibility across versions
- Add support for version information to KSB service registry

Enforcing Compatibility

- Define a set of rules for "evolving" services
  - Only add operations and data elements
  - Never remove, but can deprecate
  - If major changes needed, a new service must be created
- Implement automated tests against various services which can be run against later versions
- Operationalize a governance process for service apis

Kuali Rule Management System (KRMS)

- KRMS is a new module in Rice 2.0
- Implements a Business Rule Management System (BRMS)
- BRMS - a system used to define, deploy, execute, monitor and maintain business rules
- Business Rules – decision logic that is used by operational systems within an organization or enterprise

Motivations – Kuali Coeus

- Functional equivalence with MIT Coeus
- Workflow Rules
- Notification Rules
- Validation Rules
- Questionaire Rules
- GUI for maintaining rules
- Integrates with data in Coeus database
- Supports custom KC "Functions"

Motivations – Kuali Student

- Kuali Student also has needs for a BRMS
- Course Prerequisites
  - Student needs courses <course list>
  - Student needs a minimum GPA of <average>
  - Student must have permission from advisor
  - Etc.
- Workflow Routing
- Already implemented a repository for rules, but needed an execution engine

Overall Requirements for KRMS

- General enough to be used in many different cases
  - But must satisfy at least KC and KS requirements
- Rule Repository
- Execution Engine
  - Must be able to track execution plan and provide information back to caller for decision support
- Maintenance GUI
- Extensible and Pluggable

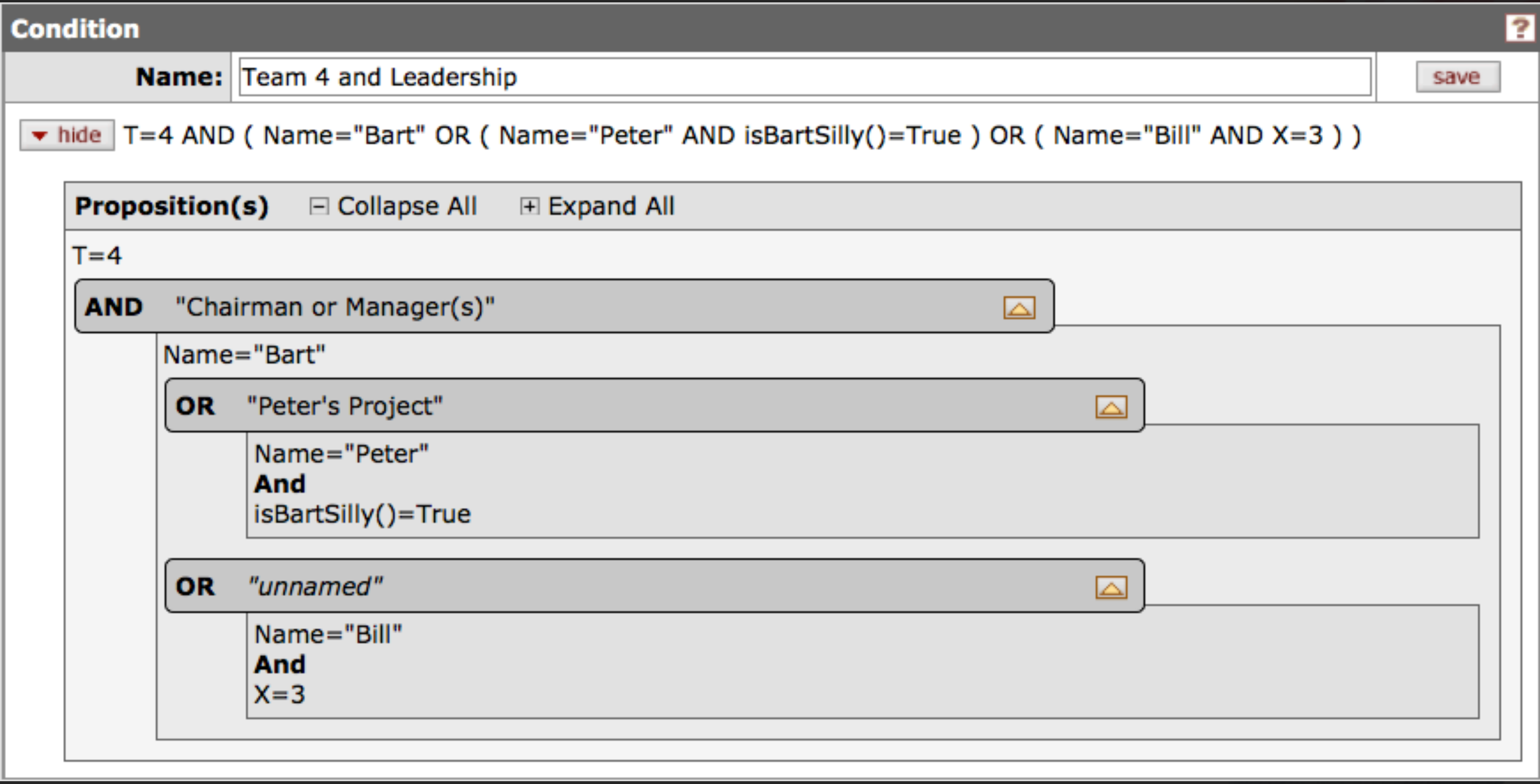
Terminology

- Proposition – a function that resolves to true or false
  - amount > $1000
- Compound Proposition – a kind of proposition that groups other propositions joined by AND or OR operator
  - ((amount > $1000) AND (category = “other”) AND …)
- Action - Executed if a rule evaluates to “true”
  - Notify unit coordinater
  - Route an approval request
  - Generate a validation error

Terminology

- Rule – a proposition linked with a list of actions to execute if proposition evaluates to “true”
- Agenda – execution plan for a set of rules
  - Controls the execution flow of rules in the agenda when it is executed
  - Can optionally contain conditional branching
- Term – a pieces of business data that can be used in the construction of propositions
- Fact – an instance of a term
- Context – a domain in which rules run

KRMS UI: Rule Editor



Condition

Name: Team 4 and Leadership

save

hide T=4 AND (Name="Bart" OR (Name="Peter" AND isBartSilly()=True) OR (Name="Bill" AND X=3))

Proposition(s) Collapse All Expand All

T=4

AND "Chairman or Manager(s)"

Name="Bart"

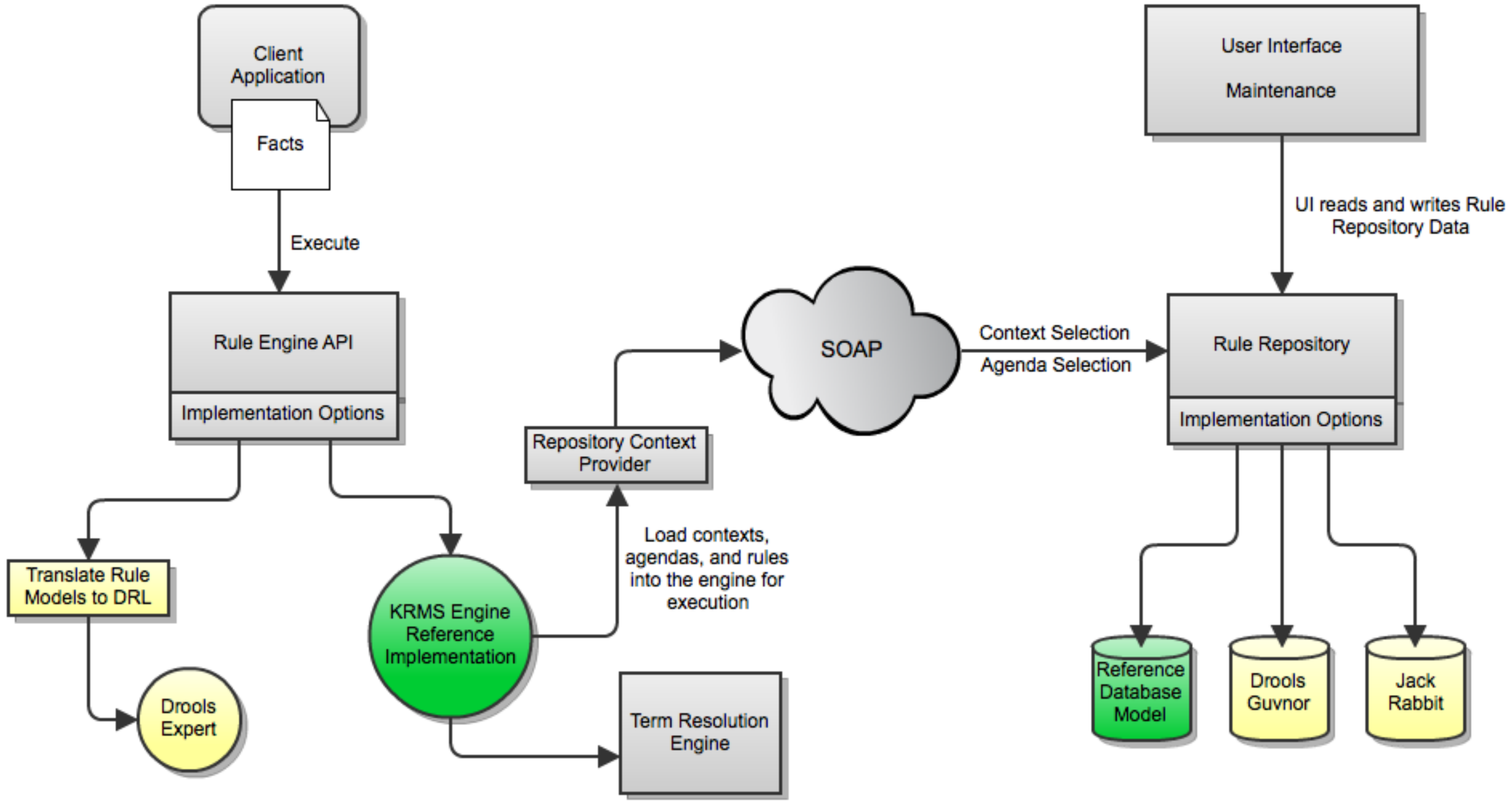
OR "Peter's Project"

Name="Peter"
And
isBartSilly()=True

OR *"unnamed"*

Name="Bill"
And
X=3

KRMS - Architecture



Integration

- Kuali Student
  - KRMS has it's own remotely accessible repository, but KS will be integrating with their pre-existing "Statement Service"
- Kuali Coeus
  - will build custom integration with their questionnaire component
- KRMS will integrate with other portions of Rice via custom rule "actions"
  - KEW
  - KEN
  - KRAD (for validation)

Kuali Rapid Application Development (KRAD)

- KRAD is intended to be a replacement for the Kuali Nervous System (KNS)
- KNS was created by the Kuali Financial System team early on in the project to create a development framework to build functionality quickly
- Extracted and included as part of the first release of Kuali Rice

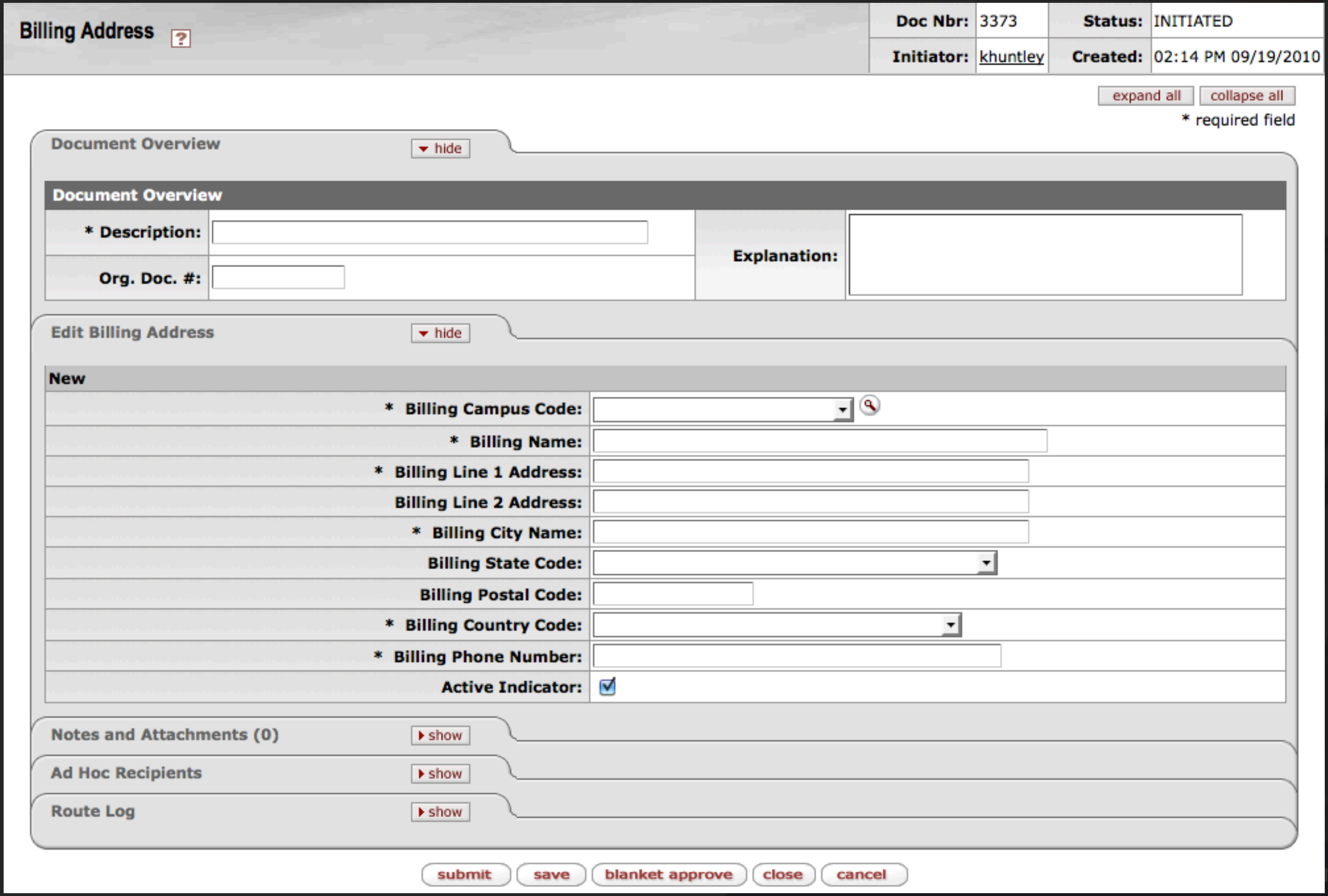
Legacy KNS

- Provides reusable code, shared services, integration layer, and a development strategy
- Provides a common look and feel through screen drawing framework
- Promotes a document (business process) centric model with workflow as a core concept
  - Built-in integration with KEW
- Provides built-in integration with Kuali Identity Management for authorization

KNS Core Concepts

- Business Objects – represents the data model for the application
- Lookups – allows for performing searches against business object data
- Inquiries – displays detailed information about business objects
- Documents – data entry (create and edit, can interact with workflow)
- Data Dictionary – defines metadata about business objects, lookup, inquiries, and documents

Sample KNS Screen



Billing Address

Doc Nbr:	3373	Status:	INITIATED
Initiator:	khuntley	Created:	02:14 PM 09/19/2010

expand all collapse all

* required field

Document Overview hide

Document Overview

* Description:		Explanation:	
Org. Doc. #:			

Edit Billing Address hide

New

* Billing Campus Code:	
* Billing Name:	
* Billing Line 1 Address:	
Billing Line 2 Address:	
* Billing City Name:	
Billing State Code:	
Billing Postal Code:	
* Billing Country Code:	
* Billing Phone Number:	
Active Indicator:	✓

Notes and Attachments (0) show

Ad Hoc Recipients show

Route Log show

submit save blanket approve close cancel

Why KRAD?

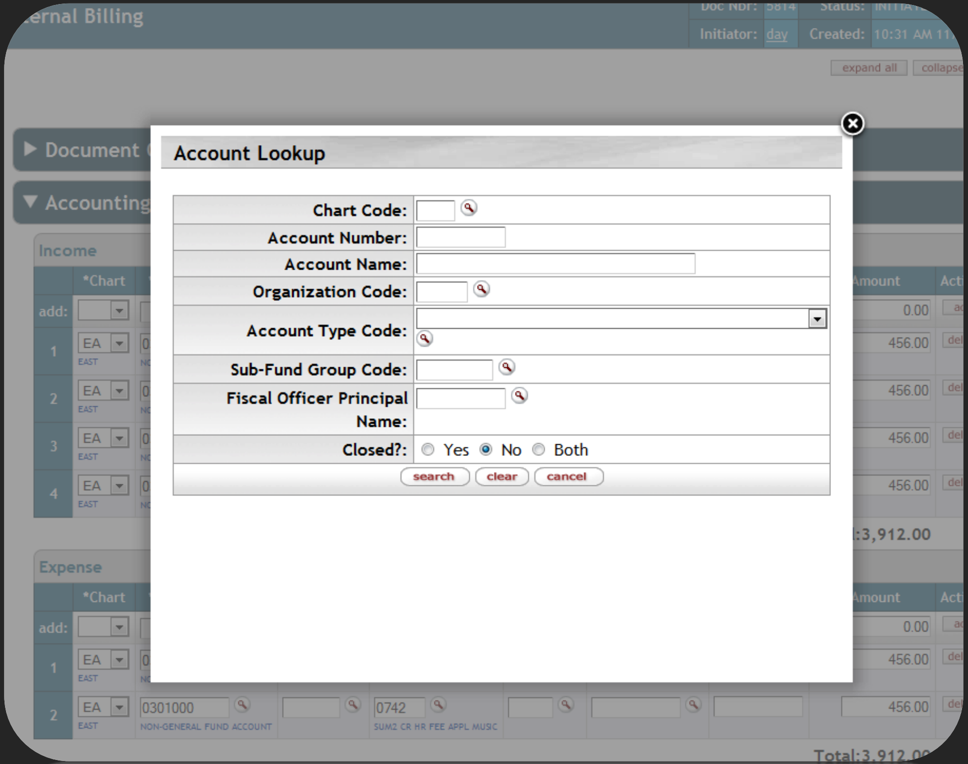
- KNS is Struts 1.x based
- Very little built-in rich user interface support
- User experience is designed more for administrative users
- Only has built-in support for a small set of screen types
- Note however, most of the core concepts from KNS are still relevant in KRAD

Why KRAD?

- Kuali Student has a wider variety of UX (user experience) requirements
- Need better support for "self-service" screens for which the KNS is not well suited
- A need for Web 2.0 and other Rich Internet Application features
- Support for more complex types of screens and layouts

KRAD Features – Rich UI

- Lightbox support for Inquiries, Lookups, Confirmations, and expanded Text Areas

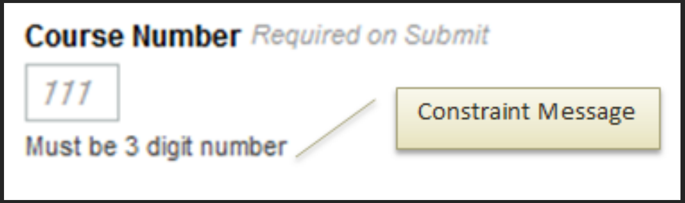


Account Lookup

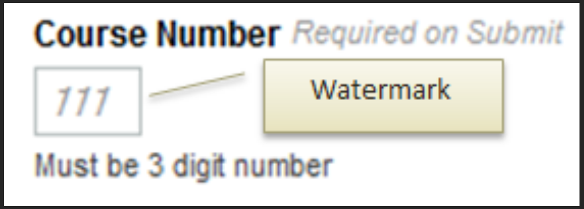
Chart Code:	
Account Number:	
Account Name:	
Organization Code:	
Account Type Code:	
Sub-Fund Group Code:	
Fiscal Officer Principal Name:	
Closed?:	Yes [✓ No] Both

KRAD Features – Rich UI

- Constraint Message – displays field restrictions

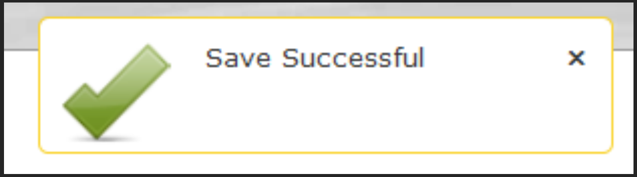


- Watermark – displays in text field (ex. date format)



KRAD Features – Rich UI

- Growls – notifications about events
- Built in growls for Save & Route



Save Successful

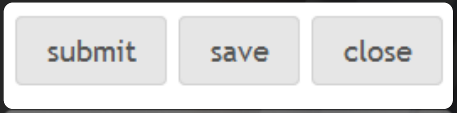
- Other Messages:
  - Roll over field level help
  - Always displayed field Summary
  - Page submit/load notification

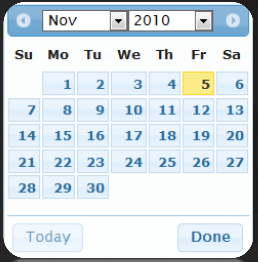
KRAD Features – Rich UI

- Progressive Disclosure
- Show information when needed
- Show/Hide
  - Sections (Tabs)
  - Groups (Parts of Tab)
  - Fields
  - Field Group (Grouping of fields)
- Server & Client side support

KRAD Features – Rich UI

- Client side validation
  - Automatic translation of dictionary validation to client validation script
- Improved Navigation (breadcrumbs)
- Text based button generation
- Auto-complete Fields
- Improved Calendar widget



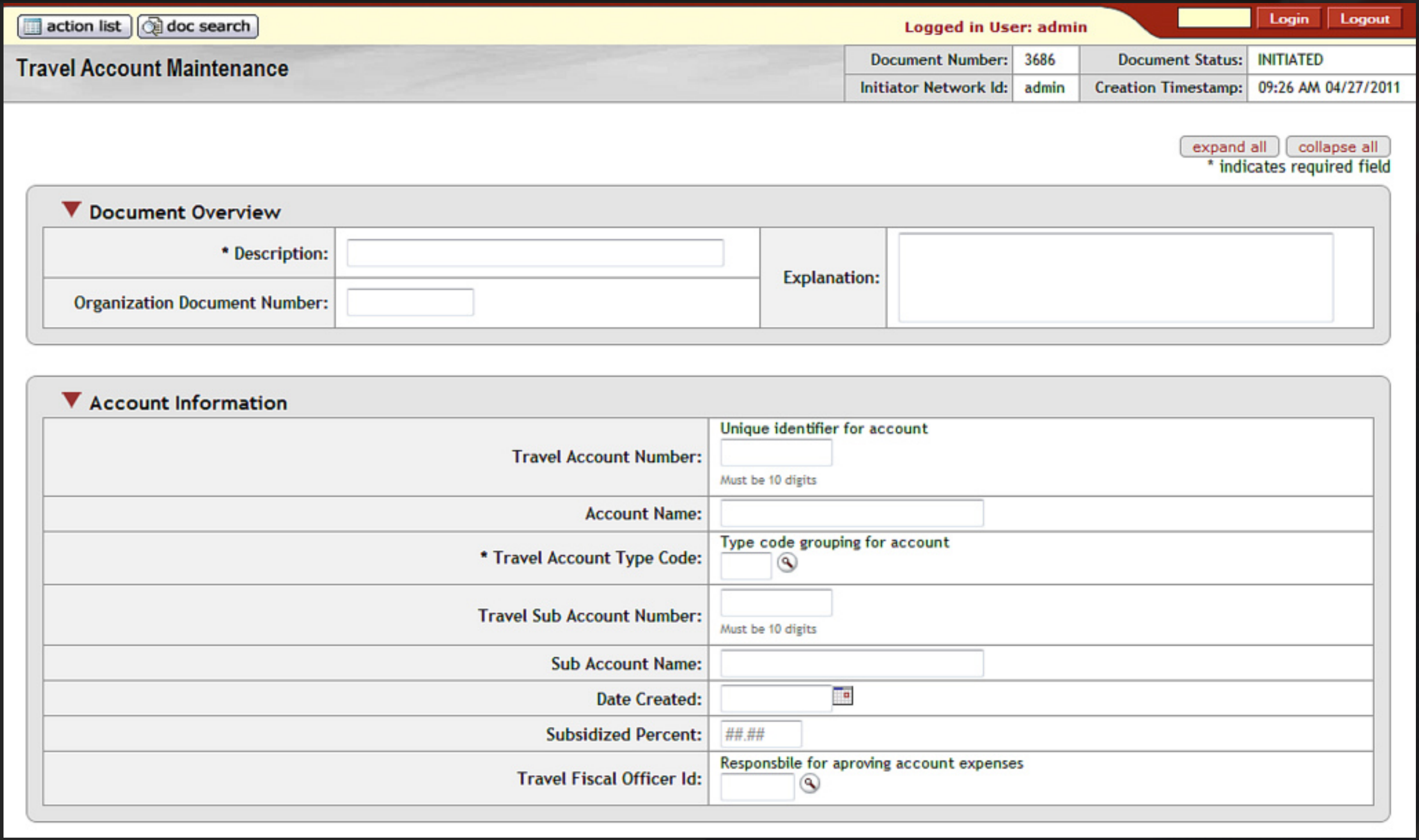


Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

KRAD Technology

- Spring MVC as the model-view-controller framework for KRAD
- Apache Tiles as the templating engine
- Fluid Skinning System for CSS
- jQuery as the javascript library
  - Including jQuery UI
  - And other plugins providing functionality like AJAX

Sample KRAD Screens – KNS Look and Feel



Travel Account Maintenance

Document Number:	3686	Document Status:	INITIATED
Initiator Network Id:	admin	Creation Timestamp:	09:26 AM 04/27/2011

* indicates required field

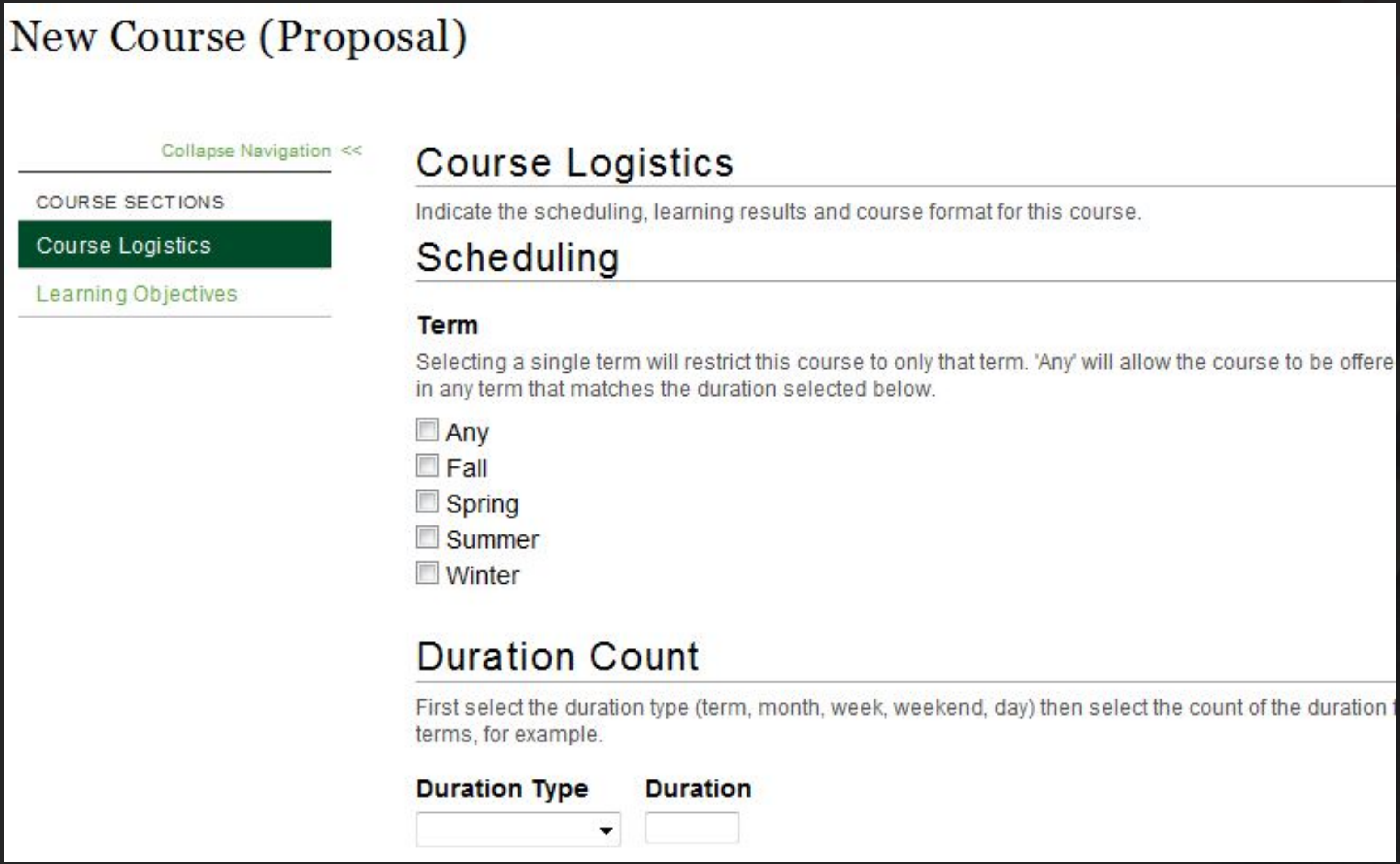
Document Overview

* Description:		Explanation:	
Organization Document Number:			

Account Information

Travel Account Number:	Unique identifier for account Must be 10 digits
Account Name:	
* Travel Account Type Code:	Type code grouping for account
Travel Sub Account Number:	Must be 10 digits
Sub Account Name:	
Date Created:	
Subsidized Percent:	##.##
Travel Fiscal Officer Id:	Responsbile for aproving account expenses

Sample KRAD Screens – KS Look and Feel



New Course (Proposal)

Course Logistics

Indicate the scheduling, learning results and course format for this course.

Scheduling

Term

Selecting a single term will restrict this course to only that term. 'Any' will allow the course to be offere
in any term that matches the duration selected below.

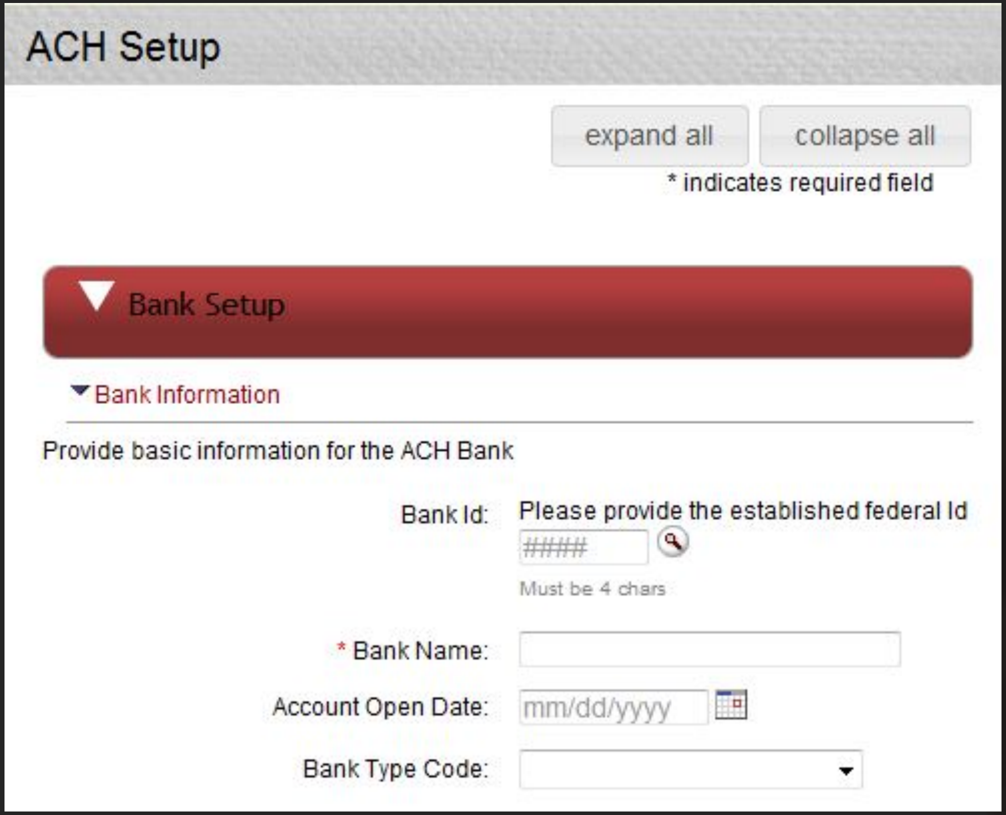
- [ ] Any
- [ ] Fall
- [ ] Spring
- [ ] Summer
- [ ] Winter

Duration Count

First select the duration type (term, month, week, weekend, day) then select the count of the duration
terms, for example.

Duration Type **Duration**

Sample KRAD Screens – Admin Look and Feel


ACH Setup

expand all | collapse all

* indicates required field

Bank Setup

Bank Information

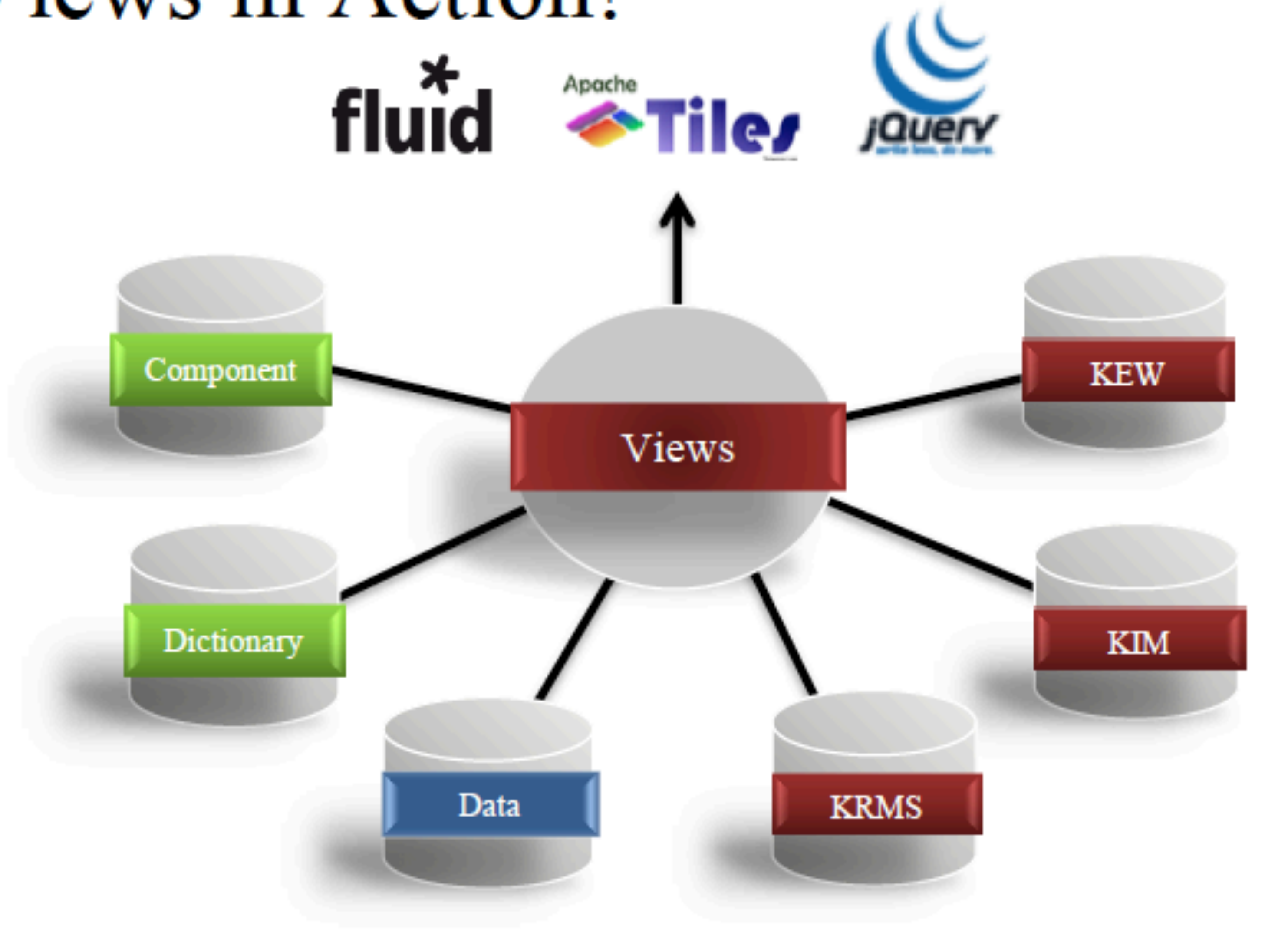
Provide basic information for the ACH Bank

Bank Id:	Please provide the established federal Id ##### Must be 4 chars
* Bank Name:	
Account Open Date:	mm/dd/yyyy
Bank Type Code:	

KRAD Views

- Full-working user interface functionality solutions that can be modified as needed
- Encompasses much of the existing KNS functionality such as lookups, inquiries, and maintenance
- Defined declaratively in Spring configuration as part of KRAD Component Framework
- Backed by Spring MVC
- Integrated with other Rice modules such as KEW, KIM, and KRMS

Views in Action!



After Rice 2.0

- Kuali Rice has a roadmap committee which works with project investors to assemble and maintain the Kuali Rice Roadmap
- Kuali Rice 2.1
  - Continued work on KRAD features
  - XML Data Import and Export tools
  - KEW workflow engine escalation

After Rice 2.0

- Kuali Rice 2.2
  - Implement KEW GUI for designing workflow processes
- Kuali Rice 2.3
  - Additional RAD Tools for application development
  - Update of Accessibility Standards
    - Kuali Rice has a dedicated UX Architect now
  - Batch Scheduler and Monitor

Questions?

- Thanks for Coming!
- Questions???
- http://rice.kuali.org
- Kuali Days 2011 – November 14-16 - Indianapolis, IN
  - Call for proposals open now!
  - http://www.kuali.org