

Automating the Enterprise

Workflow Automation with Open Source Tools

Schedule

- 8:30 - 10:00 First half
- Break: 10:00 - 10:30
- 10:30 - 12:00 Second half
- 12 - 1:30 Lunch Exhibit Hall 2

Slides?

- Yes, slides will be available after tutorial.

Topics

- Introduction
- Business Process Management (BPM)
- Automating Business Processes
- Workflow Components
- Starting Your Own BPM

Introduction

About Me

- Jim Brandt
- Lead Developer, Synacor, Inc.
- President, The Perl Foundation
- "mod_perl 2 User's Guide"

Background

- Built workflow system for University at Buffalo
- E-requisition: all campus requisitions
- EPTF: Personnel Transaction Forms (HR)

Background

- Initially implemented workflow with a commercial BPM system
- We built the requisition app to talk to the purchased workflow app
- Faced double upgrade (two major versions)
- Eventually replaced commercial product with open source product

Business Process Management (BPM)

AKA

- Business Process Automation
- Business Process Modeling
- Workflow
- Workflow Automation

Borrowed Ideas

- IT ticketing system
- Email application
- To Do tracker
- Business application
- Reporting engine

IT-centric Definition

- Analyzing business processes
- Building an engine to manage process status
- Manage the business rules for moving from one step to another (actions)
- Examine the steps and automate where possible
- Integration between different systems

Part Human

- Facilitate communication between people involved in the process (visibility)
- Single, unified view of the process and the steps in the process (transparency)
- Workflow processes (steps) not all automated
- People still the core of the process

Part Computer

- Automate steps where possible
- Automate communication between disparate systems
- Track/audit steps and people
- Reporting and analysis

BPM Advantages

- Fully document your business process
- Can keep business rules separate from tools
- Automate some parts
- Can discover the state of any work in the system (reporting)
- Can expose parts to external systems (including other companies)

A Business Process: Purchase Requisition

Paper-based Purchase

- Get a purchase form.
- Fill it out.
- Mail to appropriate approvers (boss, department head, etc.).
- Mail to purchasing.
- Purchasing routes it, possibly keying information in various systems.

Electronic Purchase

- A quick tour of an electronic requisition



Initiate

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process:

Workflow Manage Tasks Reports

Initiate Inbox Completed Queue User Preferences

Select Process

Set Process

Process Name

[Print Purchase Order](#)

[Purchase Requisition \(eReq\)](#)

UB eRequisition

Doc Type:	Regular	Incident #:	Unassigned	Creator:	Brandt, Carl
Created:	04/14/2009	Requisition #:	Unassigned	Last Approver:	
Total:	0.00	PO #:	Unassigned	Supplier:	

[Order Details](#)
[Order Items](#)
[Supplier](#)
[Ship to](#)
[Funding](#)
[Notes](#)
[Attachments](#)
[Summary](#)
[My Actions](#)

*Funding Source:

*Ordering Instructions:

Sponsor Defined Category: ☐ (If applicable for RF Acct)

Commodity:

Buyer:

Contract#:

Group #:

Select when applicable for animal purchases only

Lab Animals:

IACUC Number: (9 digit field required for Lab Animals: 3 alpha, 5 numeric, Y or N.)

Animal Care Plan:

Animal Care Plan Number:



UB eRequisition

Doc Type:	Regular	Incident #:	Unassigned	Creator:	Brandt, Carl
Created:	04/14/2009	Requisition #:	Unassigned	Last Approver:	
Total:	0.00	PO #:	Unassigned	Supplier:	

Order Details Order Items Supplier Ship to Funding Notes Attachments Summary **My Actions**

*Select Action

Route to Approver (select recipient below)

Send to Approval Group

Select One:

Or, send to Approval Person

Select one:

Save and Finish

Select Action

Your selected action will result in routing the incident forward (any submit or approval action), backward (reject), or canceling (cancel). Saving your work (Save without Submit) is also possible. Note that not all actions are possible at each step in the process.

Send To

ONLY REQUIRED WHEN ACTION IS "ROUTE TO APPROVER" AND IS IGNORED FOR OTHER ACTIONS. Select either a group or a person to route to. Should you select both a person and a group, the person will be utilized as it is more specific. Note that sending to a group may result in faster response.

Save and Finish

Based on your selections, clicking Save and Finish will route your work to the appropriate next step in the process.

Inbox

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process: Ereq

[Workflow](#)
[Manage Tasks](#)
[Reports](#)

[Initiate](#)
[Inbox](#)
[Completed](#)
[Queue](#)
[User Preferences](#)

Ereq 
 Set Process
 Sort by: Incident  Descending 
 Limit to: 10 results per page
 Set Display

1 Incidents Found 1

Incident	Summary	Step	Requested Delivery	Start Time	Owner	Initiator	Supplier Name	Hold Reason	History
100306	Req dly not specified	Begin	None	11/09/2007 12:12:46PM	Carl Brandt	Carl Brandt			History

1 Incidents Found 1

Completed

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process: Ereq

[Workflow](#)
[Manage Tasks](#)
[Reports](#)

[Initiate](#)
[Inbox](#)
[Completed](#)
[Queue](#)
[User Preferences](#)

Ereq
 Sort by: Incident
 Show days of data
 Limit to: results per page

Incident number From: To:

1 Incidents Found 1

Incident Complete	Incident	Req ID	PO #	Step	Requested Delivery	Time Completed	Owner	Initiator	Supplier Name	Hold Reason	History
	100306	118737		Begin	None	04/14/2009 12:36:06 PM	Carl Brandt	Carl Brandt			History

1 Incidents Found 1



Completed

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process: Ereq

Workflow

Manage Tasks

Reports

Initiate

Inbox

Completed

Queue

User Preferences

Ereq

Set Process

Historical status for incident #100306

Step	Client	Completion Time	Action	Status
Begin	Carl Brandt	04/14/2009 12:36:06 PM	Route to Approver (select recipient below)	Completed
Approver1	Gregory Howland	*****	*****	Active

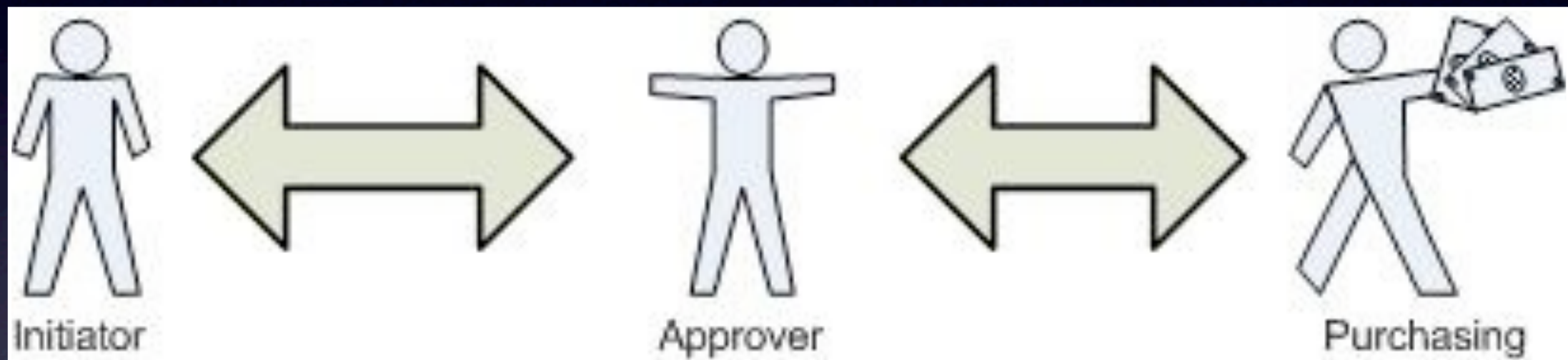
Workflow Components

Workflow Components

- Model (Map, Business Process)
- Workflow Engine
- Workflow Client
- Workflow API: Describe the workflow
- Workflow API: Act on incidents
- Reporting

Model

- Business process described in a configuration or model
- Also called a workflow or map
- Determines how incidents (tasks, processes) move from step to step
- These are your business rules



Simple Workflow

Formal Tools

- Can be designed using standards like Business Process Modeling Notation (BPMN)
- Eclipse plug-ins available
- Can export Business Process Execution Language (BPEL) to run in some engines

GUI Design

- Idea: Tech-savvy business users can use the GUI to design the business process
- The tool converts the design into something an engine can run
- Not yet realistic in my experience
- Can still be a design tool

Workflow Engine

- Implements the model
- API to examine incidents (what state? what actions available?)
- API to create and operate on incidents
- Steps (maybe automated) to run when acting on incidents

UB eRequisition

Doc Type:	Regular	Incident #:	Unassigned	Creator:	Brandt, Carl
Created:	04/14/2009	Requisition #:	Unassigned	Last Approver:	
Total:	0.00	PO #:	Unassigned	Supplier:	

[Order Details](#)
[Order Items](#)
[Supplier](#)
[Ship to](#)
[Funding](#)
[Notes](#)
[Attachments](#)
[Summary](#)
[My Actions](#)

*Select Action

Route to Approver (select recipient below) 

Send to Approval Group

Select One: 

Or, send to Approval Person

Select one: 

[Save and Finish](#)

Select Action

Your selected action will result in routing the incident forward (any submit or approval action), backward (reject), or canceling (cancel). Saving your work (Save without Submit) is also possible. Note that not all actions are possible at each step in the process.

Send To

ONLY REQUIRED WHEN ACTION IS "ROUTE TO APPROVER" AND IS IGNORED FOR OTHER ACTIONS. Select either a group or a person to route to. Should you select both a person and a group, the person will be utilized as it is more specific. Note that sending to a group may result in faster response.

Save and Finish

Based on your selections, clicking Save and Finish will route your work to the appropriate next step in the process.

Application Tool or Stand-alone App

Embedded or Stand-alone

- Workflow can be integrated with a work tool (application)
- Can be a stand-alone system interfacing with work tools

Why Built-in Workflow

- Because customers asked for it?
- Maybe

Why Built-in Workflow

- Key feature for the application?
- Maybe

Why Built-in Workflow

- Lock you in by getting all of your business rules in their product?
- Probably

Why Built-in Workflow

- All of the above
- If most work is done in a product (ERP)
- Internal APIs will be robust
- External APIs could be iffy (if available)
- Could be a step toward stand-alone workflow

Stand-alone Workflow

- De-couple business rules from other software products
- Allow communication between different products in your environment
- Single source for reporting
- Re-use with different types of work across the company

Hybrid Approach

- Have some stand-alone workflow
- When incidents pass to a product, launch in-product workflow
- Pass back to stand-alone system when steps are complete

Hybrid Approach

- Stepping stone when some dependent systems are missing
- Work with systems that have limited APIs

Hybrid Approach

- Can lose visibility of work in the product
- Could become a black hole for incidents
- Reporting less effective

Open Source Workflow

- Your business rules remain yours
- Know what the engine is doing
- Own the database

Evaluating BPM Products

Evaluating Options

- Evaluation criteria for BPM tools
- Vendor product may be a good solution
- Open source allows you to try
- Evaluation criteria valid for both

Initial Implementation

- Needed workflow and requisition apps
- Build one, buy the other
- Full evaluation to pick a strong vendor

Initial Implementation

- Gartner and other industry ratings
- Product demos
- Reference calls

Rapid Deployment

- Consulting to build our processes
- Provided Workflow client (inbox, etc.)
- Visual (GUI) design tool
- Extensive delivered APIs (SOAP, database, etc.)

Evaluation Item

- Consulting/Professional Services
- Was helpful, as we were new to BPM
- Experienced third-party analyst can be good resource
- They know the product
- Focus all team members

Ongoing Support

- Problems with support
- Regression on releases
- Often external mailing list more helpful

Evaluation Item

- Ask other customers about support
- Find out about mailing lists and community support during evaluation
- Investigate paid support for open source systems

Additional Issues

- Workflow client not accessible (very IE-centric)
- Server had stability issues and would bring down client
- Server had performance problems under load

Evaluation Item

- Ask about accessibility (run checks in demos)
- Ask about performance and throughput on your planned hardware (duplicate with prototype install)

API Trouble

- Some APIs didn't work well
- Important SOAP API returned ambiguous results

API Trouble

- What does a 'true' return code mean to you?
 - Task completed?
 - Task request received?
- Theirs was number 2

Evaluation Item

- Ask about and test the APIs you plan to use most
- Get a reference who is using the APIs you plan to use

Changing the Model

- Difficult to change the model when processes were active
- Database design confusing; hard to troubleshoot and fix things
- General poor database tuning

Evaluation Item

- Ask about changing the model and possible impacts
- Ask about policy for the database (look but don't touch?)
- How many customers using your server, DB, operating system?

Open Source

- Facing a double version upgrade
- Feedback on the mailing list indicated each was significant work
- Prototype open source Perl solution

Use Evaluation Items

- Used these evaluation items in prototype
- Had already built separate accessible workflow client
- Created simplified sample workflows and benchmarked (10x improvement)
- Made API calls transactions

Use Evaluation Items

- Could see the database schema
- Simple enough to handle model changes
- APIs were Perl, using established CPAN modules (SOAP, DBI, etc.)

Use Evaluation Items

- Mailing list was not active, but module was maintained
- Willing to accept patches

Use Evaluation Items

- Download software and build simple workflows
- Use these evaluation criteria
- Add your own criteria

Please submit
evaluations

Topics

- Introduction
- Business Process Management (BPM)
- Automating Business Processes
- Workflow Components
- Starting Your Own BPM

Workflow Engine

Workflow Components

- Model (Map, Business Process)
- Workflow Engine
- Workflow Client
- Workflow API: Describe the workflow
- Workflow API: Act on incidents
- Reporting

Workflow Engine

- Core rules defining the business process
- API to examine incidents
- API to create and operate on incidents
- Steps (maybe code) to run when acting on incidents

Workflow Factory

- The Workflow.pm module from CPAN
- Singleton that loads the configuration
- Core methods to get a workflow object

```
use Workflow::Factory qw( FACTORY );

# Add XML configurations to the factory
FACTORY->add_config_from_file(
    workflow => 'workflow.xml',
    action   => [ 'myactions.xml', 'otheractions.xml' ],
    validator => [ 'validator.xml', 'myvalidators.xml' ],
    condition => 'condition.xml',
    persister => 'persister.xml' );

# Create a new workflow of type 'MyWorkflow'
my $wf = FACTORY->create_workflow( 'MyWorkflow' );

# Fetch an existing workflow with ID '25'
my $wf = FACTORY->fetch_workflow( 'MyWorkflow', 25 );
```

Configuration

- Workflow.pm configuration is XML or Perl data structures
- TODO: Interface in some way with BPEL
- TODO: Graphical representations of configs

Main Config

Building the model

- The rules for the workflow
- All of the states the process can be in
- The actions you can take from each state

Building the model

- Keep states and rules for transition in the model
- Keep tool-specific items in the external tools
- Build mapping (workflow_id to req_id)
- Try to maintain loose coupling

```
<workflow>  
<type>Requisition</type>  
<observer  
  class="Workflow::Observer::  
NotificationObserver" />  
<description>Electronic Requisition  
  </description>  
<persister>MainDatabase</persister>
```

Basic Config

- Some basic configuration of the main workflow
- The type tag allows you to build multiple types
- Can load these and run many workflow types from a single engine
- Persister is the data source (DB, SQLite)

```
<state name="Approval"  
description= "Approve Req">  
  <action...>  
  <action...>  
  <action...>  
</state>  
<state name="AccountCheck"  
description="Check account balance"  
autorun="yes">
```

States

- State tags define process states
- States have one or more actions to perform to get to another state
- Autorun states run as soon as they are called (this is your automated workflow)
- A state can have many possible actions

```
<action name="SubmitForApproval">  
  <resulting_state return="Approver"  
    state="Approver" />  
  <resulting_state return="Buyer"  
    state="Buyer" />  
  <condition name="Authorize" />  
  <condition admin="$context->{user} ne  
    'admin' " />  
</action>
```

Actions

- How to get from one state to another
- Resulting state is the state the workflow will be in if executed successfully

Conditions

- Conditions test something to verify you should get to the next state
- Workflow exposes only actions you can execute based on conditions
- Conditions can be in-line or in a module

Action Config

```
<actions>  
<type>Requisition</type>  
  <action name="SubmitForApproval"  
    class="MyApp::Action::SFA">  
</action>  
  <action name...  
  <action name...
```

Actions

- Map your action to Perl code (finally!)
- Workflow looks for sub 'execute' and runs it
- Just return for success
- If you have multiple possible next states, return a string with the destination state
- Throw Workflow::Exception to fail

```
package MyApp::Action::SFA;

use base qw( Workflow::Action );
use Workflow::Exception qw( workflow_error );

sub execute {
    my ( $self, $wf ) = @_;
    my $context = $wf->context;
    ...
    return "Approver";
}
```

Actions

- You get a Workflow object
- On success, workflow will change the state
- You change your data (extra data)
- You call `add_history` for updates

Workflow::Context

- Module to store data to pass between Workflow components
- Set it yourself when you create the workflow
- Access in Actions, Conditions, Validators
- Get data auto-loaded with ExtraData

Action Types

- Give actions a type if you have multiple workflows with the same actions, but different behaviors
- Generic actions like Submit or Cancel might act differently in different Workflows

```
<action name="AccountCheck"  
  class="MyApp::Action::AccountCheck">
```

```
<field name="acctnum" label="AcctNum"  
  description="Charge acct num."  
  is_required="yes" />
```

Action Fields

- Define fields needed for your action
- External apps can use the API to read possible fields
- Missing required fields will throw an exception

```
<action name="AccountCheck"  
  class="MyApp::Action::AccountCheck">
```

```
  <validator name="purchase_limit">  
    <arg>$total_cost</arg>  
  </validator>
```

Validators

- Can run validators as part of the workflow
- Many ways to do validation, esp. from the web
- Advantage is core requirements are "documented" with the workflow
- Less code in action
- Reusable

Persisters

(Data Store)

```
<persister name="MainDatabase"  
  class="Workflow::Persister::DBI"  
  driver="MySQL"  
  dsn="DBI:mysql:database=workflows"  
  user="wf"  
  password="mypass"/>
```

Persister Config

- Name to use in workflow config
- Details to initiate data store
- Subclassable with these options:
 - DBI, so anything with a DBI driver
 - File-based persister

Workflow Data

- Main workflow table to store the basics
 - workflow id, type, state, timestamp
- Workflow history tracks all activity on an incident, with user if you have it
 - Must call manually from actions via `add_history` method

Workflow Data

- Tables are simple enough to update when needed
- Possible to modify the model without damaging in-process incidents

More Data

In persister config

extra_table="workflow_ticket"

extra_data_field="ticket_id, due_date"

- Another table to store workflow details
- Workflow will load fields into context
- Need to manually write data back

Conditions

```
# workflow_conditions.xml
<conditions>
  <condition
    name="IsAdminUser"
    class="MyApp::Condition::IsAdminUser">
    <param name="admin_group_id"
value="5" />
  </condition>
```

```
# workflow.xml
```

```
<workflow>
```

```
  <state>
```

```
    ...
```

```
    <action name="SomeAdminAction">
```

```
      ...
```

```
      <condition name="IsAdminUser" />
```

```
    </action>
```

Conditions

- Enforce rules for actions
- For a given context, determines if an action is available
- Define in condition config and map to package
- Apply to selected actions in main workflow config

```
package MyApp::Condition::IsAdminUser;
use base qw( Workflow::Condition );
use Workflow::Exception;

sub evaluate {
    my ( $self, $wf ) = @_;
    my $current_user = $wf->context-
>param( 'current_user' );
    unless ( $self->is_admin($current_user) ) {
        condition_error "Not an admin user";
    }
}
```

Conditions

- Create an 'evaluate' sub
- Throw a condition error if it fails
- return if it's OK
- By default, conditions are cached

Observers

Watching Workflow

- Trigger related events
- Notifications (email)
- Watch for certain conditions (req over \$100,000)
- Can run code on:
create, fetch, execute, state change

In main workflow config

```
<workflow>
```

```
  <type>Requisition</type>
```

```
  <description>This is...</description>
```

```
  <observer class="SomeObserver" />
```

Or

```
  <observer sub="SomeOther  
Observer::Functions::other_sub" />
```

Two Methods

- Can define a class, workflow will execute the sub 'update'
- Define a subroutine in a class, workflow will execute that subroutine
- Loaded when the engine is started

Limitations

Workflow.pm

Limitations

- Can't do parallel workflows
- No visualization of workflow or actions
- Configuration in XML (could be a plus)

Workflow Client

Workflow Client

- Primary user interface to workflow system
- Start with stock web framework
- Authenticated, system knows the current user
- Interface with some permission system (org chart, LDAP, etc.)
- Options appropriate for the user/role



Initiate

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process:

Workflow Manage Tasks Reports

Initiate Inbox Completed Queue User Preferences

Select Process

Set Process

Process Name

[Print Purchase Order](#)

[Purchase Requisition \(eReq\)](#)

Starting a Task

- Tab to show all tasks available
- Links to workflow-integrated apps
- Simple set of links
- Context sensitive for the current user
- May not actually start the workflow

Inbox

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process: Ereq

[Workflow](#)
[Manage Tasks](#)
[Reports](#)

[Initiate](#)
[Inbox](#)
[Completed](#)
[Queue](#)
[User Preferences](#)

Ereq
 Sort by: Incident
 Limit to: results per page

1 Incidents Found 1

Incident	Summary	Step	Requested Delivery	Start Time	Owner	Initiator	Supplier Name	Hold Reason	History
100306	Req dly not specified	Begin	None	11/09/2007 12:12:46PM	Carl Brandt	Carl Brandt			History

1 Incidents Found 1

Inbox

- Basic email inbox metaphor
- Work that has been routed to you
- Links to the appropriate next step
- Links to history for context

Inbox

- Direct query against workflow tables
- All active steps owned by current user
- Other information as desired for context
- May be different for different workflow types

Groups

- Way to route work to a group of people
- Similar to inbox, but for many people
- Allow users to 'Take' an incident
- Org chart or other system to determine group membership

Completed

[Refresh](#) | [FAQ](#) | [Help](#)

Active Process: Ereq

Workflow | Manage Tasks | Reports

Initiate | Inbox | **Completed** | Queue | User Preferences

Ereq
 Sort by: Incident
 Show 15
 Limit to: 10 results per page

Incident number From: To:

1 Incidents Found 1

Incident Complete	Incident	Req ID	PO #	Step	Requested Delivery	Time Completed	Owner	Initiator	Supplier Name	Hold Reason	History
	100306	118737		Begin	None	04/14/2009 12:36:06 PM	Carl Brandt	Carl Brandt			History

1 Incidents Found 1

Completed

- Tasks that you have worked on, but are no longer yours
- Can track where a task is
- Might show:
 - only active incidents
 - incidents for a certain time period

Completed

- More queries against workflow tables
- Similar customization as inbox
- May have different columns that are relevant when task has passed you

UB eRequisition

Doc Type:	Regular	Incident #:	Unassigned	Creator:	Brandt, Carl
Created:	04/14/2009	Requisition #:	Unassigned	Last Approver:	
Total:	0.00	PO #:	Unassigned	Supplier:	

[Order Details](#)
[Order Items](#)
[Supplier](#)
[Ship to](#)
[Funding](#)
[Notes](#)
[Attachments](#)
[Summary](#)
[My Actions](#)

*Select Action

Route to Approver (select recipient below) 

Send to Approval Group

Select One: 

Or, send to Approval Person

Select one: 

[Save and Finish](#)

Select Action

Your selected action will result in routing the incident forward (any submit or approval action), backward (reject), or canceling (cancel). Saving your work (Save without Submit) is also possible. Note that not all actions are possible at each step in the process.

Send To

ONLY REQUIRED WHEN ACTION IS "ROUTE TO APPROVER" AND IS IGNORED FOR OTHER ACTIONS. Select either a group or a person to route to. Should you select both a person and a group, the person will be utilized as it is more specific. Note that sending to a group may result in faster response.

Save and Finish

Based on your selections, clicking Save and Finish will route your work to the appropriate next step in the process.

Actions

- Tab with available actions for the current workflow state
- May need to build this in the workflow client
- Call to workflow engine for available actions

Building APIs

Query API

- Methods to:
 - get valid actions
 - get information needed to perform actions
 - get incident metadata (current owner, past owners, current state, etc.)

Perl Solution

- Create a module to provide the API
- Other Perl code can use module directly
- SOAP interface with `Apache::SOAP` (`mod_perl`)
- RSS interface with `XML::RSS` (`inbox`, etc.)

Workflow.pm

Wrap an API around these methods

```
my $wf = Workflow::Factory-  
>create_workflow( 'Ticket' );
```

```
my @actions = $wf->get_current_actions;
```

```
my @fields = $wf-  
>get_action_fields( 'Submit' );
```

Topics

- Introduction
- Business Process Management (BPM)
- Automating Business Processes
- Workflow Components
- Starting Your Own BPM

Is it for me?

When to consider BPM

- Have business processes that:
 - Involves several products/people/dept.
 - Has steps that can be automated
 - You want documented and audited

What's Available

- Workflow.pm module
- CPAN (for integration)
- Many web frameworks

Not Available

- The user client (inbox, completed, etc.)
(web framework of choice)
- External API (SOAP, web services) (small module and Apache::SOAP)
- Reporting (web framework)
- Interfaces to other apps (CPAN)

What's Available

- If you have Perl apps, can embed Workflow.pm to offer workflow
- Can be completely hidden

What's Available

- Intalio: <http://www.intalio.com>
- ProcessMaker: <http://www.processmaker.com>
- Cute:flow: <http://www.cuteflow.org>

Might be available

- An internal organizational chart for authorization
- Existing applications that have some API or ability to extend.
- Documentation on internal processes
- Mini-workflows you can tap into

The Project

BPM Project

- Everyone has a different idea what your business process are...
- ...even people who sit next to each other in the same department
- This could be a concern

BPM Project

- If you're asked to estimate the tech part:
 - The technical part won't take the most time
 - It's very doable
 - Interfaces with external apps take time
 - Building supporting systems you didn't know you needed takes time

Changing Processes

- Forces you to examine your processes
- "We've always done it that way."
- Some may need to be changed

Changing Processes

- Another dimension to the project (and more time)
- Training on the new system, and new processes
- Can cause resistance
- Can reduce unjustified complexity

Buy-in

- Need buy-in at the 'C' level
- Can create mistrust

Buy-in

"Before we replace you, please teach this robot to do your job."

Buy-in

"Let's automate the boring part of your job so you can do something fun, like think of ways to make things better."

Buy-in

- If successful, the process owners will become the biggest advocates
- Should produce time-savings
- Digitized everything

Questions

(Yes, slides will be available after tutorial.)