ThoughtWorks®

Rocking the Enterprise with Ruby

Sudhindra Rao
Munjal Budhabhatti

Ruby in the Enterprise

"Enterprise software is a system that is used to support a business and its growth."

Ruby in Enterprise

- Domain
- Technology and Delivery
- Architecture
- Integration with the Enterprise
- OSS

Understanding the problem domain

- Server and network management
- Location and Power management
- Address management
- Monitoring and Diagnosis
- Customer Support
- Back Office Support

IP Automation

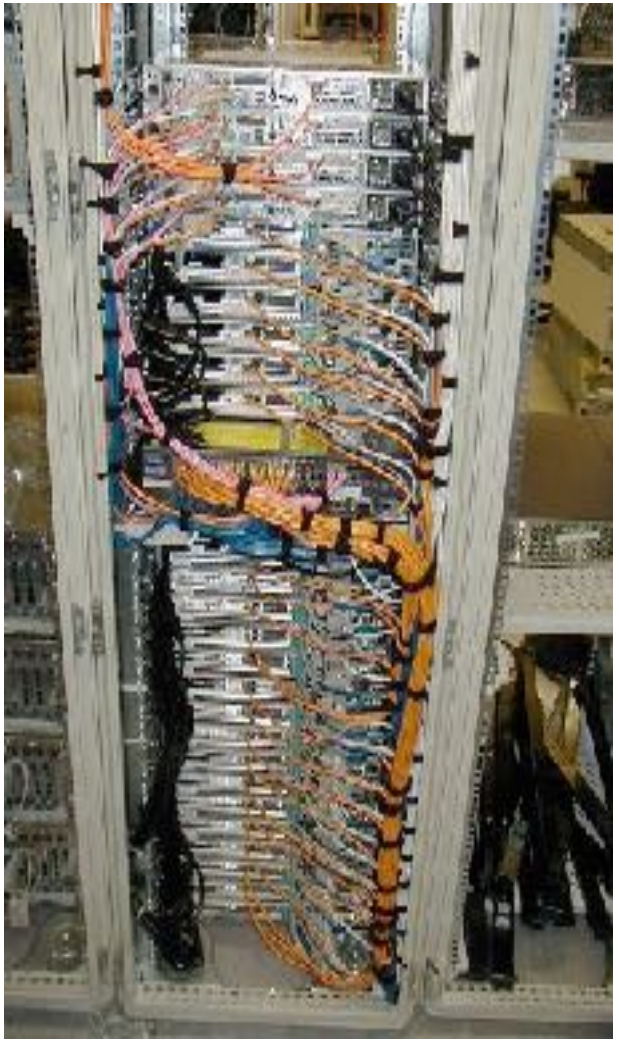
- Limited address space with IPv4
- IP Blocks management
- Governing body compliance
- Public, Private IPs
- Public, Private Networks
- IPv6 support

DNS Automation

- Huge database
- Security concerns over routing
- Legal implications
- Accuracy
- Systems support
- Always on

Network Device Manager

- ruby DSL for network devices
- Commission new devices
- Intelligent command chains
- Enhanced logging for debug support



Technology

- Ruby
  - ease of use and refactoring
  - natural language syntax to support varied applications
  - allows iterative addition to features
  - expertise - ThoughtWorks and Rackspace
  - Rails - naturally built for web applications
- Fast ramp-up for developers
- Spike for first small project - achieved goal at 40% cost of off-the-shelf product

Delivering Continuously

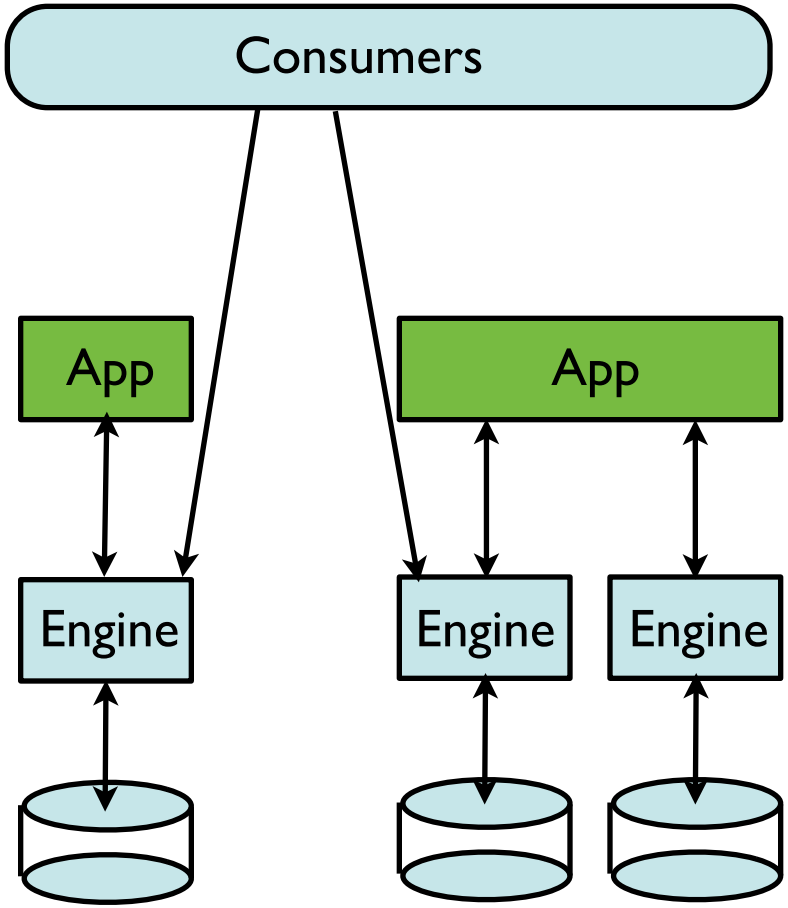
- Discipline
  - Identify focused user base
  - Strict control on features
  - Delivering frequently - 2 weeks
  - Small testable features
  - Consistent code coverage
- Feedback
  - Adapt architecture
  - RESTful interface

Architecture

- Remodelling to adapt to usage changes
  - Physical Space Project
    - Combined concepts of Physical and Virtual world
    - Data collection, usage, and consistency issues
  - Redesign
    - Identify User groups and Patterns
    - Split the domain
- Changing Domain abstractions
- Distributed transactions
- Pluginization to reuse functionality

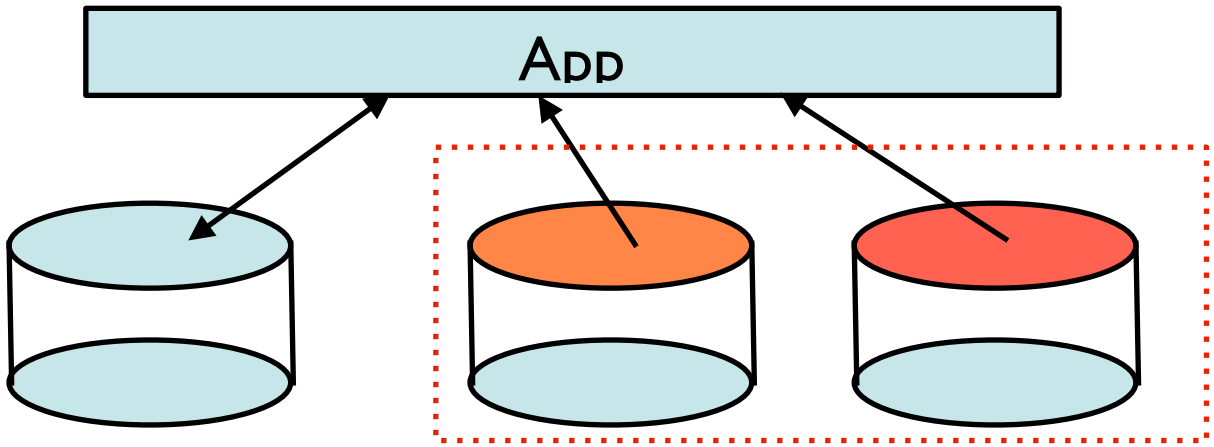
Design

- REST Engine
- Web based UI - Consumer
- Rails templates and resource_full
- Distributed testing
- Selenium suite



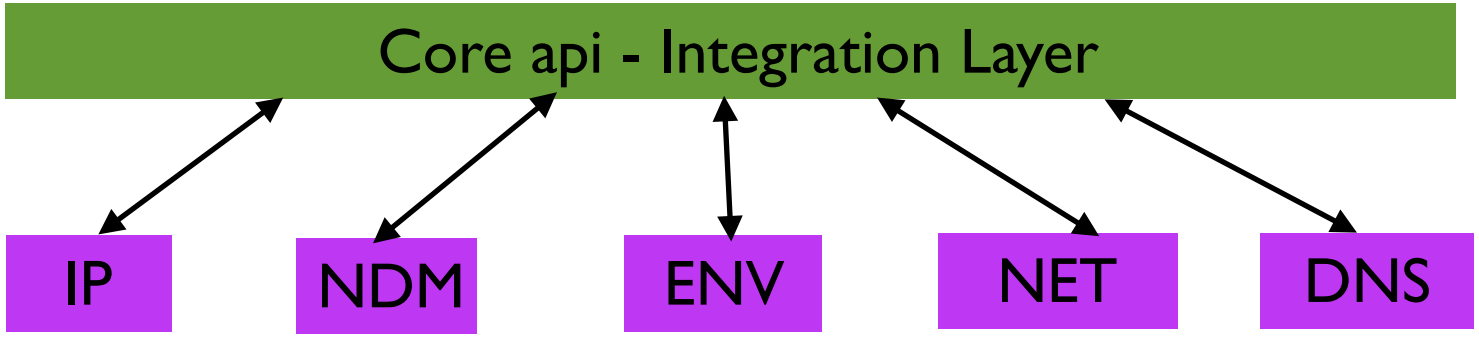
Pluginized Models

```
def setup_load_paths
  extra_paths = Dir.glob(plugin_root + "/app/*/")
  ActiveSupport::Dependencies.load_paths.unshift(*extra_paths)
  $:.unshift(*extra_paths)
end

if should_enhance_rake_tasks?
  Rake::Task["db:drop"].enhance(["#{plugin_name}:db:drop"])
  Rake::Task["db:create"].enhance(["#{plugin_name}:db:create"])
end
Rake::Task["db:migrate"].enhance(["#{plugin_name}:db:migrate"])
```



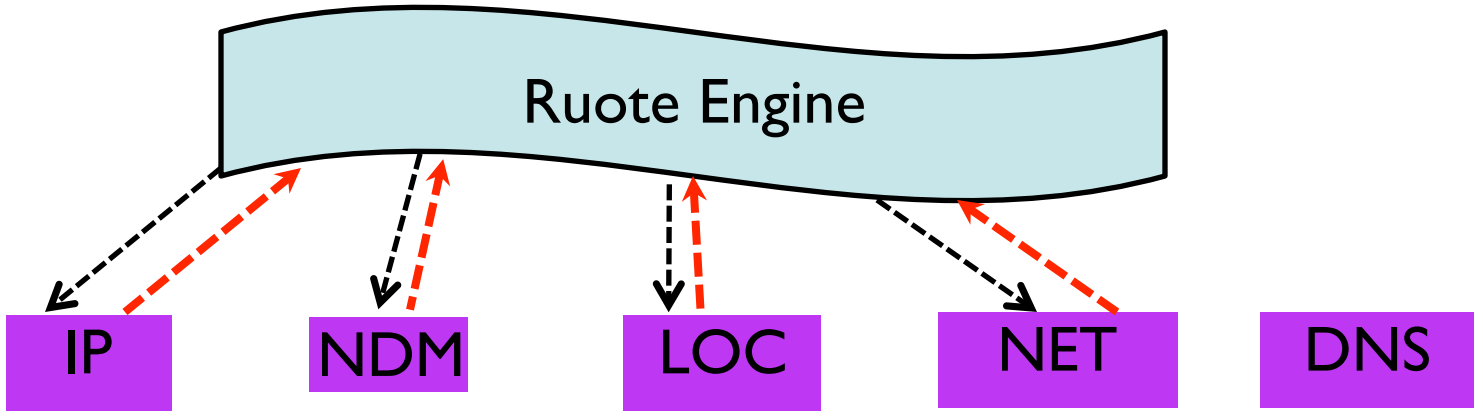
Integrating with the Enterprise

- Comprehensive reporting
- Repurpose with API
- Workflow engine
- Enterprise Message System



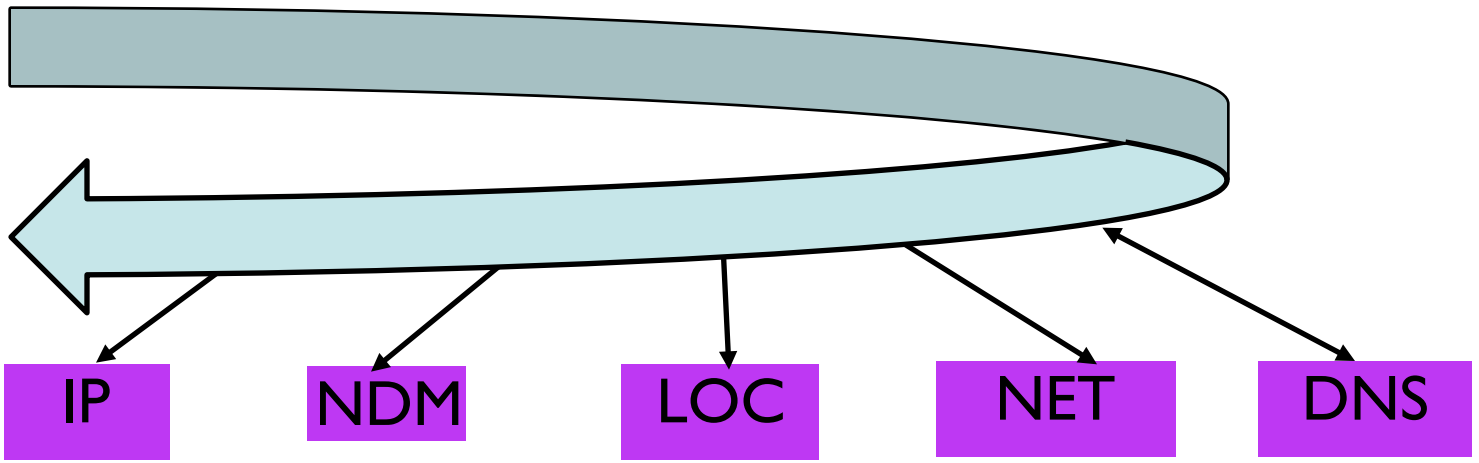
Workflow Engine

- Ruote - DSL based workflow engine
- RESTful services orchestration and Rich abstractions
- Example



Enterprise Message System

- RabbitMQ - a highly reliable, available, and scalable EMS
- Decoupling when Integrating with external systems
- Distributed transactions



OpenSource

- resource_full
- distributed testing
- cc monitor

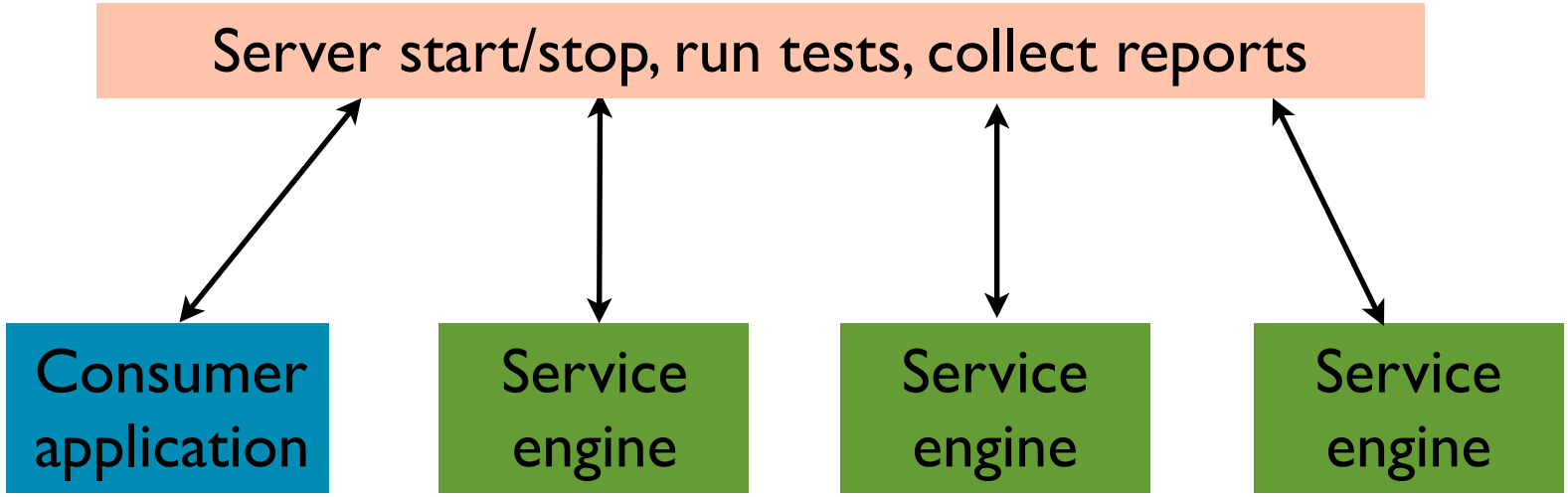
resource_full

- Fully compliant ActiveResource Built on ActionController
- RESTful parameter queryability, paging, etc.
- template for making engines REST easy
- template for REST consumers
- Typical functionality Out of the box
- Extend to return different stream types - json, xml
- http://github.com/bguthrie/resource_full

resource_full

```
class UsersController < ResourceFull::Base
  identified_by :username, :unless => lambda { |id| id =~ /^[0-9]+$/ }

  queryable_with :city, :state, :from => :address
  queryable_with :name, :columns => [:first_name, :last_name]
  queryable_with :email_address, :fuzzy => true
  queryable_with :is_active, :scope => :active

  orderable_by :city, :from => :address

  responds_to :html
  responds_to :xml, :only => [:read, :update]
end

class AddressesController < ResourceFull::Base
  nests_within :users
  queryable_with :city, :state
end
```

Distributed Testing



Distributed Testing

```
#Plugin code
require rails_root("/vendor/plugins/integration/lib/integration/helper")
include Blackbox::Integration::Helper
start_dependencies( :app1, :engine1, :location_engine )


#spec across engine and application
it "should find all data_centers" do
  data_center1, data_center2 = engine_drb.setup(<<-DATA, __FILE__, __LINE__)
    data_center1, data_center2 = create! :data_center, :quantity => 2
    [data_center1, data_center2]
  DATA
  DataCenter.find(:all).should == [ data_center1, data_center2 ]
end
```

Distributed Testing

- DRb based integration testing
- Integrating multiple engines to run a test that spans the domain
- Unit tests/functional tests to ensure quality of each engine
- Domain level testing
- Functional testing
- Transaction testing

CC Monitor

- Simple dashboard for all your builds
- Ramaze application
- Instant Feedback
- Cruise, Bamboo,

http://github.com/betarelease/cc_monitor

Secrets of Success

- Continuous testing and refactoring
- Managed Scope and timeline
- Ruby
- Enterprise Ready??

Photo Courtesy

- http://royal.pingdom.com/2008/01/24/when-data-center-cabling-becomes-art/
- http://www.flickr.com/photos/chrisdag/865724585/
- http://www.flickr.com/photos/digitalslurp/208731724/
- http://www.flickr.com/photos/tim_d/184018928/

Questions?

ありがとうございます。

sudhindra.rao@thoughtworks.com @sudhindraRao
munjal@thoughtworks.com @munjal