

?

200 : 2,000

70%

Fanatical Support

How do U compete when your software development team is 200 people vs 2000's?

How do U reduce 70% of your datacenter operation costs and make your data center more efficient?

How do U make your workforce more efficient at work when you are know for "Fanatical Support"

The image shows the Ruby logo, which consists of the Japanese characters 'ルビー' (Rubi) in a stylized, orange-red font. The characters are centered on a dark, gradient background that transitions from black at the top to a dark blue-grey at the bottom.

ルビー

I can picturize Matz or Ola smiling to themselves and saying: Use Ruby you dumb ass!

How Ruby Helps Rackspace challenge Amazon

Munjal Budhabhatti
ThoughtWorks, Inc

Thank for coming to my talk.

Introduce yourself

crappy dev to super crappy delivery principal

ThoughtWorks has been working with Rackspace, one of the world's leading hosting companies, on a large scale data center automation project using Ruby. I want to share some insights on how Ruby enabled Rackspace, with a small development force, to challenge Amazon.

ThoughtWorks®



Big thanks goes to ThoughtWorks for letting me fly and share some time with y'all.

Big thanks goes to Rackspace.

Rackspace, huh?

- 👁 One of the leading web hosting providers in world
- 👁 Managed Hosting, Cloud, Cloud Apps
- 👁 Fanatical Support
- 👁 200 developers

So what is Rackspace? How many of you know about Rackspace?

Rackspace is an internet web hosting company hence the domain is pretty awesome and challenging: Computer Networking. How many of you have read Tanenbaum? It's way way more complex than that!!

Amazon, oh yeah!

- 👁 Book seller to Online seller.
- 👁 Built cloud for resource fluctuations
- 👁 EC2, SW3, AWS
- 👁 Low price, easily scalable
- 👁 2000+ developers

Why did Amazon built cloud

Amazon allows bootstrapping startup ideas quickly. Very low investment and it can scale according to your requirements.

Fanatical Support

- 👁 Fanatical support vs No Support
- 👁 200 developers vs 2000+ developers
- 👁 And being cost effective?!!
- 👁 Really now?

So the question here is:

On one hand you have Rackspace that provides Fanatical support but has only 200 devs.

On the other hand you have Amazon that provides no support and has 2000+ devs.

In addition, Rackspace has to be price sensitive. Really now? So how do you do it:

How does Rackspace do it?

- 👁️ Lean Agile in Product development
- 👁️ Technology
- 👁️ Fanatical Culture

I think it has a lot of things to do with very lean agile...

Ruby allows Rackspace to...

- 👁 Program domain quickly
- 👁 Tailor business and reduced repetitive tasks
- 👁 Ease of Refactoring
- 👁 Powerful integration
- 👁 Realtime manipulation of interdependent systems

- 1) Developers investing time learning and programming the domain and not the framework.
- 2) DSLs allowed SMEs, instead of developers, to easily tailor business logic while reducing everyone's work load.
- 3) Ease of refactoring for major domain changes such as ipv6, architecture.
- 4) Powerful Enterprise integration with disparate system via Atom, REST, custom message queue, and even SOAP.
- 5) Realtime manipulation of interdependent systems for integration testing.

Understanding Domain

- 👁 Server and Network management
- 👁 Location and Power management
- 👁 VM management
- 👁 Monitoring and Diagnosis
- 👁 Customer Support
- 👁 Back Office support



In order to appreciate the complexity of the business let us quickly try to understand domain

What does a data center contain?

How do you locate a server in a data center?

Rackspace is known for its Fanatical support. How do you find a rough server quickly?

You don't want your best and brightest people to work on non-challenging stuff

AutoHost

- 👁 Discovery & Provisioning of a server took 25 days
- 👁 Ruby/Rails for Datacenter automation
 - 👁 IP Automation
 - 👁 Dns Automation
 - 👁 Location and Power manager
 - 👁 Network device Manager
- 👁 Pure XP. Just devs & SMEs. No BAs/QAs.
- 👁 Try that with Static ceremonial language

History

Project details

How we did it? Using XP. Would that be possible in any other ceremonial language as Java or .NET?

Advantages?

- 👁 25 days to 18 mins
- 👁 Reallocated data center engineers to provide Fanatical support
- 👁 Happier customers
- 👁 70% reduction of datacenter cost
- 👁 Start earning revenue sooner

How did Ruby help?

- 👁 Programming Domain instead of worrying about frameworks
- 👁 Smaller development team
- 👁 Delivering frequently - once every 2 weeks
- 👁 Faster feedback

Domain Specific Language

Domain Specific Language is a computer language that's targeted to a **particular kind of problem** rather than a general purpose language that's aimed at **any kind of software problem**.

Martin Fowler

What is a DSL?

Network Device Manager

- Ruby DSL for network devices
- Commissions new devices
- Intelligent command chains for different OSes
- Enhanced logging for debugging support

We created Ruby DSL for Data scrapping and custom script execution
Has conditional and for loops
Configure, diagonize, status of networking device
Intelligent command chains – Configurable to build command structures to support various hardware and OS combinations

Network Device Manager

Firewall tasks:

```
Remove Static NATs  
save
```

Switch tasks:

```
foreach interface name for port name  
  Disable the port  
  Clear the port description  
end
```

Please look at this example:

This is one of the ways that you can disable access to customer's server

Advantages?

- 👁️ SMEs tailor business logic
- 👁️ Repetitive mundane tasks automated
- 👁️ Significant reduction in training time for new network engineers
- 👁️ Easily add new networking devices

Refactoring domain changes

- 👁 Bridge domain to virtual L2 network
- 👁 ipv4 to ipv6
- 👁 Refactoring architecture

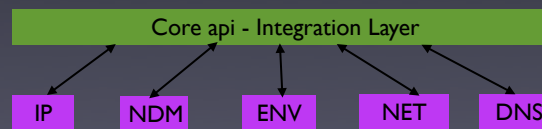
We used bridge domains to creating a seamless network for a customer in two different sections of datacenters or two different datacenters. This is quite complex set of configuration. With the need of virtual L2 networks such a complex configuration is no more required. Refactoring such domain changes was quick and painless.

Advantages?

- 👁️ Quickly refactor domain changes
- 👁️ Protect customer's investments
- 👁️ Reduce time to market

Enterprise integration

- 👁 Comprehensive reporting
- 👁 Repurpose with Api
- 👁 Workflow engine
- 👁 Enterprise Message Bus



Difference between Ruby and smalltalk

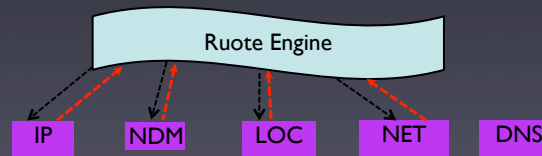
Easily integrate with disparate systems using REST, ATOM, Message queue, and SOAP

Reporting to other departments and data ware house

Orchestration of RESTful services for rich behavior

Workflow Engine

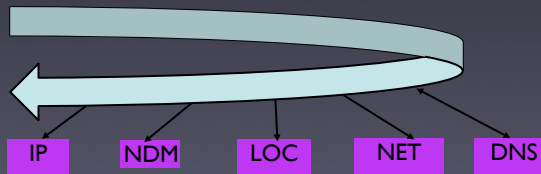
- 👁 Ruote - DSL based workflow engine
- 👁 RESTful service orchestration and rich abstraction
- 👁 Example



What is one click? Instead of set of commands in different applications and keeping a note in a piece of paper, have one click do a task.
4.5 days to 18 minutes

Enterprise Message Bus

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



Decoupling with external systems such as Core, Cloud

Advantages?

- 👁 Powerful enterprise integration
- 👁 Support legacy systems
- 👁 Easier reporting

Enterprise int with disparate system via Atom, REST, custom message queue, and even SOAP.

Support legacy systems & iterative replace it with new system instead of a big bang approach

Let devs concentrate on their expertise and not write custom reports. Each department created its own fancy report

Ruby really rocks...

- 👁 Program Domain quickly
- 👁 DSLs
- 👁 Refactor domain
- 👁 Enterprise integration

allowed twisting and splitting applications
allowed merging domain concepts via modularization
allowed for simple design – thus making it amenable to change
availability of libraries to support enterprise needs

Ruby philosophies and communities

- 👁 Secret sauce - community
- 👁 Something to change and learn

Why is Ruby so successful? I think the secret sauce is its community. It you all.

Community suggest the best practices: using Rspec, using CoffeeScript over Javascript etc. List is endless. Devs spend way less time on looking up frameworks.

However, Sometimes our community looks down on other community and doesn't accept if "It's not made here"

ARIL is a brilliant implementation. Question is: why it was not done before?

Questions and Discussions

Thank You

Munjal Budhabhatti

@munjal

munjal@thoughtworks.com