Enterprise Application Mashup
Architecting the Future

Eugene Ciurana
Director, Systems Infrastructure
LeapFrog Enterprises, Inc.

eugenex@leapfrog.com
pr3d4t0r @ irc://irc.freenode.net ##java, #esb, #awk, #security

Don't forget the evaluation - Thanks!

About Eugene Ciurana

- **15+ years of experience building mission-critical, high-availability systems infrastructure**
  - **Not an end-user applications kinda guy**
- **10+ years of Java work**
- **Engaged by the largest companies in the world**
  - **Retail**
  - **Finance**
  - **Oil industry**
- **Background ranges from industrial robotics to on-line retail systems**

What prompted this presentation?

- **Cost-conscious enterprises demand More, Fast, Cheap, and Now**
- **Outsourcing overseas is the result of low ROI from programming efforts**
- **IT vendors see this as a chance to sell you their wares...**
  - **...and to lock you in!**
- **The business demands more applications; you must...**
  - **...balance cost**
  - **...leverage open-source technologies**
  - **...decide between vendor offerings**
- **And you get to assemble the whole thing!**

What you'll learn from this...

- **Plan a 5-year outlook for your company**
  - **Defy conventional wisdom**
- **Discard outdated policies toward open-source**
  - **Very common in large enterprises**
- **Adopt best-of-breed applications, wherever they may come from**
  - **Commercial offerings**
  - **Open-source projects**
- **Embrace upstart open-source technologies**
  - **Mule**
  - **Codehaus Xfire**
  - **Terracotta DSO**
  - **Wicket**
- **Welcome change**
  - **Put Java 6 and Java 7 in your roadmap TODAY**

What you'll learn from this...

How to shift your strategy from

CUSTOM CODING

to

SMART CODING AND INTEGRATION

Enterprises Today

- **Everyone is cost-conscious**
- **The system architecture life cycle**
  - **Today's new system may become tomorrow's maintenance nightmare**
  - **Companies see a competitive advantage in improving the IT infrastructure**
  - **Companies see only a cost once systems enter the maintenance cycle**
- **What do smart companies do?**
  - **Cut costs by outsourcing system maintenance, operations, and other activities**
  - **Improve productivity by reassigning the engineering staff to satisfy new business requirements**
  - **Inshoring!**

Enterprises Today

- What do a lot of companies really do?

Assume that outsourcing is a business reality.

- Outsourcing "buy" instead of "build"
  - Outsource mission-critical engineering projects to third-parties
  - Lack the infrastructure to monitor and rein them in
    - Remote management is expensive
    - Deployment and integration costs are higher than proposed
  - Promote a low-morale environment

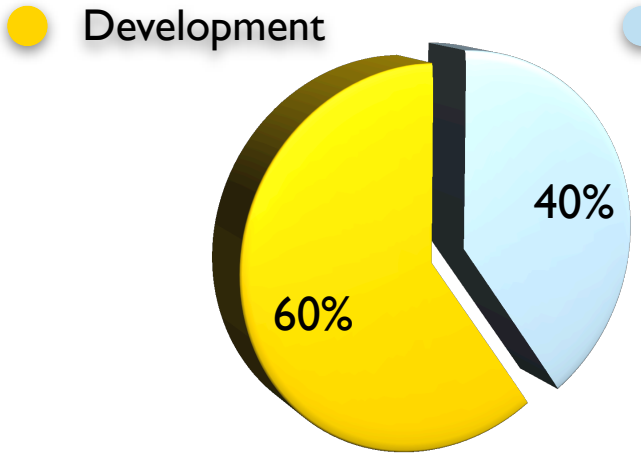
If it's going to happen anyway... how would you overcome its ill effects?

By shifting efforts toward smart coding and integration!

- How do you prevent this from happening?

Smart Coding and Integration

- **The business and engineering teams know the problem domain better than anyone.**
  - **Shift only low ROI activities out of the organization**
  - **Unleash the engineering team to work on thorny/sexy new problems**
  - **Become more competitive by adopting new technologies quickly**
    - **Tried-and-true may be your enemy**
    - **Vendor lock-in *IS* your enemy**
    - **These new technologies can be commercial, open-source, or both**



Sources:
Standish Group CHAOS 2 Report
Information Week 2000 report

Open-Source/Cutting-Edge Software

- **Larger enterprises sometimes don't benefit from using these tools**
  - **Corporate policy gets in the way**
  - **Fear of litigation**
  - **Aversion to being first**
  - **Inertia**
  - **Vendor lock-in**
- **The smart companies, though, figure out ways of implementing these**
  - **Pilot programs**
  - **Partnerships**
  - **Shift litigation through open-source subscriptions and indemnification**

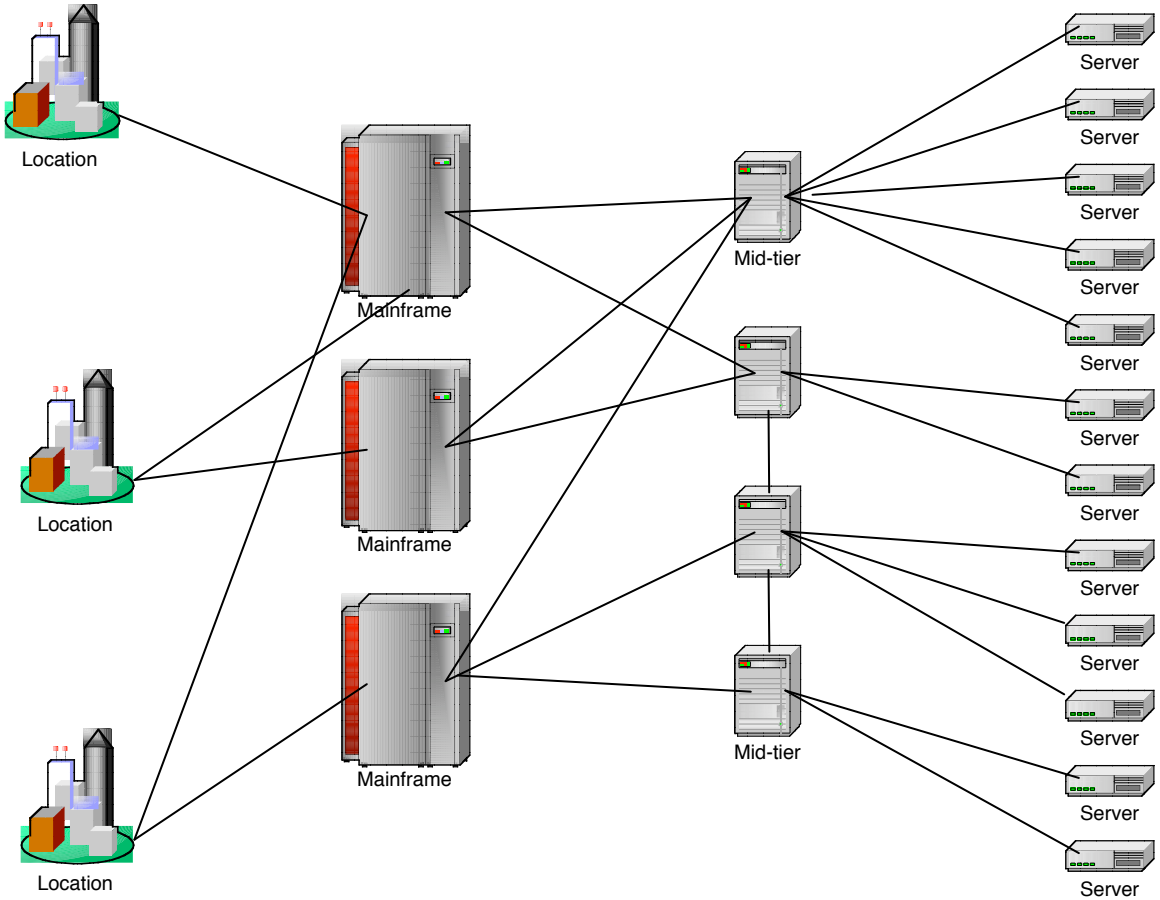
Open-Source/Cutting-Edge Software

- **How do you go about changing corporate policy?**
- **Through education!**
  - **Most large IT organizations are too busy with day-to-day business to pay attention**
  - **Many lack, or forego, ongoing formal training for their engineering and engineering management groups**
- **Remember: vendors have t… …eart!**
- **Seek education from ou… …e in that business, not in …**
  - **Open Source Initiative**
  - **Trade magazines**
  - **WWW: TheServerSide, Digg, Slashdot, eWeek, Reddit, java.net, etc.**

Are you spending at least 30 minutes researching new technologies and trends every day?

Why?

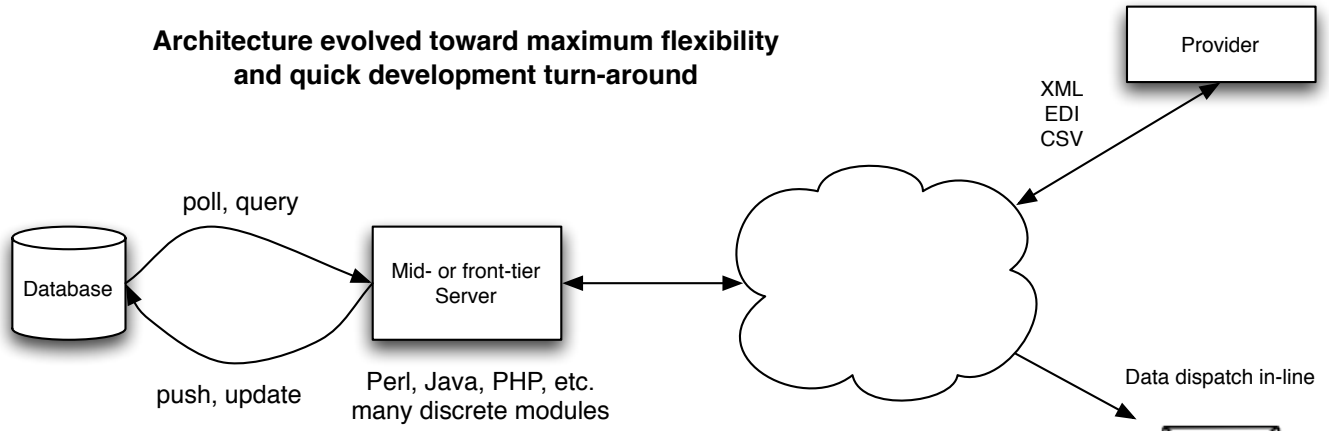
Typical Physical Architecture



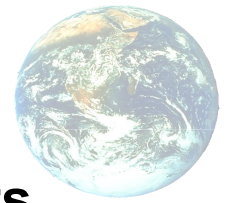
Typical Physical Architecture

- **Multiple tiers**
  - Each tier is implement on different technology
  - Coexistence of legacy and new systems
  - Different skill sets for each tier
- **Multiple physical locations**
  - Ad hoc interfaces between locations
  - Multiple protocols
- **Data and process consolidation**
  - Mainframes are at the core
  - Cross-coupling between systems makes it difficult to add new functionality
- **New integration projects == lengthy development cycle**

Typical Application Architecture



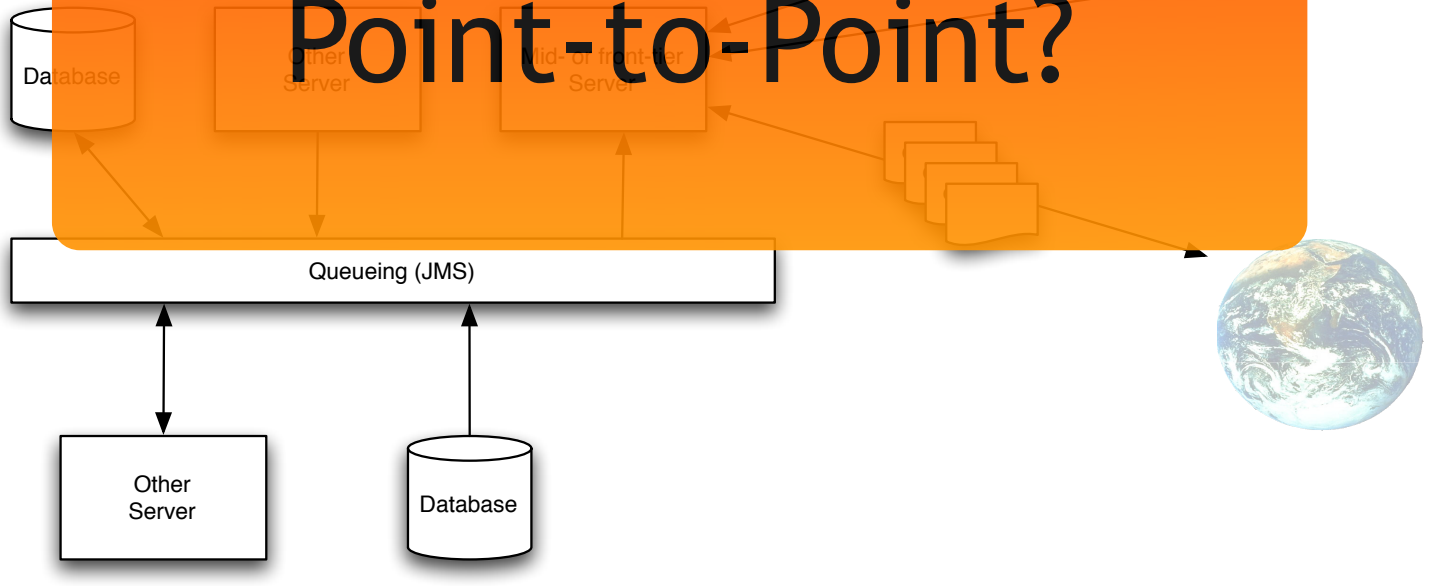
- **Systems are built "as needed"**
  - **Point-only integration**
- **New systems require new interface definition**
  - **Interface duplication or incompatibility**
- **Queuing (MQ Series, JMS, etc.) seen as silver bullets**
  - **It works as long as all the participants support queuing**



Typical Application Architecture

- **Messaging works well as long as every participating system supports it!**
- **All communications become point-to-point**
  - **Web services also become point to point**

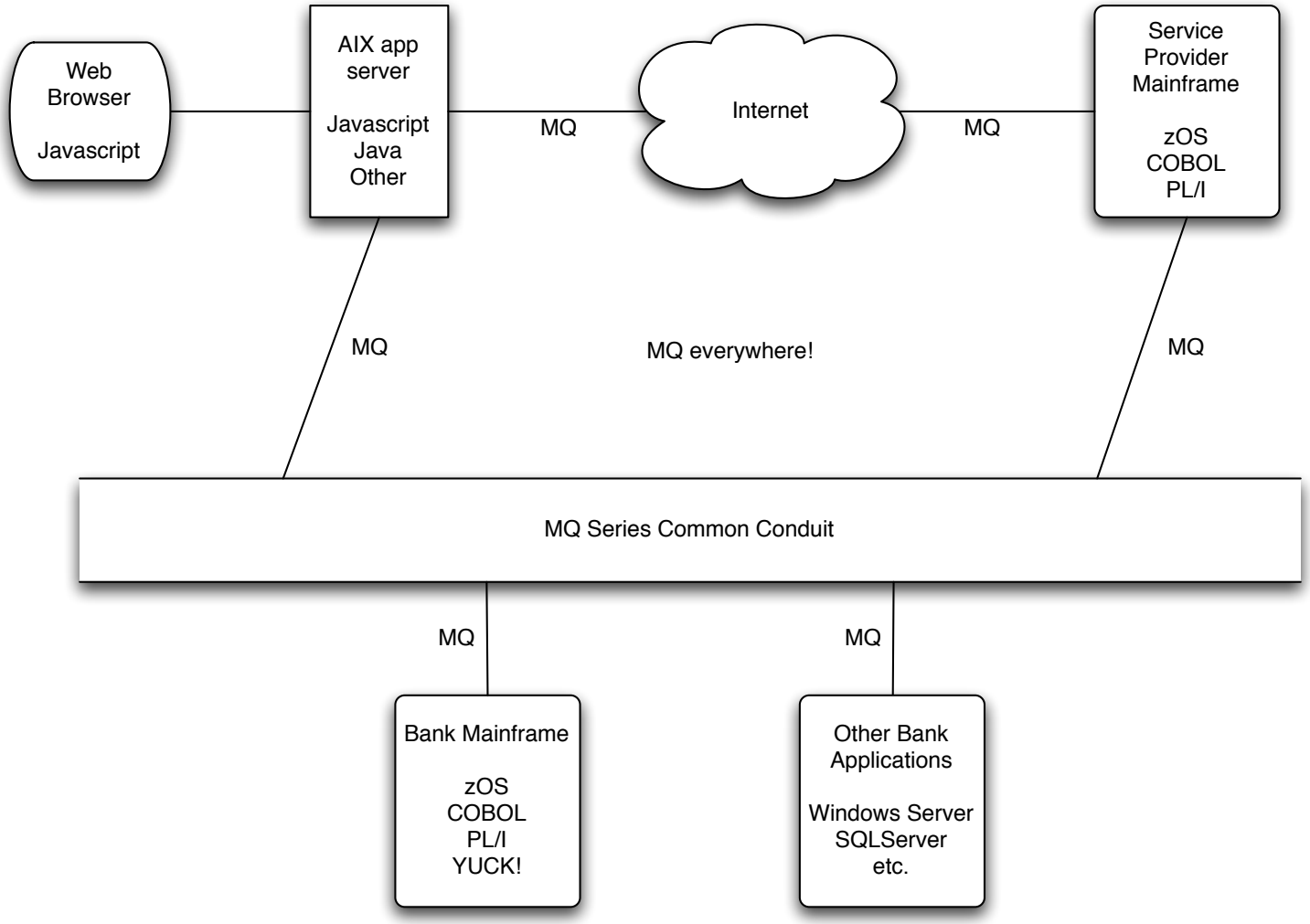
What's the problem with Point-to-Point?



The Problem of Point-to-Point: Case Study

- **Top-3 in the US financial operation/Integrion**
- **On-line banking system has a three-tier architecture**
  - **Web browser front-end**
  - **AIX/app server middle-tier**
  - **Mainframe middle-tier**
  - **Mainframe back-end**
- **MQ Series was chosen as the common conduit...**
  - **Brittle data format**
    - **COBOL-like data structures, positional**
  - **Not scaleable**
  - **Lots of processing overhead because of native-to-uniform data formats**
- **It worked, but it became very expensive to operate and enhance**

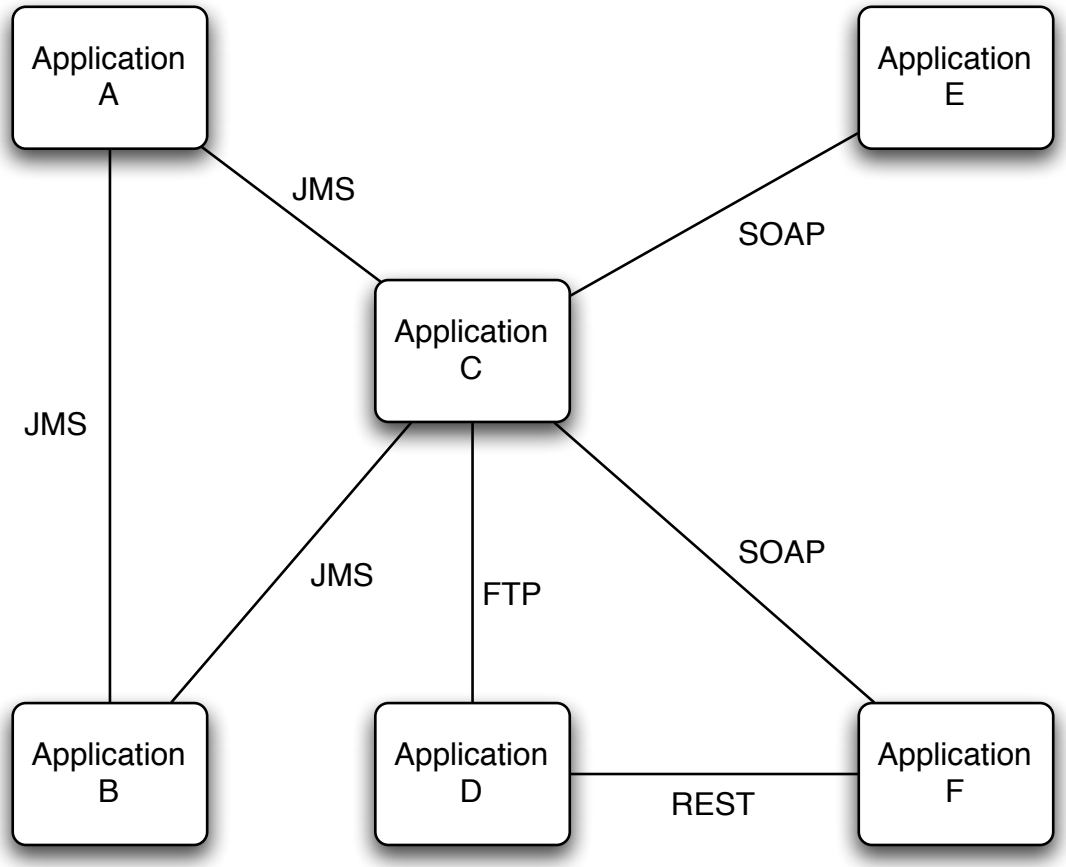
The Problem of Point-to-Point: Case Study



The Problem of Point-to-Point Integration

- **Every problem becomes a new project**
- **Different systems may support different interoperability protocols**
  - **JMS**
  - **SOAP**
  - **REST**
  - **BPEL**
  - **etc.**
- **Point-to-point integration is more expensive**
  - **Consumes more resources**
  - **Requires more regression testing**
  - **Creates lock-in/artificial dependencies between systems**

The Problem of Point-to-Point Integration



- **Applications run on different platforms**
  - **Mainframes**
  - **Windows**
  - ***NIX**
- **Adding a new application and protocol (i.e. BPEL) requires extensive changes**
- **Some applications become hubs... and potential bottlenecks.**

Solving the Application Integration Problem

- Monolithic, proprietary, and custom-built
- Commercial applications
- Open-source software
- Finding the “best of breed” for each integration component in your environment is the key to success
  - Cannibalization - does a third-party product work better than the in-house version?
  - Data-driven evaluation - define the business and tech goals, then evaluate the products; don’t be blinded by the vendor’s BS trying to sell you a marketecture
  - Open-source or commercial - both have legal exposures; educate yourself and your team

Shift development efforts toward solving domain-specific problems and unleash your software engineering talent to solve integration problems instead!

Solving the Integration Problem

- **You picked your applications**
- **You leveraged open-source options**
- **Next comes the fun part: Integrating all application and subsystems to play together**
- **Assumption: each of the systems in the mix supports at least one way of communicating with the outside world**
  - **Database**
  - **SOAP**
  - **BPEL**
  - **REST**
  - **FTP**
  - **JMS/queueing**
  - **JavaSpaces**

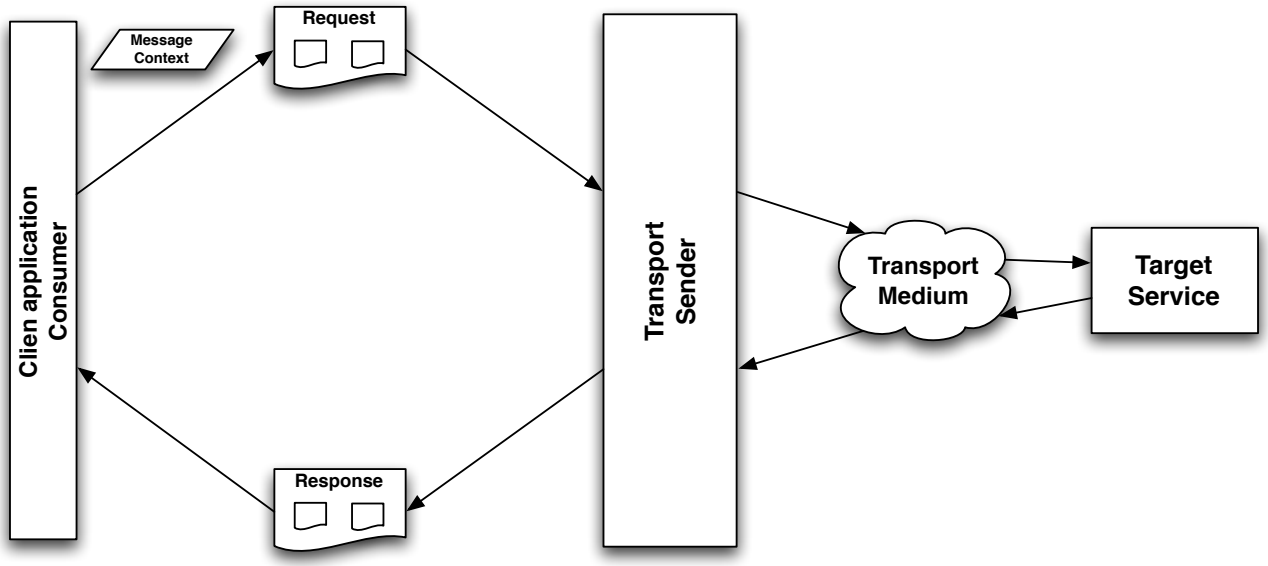
Et cetera!

Solving the Integration Problem

- **Define your Prime Directives or Policies**
- **Acquire - DO NOT BUILD!**
- **Adopt Open-source Wherever Possible**
- **The Oldest Component in your Infrastructure Should Be 18 Months Since GA**

Solving the Integration Problem

- **Problem: Structured exchange between two systems for document exchange, transactions, etc. over HTTP**
- **Solution: SOAP**



Solving the Integration Problem

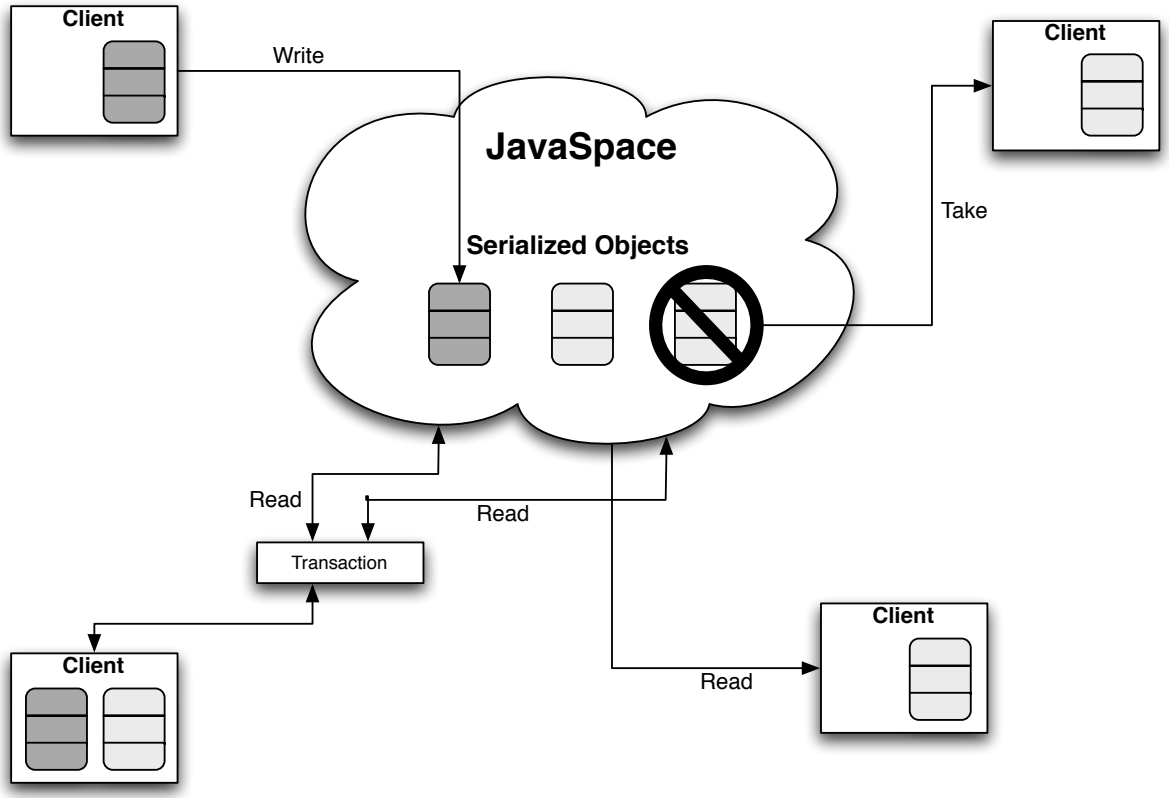
- **REST (Representational State Transfer)**
  - **Based on the concept of web resources**
  - **A resource is anything that has zero or more representations addressable through a URI**
  - **Simple messages can be encoded in the URI itself**
  - **Other messages are in XML (normalized through XML Schema or DTD) or JSON**
  - **All interfaces use HTTP (GET, POST, PUT, DELETE)**

REST is a better solution than SOAP in most cases -- remember that SOAP's main design goal was for Microsoft and IBM to have more frameworks to sell you!

Solving the Integration Problem

- **Problem: multiple Java applications must share data with one another asynchronously and without the overhead of JMS or SOAP**
- **Solution:**

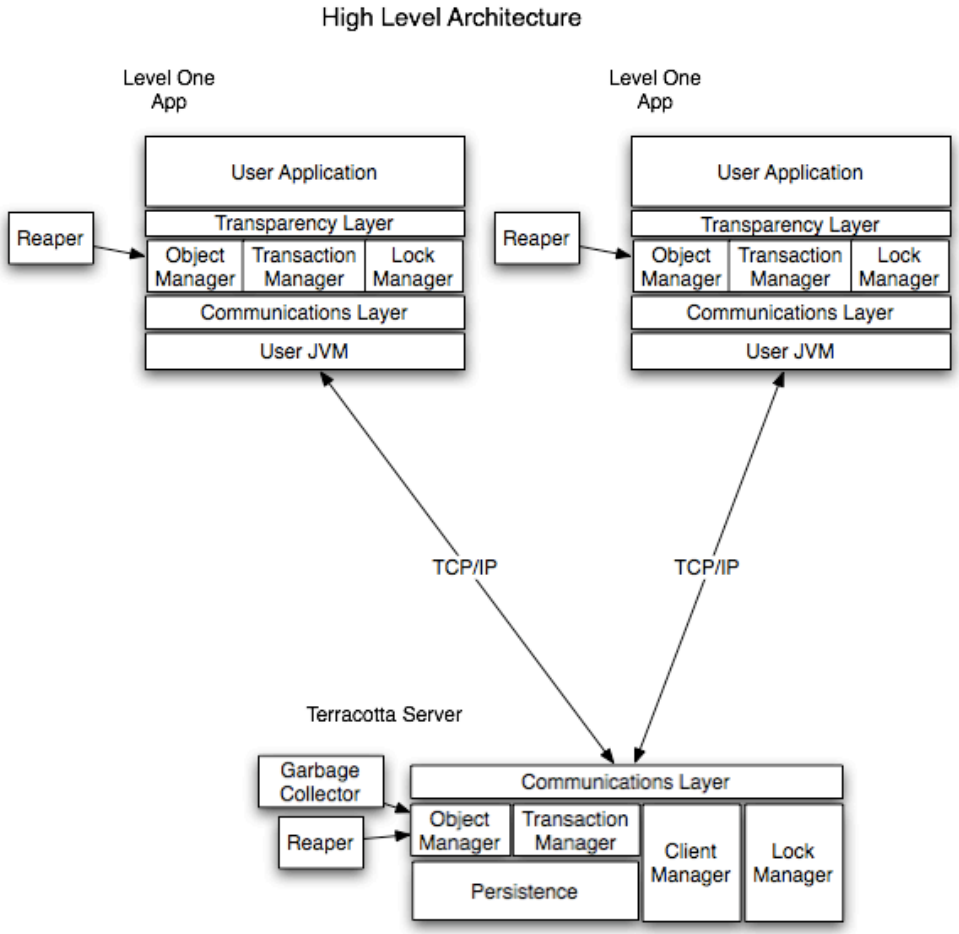
JavaSpaces



Solving the Integration Problem

- **Problem: Multiple systems must share data among them in near real-time; JavaSpaces allows for that but may prove to be too slow.**
- **Solution: Terracotta DSO**
  - **Data sharing and coordination**
  - **Transparent to the programmer**
  - **Transparent to the application (works well with third-parties!)**
  - **It works by extending the memory model from one to multiple JVMs**
  - **Safe memory access across applications through lock implementation**
- **See diagram next...**

Solving the Integration Problem



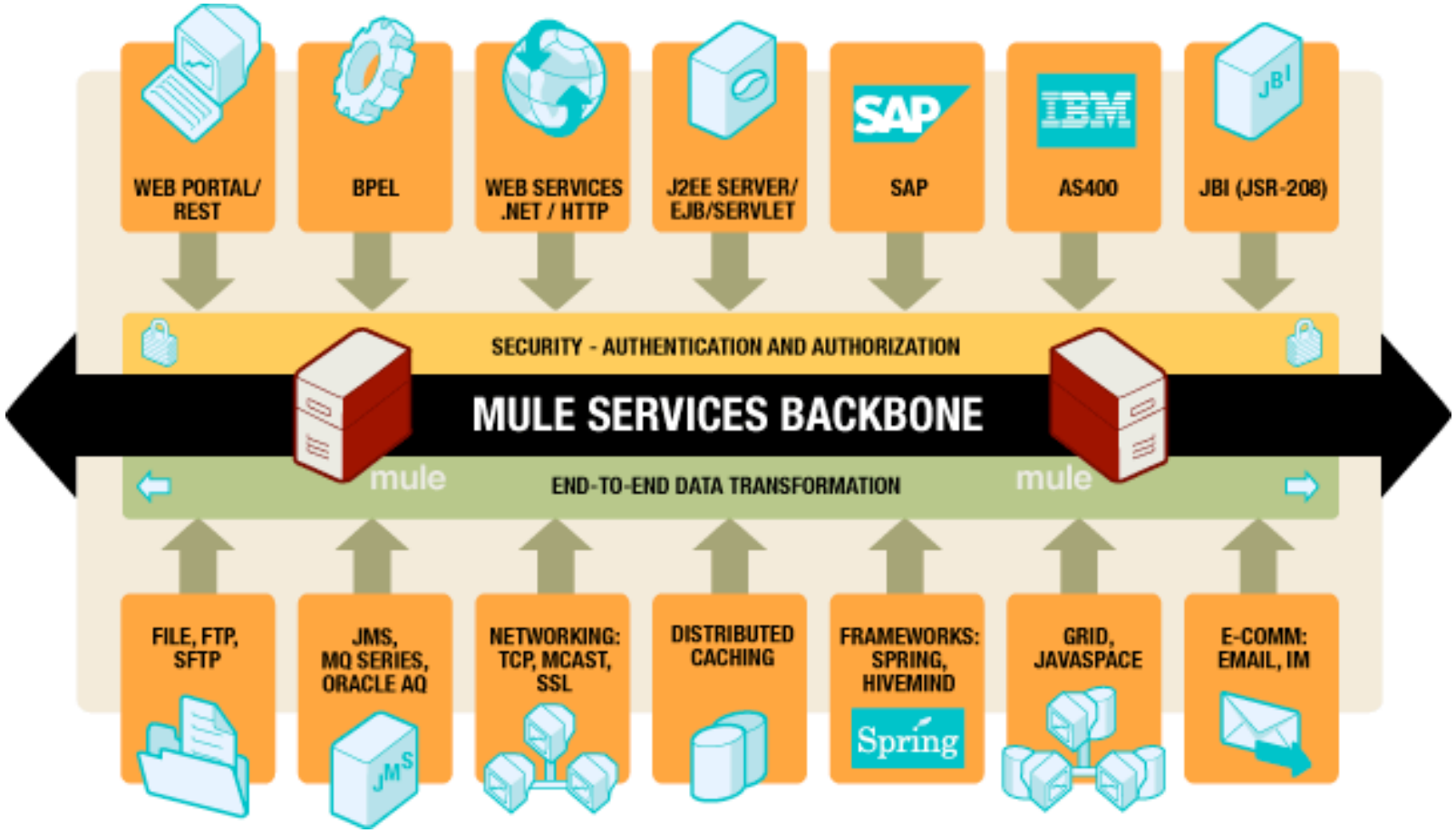
Is the Integration Problem Solved?

- **No! Different systems implement different strategies**
  - **Some do JMS, some SOAP, some do both, some do neither**
- **No solution is a one-size-fits-all**
- **Legacy systems aren't covered**
  - **COBOL, PL/I and other**
  - **EDI**
- **Non-Java systems aren't covered**
  - **Native C/C++/4GL applications**
  - **Windows Server applications**
  - **Other**
- **Third-party integration**

The Last Leg: Enterprise Service Bus

- **An Enterprise Service Bus, or ESB, is the common conduit for all applications because it "understands" all protocols**
- **Developed in parallel by Progress Software and Ross Mason**
  - **Progress came up with the term ESB**
  - **Ross developed the first true cross-platform ESB**
- **Commercial and open-source options available**
  - **Mule, OpenESB, ServiceMix**
  - **TIBCO, IBM, IONA, Progress, etc.**
- **Best of breed out of the box: Mule**
  - **Supports more transports and adapters than anyone else**

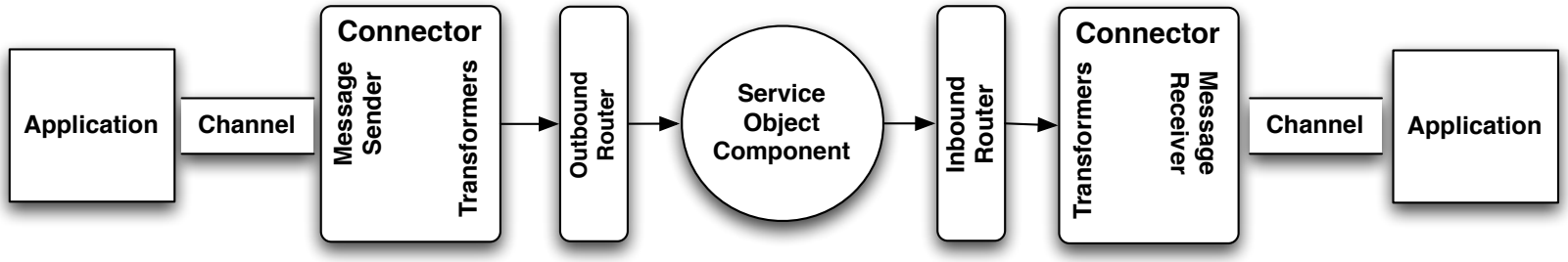
The Last Leg: Enterprise Service Bus



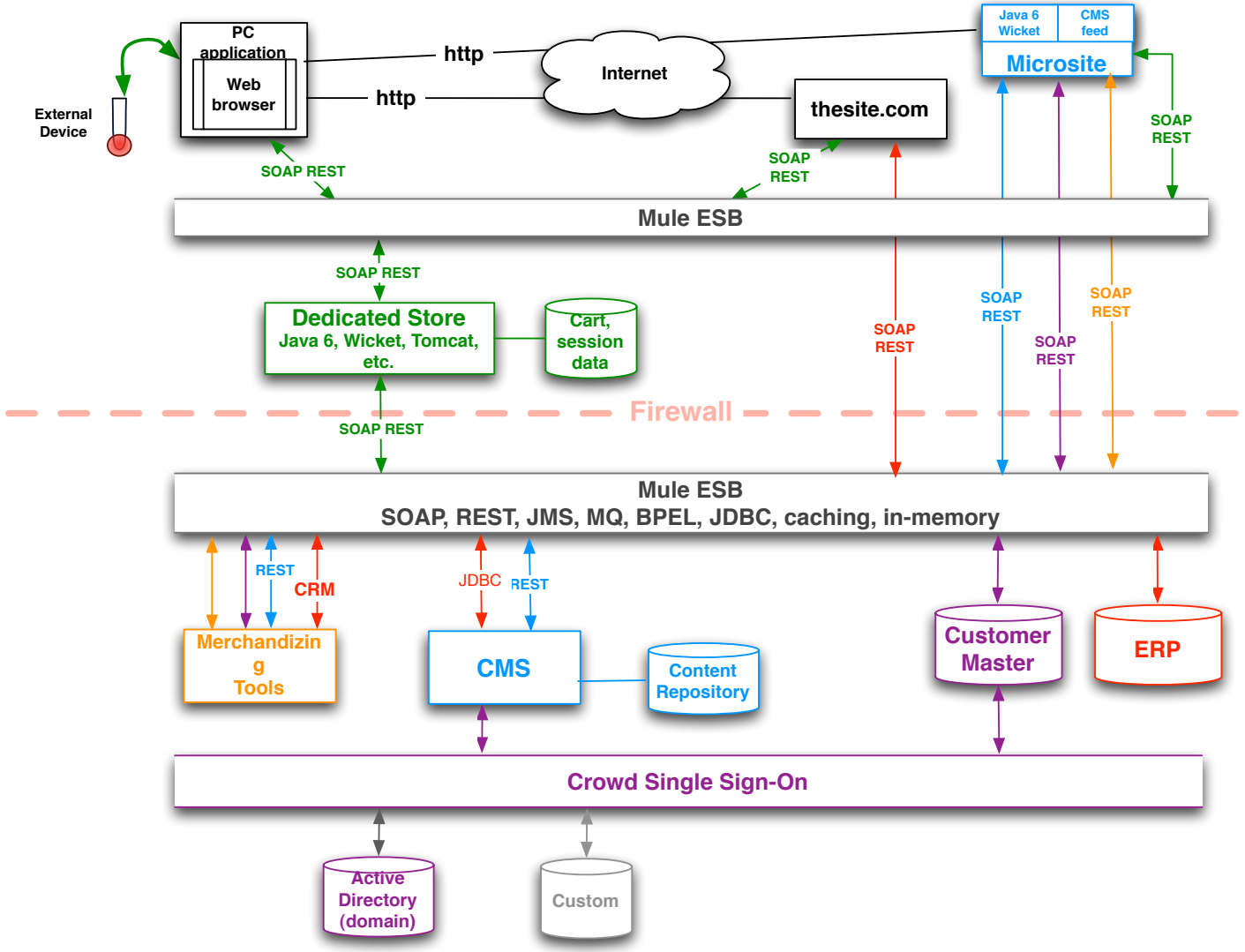
Solving the Integration Problem with an ESB

- **Encourages coexistence of heterogeneous systems**
- **Supports multiple languages, protocols, etc.**
- **Applications do not talk to one another point-to-point!**
  - **Applications talk to the ESB**
  - **ESB routes messages between applications, or “end points”**
  - **Customization occurs only at the translator level**
  - **Short development and testing cycle**
  - **Localized problem determination and resolution**
- **Commercial, legacy, and new applications coexist in the same environment**
- **Effort shifts from coding to integration and extension**

Solving the Integration Problem with an ESB



Solving the Integration Problem



Solving the Integration Problem: Fail-over

- **Goal: another instance of Mule must be ready to pick up if the first one crashes**
- **Areas of interest:**
  - **Configuration replication**
  - **Resource replication**
  - **Fail-over mechanism**
  - **Shared memory space**
- **Today: these are built on-demand and aren't part of the standard Mule project**
- **Soon: OSGi, JMX, shared memory and/or registry for replication and state**

Conclusions

- **Use the best of breed application whenever possible**
- **Recycle existing systems**
- **Shift effort from continuous development to integration**
- **Develop only strategic portions of the applications and infrastructure**
- **Leave application development to the domain experts**
  - **Specialized groups within the organization**
  - **The open-source community**
  - **Third-party vendors**
- **Apply your resources toward integration of all your software around an ESB rather than any single sub-system in your application stack**

Q&A

Thanks for coming!

http://ciurana.eu/TSSJSBarcelona

Don't forget the evaluation - Thanks!

Eugene Ciurana
Director, Systems Infrastructure
Leap Frog Enterprises, Inc.

eugenex@leapfrog.com
pr3d4t0r @ irc://irc.freenode.net ##java, #esb, #awk, #security