from applications to services

as natural as the move from albums to 'itunes'

a service-oriented state of the union at gsi commerce

Steve Buzzard

Principal Architect, GSI Commerce

March 27th, 2009

GSI Commerce® and the gsi commerce® logo are trademarks, service marks, registered trademarks, or registered service marks of GSI Commerce, Inc. or its subsidiaries or affiliates. Other trademarks contained in this presentation are the property of the respective companies with which they are associated.

This presentation is for informational and discussion purposes only and should not be construed as a commitment of GSI Commerce, Inc. or of any of its subsidiaries or affiliates. While we attempt to ensure the accuracy, completeness and adequacy of this presentation, neither GSI Commerce, Inc. nor any of its subsidiaries or affiliates is responsible for any errors or will be liable for the use of, or reliance upon, this presentation or any of the information contained in it.

The information contained in this presentation is subject to change without notice. This presentation contains proprietary and/or confidential information of GSI Commerce, Inc., its subsidiaries or affiliates and is released only under the terms of the non-disclosure agreement between our respective organizations. Unauthorized use, disclosure or dissemination of this information is expressly prohibited.

gsi commerce®

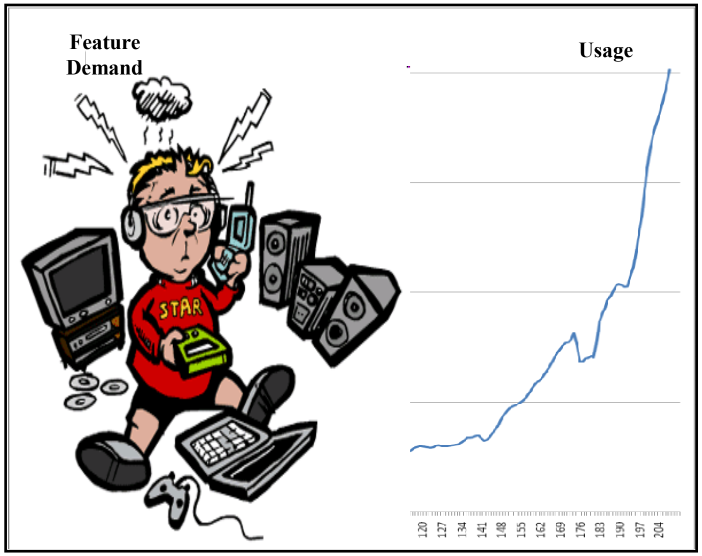
© Copyright 2009

but first, for something completely different …

Monty Python's The Architect

- **MR. TID:** Gentlemen, we have two basic suggestions for the design of the residential block of apartments, and I thought it best the architects themselves came in to explain the advantages of both designs. Mr. Wiggin will be first. Mr. Wiggin?
- **MR. WIGGIN:** Thank you. Good morning, gentlemen. As you can see, this is a twelve-story block combining classical neo-Georgian features with all the advantages of modern design. The tenants arrive in the entrance hall here, are carried along the corridor on a conveyor belt in extreme comfort and past murals depicting Mediterranean scenes, toward the rotating knives. The last twenty feet of the corridor are heavily soundproofed. The blood pours down these chutes and the mangled flesh slurps into these large containing … eh, yes?
- **CITY GENT #1:** Uh, did you say 'knives'?
- **MR. WIGGIN:** Rotating knives. Yes.
- **CITY GENT #2:** Are you, uh, proposing to slaughter our tenants?
- **MR. WIGGIN:** Does that not fit in with your plans?
- **CITY GENT #1:** No, it does not. We wanted a... simple... block of apartments.
- **MR. WIGGIN:** You see, I mainly design slaughter houses.
- **CITY GENT #1:** Yes. Pity.
- **MR. WIGGIN:** Mind you, this is a real beaut! None of your blood caked on the walls and flesh flying out of the windows inconveniencing passers-by with this one. I mean, my life has been building up to this.
- **CITY GENT #2:** Yes, and well done – err … but we did want a block of apartments.

gsi commerce's service oriented state of the union -
an unfinished drama told in six acts



- ➢ The Protagonist
  - ❖ **GSI Commerce**
- ➢ The Antagonist
  - ❖ **Multi-Dimensional Growth**
- ➢ The Conflict
  - ❖ **Scaling Usage while Satisfying Feature Demand**

- ➢Awareness
  - ❖ **Diversity of Feature Demand is a Scalability Concern Too!**
- ➢ Enlightenment
  - ❖ **From albums to 'itunes'**
    - ❖ Moving the Focus from Applications to Services
    - ❖ So the customer can build what they need when they need it
    - ❖ In case they need a slaughterhouse when you offer a webstore ☺
- ➢ Enlightenment's Challenges
  - ❖ **Operational Complexity**

gsi commerce at a glance

- **Dedicated to supporting 3rd party eCommerce businesses since 1999**

- **4000+ employees across the organization**
  - Headquarters in King of Prussia, PA
  - Offices in CA and VA and in Barcelona, Spain (Europe HQ) and the UK
  - Three Distribution Centers in KY and another in VA
  - Customer Care Centers in WI, GA, VA, and FL
  - Fully redundant data centers in VA and NJ

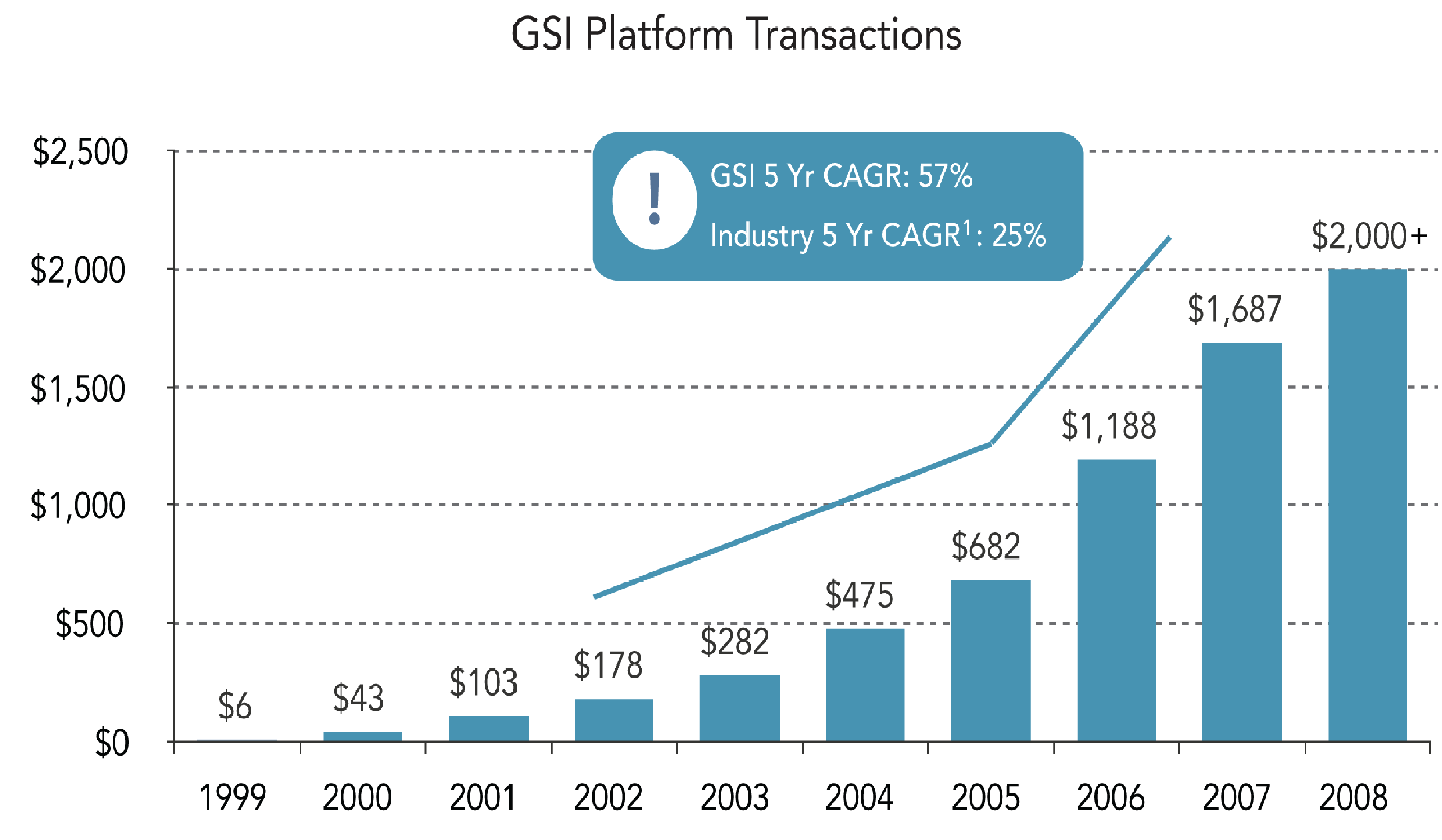
- **Provide solutions through our integrated eCommerce platform**
  - World class, highly scalable technology platform
  - Logistics and customer care
  - Marketing services

- **Publicly traded on NASDAQ ("GSIC")**

gsi commerce's corporate history -
with an emphasis on growth

1999	Founded in 1999 by CEO Michael Rubin in King of Prussia, PA	Global Sports, Inc.	SPORTS AUTHORITY	Launched 5 Sporting Goods Sites
2000	Opened fulfillment center in KY	Infrastructure scaled to support growth	fogdog SPORTS	QVC QUALITY VALUE CONVENIENCE
2001	EVERY SEASON STARTS AT DICK'S SPORTING GOODS	Added partners in entertainment	Redundant data centers	MODELL'S SPORTING GOODS Gotta Go To Mo's SINCE 1889
2002	Changed name to GSI Commerce, Inc.	Opened call center in FL	palm	ESTĒE LAUDER COMPANIES
2003	40+ partners	Entered home vertical	ACE The helpful place.	PBS
2004	Entered apparel vertical	RALPH LAUREN	kate spade NEW YORK	BURBERRY
2005	Entered jewelry vertical	ZALES THE DIAMOND STORE	Acquired Aspherio in Barcelona, Spain	Over 4 billion page views
2006	Opened 2nd call center in WI	Toys R Us	NFL	gsi interactive a division of gsi commerce
2007	Acquired ACCRETIVE commerce	Acquired Zendor in the UK	Additional warehouses and customer service locations opened in US	$1.7B in transactions through the platform
2008	Acquired e-dialog	Kenneth Cole Calvin Klein BIG LOTS!	Significant international focus & investment	Over $2B in transactions Over 25.8M orders Over 19.7B page views Over 1.5B visitors Over 305K Cyber Monday orders

we have certainly grown in transaction volume …



gsi commerce

… but even more in diversity and capability demand

5 sporting goods partners > 85+ partners in 15 categories

1999

Categories

Sporting Goods

- Small number of Sporting Goods Stores
- We did everything for each store
- We owned the inventory for all stores
- The Webstore Application satisfied pretty much any demand

2008

Categories

Apparel, Accessories & Footwear

Appliances & Tools

Baby Products

Cosmetics & Fragrances

Consumer Electronics

Entertainment Media

General Merchandise

Gifts

Home Furnishings

Jewelry

Pets

Personal Care

Specialty Foods

Sporting Goods

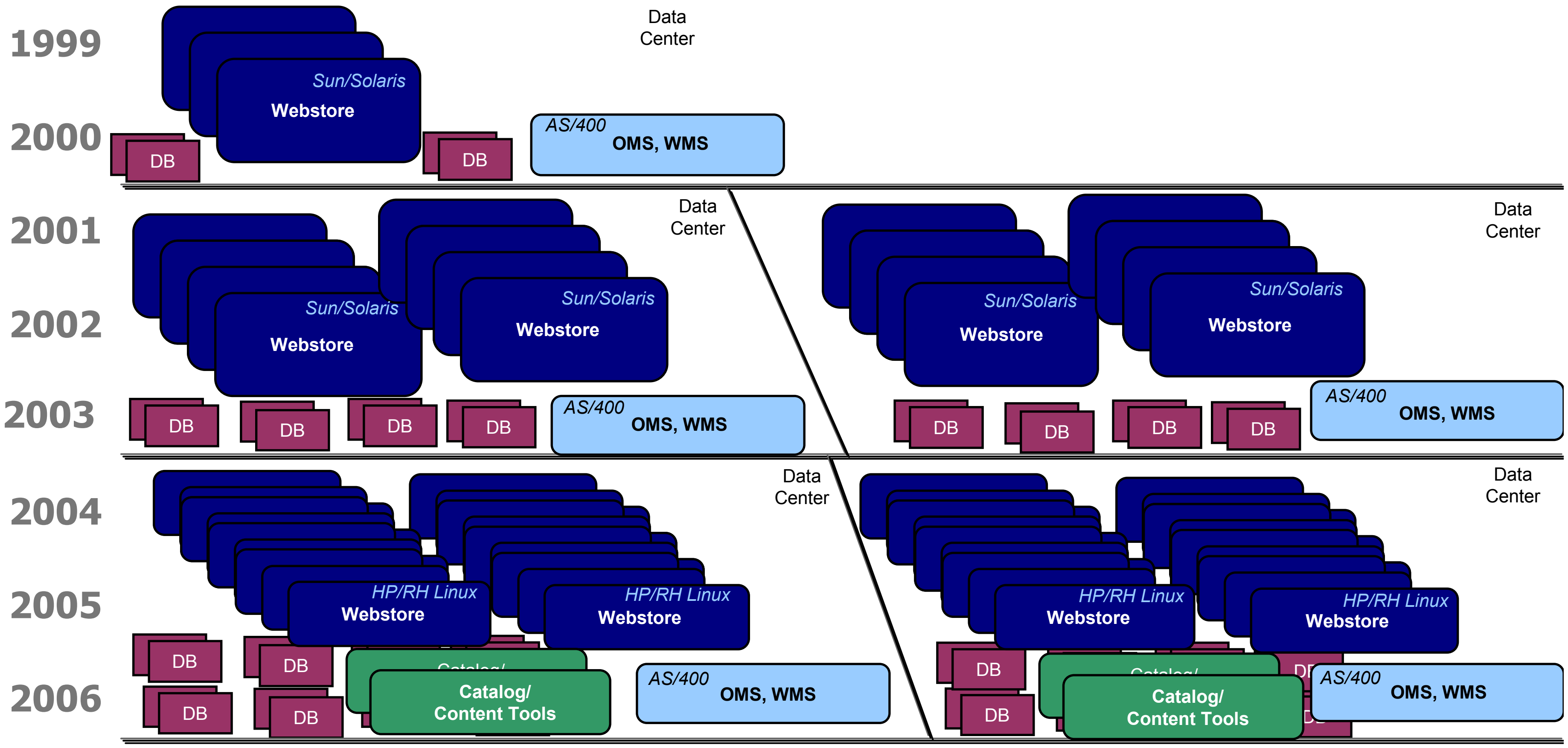
Toys & Video Games

- Sporting Goods became “All Kinds of Things”
- “We do everything” became just one of many partner models
- Partner needs and desires had become increasingly sophisticated

- **By 2006, scalability needs had expanded not just in terms of size but perhaps most of all in functional diversity**

gsi commerce ecommerce platform history -

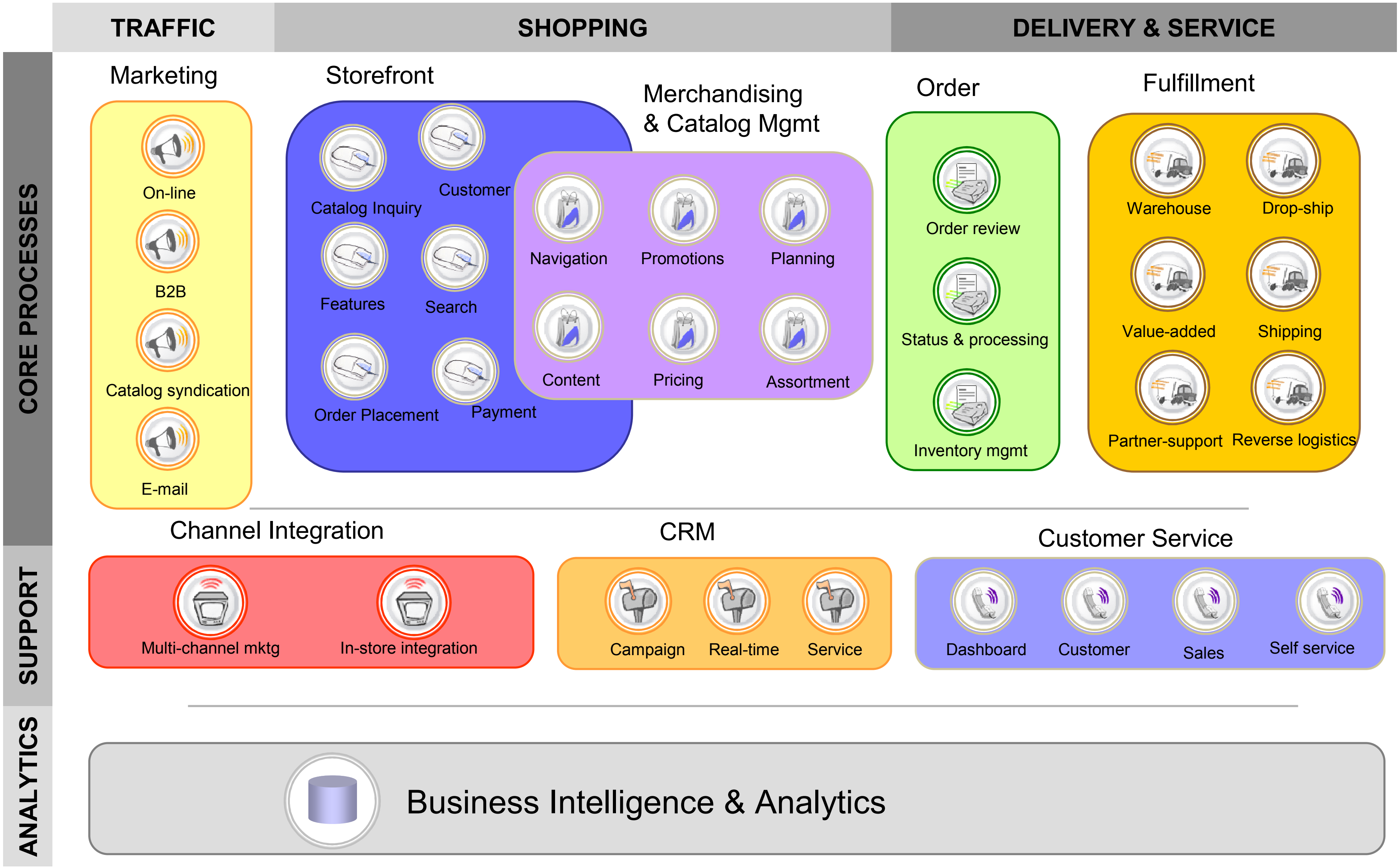
scaling to meet traffic growth, but something was missing …



- ➢ This scaling solved the “we’re getting a lot more traffic to the webstore” problem
- ➢ We added catalog and content management tooling to help solve the “our partners want more control” problem
- ➢ We were then confronted with the “we build our own applications too and need access to your capabilities programmatically” problem
- ➢ **Continuing to add applications to satisfy each flavor of each partner need would itself clearly not scale**

gsi commerce

awareness -
our logical services …



TRAFFIC | SHOPPING | DELIVERY & SERVICE

CORE PROCESSES

Marketing
- On-line
- B2B
- Catalog syndication
- E-mail

Storefront
- Catalog Inquiry
- Customer
- Features
- Search
- Order Placement
- Payment

Merchandising & Catalog Mgmt
- Navigation
- Promotions
- Planning
- Content
- Pricing
- Assortment

Order
- Order review
- Status & processing
- Inventory mgmt

Fulfillment
- Warehouse
- Drop-ship
- Value-added
- Shipping
- Partner-support
- Reverse logistics

SUPPORT

Channel Integration
- Multi-channel mktg
- In-store integration

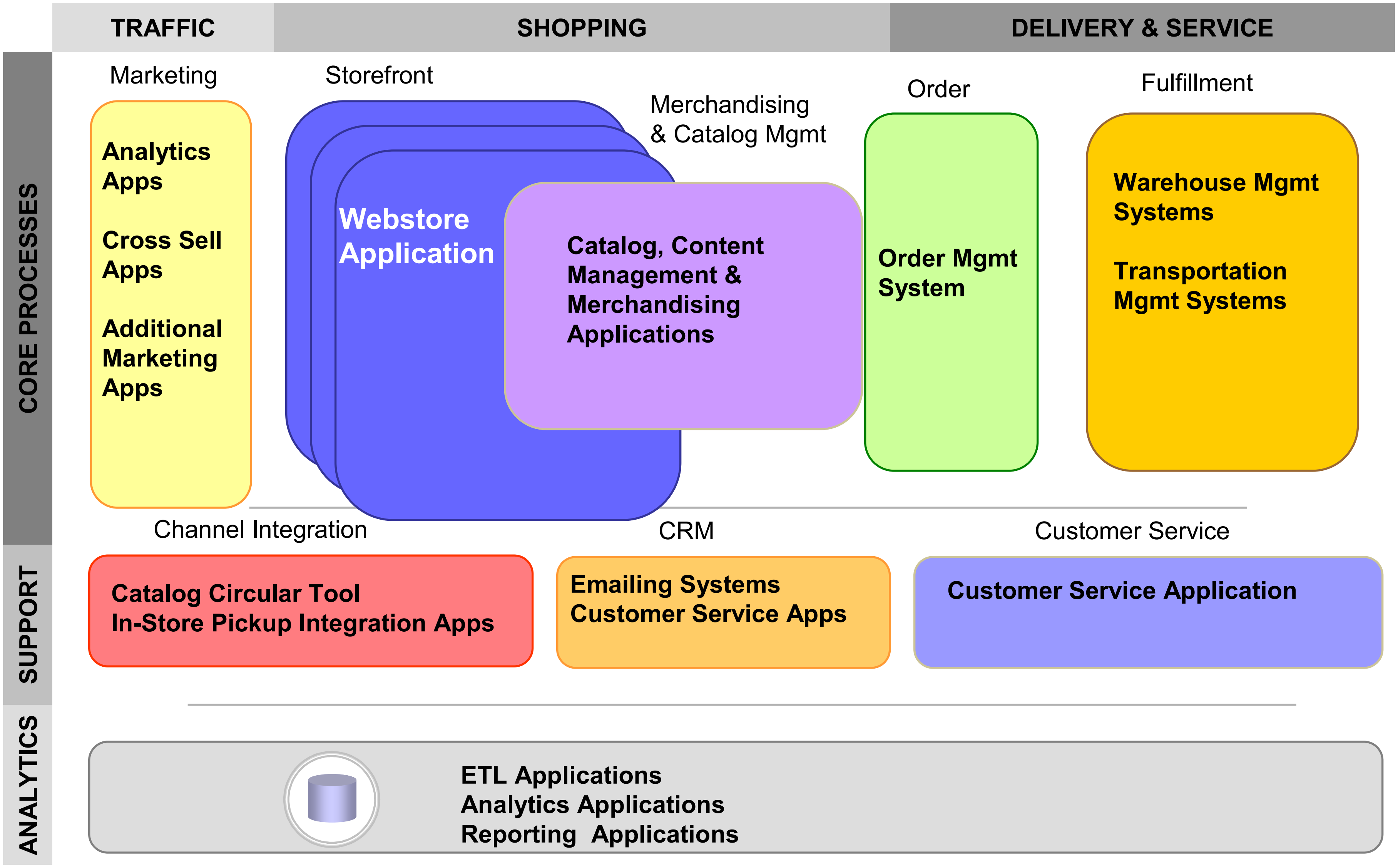
CRM
- Campaign
- Real-time
- Service

Customer Service
- Dashboard
- Customer
- Sales
- Self service

ANALYTICS

Business Intelligence & Analytics

… were locked within our applications and systems –
*impeding our ability to **scale in diversity of feature demand***



TRAFFIC

SHOPPING

DELIVERY & SERVICE

CORE PROCESSES

Marketing

Analytics Apps

Cross Sell Apps

Additional Marketing Apps

Storefront

Webstore Application

Merchandising & Catalog Mgmt

Catalog, Content Management & Merchandising Applications

Order

Order Mgmt System

Fulfillment

Warehouse Mgmt Systems

Transportation Mgmt Systems

SUPPORT

Channel Integration

Catalog Circular Tool
In-Store Pickup Integration Apps

CRM

Emailing Systems
Customer Service Apps

Customer Service

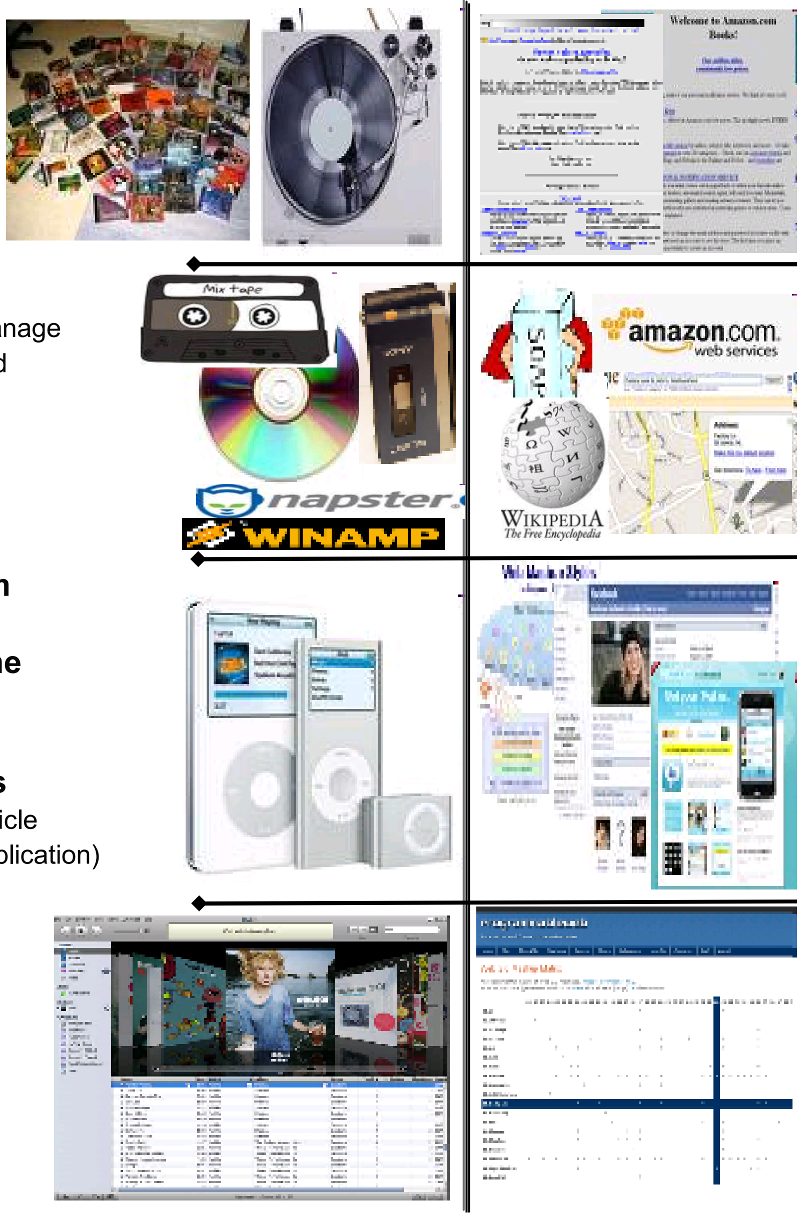
Customer Service Application

ANALYTICS

ETL Applications
Analytics Applications
Reporting Applications

the dawn of enlightenment –
from albums to 'itunes' (or how a good thing became better)

- **Albums (CDs) are much like applications**
  - The good ones are classic but they're rarely **exactly** what you want
- **Albums in the 1960s were like Web Applications in the 1990s**
  - If you only liked some of its songs (features), you lived with the rest
- **The 1970s, 80s, and 90s brought about more choice in music …**
  - Everyone had "awesome mix" tapes, but they were time consuming to make
  - Walkmans made you mobile - CDs made your music collection much easier to manage
  - Toward the end of this era, the MP3 began its ascent, mainly with the techie crowd
- **Likewise, Web Applications in early 2000-2002 involved more choice**
  - Web Services appeared and the read-only web became a bit more read-write
  - Consumers (and clients/business partners) were getting more savvy
- **Then … BAM! Apple brought digital music delivery to the mainstream**
- **POW! The groundswell of the Participatory Web brought an avalanche of services not just on the web but of the web**
- **Now, 'traditional' Web Applications (and Albums/CDs) are still with us**
  - But it's the service (the digital song) that is becoming the mainstream delivery vehicle
  - The playlist (the service based application) is your (customer's) perfect album (application)
  - Services aren't as far along on this journey, but they'll quickly catch up
- **<u>If the digital song is the service, iTunes is its most successful service registry</u>**
  - Certainly not the only one (Napster, Amazon, givembeats, etc.) but to most it is
  - It's also a 'mashup' editor (in a couple of senses!), a BPM tool, a browser



we first exposed services directly from our webstore – *many square pegs from a decidedly application-specific round hole*

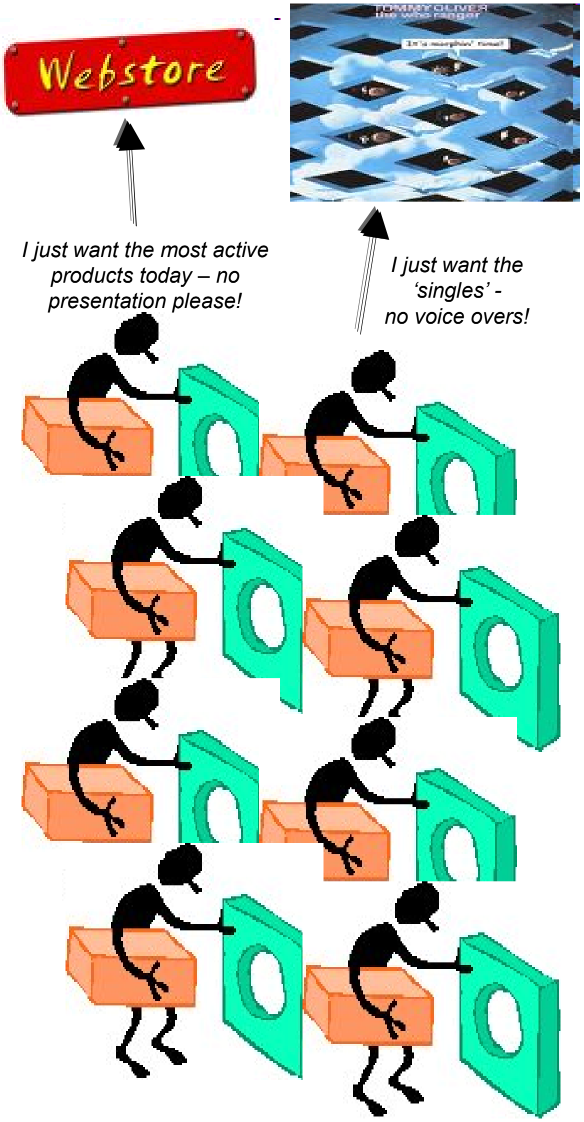
- **Some applications, like our Webstore, are more like Concept Albums**
  - Meant to be used (listened to) as a whole – like the Who's 'Tommy'
  - Sometimes containing application context (voice over, thematic backing tracks)
  - Containing songs are often very different from the versions released as 'singles'

- **Services are simply songs we want/need for our own playlist**
  - As consumers, we may not want the presentation/concept (we likely either have our own or want none at all) – maybe we want the 'single' version

- **Extracting Services requires knowledge of the Application**
  - Not simply by the provider but by the consumer as well
  - "The songs are trapped in their conceptual album window dressing"

- **From an operational perspective, this compromise between service and application has repercussions**
  - Configuration/set-up for one that likely doesn't apply to the other
  - Monitoring, management, and tuning applied to one can negatively affect the other

- **It was clear the Webstore would need change to consume our services, not contain them**
  - We needed to separate the services from the application – people needed both but they didn't need a hybrid

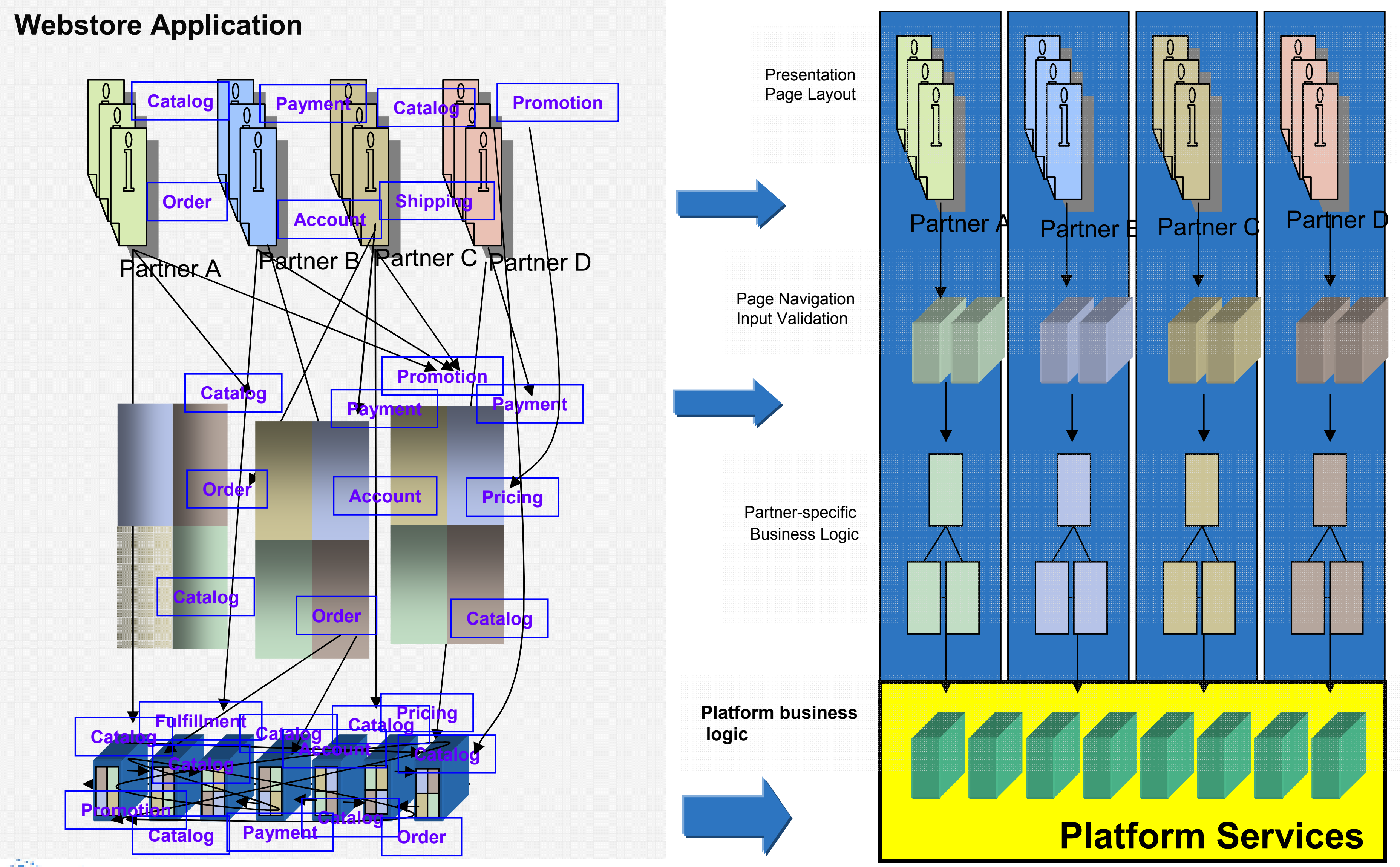


I just want the most active products today – no presentation please!

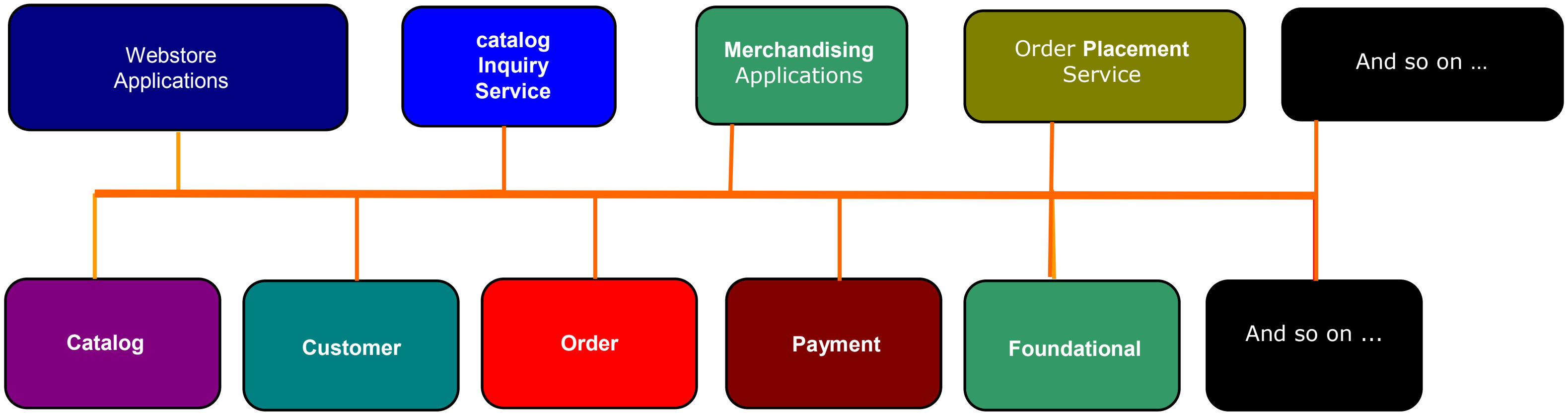
I just want the 'singles' - no voice overs!

we needed to free the services from the webstore …

… along with their underlying components



we needed a strategy – *balance applications with components and services starting with the webstore*

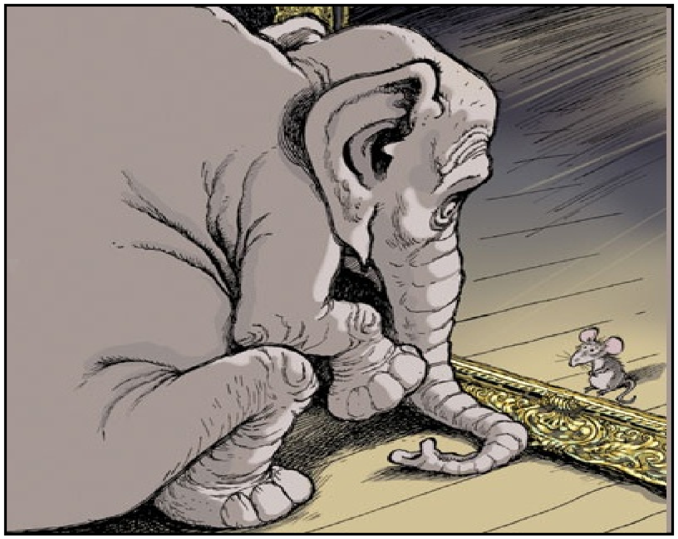


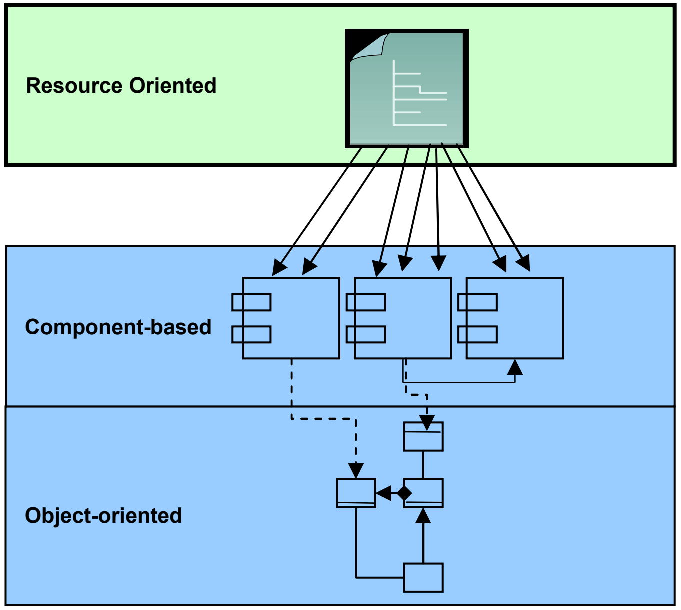
- **Design the Service Contracts**
  - Logical Resources, Representations and Messages
  - SLAs
  - Security Concerns
  - Community Concerns
- **Construct the Services**
  - Using the underlying components when available
- **Refactor the Services**
  - The implementation to use components if appropriate
  - The interface based on consumer feedback

- **Decompose the Webstore into Components**
- **Start with the Catalog components (Products, Categories, etc.)**
- **Continue with Orders, Customers, Payment, etc.**
- **Factor out Foundational Component Capabilities**
  - Persistence support, Configuration, Instrumentation, etc.
- **Components have contracts with consumers too**
  - Can be tested standalone, and change impact can be better isolated

digging deeper -
distinct design approaches were needed for the large and the small

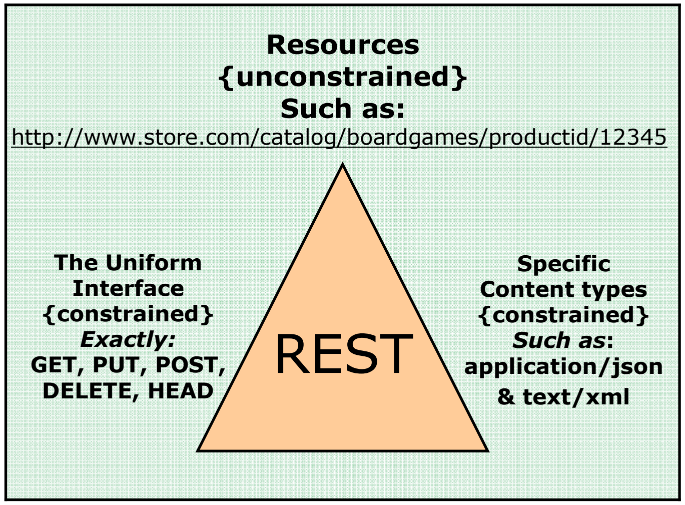
- **String theory aside, physicists have yet to unify general relativity & quantum mechanics …**
  - … so why do we so often think one size fits all with software?
- **REST and Messaging for Distributed Services: “the large”**
  - Coarse Grained
  - Data centric
  - The request/reply payloads as the engine of application state
  - Stateless between service requests
  - Naturally Resource Oriented
- **Domain Driven Design for Objects & Components: “the small”**
  - Naturally tightly coupled clusters of classes making up the implementation is object oriented
  - Behavior-centric, rather than state/resource centric
  - Often stateful
  - Relatively fine grained
  - Often exposing rich sets of operations





the large –
service oriented design at gsi commerce

- **A Consumer Says “Give me product details on Plasma TVs”**
  - Procedural
    - Product.getDetails(ProductType.TELEVISION, ProductSubType.PLASMA, “Radioshack”)
  - RESTful
    - GET /store/radioshack/catalog/television/plasma?type=detail
  - Messaging
    - <ProductDetailRequest><Store>RadioShack</Store>…
  - What’s more Natural? More Practical? Depends on the consumer!
- **For Distributed Services, Messaging and RESTful HTTP are generally our first choice**
- **Services whose primary consumers are “of the Web” seem to naturally think in RESTful terms**
  - AJAX, Flash
  - Generally talking via JSON
  - Rarely use SOAP (SOAP isn’t usually very RESTful)
- **Integration Services naturally communicate via Messaging**
  - Generally talk via XML
  - May use SOAP (in which case, please, doc/literal)
- **Messaging can be RESTful**
  - Atom feeds are a good example
- **We provide SOAP for those who need it**
  - Usually server-side/integration scenarios
- **Security? Good old SSL when its needed**
  - Usually with an API key for external consumers so we can track usage
  - WS-Security for partners and clients that need it: useful for integration collaborations



Resources
{unconstrained}
Such as:
http://www.store.com/catalog/boardgames/productid/12345

The Uniform Interface {constrained}
Exactly:
GET, PUT, POST, DELETE, HEAD

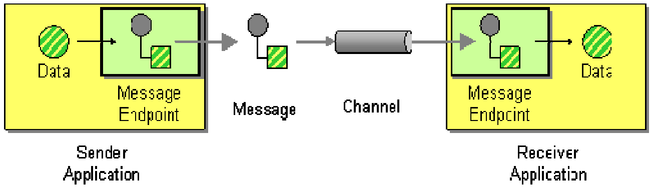
REST

Specific Content types {constrained}
***Such as*:**
application/json & text/xml

JAX-RS/ Jersey

Mule ESB

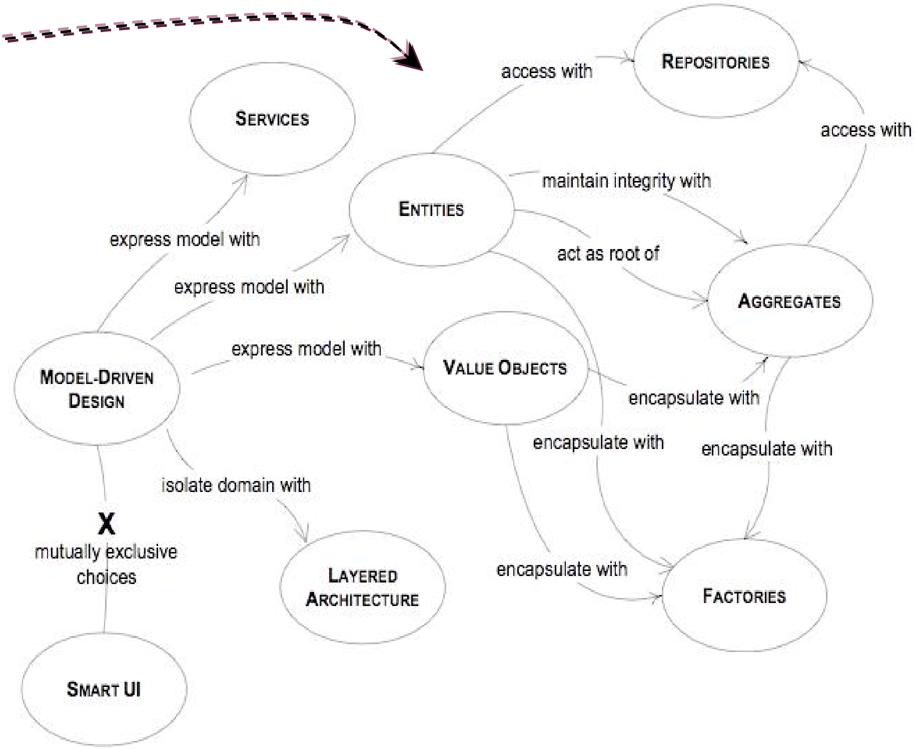
Programming *with* the Web – ***not against it***



the small –
component based development at gsi commerce

Design

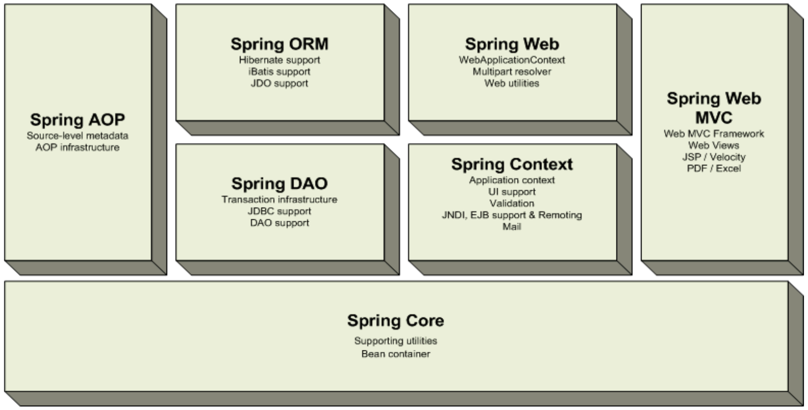
- **Domain Driven Design as overarching philosophy**
- **Extensibility Principles**
  - Extend Service, Domain and Repository behavior via Interface Inheritance, Delegation and Composition
  - Push Extensibility outward toward (and to) the client
  - Annotate All Extensibility Points
- **Maintainability Principles**
  - Streamline Associations
  - Streamline State
  - Streamline *Implementation* Inheritance
  - Streamline Visibility
  - Streamline Mutability
  - Annotate Concurrency Semantics
  - Unit Tests for virtually everything and anything



REPOSITORIES
SERVICES
ENTITIES
AGGREGATES
VALUE OBJECTS
MODEL-DRIVEN DESIGN
LAYERED ARCHITECTURE
FACTORIES
SMART UI
access with
access with
maintain integrity with
act as root of
express model with
express model with
express model with
encapsulate with
encapsulate with
encapsulate with
encapsulate with
isolate domain with
X
mutually exclusive choices

Technologies

- **Spring DI for decoupling dependencies**
- **DAOs with Spring JDBC Templates**
  - Initial component-based persistence encapsulation
  - Focus was on solidifying component responsibilities and boundaries
  - Looking now at full OR/M solution as next step
- **Spring Modules AOP-based Caching**
  - Allows multiple cache implementations to be plugged in
  - Cacheable Aspects on CRUD operations of our Entity Repositories
  - EHCache is our current cache provider (local cache/overflow to disk)
- **Spring Security**



Spring AOP
Source-level metadata
AOP infrastructure

Spring ORM
Hibernate support
iBatis support
JDO support

Spring DAO
Transaction infrastructure
JDBC support
DAO support

Spring Web
WebApplicationContext
Multipart resolver
Web utilities

Spring Context
Application context
UI support
Validation
JNDI, EJB support & Remoting
Mail

Spring Web MVC
Web MVC Framework
Web Views
JSP / Velocity
PDF / Excel

Spring Core
Supporting utilities
Bean container

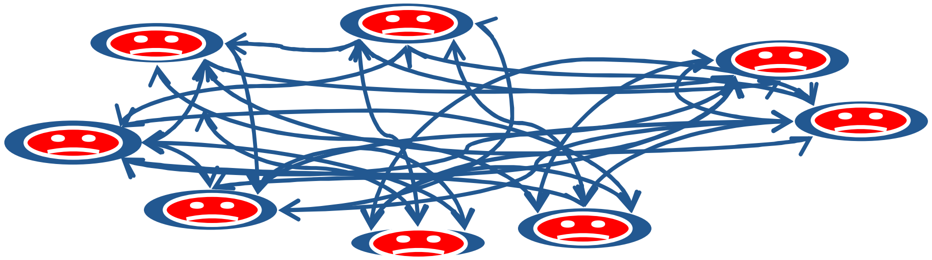
Future Directions

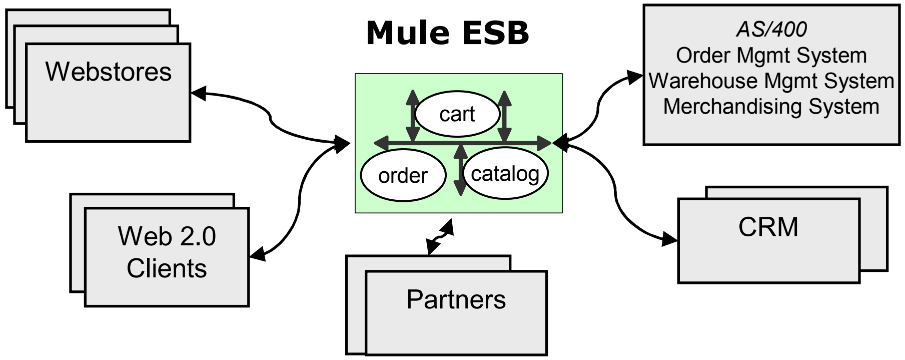
- OSGI
- Complete Object/Relational Mapping (OR/M) Solution
- Distributed Caching

a big bang approach to services did not make sense …

… so we needed an integration solution

- **We identified our integration needs as we grew our service ecosystem**
  - Service Interface Translation to support our divergent partner desires without it bleeding into the core of our platform
  - A way to expose "legacy" capabilities as services until we could get to componentizing them (or because it makes no sense to componentize them)
  - A way to compose aggregate services from discrete ones
  - A way to abstract a heterogeneous smorgasbord of backend integration solutions
- **The name Enterprise Service Bus (ESB), like SOA, ceases to have much meaning these days**
  - We simply needed a good integration solution
- **Mule ESB gave us just what we needed without giving us a lot of vendor baggage**
  - Its event driven model, based on Enterprise Integration Patterns, was familiar and intuitive
  - Its integration with Spring was nice as well
  - It could be used as an ESB but also simply as an embeddable integration framework





Mule ESB

Webstores

Web 2.0 Clients

Partners

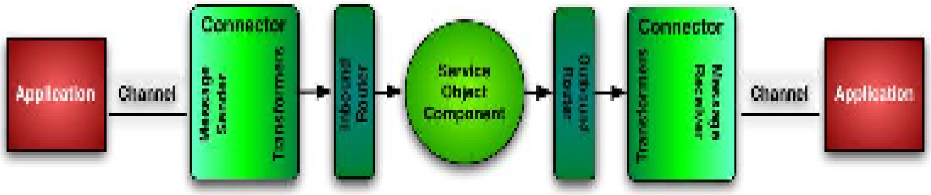
cart

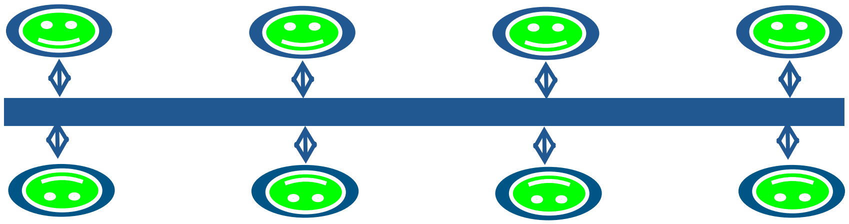
order

catalog

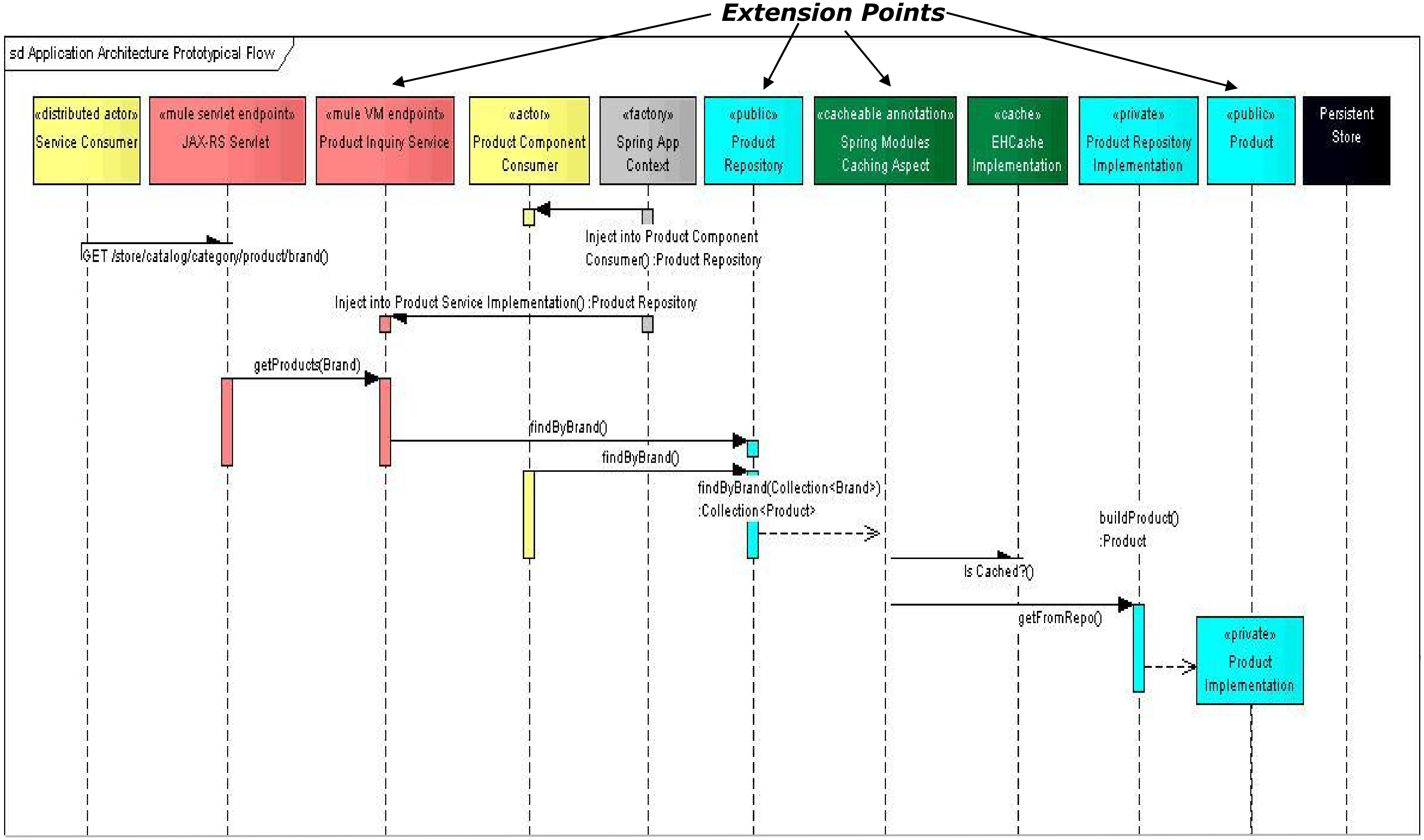
AS/400
Order Mgmt System
Warehouse Mgmt System
Merchandising System

CRM

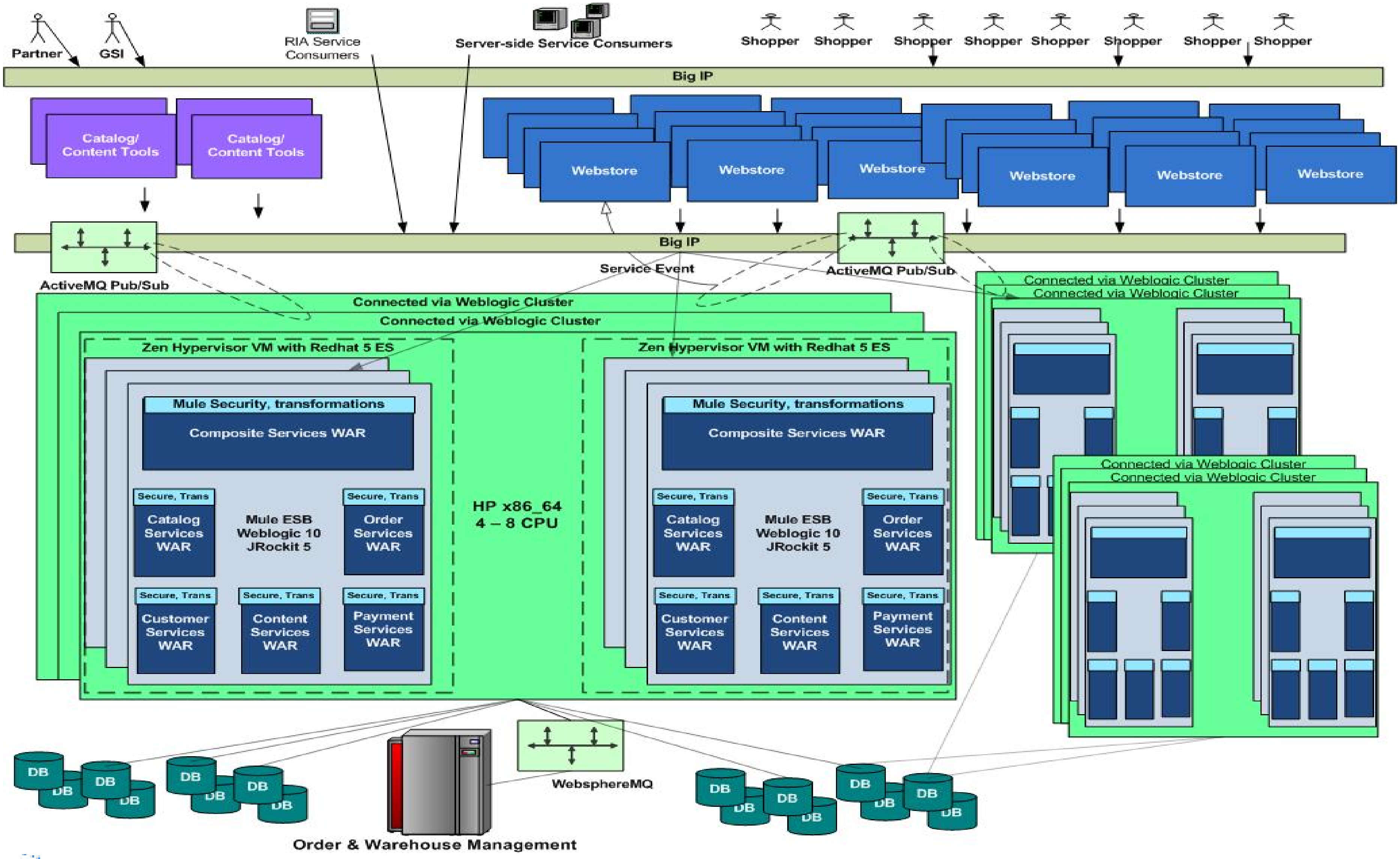




the logical …


Extension Points
sd Application Architecture Prototypical Flow
«distributed actor» Service Consumer
«mule servlet endpoint» JAX-RS Servlet
«mule VM endpoint» Product Inquiry Service
«actor» Product Component Consumer
«factory» Spring App Context
«public» Product Repository
«cacheable annotation» Spring Modules Caching Aspect
«cache» EHCache Implementation
«private» Product Repository Implementation
«public» Product
Persistent Store
Inject into Product Component Consumer() :Product Repository
GET /store/catalog/category/product/brand()
Inject into Product Service Implementation() :Product Repository
getProducts(Brand)
findByBrand()
findByBrand()
findByBrand(Collection<Brand>) :Collection<Product>
buildProduct() :Product
Is Cached?()
getFromRepo()
«private» Product Implementation

… and the physical

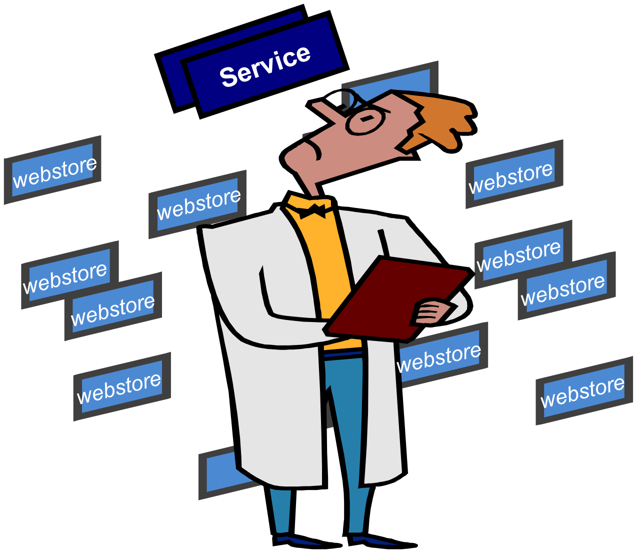


gsi commerce

the downside of services?

two words: operational complexity

- We deploy releases for over 85 partners every two weeks
- We have a large number of projects with significant changes flowing through development, integration, QA, and UAT environments at any given time
- We have 100s of machines and many 100s of application server instances across two data centers for production roll out
- 95% of all of this activity is driven by what is primarily one application we all know very well: the Webstore
- Once we cracked open the Webstore and decomposed its parts into services, we gave birth to new deployment units that were most decidedly not web stores
- The Enterprise Service Bus helps hide this complexity from Service Consumers but not from administrators and release managers



gsi commerce

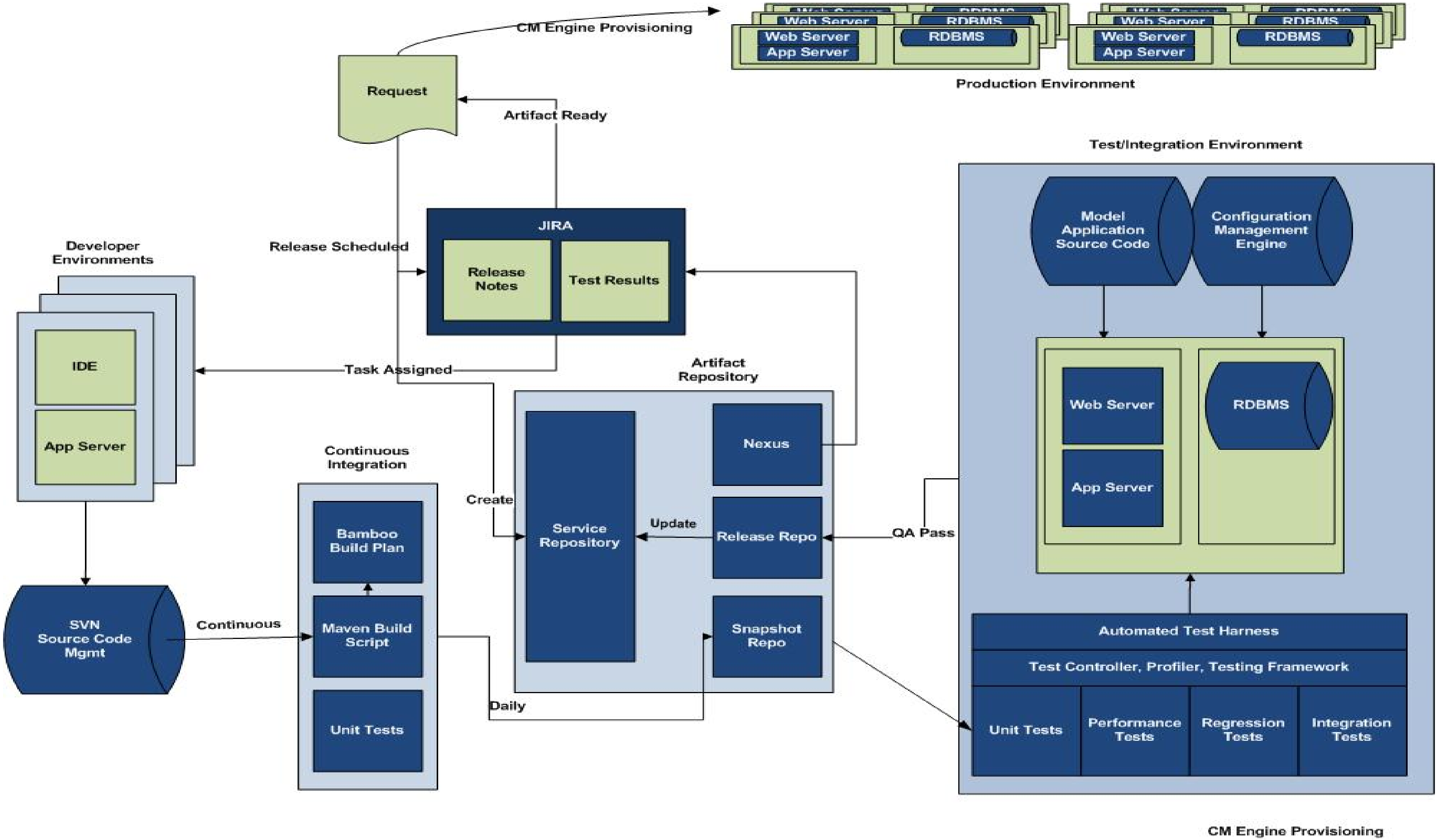
how do we tackle this complexity?

policies applied through automation through service lifecycle …

Policy	Requirement	Managing System
Source Code Management	• Component and Service Source Control – Single Component Library / Single Module	**Subversion**
Request Tracking	• Component and Service has Issue Tracking System to manage requests	**JIRA**
Build	• Components and Service have a Maven build that implements site, install, and test targets	**Maven**
Deployment	• Services are capable of configuring & deploying themselves using the standard script-driven engine	**GSI Configuration Management Engine**
Continuous Integration	• Components & Services have Continuous Integration plans for each scheduled version	**Bamboo**
Code Coverage	• Components and Services have 90% Unit test coverage	**Cobertura, JUnit**
Documentation	• Components & Services have supporting rollout/usage documentation, published to Mule Galaxy	**Confluence, Mule Galaxy Repository**
Versioning	• Component versions that are QA passed are published to the Maven Release Repository • Versioned Service Metadata is auto published to Mule Galaxy Repository	**Nexus, Mule Galaxy Repository**
Code and Module Structure	• Services & Component Implementation Idioms & Project Module makeup meet GSI Minimum Application Requirements	**Maven, Checkstyle, Cobertura**
Production Monitoring and Management	• All components have associated JMX beans, for runtime production behavioral analysis	**Mule ESB, Mule HQ, Ganglia, Nagios, JManage**
SLA Compliance	• All E-Commerce Web Capabilities, including our Web Services, are continuously monitored and measured at numerous locations across the internet	**Gomez**

service governance -

… but policies must be baked into daily practice

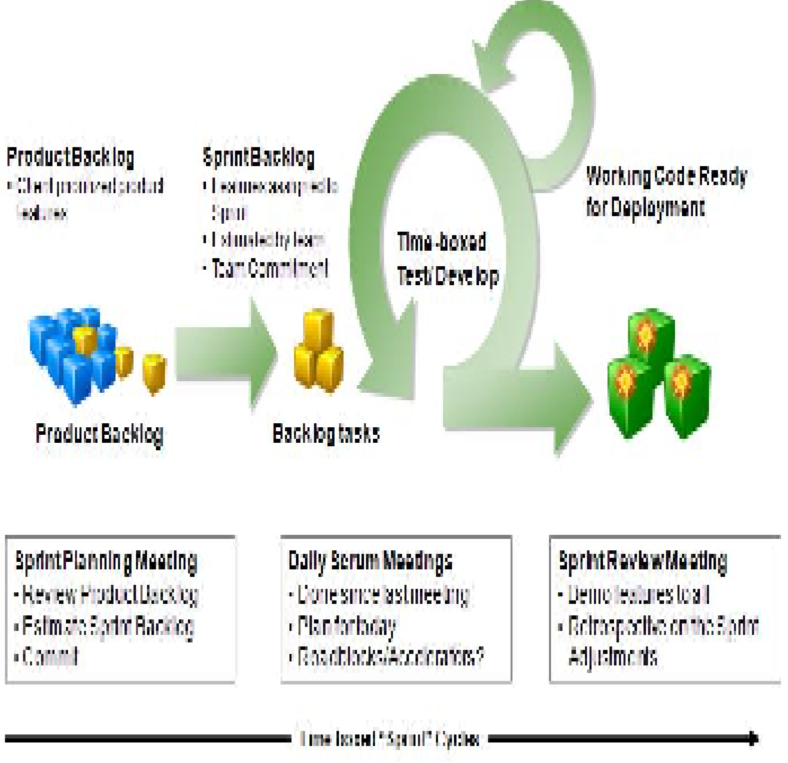


gsi commerce

daily practice – the process

agility under the waterfall

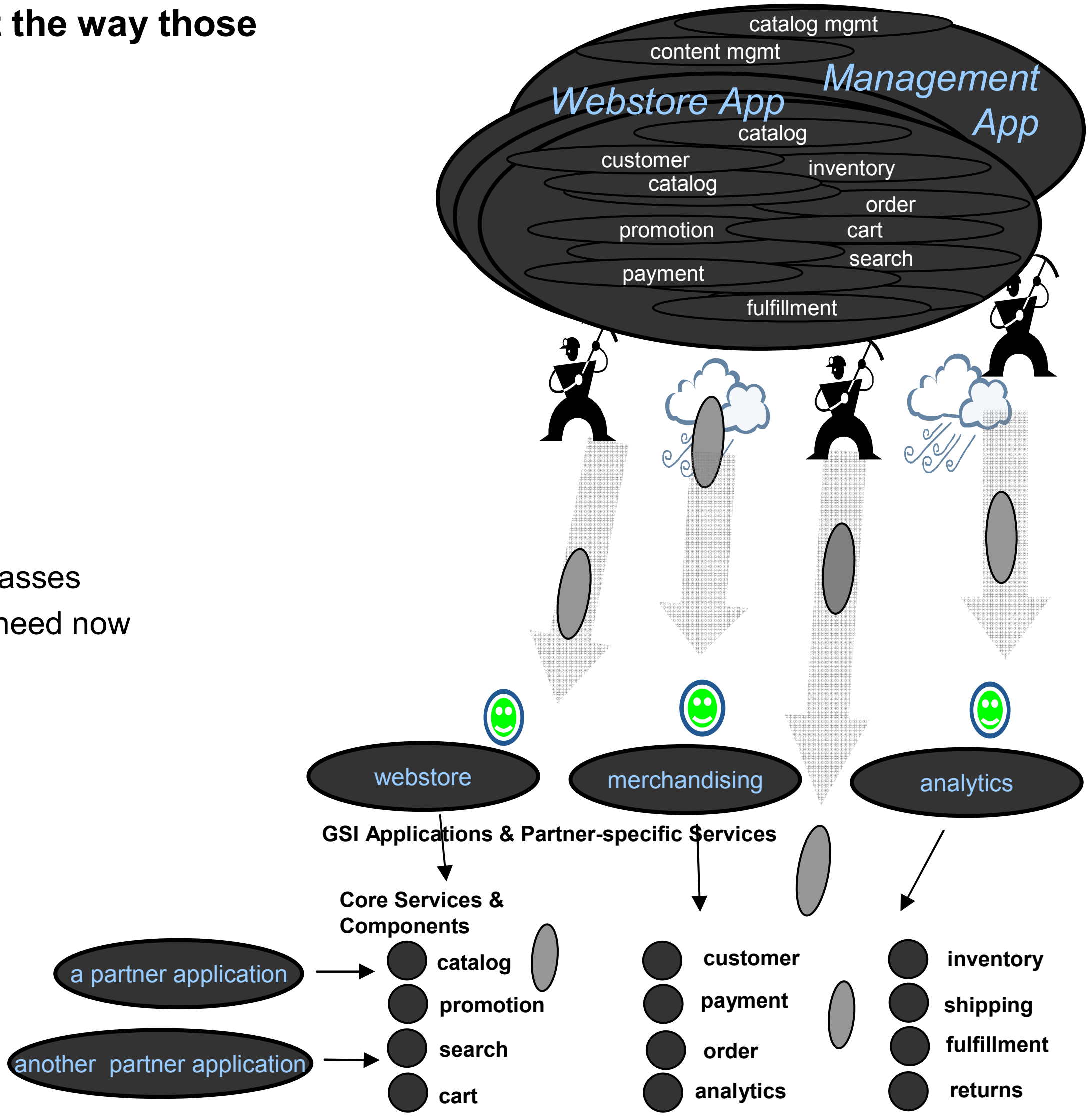
- **Deliverables, Resource/Time Estimates and major milestone artifacts are agreed upon for each release**
  - Formal backlog of tasks prioritized based on this
- **Each release project has macro-level waterfall phases**
  - Requirements, Design, Implementation and Test
  - Each phase ends with formal release of agreed upon deliverables
- **Each phase in turn consists of a series of iterative, two week sprints**
  - Each sprint involves tasks with elements of requirements, design, implementation and test
    - The relative **focus** on each varies based on the phase
  - Culminating with demonstrations to the business stakeholders
  - The sprint deliverables must be production ready
    - We have a strict definition of what 'done' means
- **Management, development, test, requirements & architecture members all actively involved in each sprint**
- **Focused on peer review, refactoring and, above all, tests**



gsi commerce

summary

- **You need to evolve not only capabilities but the way those capabilities are exposed**
  - ❖ Scalability exists at multiple dimensions
  - ❖ Functional diversity has scale as well
- **Decompose your …**
  - ❖ Capabilities into components
- **Compose your …**
  - ❖ Components into Services
- **Allows you to …**
  - ❖ Quickly construct applications needed for the masses
  - ❖ Give partners building blocks for the apps they need now
    - ***Give them iTunes in addition to Albums!***
- **But be sure you …**
  - ❖ Automate the build-test-provision/deploy cycles
  - ❖ Are relentless in refactoring
  - ❖ Are fanatical about consistency
    - ***It reduces technical debt***
    - ***It slows the architectural decay***
  - ❖ And it never hurts to say too much …
    - ***Test, Test, Test***
    - ***Automate, Automate, Automate***



thanks for listening!

questions?

mailto: buzzards@gsicommerce.com