SOA: Beyond the Hype

Rob Richards
3/15/2007
rrichards@ctindustries.net
http://xri.net/=rob.richards

OASIS SOA Definition

"Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations."

OASIS Service Oriented Architecture TC "OASIS Reference Model for Service Oriented Architecture V 1.0", http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf, 2 August 2006

The SOA Hype

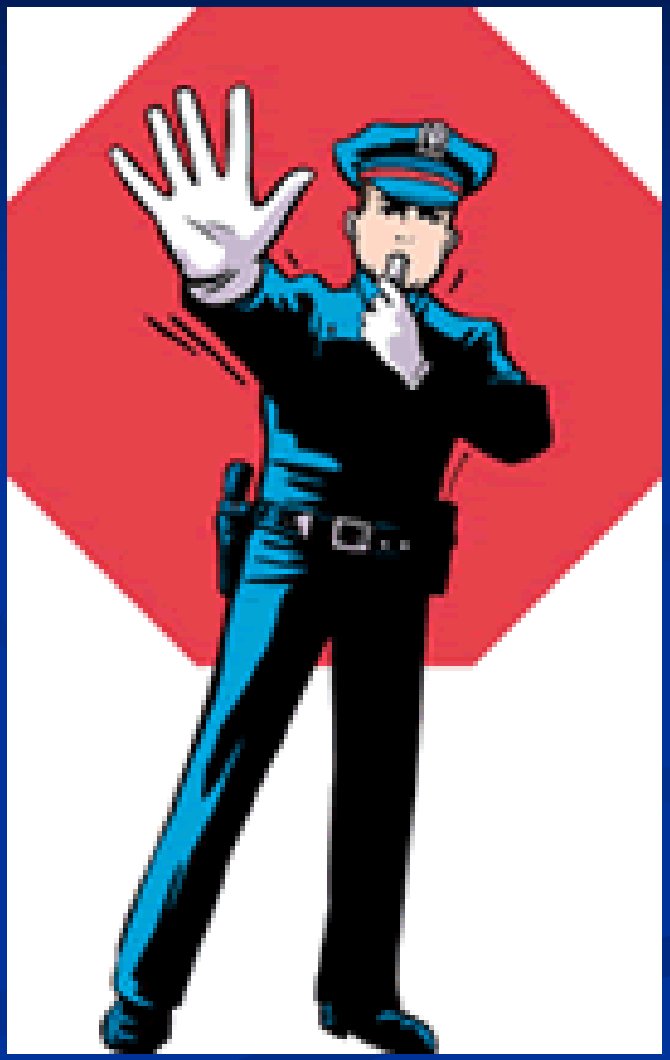
- Increased Return on Investment (ROI)
- Customer Retention
- Faster time to market
- Seamless Interoperability
- Decrease development time
- Simplify System maintenance
- Business Agility
- Re-usability

SOA Facts
www.soafacts.com

- Guns don't kill people, the SOA WS-* stack kills people.
- SOA is being used in the developing world to solve hunger. Entire populations will be fed on future business value.
- Not content to just best sliced bread, SOA is actually the best thing since beer, wine, coffee, ice cream, chocolate... oh, and sliced bread.
- Dante has a special level in hell for consultants whose resumes do not say SOA.
- SOA - building contractor jobs, one Visio slide at a time.
- The Answer to the Ultimate Question about Life, the Universe, and Everything is ..... SOA

STOP!

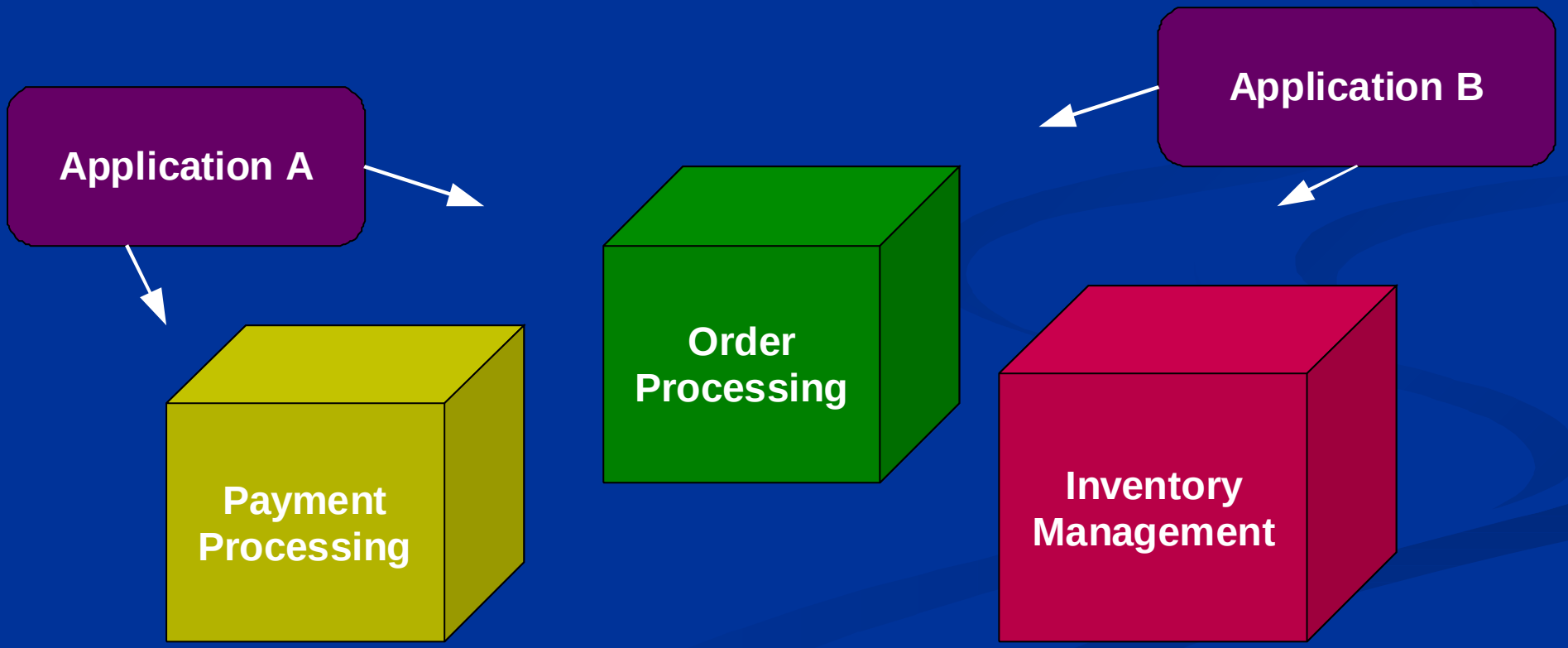
Let's take a step back and get a better understanding of exactly what Service Oriented Architecture is all about before betting the business on it.



What is SOA?

Simplified and Generalized Answer

Service-oriented architecture is a style of building applications based on independent and re-usable building blocks that provide some specific functionality.



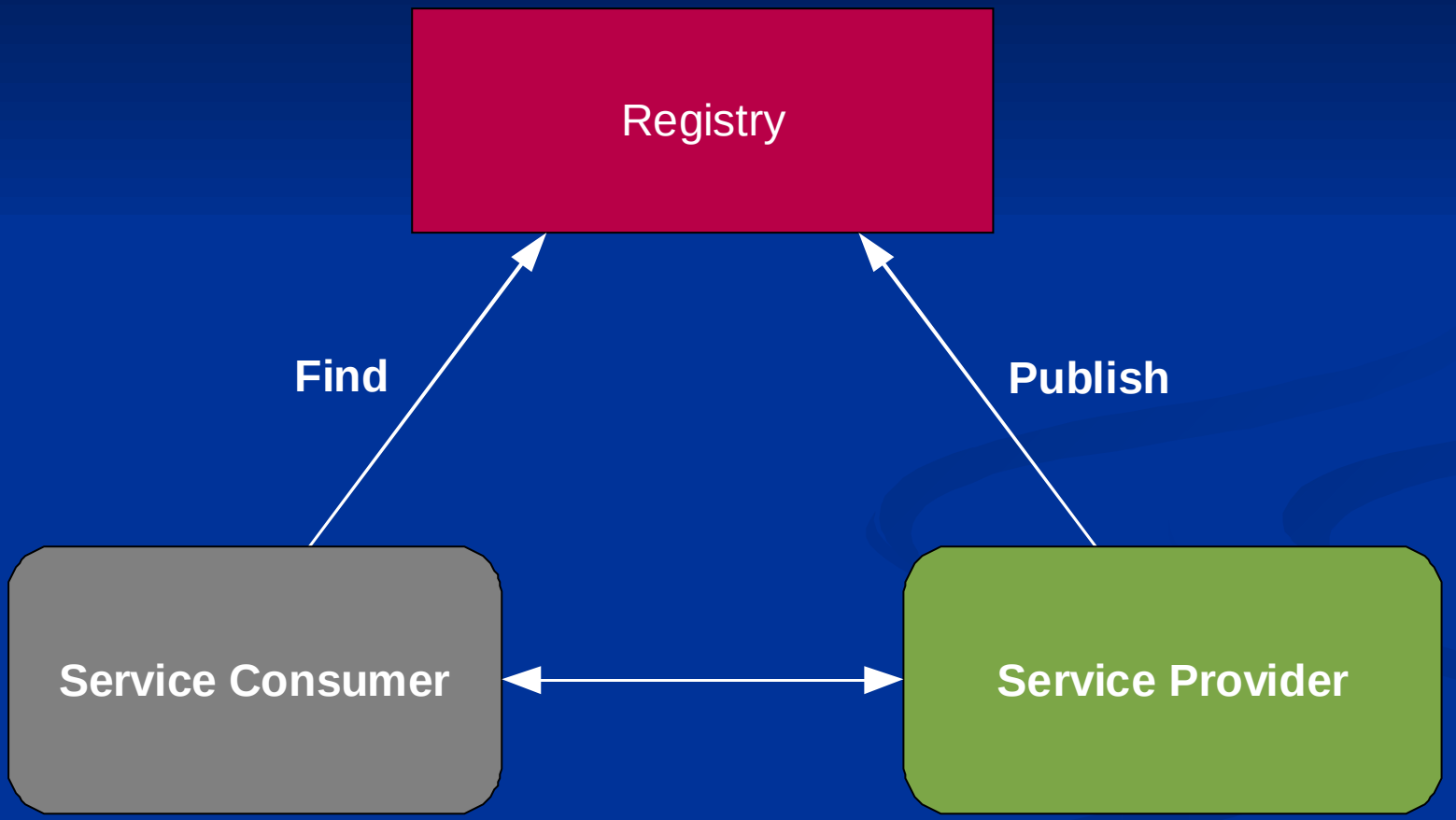
SOA Is a New Technology!

- It's not new
  - EJB
  - DCOM
  - CORBA
- It's not a technology
  - It is an architectural style
  - Vendors like to hawk their wares
- It all depends upon your definition

Services

- A collection of related endpoints
- Loosely Coupled
- Well Defined Interfaces
- Interface Granularity
- Reusable
- Discoverable

SOA Model



Common Technologies

- SOAP
- Web Service Definition Language (WSDL)
- Universal Description Discovery and Integration (UDDI)
- WS-?
  - WS-Security
  - WS-Address
  - WS-xxxxxx

May often hear this referred to a WS-*

What About REST?

- Resources rule
- Uniform interface
- Data caching
- Data centric
- Lightweight and easy to use
- No formal contract

SOA and REST

SOA

Order Processing

getOrder(o_id)
getOrders(c_id)
addOrder(c_id, Order)
updateOrder(Order)

Customer Management

getCustomer(c_id)
getCustomers()
newCustomer(C)
updateCustomer(C)

REST

/orders

/orders/{o_id}

/customers

/customers/{c_id}

/customers/{c_id}/orders

GET
PUT
POST
DELETE

SOA: More than WS-*

- Many definitions of SOA do not preclude REST
- Different jobs require different tools
- Common Goals
  - Distributed
  - Interoperability
  - Reusability
  - Loosely coupled

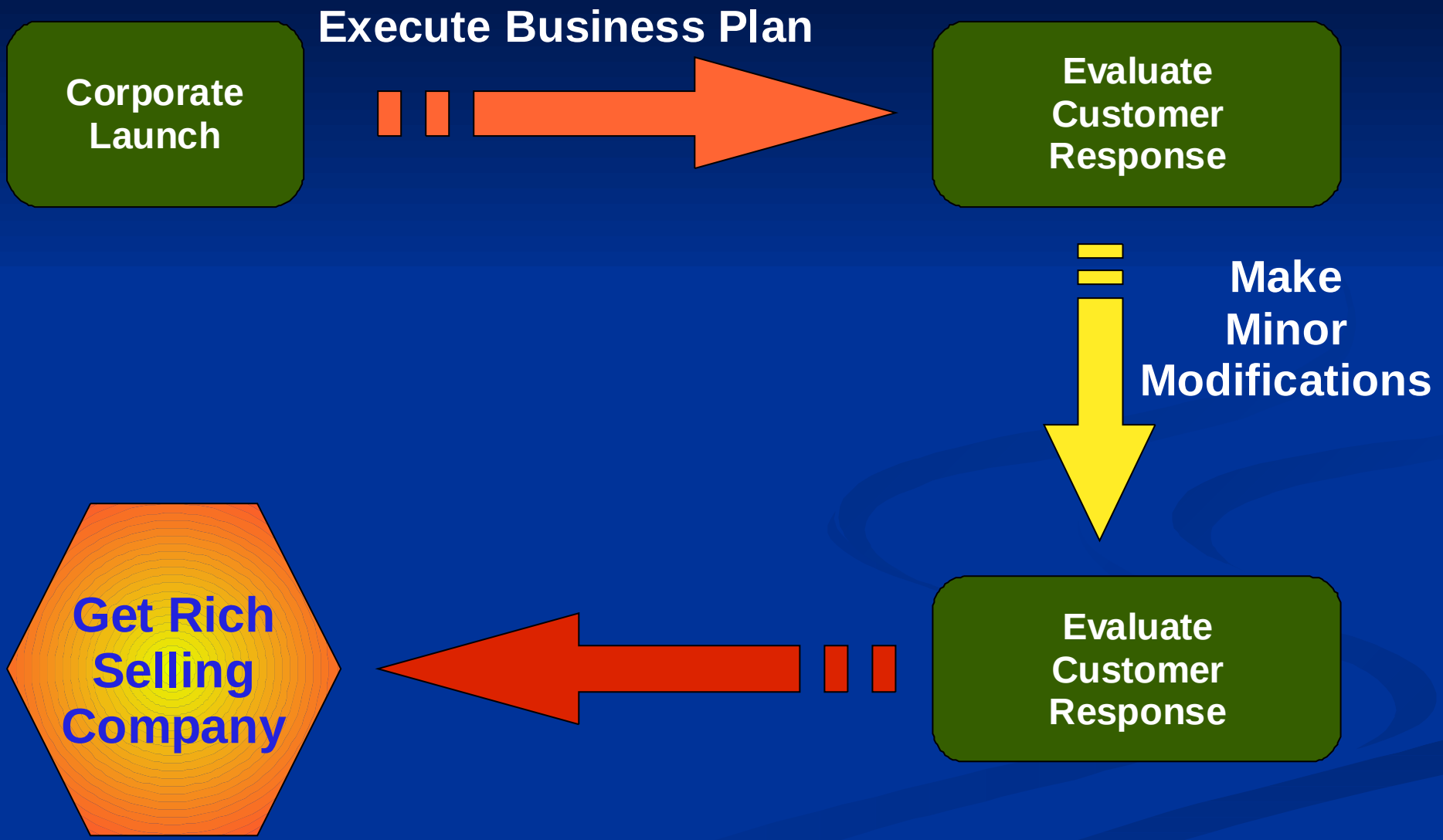
Arguments for WS-* Stack

- Existing architecture
- Legacy System Integration
- Granular Security Requirements
- Two-phase commits
- Asynchronous Messaging
- Target Consumers

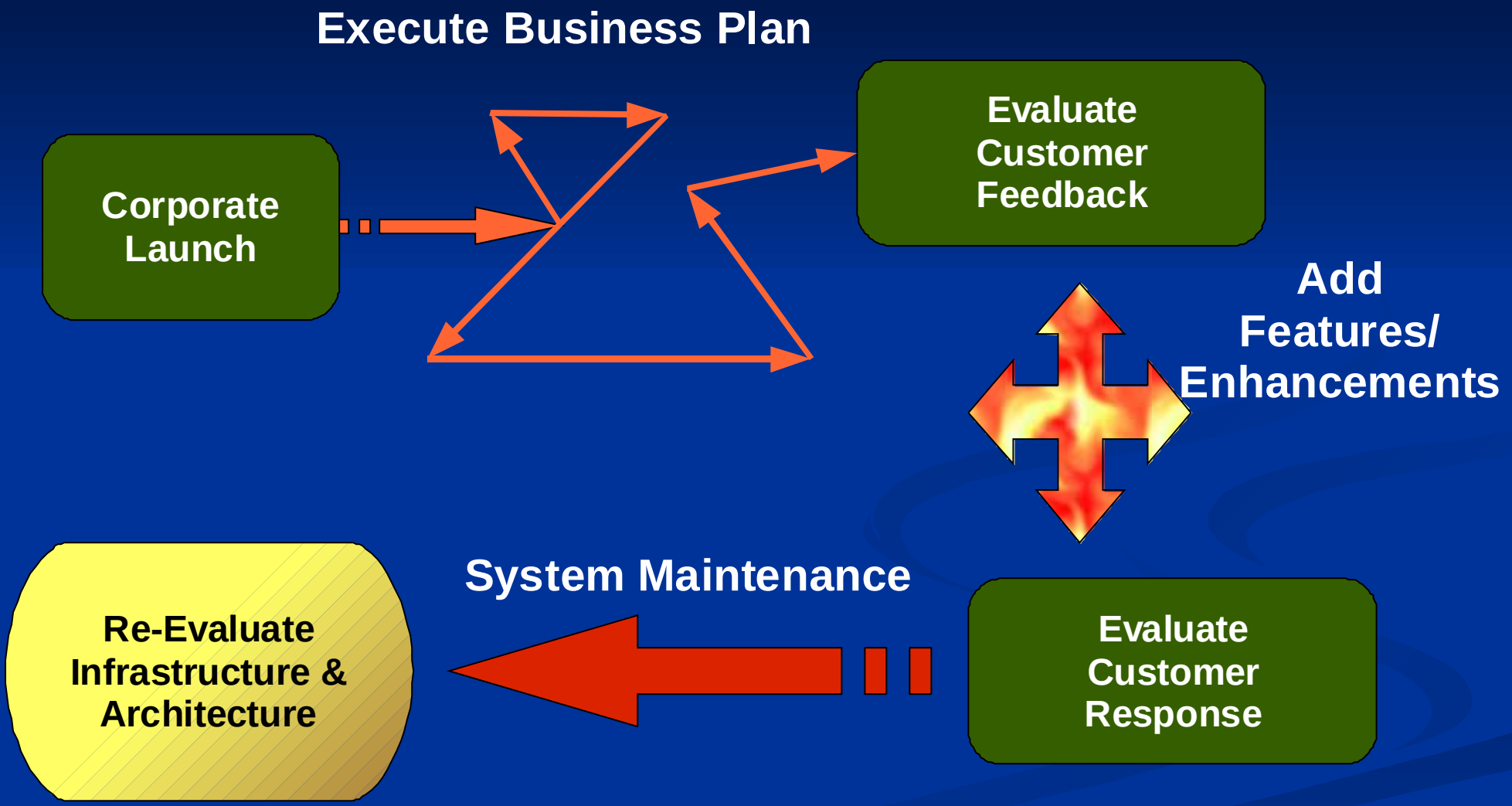
Implementing SOA
A Continuous Cycle

- Clearly define why you are going to implement SOA
  - Everyone else is doing it so why shouldn't we?
  - Tangible reasons that would be beneficial to the company
- All levels of the organization must see the benefits
  - Impacts many areas of a company from finance to operations
  - Unless everyone buys into the idea it will fail from the start
- Assess and evaluate the current infrastructure
  - Visualize the infrastructure by areas of functionality
  - Are there areas that will benefit the most?
- Make a Plan
  - Clearly identify what and how you plan on implementing
  - DO NOT START OFF TO BIG!

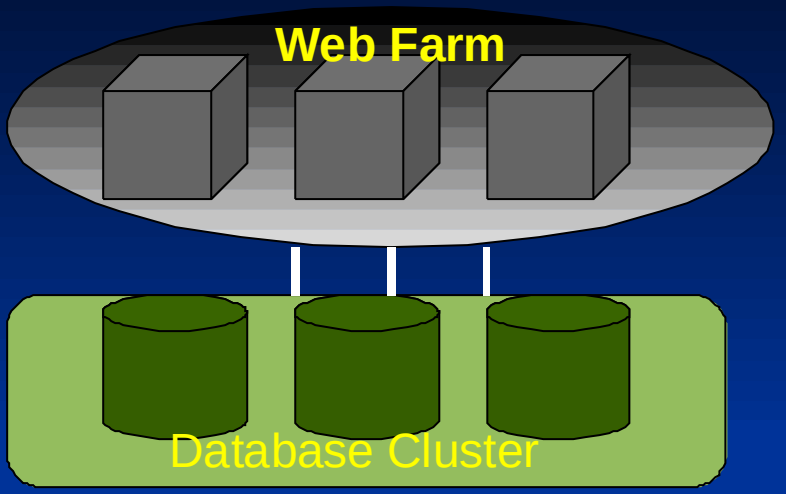
Ideal Development Path



Development Reality



Initial Architecture



Hosting Facility

DSL

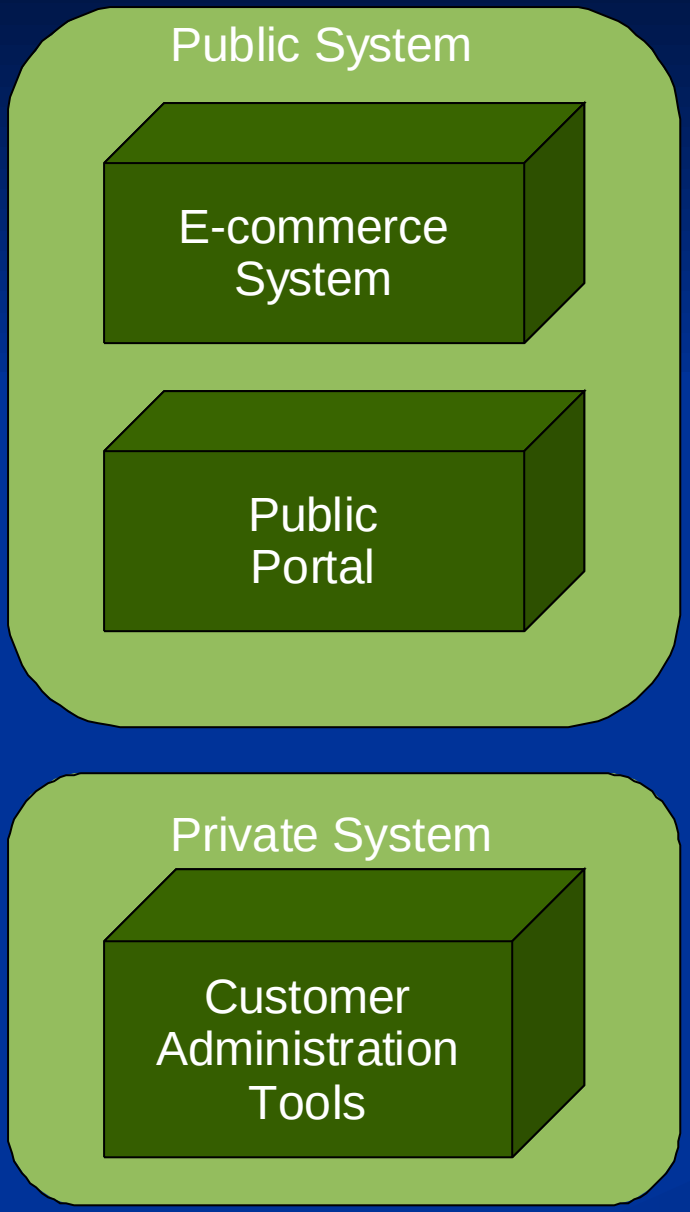
Local Office

Financial Application

Customer Relationship Application

Salesperson Management Application

Initial Software Architecture

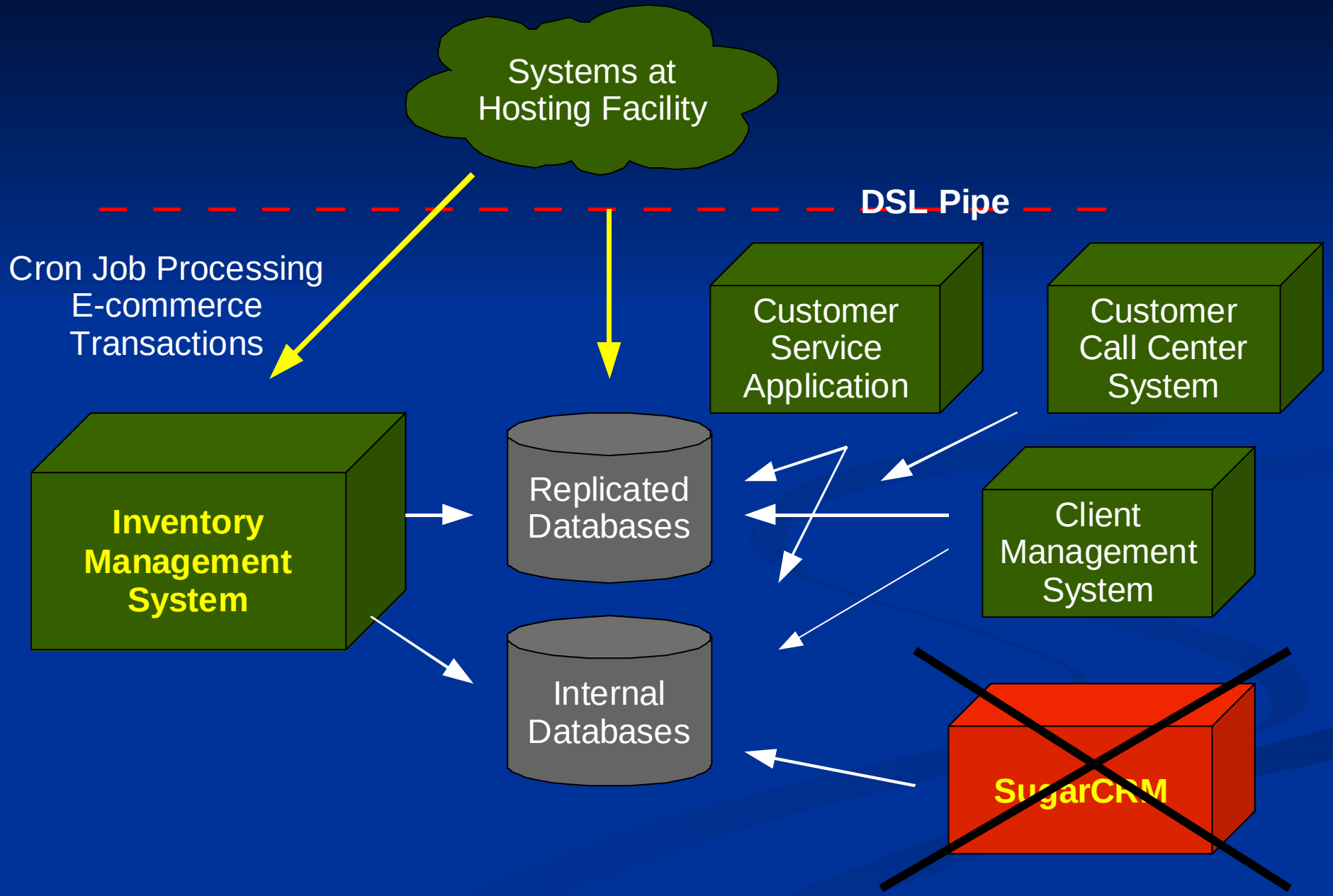


- Tightly coupled
- Re-Usability
  - Object Oriented coding
  - Organized procedural code
- Agility based on how fast we can code and get things tested
- Local office used third party stand alone applications

The Growing Pains

- Customer base is growing
- Organization is growing
- Existing infrastructure no longer meets company needs
  - Need more robust Customer Management system
  - Need more robust SalesPerson Management system
  - Business requires tighter integration into public web applications
- Keep expenses down
- **Time to market is critical!**

Git 'Er Done!


Systems at Hosting Facility
DSL Pipe
Cron Job Processing E-commerce Transactions
Customer Service Application
Customer Call Center System
Inventory Management System
Replicated Databases
Client Management System
Internal Databases
SugarCRM

So Where's the SOA?

- Potential Disadvantages
  - We now have duplicate code
  - Multiple systems performing similar functionality
  - Potential higher maintenance costs
  - More systems to administer
- Potential Advantages
  - Systems were rolled out quick
  - Minimal cost to develop applications
  - No impact on rest of organization

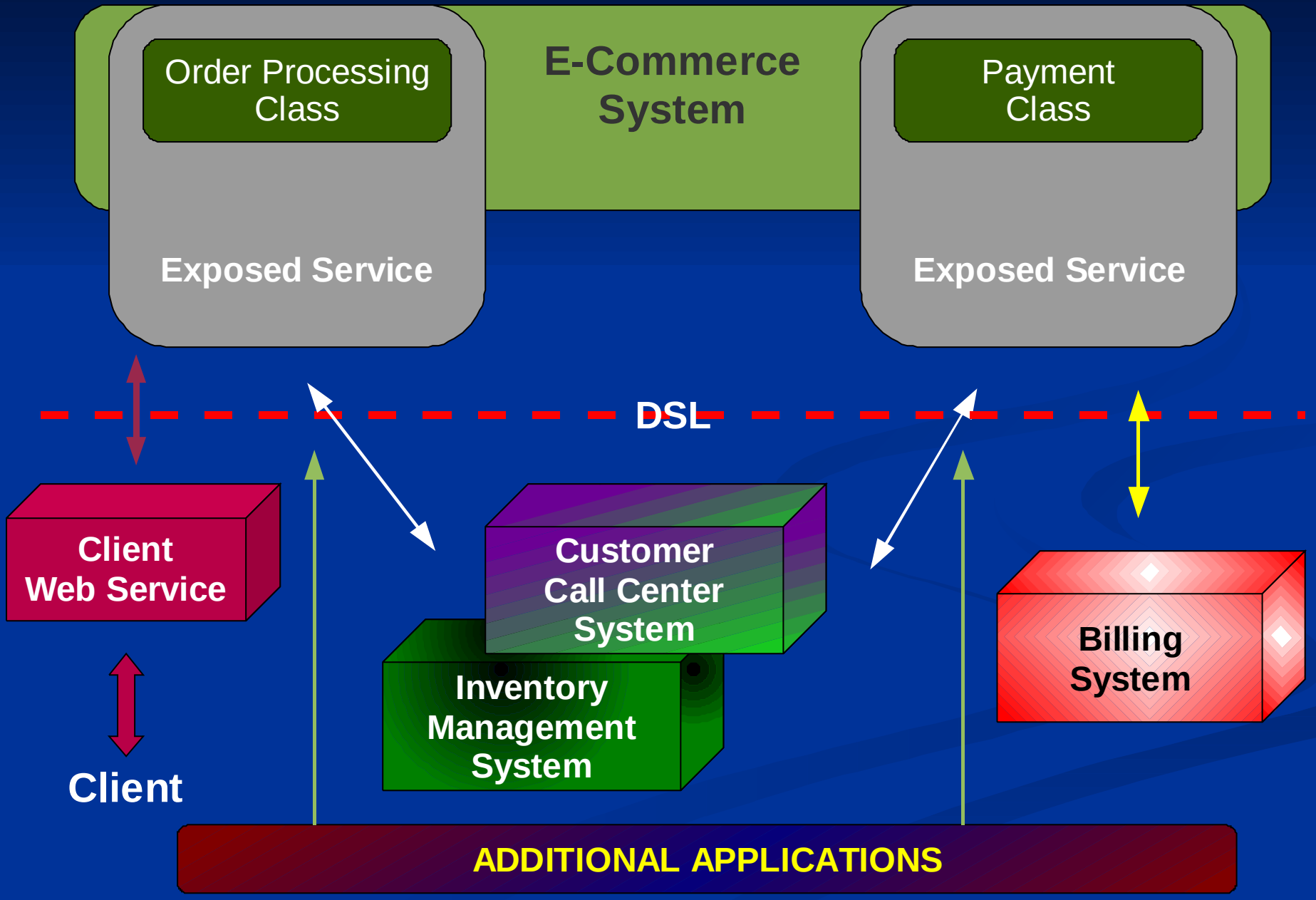
Things to Consider

- Justification or Mandate?
- Existing problems or challenges?
- Is it cost effective?
- It is time effective?
- Impact on current projects and applications
- Plan before you execute
  - What do you expect to achieve?
  - Not everything should be a service

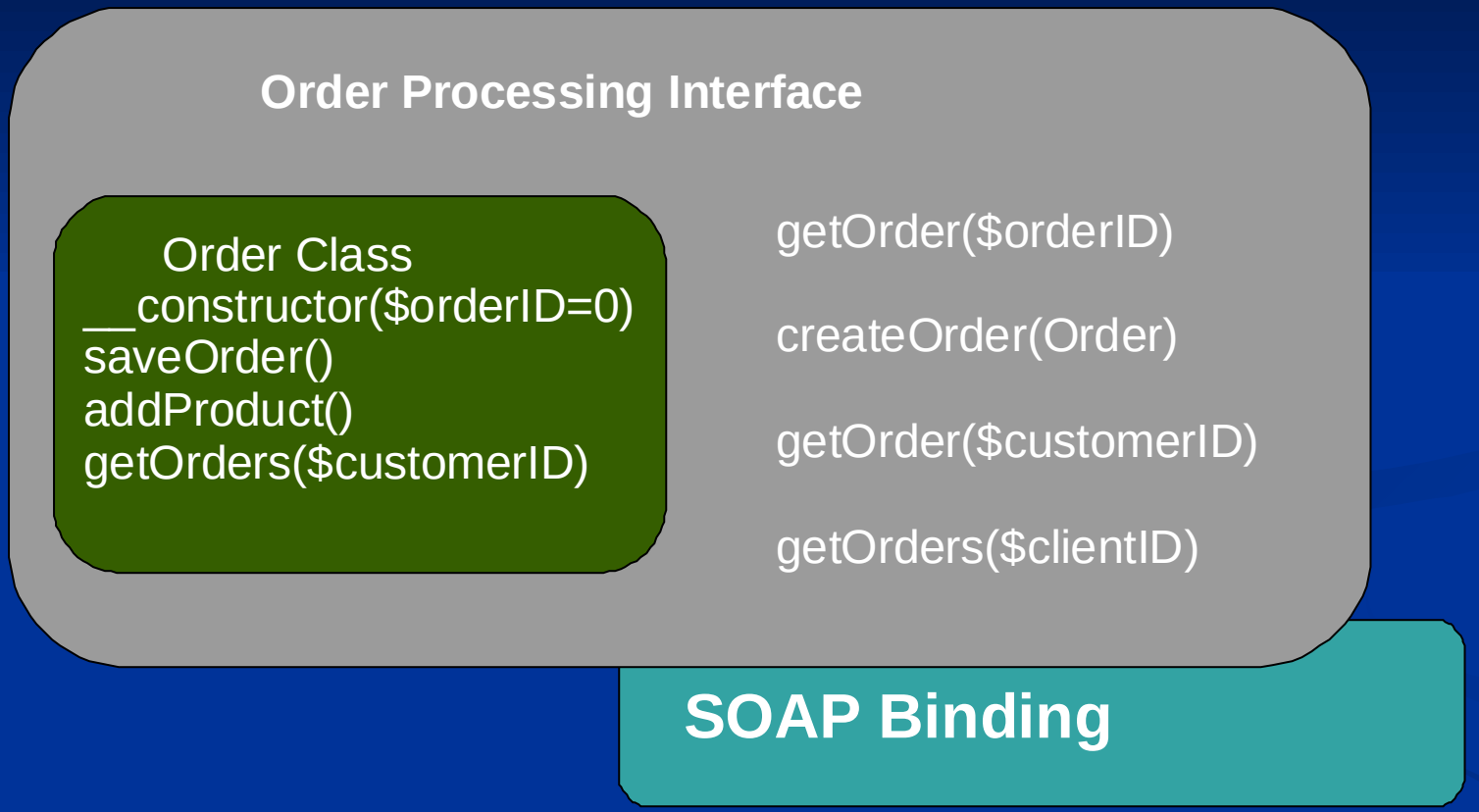
Finally implementing SOA

- Order Processing
  - Order information must be exposed to our clients
  - Orders must be accessible from a number of different systems
  - 100% uptime is critical for public E-commerce system
- Payment Processing
  - Multiple applications process payments
  - Need to be able to globally swap out underlying processor being used
  - 100% uptime is critical for public E-commerce system

The Architecture



Order Processing Service



Why SOAP?

- Existing system already tightly coupled
- Simple code replacement with SoapClient calls
- Developers comfortable with SOAP
- Majority of orders handled via E-Commerce system
  - SOAP overhead has minimal impact on data traffic
  - SOAP overhead had minimal impact on system resources
- No real arguments against using REST

Look before you Leap!

- Is the service critical to the business?
  - What happens in the event of network problems?
  - What happens in the event of hardware problems?
- Are there any security issues?
  - Is HTTPS sufficient?
  - Does WS-Security need to be leveraged?
  - Are there any other authentication factors to consider?
- How do you keep track of all the services?
  - Are they tracked manually?
  - How do you know which ones are available?
  - Do you implement a UDDI Registry?

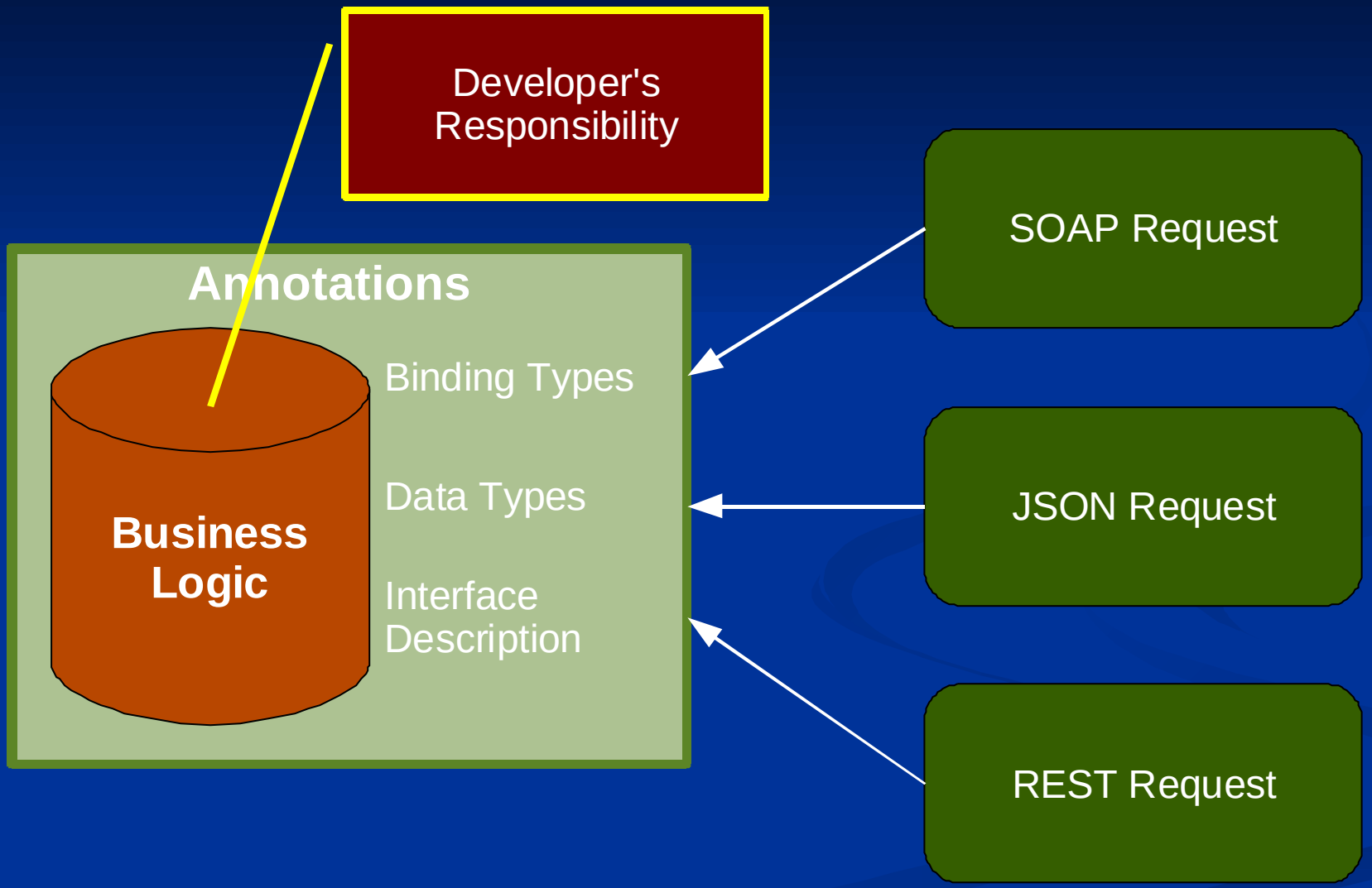
Implementing SOA in PHP

- REST
  - XML data handled by any of the XML extensions
  - JSON data can be handled by PHP 5.2 by default
- SOAP
  - ext/soap – built in SOAP support
  - Axis2 – WSF for PHP currently in development will also support pluggable WS-* stack
  - Other alternatives (pear/SOAP, nuSOAP)
- Service Component Architecture
- Simple Asynchronous Messaging

Service Component Architecture (SCA)

- Project by the Open Service Oriented Architecture (OSOA) collaboration - http://www.osoa.org/display/Main/Home
- Allows the developer to concentrate on the business logic rather than how it is all connected together
- http://www.osoa.org/display/PHP/SOA+PHP+Homepage
- Combined with the SDO extension
- Pecl repository: http://pecl.php.net/package/SCA_SDO
- Still in development

SCA


Developer's Responsibility
Annotations
Business Logic
Binding Types
Data Types
Interface Description
SOAP Request
JSON Request
REST Request

SCA Component

```
include "SCA/SCA.php";

/**
 * @service
 * @binding.ws
 * @binding.jsonrpc
 * @binding.rest.rpc
 */

class RobsService {

    /** Find out who I am.
     * @return string Who I am.
     */
    public function whoAmI()
    {
        return "I am Rob Richards";
    }
}
```

Calling the Component

```
include "SCA/SCA.php";
$url = 'http://localhost/robsService.php';

/* Use SOAP Binding */
$robservice = SCA::getService($url.'?wsdl');
$whoami = $robservice->whoAmI();

/* Use JSON Binding */
$robservice = SCA::getService($url.'?smd');
$whoami = $robservice->whoAmI();

/* Use SOAP Binding */
$client = new SoapClient($url.'?wsdl');
$whoami = $client->whoAmI();

/* Use REST Binding */
$whoami = file_get_contents($url.'/whoAmI');

/* Use Local Binding */
$robservice = SCA::getService('/home/rrichards/robweb/robsService.php');
$whoami = $robservice->whoAmI();
```

SAM - Simple Asynchronous Messaging

- Homepage: http://project-sam.awardspace.com/index.htm
- PECL Repository – http://pecl.php.net/package/sam
- MQTT (MQ Telemetry Transport) Messaging Protocol
- Optionally interfaces with IBM Messaging and Queuing using XMS
  - WebSphere Application Server
  - WebSphere MQ
  - Other support IBM middleware products
- Pluggable architecture for additional messaging systems
- Add/Retrieve message capabilities
- Publish/Subscribe capabilities

SAM - WebSphere MQSeries Messaging Server Example

```
/* Code assembled from SAM documentation: http://project-sam.awardspace.com/docs/ref.sam.html */
$conn = new SAMConnection();
$conn->connect(SAM_WMQ,
                    array(SAM_HOST => 'host.example.com',
                          SAM_PORT => 1506,
                         SAM_BROKER => 'examplebroker'));

$msg = new SAMMessage('This is a simple text message');
$msg->header->SAM_REPLY_TO = 'queue://receive/test';

if ($correlid = $conn->send('queue://send/test', $msg)) {
  $resp = $conn->receive('queue://receive/test',
                    array(SAM_CORRELID => $correlid));
} else {
  echo "Send failed ($conn->errno) $conn->error";
}
```

Implementing SOA
A Continuous Cycle (DON'T FORGET)

- Clearly define why you are going to implement SOA
  - Everyone else is doing it so why shouldn't we?
  - Tangible reasons that would be beneficial to the company
- All levels of the organization must see the benefits
  - Impacts many areas of a company from finance to operations
  - Unless everyone buys into the idea it will fail from the start
- Assess and evaluate the current infrastructure
  - Visualize the infrastructure by areas of functionality
  - Are there areas that will benefit the most?
- Make a Plan
  - Clearly identify what and how you plan on implementing
  - DO NOT START OFF TO BIG!

QUESTIONS?

SOA: Beyond the Hype

Rob Richards
rrichards@ctindustries.net
http://xri.net/=rob.richards