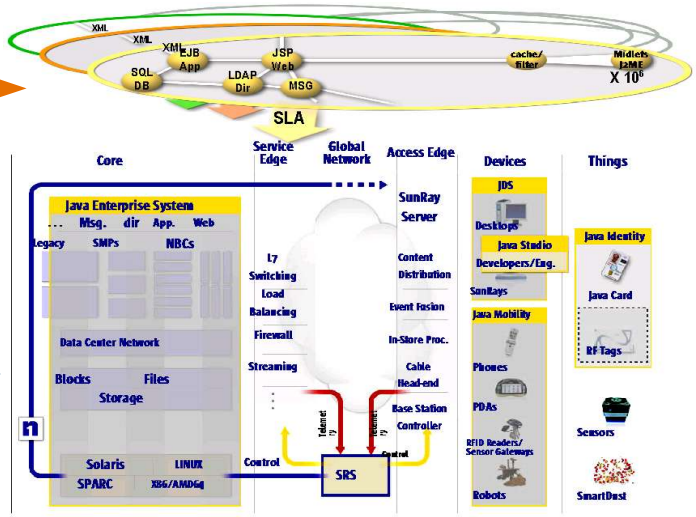
Patterns for SLM

- Refactoring & Integrating Services & Infrastructure

James Baty
Distinguished Engineer, VP & CTO
Sun Microsystems
Global Sales & Client Solutions
September 28, 2005

Shifting to the Network Enterprise

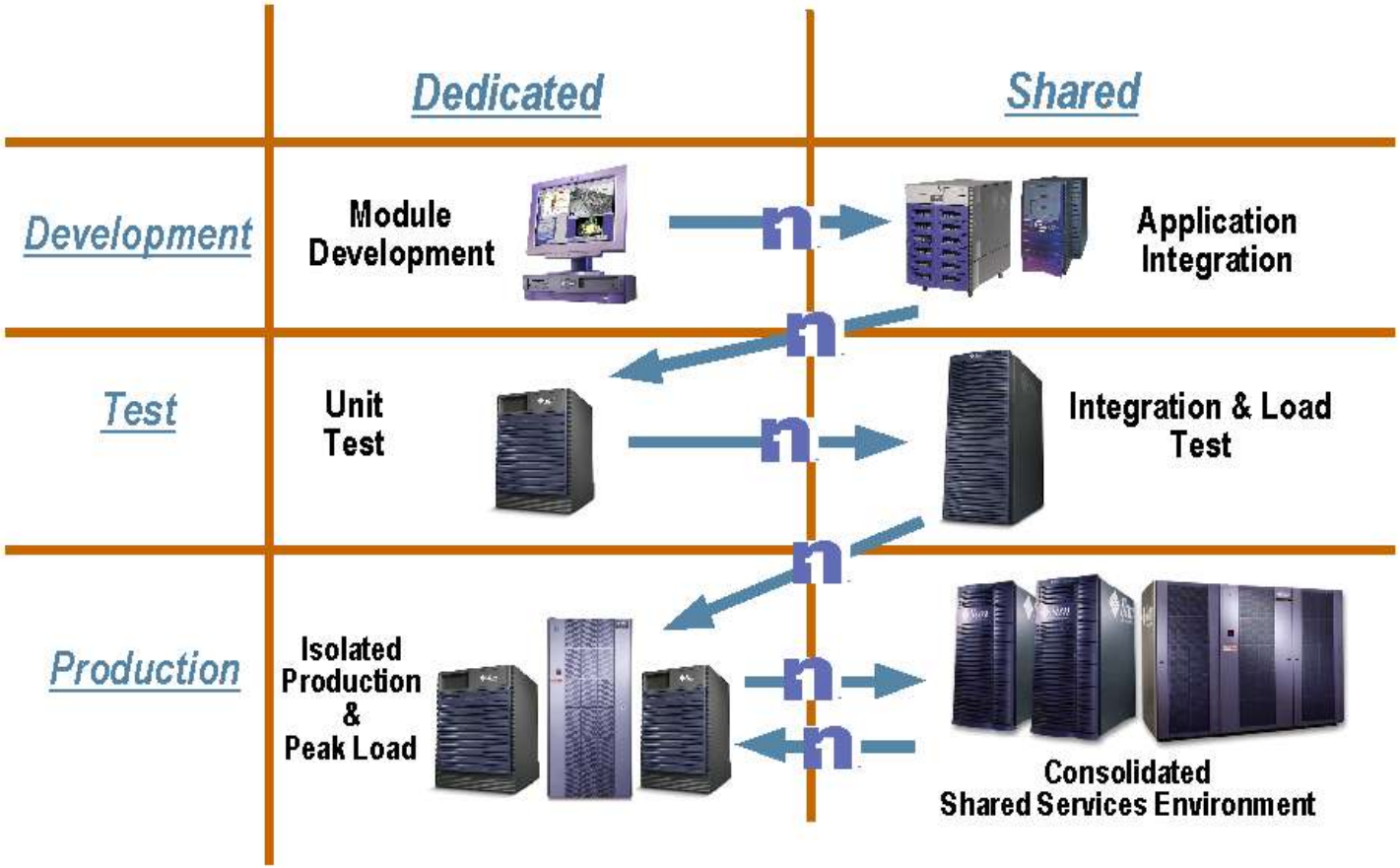
- Applications are evolving toward assembly (via coordination / orchestration) of highly distributed services
- The 'Data Center' is assembled from heterogeneous compute & storage resources and the interconnecting network



- Predictable QoS is never more important & more complex
- Achieving SLM requires real time architectural synchronization between the distributed fabrics of Services and Systems
- Network virtualization is key, patterns are the methodology, but the knowledge engineering is challenging

Goal – enable dynamic application lifecycle mobility over virtual platform

To manage TTM, Response, Throughput, Availability ...



Patterns (SW & HW) for Service levels

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.
(M. Fowler)

Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others. (A. Singh)

We virtualize IT for business goals . We use patterns:

- to refactor functionality, technology & processes.
- to continuously/dynamically optimize applications & resources for Service Levels

Design Patterns are recurring solutions to software design problems you find again and again in real-world application development. (GOF)

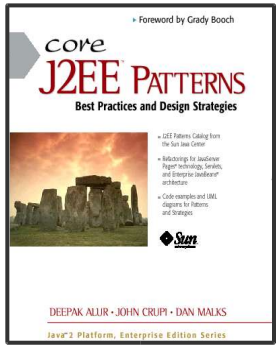
Optimization is the procedure or procedures used to make a system or design as effective or functional as possible. (Lexico)

Key 'technology' - Design Patterns

Higher level abstraction

Core J2EE Patterns

Basic design elements



Intercepting Filter
Front Controller
Composite View
View Helper
Service to Worker
Dispatcher View
Business Delegate
Service Locator
Session Façade
Value Object
Composite Entity
Value Object Assembler
Value List handler
Data Access Object
Service Activator

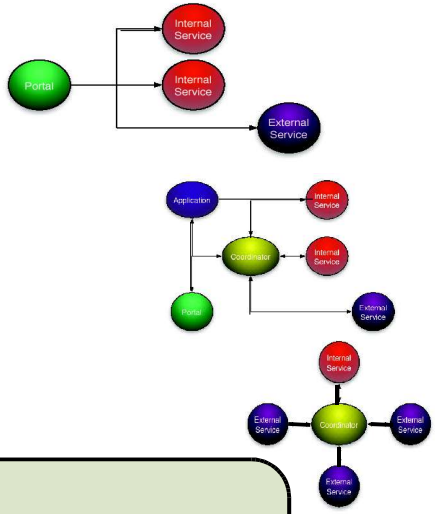
Sun ONE Patterns

fr. Web Services Use Cases

Create Service
Assemble Service
Deploy Service
Register Service
Discover Service
Consume Service
Authenticate Service
Authorize Service
Monitor Service

SOA Patterns

fr. business modeling



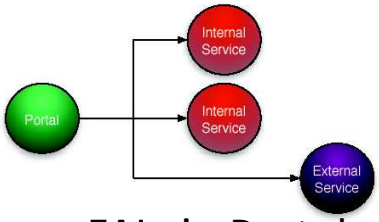
Pattern Definition Template

- ***Name*** *– unique, descriptive name*
- ***Problem*** *– design problem to be solved*
- ***Context*** *– environment of pattern*
- ***Forces*** *– reasons & motivation for selection*
- ***Solution*** *– describe approach*
- ***Strategies*** *– different ways to implement*
- ***Consequences*** *– pros & cons, trade-offs*

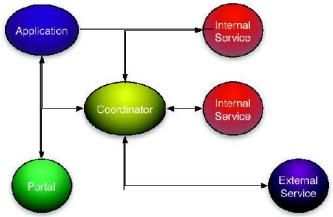
Encapsulating SOA Architecture as Patterns

- Capture common business requirements
- Represent common architectures
- Support modular design

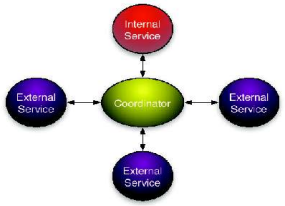
Fundamental Patterns



EAI via Portal

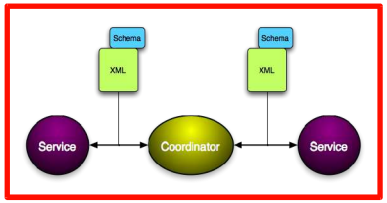


EAI w/ Coordination

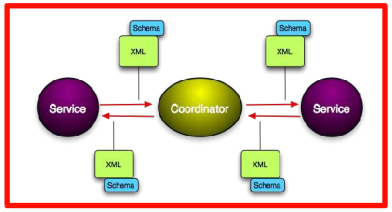


B2B Style

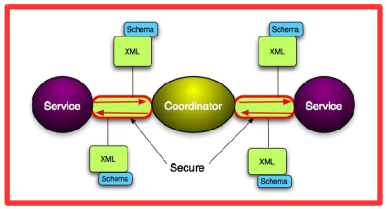
Functional Elaboration



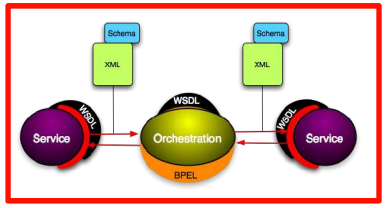
Document based



Mostly Async



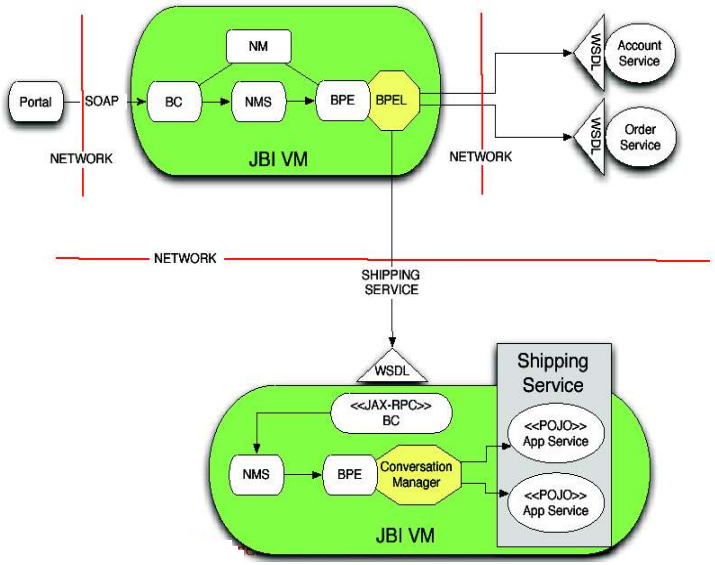
Secure / Identity



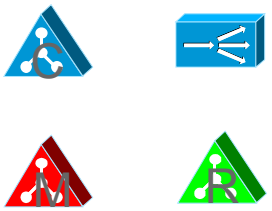
BPEL Orchestrated

Putting it all together

combining patterns /styles / rules to develop integrated design

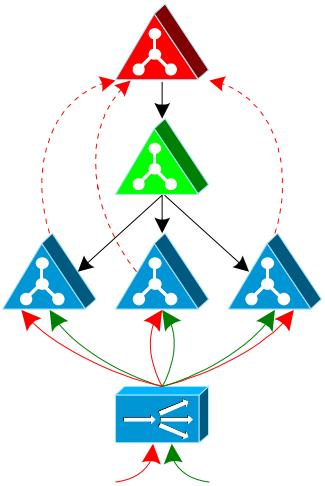


Now Extending Patterns to Infrastructure
- e.g., Directory services architecture



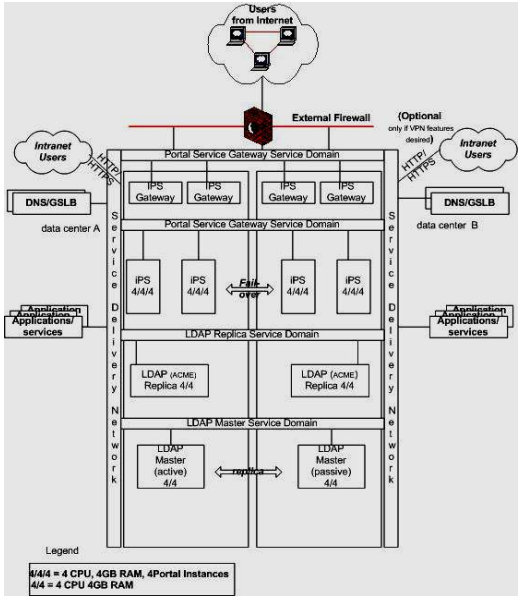
Core Directory Elements

(Master, Replica, Consumer, Load Balancer)



N-Tier Micro-Architecture

Pattern

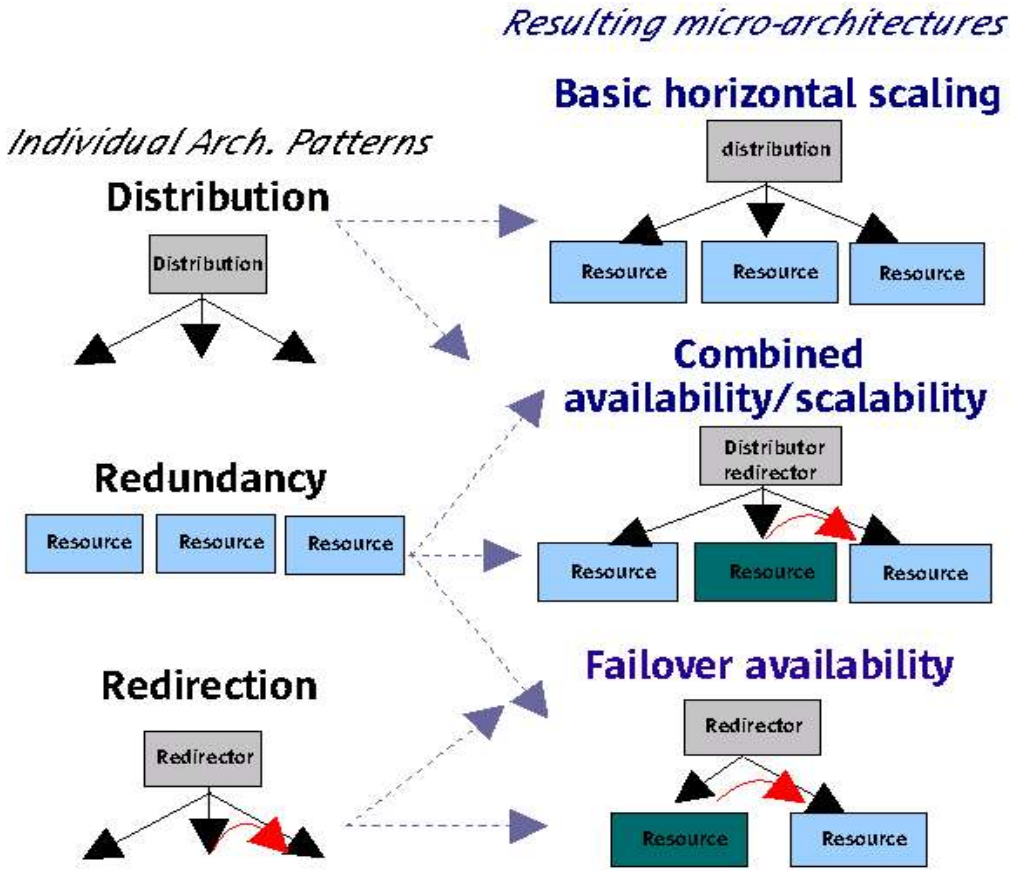


Deployment/Build Pattern

Composition and implementation

Combining tier 0 patterns for SLM

- combine basic design elements to develop component micro-architectures
- increasing Service Level control
- assemble micro-architecture into larger designs
- implement in alternative technologies



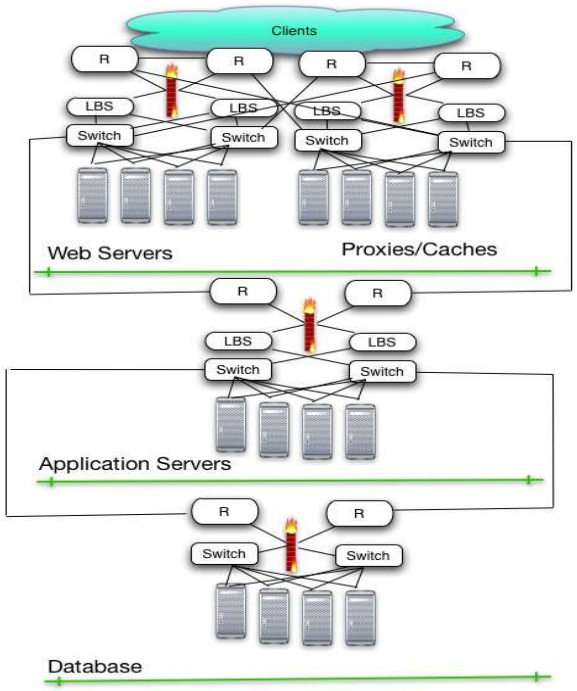
A brief history of "load balancing"

- Ancient history – hacks
  - Lbnamed, RR-DNS
  - Simple load balancing
- Recent past – HW/SW products
  - Resonate, Local Director
  - Complex policies + failover
- Today – Network function
  - Alteon, Arrowpoint
  - Sun Secure Application Switch
  - Add SSL & wireline speed

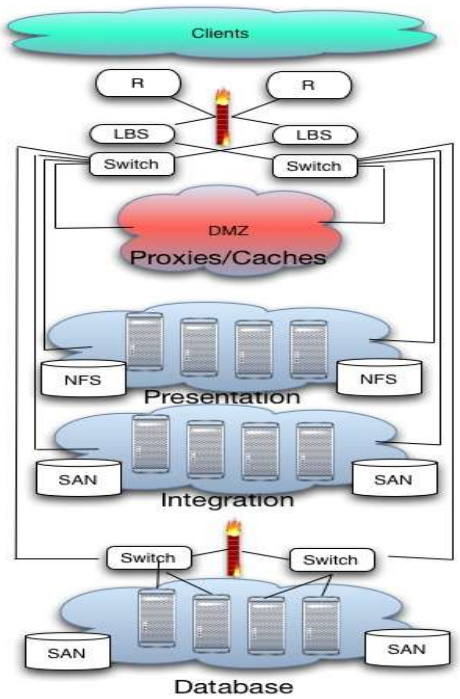
Same pattern – different strategies

Moving from 'static' to virtual architecture enables implementation of a service model

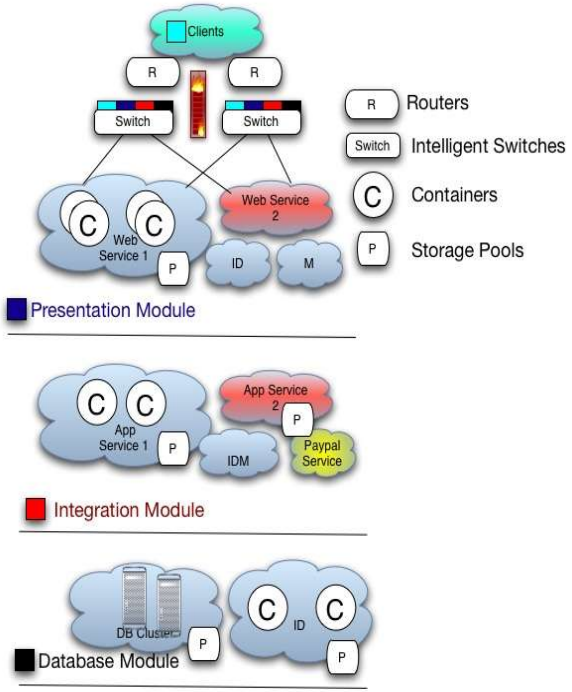
Hard / static wired
bastion hosts
direct storage



VLAN
virtual tiers
SAN



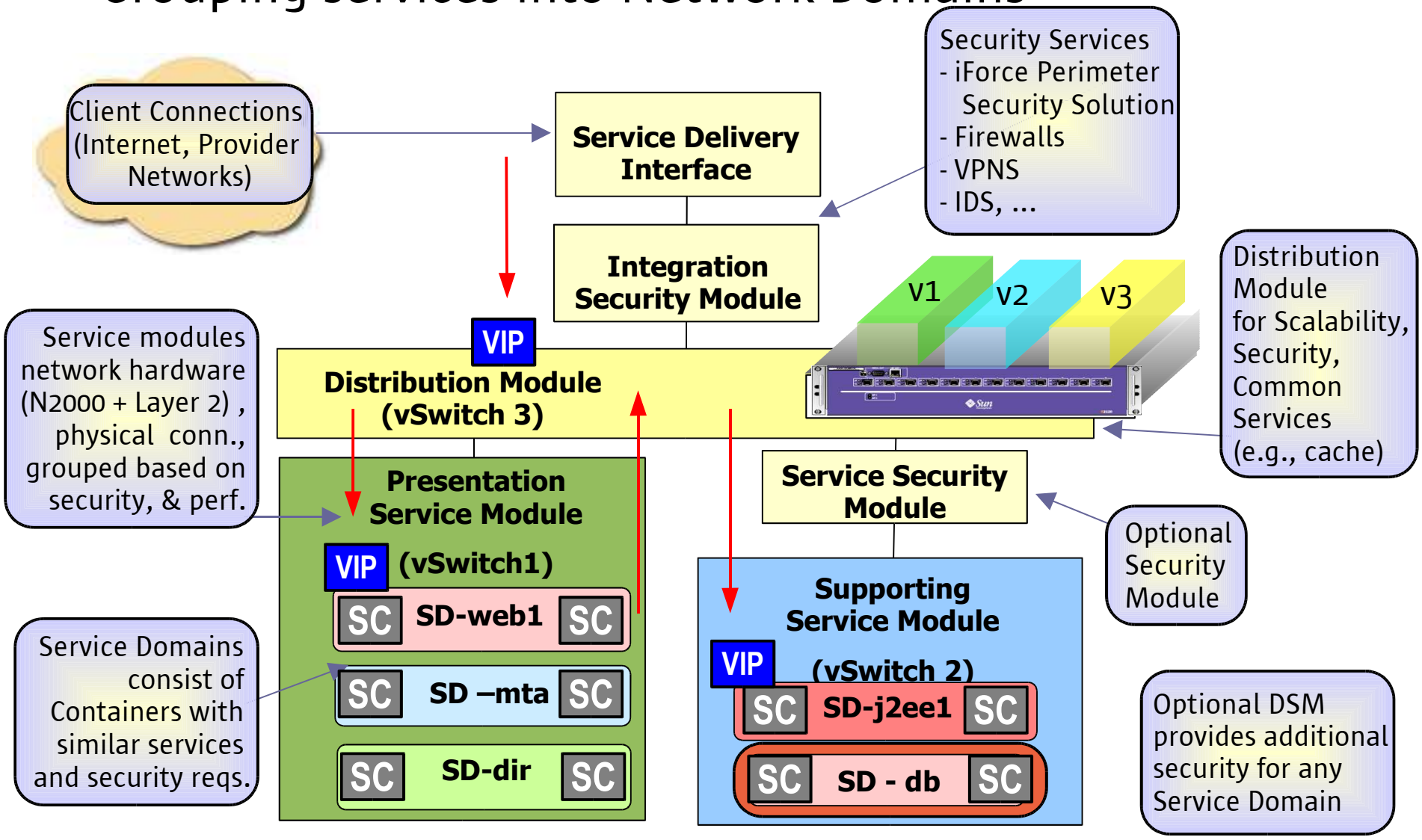
VLANs/Load Balancing
'zones' w/ svc modules
SAN Pools



Increased Virtualization

Deploying Patterns - Service Delivery Network

- Grouping services into Network Domains



3 Issues –
1- heavy lifting - Harvest & refactor Admin Use Cases / Patterns

- Concentrate on "what", not "how"
- Capture requirements in terms of behavior
- Identify clear roles and responsibilities
- Abstract design from implementation

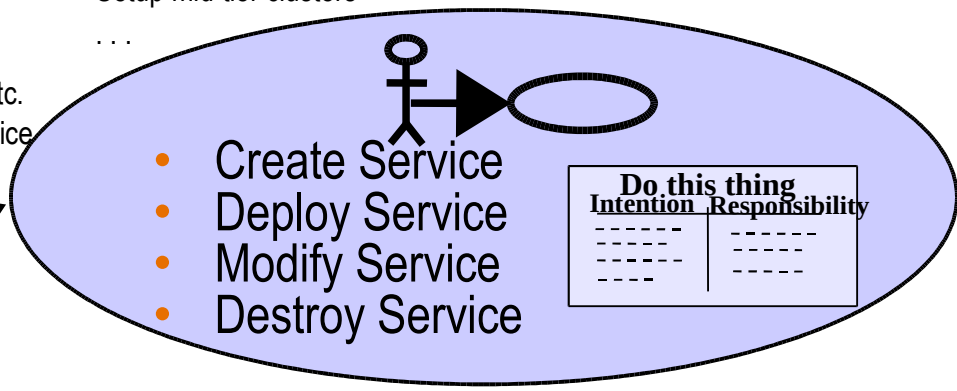
For Each Server

Provision server (physically acquire)
Connect to network — acquire IP address, etc.,
NIS name, DNS name, LDAP name, etc.
Install OS and relevant patches (the latter sometimes takes longer than OS)
Install and configure Volume Management (optional)
Install and configure 3rd party file systems (optional)
Install clustering software (optional)
Install management framework probes/agents, etc.
Install Application software (traditionally to local storage)
Tune O/S for software (rare these days except for DB)
Configure application software part 1 — bind to the O/S, use IP addresses, etc.
Configure application software part 2 — give it an identity in terms of the service (database name, etc.)
Start the application
. . .

Move: from this to this

For Each Service

Determine and set up range of IP addresses
Physically or logically setup LAN
Physically and logically layout storage
Setup firewalls
Setup load balancing clusters
Setup HA clusters
Setup mid-tier clusters
. . .

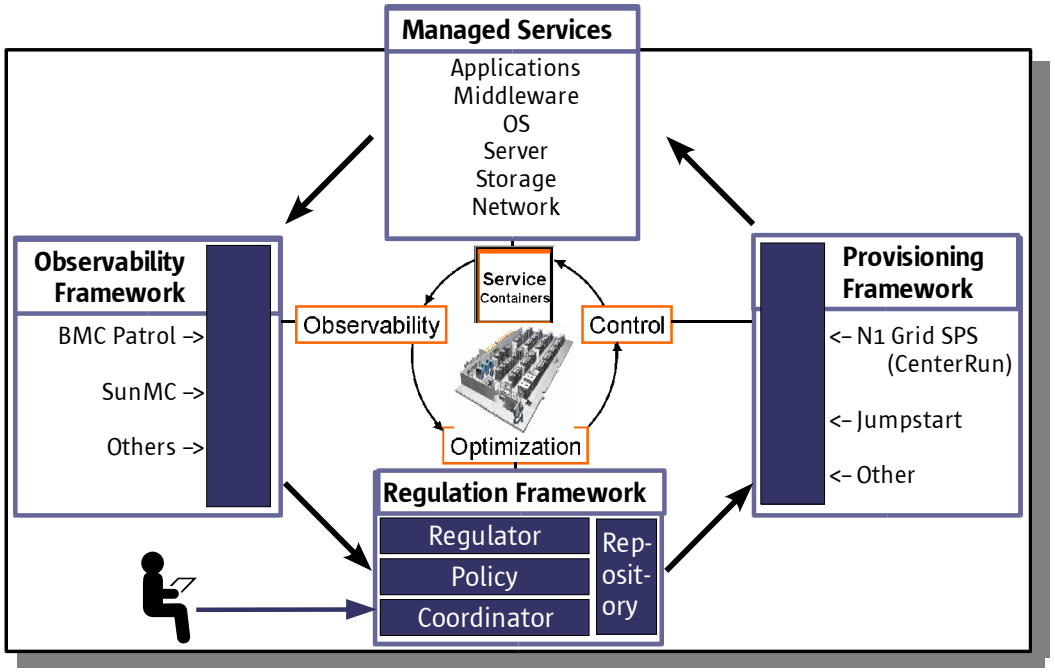


3 Issues

2 – Grand Challenge – Integrating Service Configuration & Deployment

Service Containers
– The Target -- That which is being controlled, observed and optimized.

Observability
-- Visibility of changes in the environment.



Control
- Resource Allocation (Static/Dynamic Deployment).

Optimization
– Regulation/Governance
-- automated decision making
-- service level arbitration

3 Issues

3 – Conceptual Barrier - Identifying "Context & Forces"

Context – environment of pattern
Forces – reasons & motivation for selection