Paremus

Why has enterprise SOA delivered so little?

- The IT industry has over hyped WS-*
- SOA - is not just about Web Services!
- An WS-* based “SOA strategy” de-couples applications, but...
  - leaves these same old *monolithic* business and infrastructure services in place
  - using the same old approaches to application availability and manageability
  - And so business systems remain as ***change resistant*** as ever.
- WS-* is not the Silver Bullet, but rather only a part of the final solution.

Fundamentals

What do we actually want?

- Business Agility
- Service Availability
- Reduced Operation Cost

As the Operations Manager...

- “Lights Out” or “Black Box” Data Centre...
  - Automatic service recovery from infrastructure failures.
  - Massive *dynamic* scalability.
  - Rapid deployment / rollback of all business applications.
  - Embedded Audit, Security and Configuration Management
- The ability to run across commodity hardware
- A software stack that doesn’t cost more than the commodity hardware!

Adding up to a significant reduction in operational risk and large OPEX savings

As the Development Manager...

- Ability to rapid assemble all business applications from re-usable standards-based service components - ***just like LEGO***!
- A runtime that isn't a **Hammer**; that doesn't treat all business applications as **Nails**; instead, one that optimally adapts to each application's runtime requirements.
- An approached based on Industry Standards - ensuring longevity and avoiding vendor / architectural lock-in.

Service Oriented Infrastructure

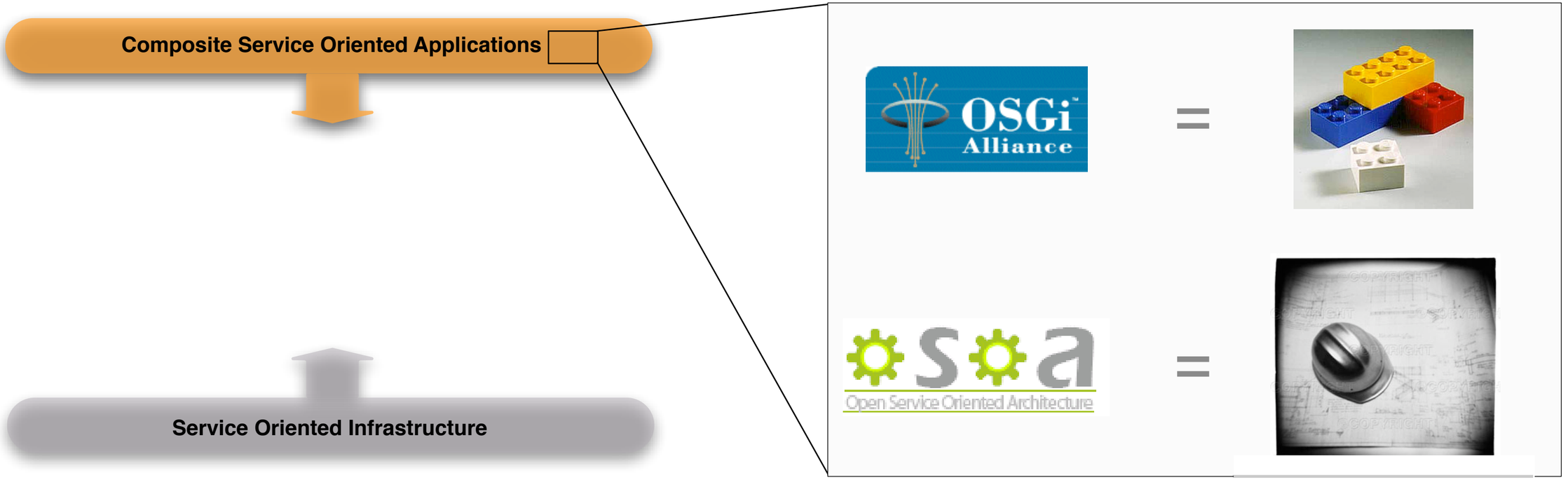
Hardware: Network, Storage, Compute

OS - Native: Linux, Solaris, Windows, OSX,

Virtualization - VLan, VStorage, VMware, Xen

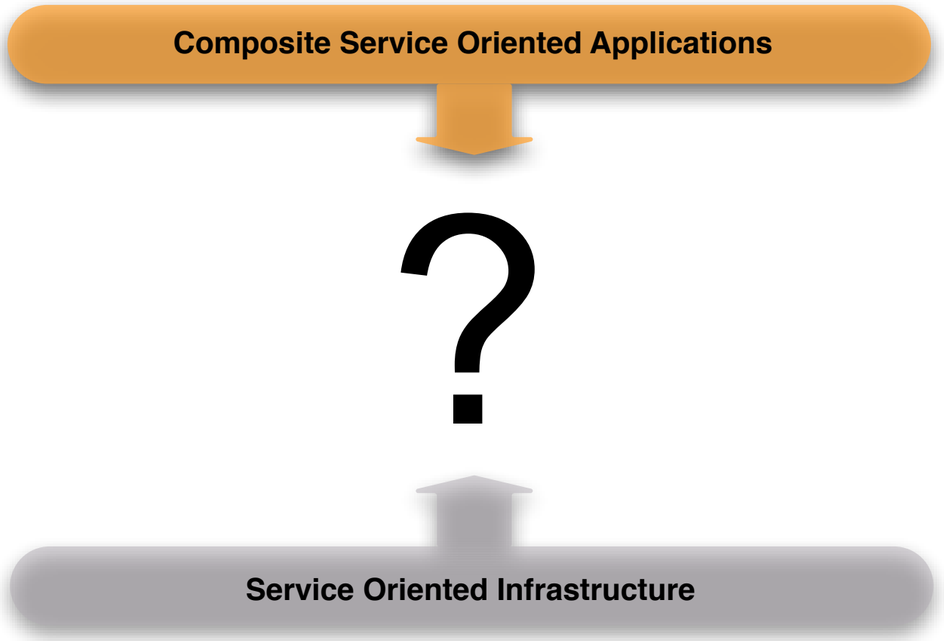
SOI management - (OS & Software deployment / configuration) - HP-OpenView, IBM Tivoli, Sun, Opsware, Cassatt, 3Tera, EverGrid, Enigmatec, Veritas, DataSynapse...

Dynamic / Composite SOA



OSGi - The most important standard of the Decade! (SD Times 1st June 2007)

Question



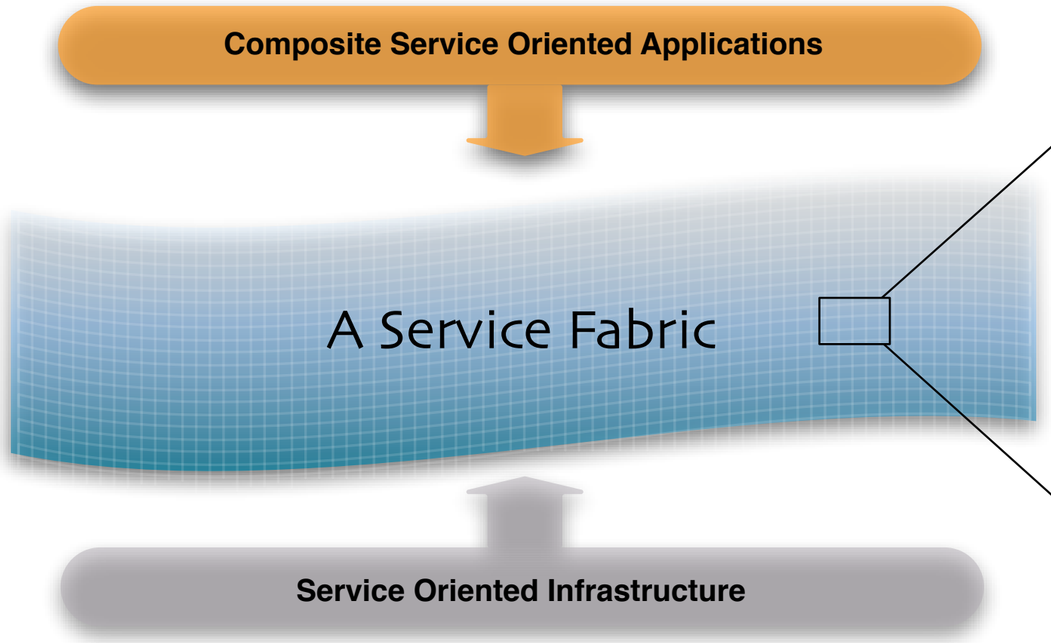
How do we map

- dynamic / composite service oriented applications

to

- anonymous & potentially ***volatile*** runtimes

The Answer

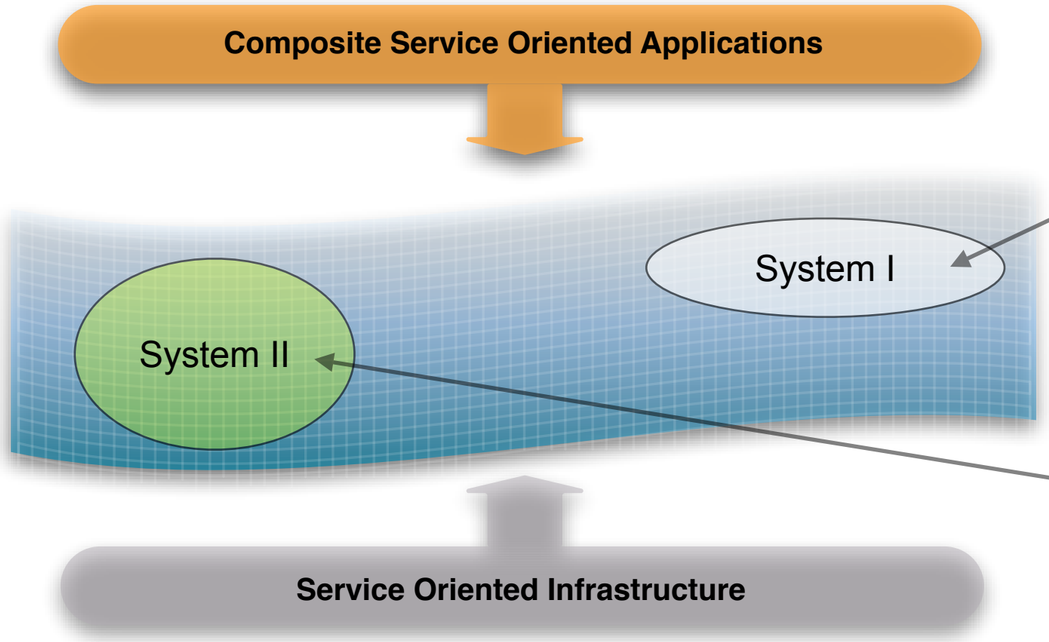


- Dynamically Assembles and Maintains composite applications
- Insulates these applications, and itself (!) from resource failures
- SELF-* (healing, managing, protecting, scaling)
- Adapts to application requirements - Low Latency Messaging, High Computational Throughput, Transactional Grid, ESB & CEP

A Service Fabric != Data/Compute Grid, or an ESB, or a WS-SOA

A Service Fabric >> ∑(Data/Compute Grid+ ESB + WS-SOA)

The Service Fabric



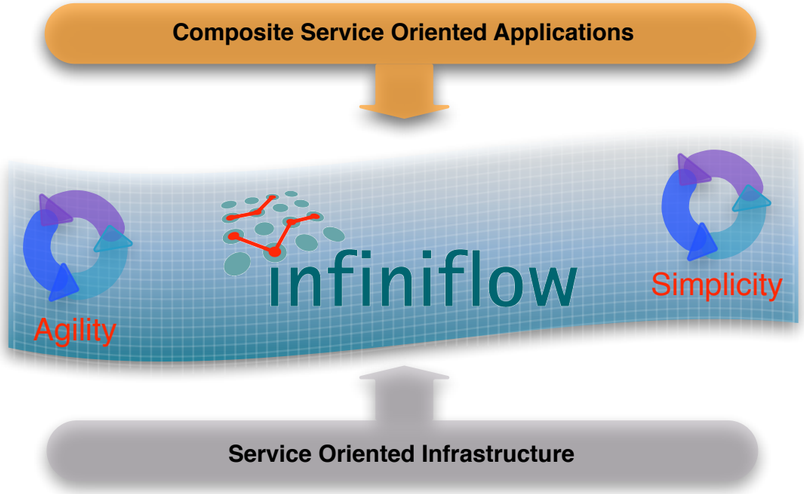
A Composite Spring based Web Service

A Java POJO based Grid Service

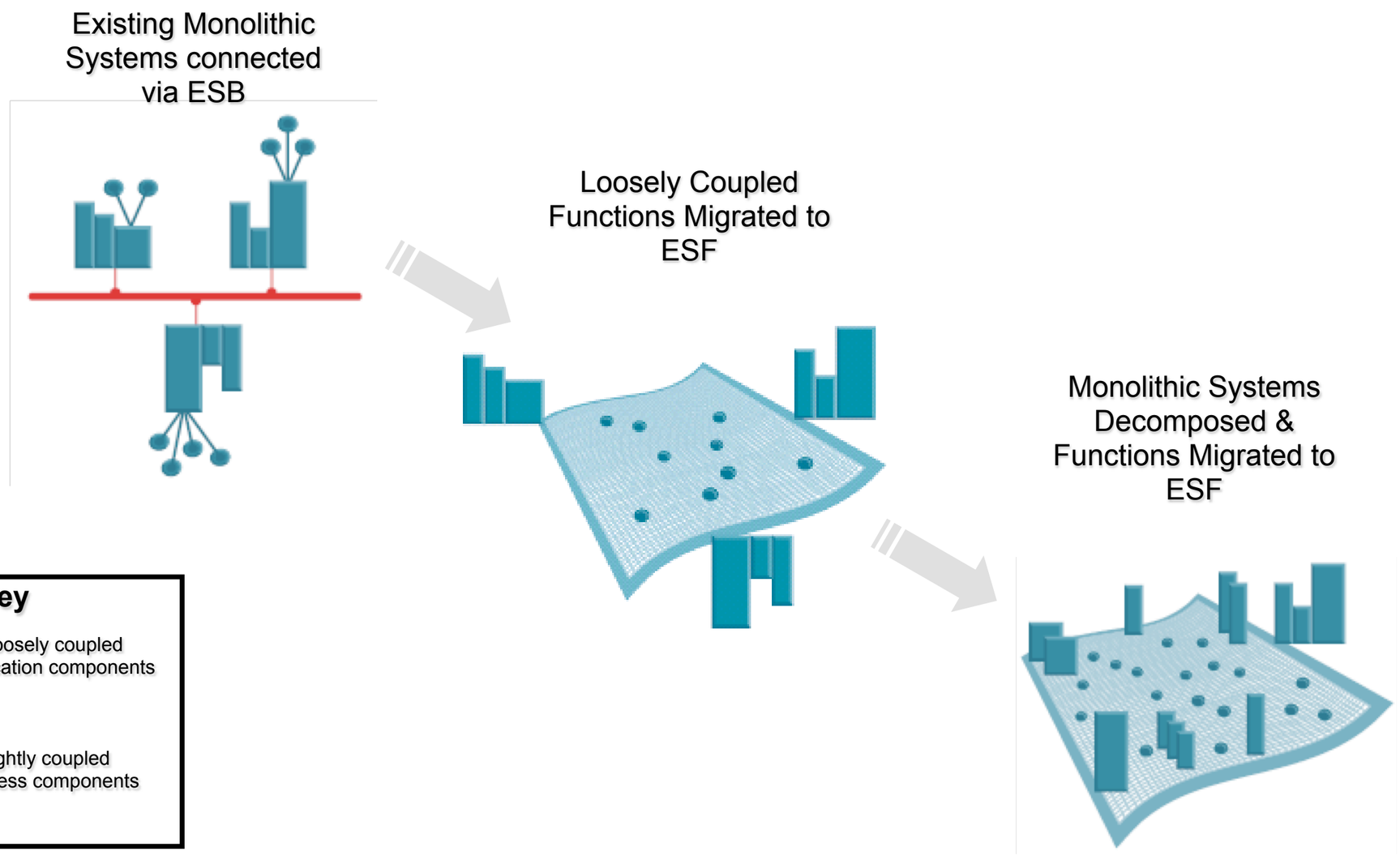
A Service Fabric simultaneous supports many applications, each with its own unique service requirements

Infiniflow - Enterprise/Utility Service Fabric

- Massively Scalable
  - ~10,000 nodes per fabric instance
- Autonomic operation
  - Self-* (Managing, Auditing, Provisioning, Healing, Scaling)
  - Dynamically Provisions applications across 100's of anonymous nodes in Seconds.
- Virtual Resource Market
  - Dynamically maps most *cost effective* physical resource to user/ service requirements
- Self-Similar - Model Driven / Modular Design
  - All runtime systems are dynamically assembled from SCA descriptions
  - All runtime services are dynamically assembled from OSGi service components



But How Do We Get There from Here?



Infiniflow Market Acceptance

- Implementations to date include:
  - Defense
  - Finance
  - Business / Web 2.0
  - ISV OEM's
- Related Open Source Project 

Thank You

Richard Nicholson (Paremus CEO)

see http://www.paremus.com/downloads/downloads.html