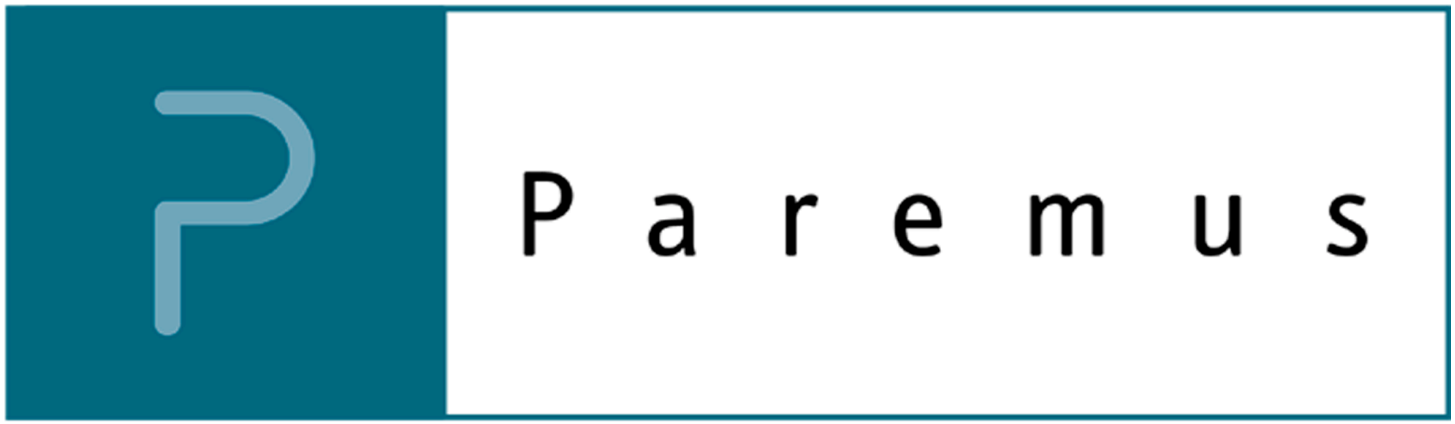



Intelligent Information Engineering

David Savage
Paremus Software Engineer

*10th Jini Community Meeting*
*14th September 2006*

# Towards SCA – The Newton Approach

- Acronym city, a brief guided tour
- The Newton Component Framework
  - Concepts
  - Core Components
- Building a Service Orientated Architecture using Newton
- Demo
- Questions

# Acronym City – A Brief Guided Tour

## OSGi - Open Services Gateway Initiative

- Not OGSI!
- IBM, BEA, Oracle, Sun, Nokia, Motorola
- Improved class loading model
- Bundle life-cycles
- Enterprise OSGi - Adobe Systems, BEA Systems, CA, Cognos, Compuware, Exadel, Google, IBM, Intel, IONA Technologies, Oracle, Paremus, SAP, SAS Institute, Siemens, Sybase, Interface-21, and VeriSign
- http://www.osgi.org

# Acronym City – A Brief Guided Tour

## SCA - Service Component Architecture

- BEA, IBM, Iona, Oracle, SAP, RedHat, Sybase, Tibco, Interface-21
- Extensible set of specifications to define SOA systems
- Current specs provide WS*, Java, C++, BPEL bindings.
- http://www.osoa.org

# Towards SCA – The Newton Approach

- Acronym city, a brief guided tour
- The Newton Component Framework
  - Concepts
  - Core Components
- Building a Service Orientated Architecture using Newton
- Demo
- Questions

# Newton Introduction


- A distributed component framework
- An open source project set up by Paremus, hosted on CodeCauldron.org
- Combines Jini, OSGi and SCA
- Why this combination?

# Jini

- Discovery, Leasing, Transactions, Security

# OSGi

- Peer classloading, Bundle-lifecycles, security

# SCA

- Extensible, applicable, XML as static structure, not implicitly as a transport mechanism

# Newton – Features

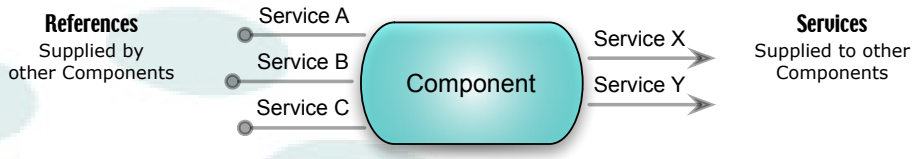
- POJO / IoC
- Dynamic Install
- Garbage Collection
- Log Management
- ROC

# Newton Bundle & Component Model

## OSGi Bundles

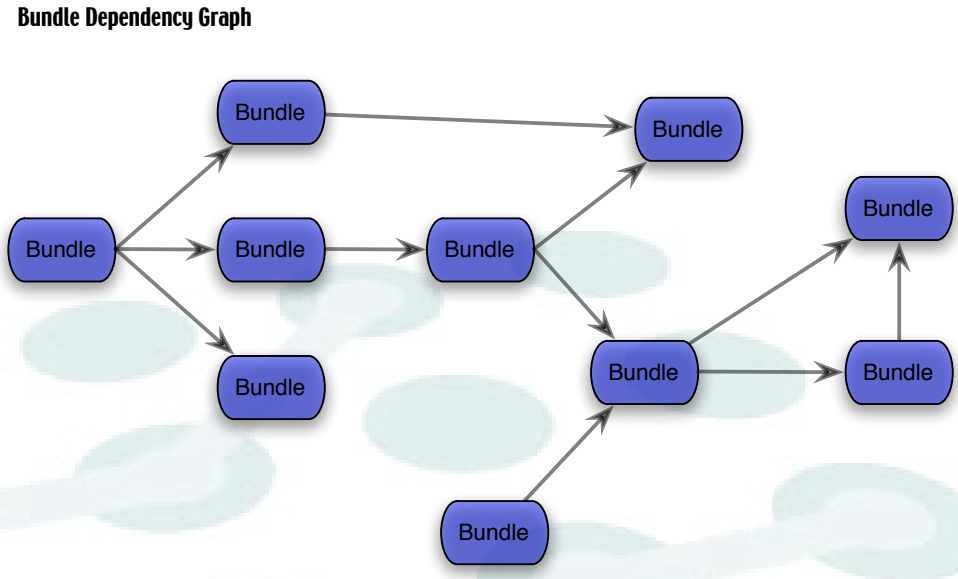

- An OSGi bundle provides:
  - Unit of deployment
  - Well defined code and service lifecycle
  - Peer classloading model for sharing code
  - Registry for sharing services
- Metadata
  - Embedded in bundle manifest
  - Includes Inter-bundle code dependencies
  - Includes Bundle location - e.g. for installs
  - Exportable to OBR format

## Newton Composites - Analogous to OSGi Bundles




- Services annotated by *name* = *value* attributes
- Reference filtered using LDAP expressions over attributes
- Reference cardinality constraints
- Services and references are dynamic
  - They can come and go at runtime
  - Attributes and filters can change at runtime

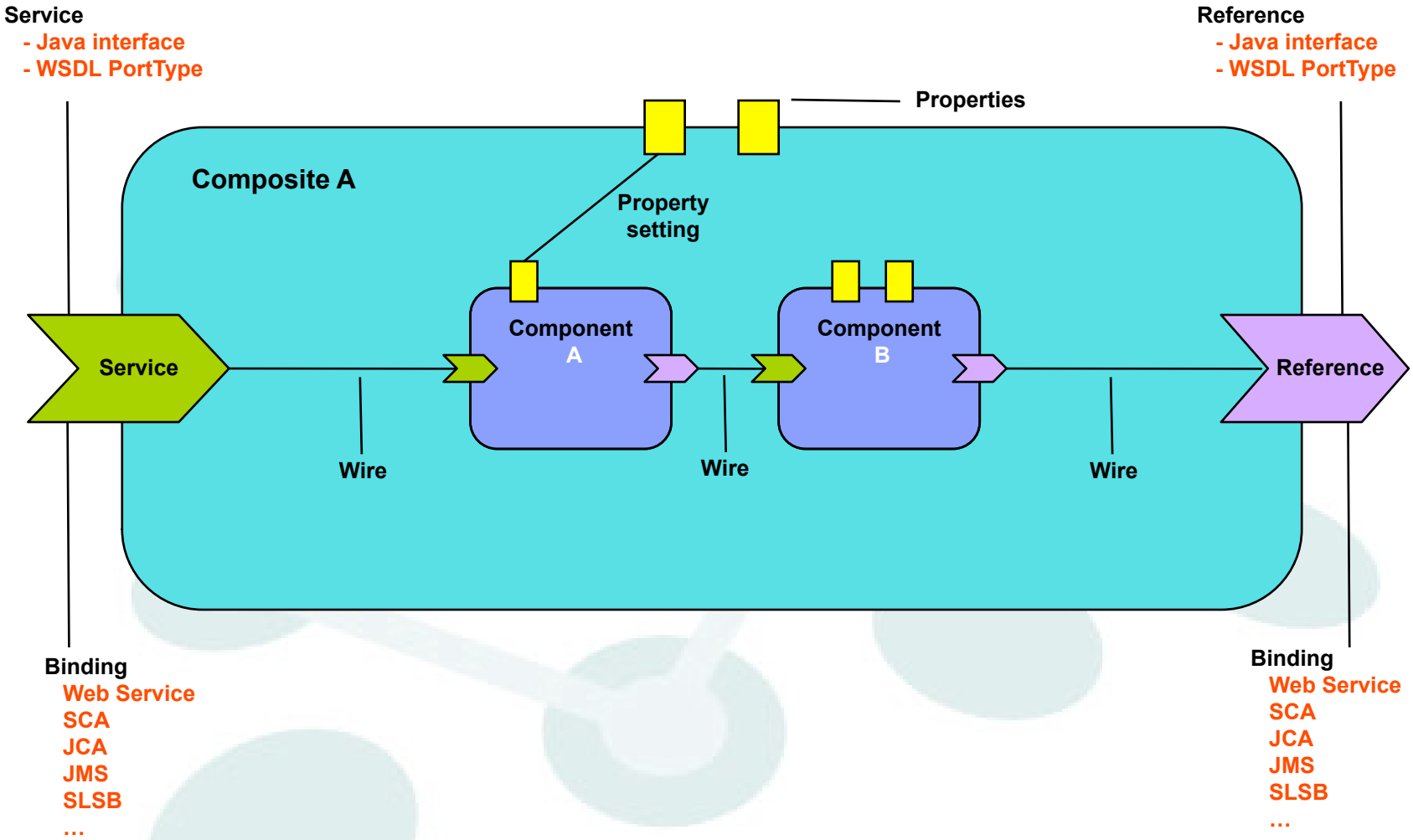
# Newton Binding Services




**Newton Binding Service -** (3 Levels one consistent model).

- Internal
  - Private to each factory bundle - not published
  - IoC / DI container style assembly of a static graph of POJO's
- OSGi
  - Component graph built dynamically out of services which are published at the OSGi registry level.
  - Service bindings *may* ***come*** *and* ***go*** over the lifetime of a component. Garbage collection behaviors.
- Jini
  - Dynamically maintained graph built out of services published in the Jini Registry
  - Handles ***unexpected service disconnect*** *or* ***failure***
  - Hides Jini plumbing code

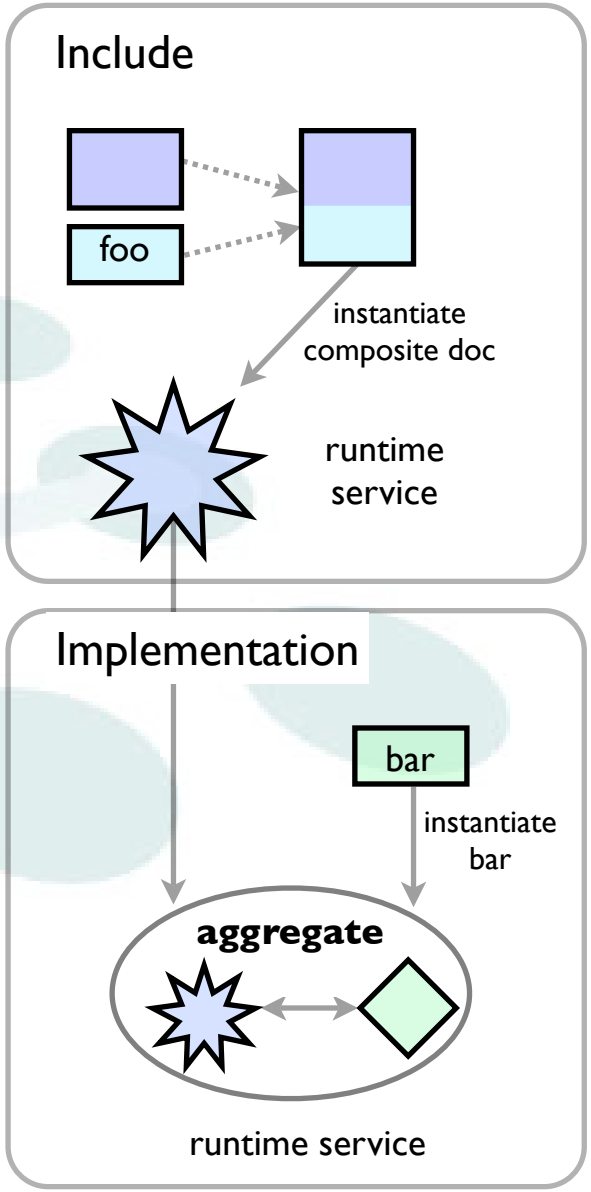
# A SCA Composite



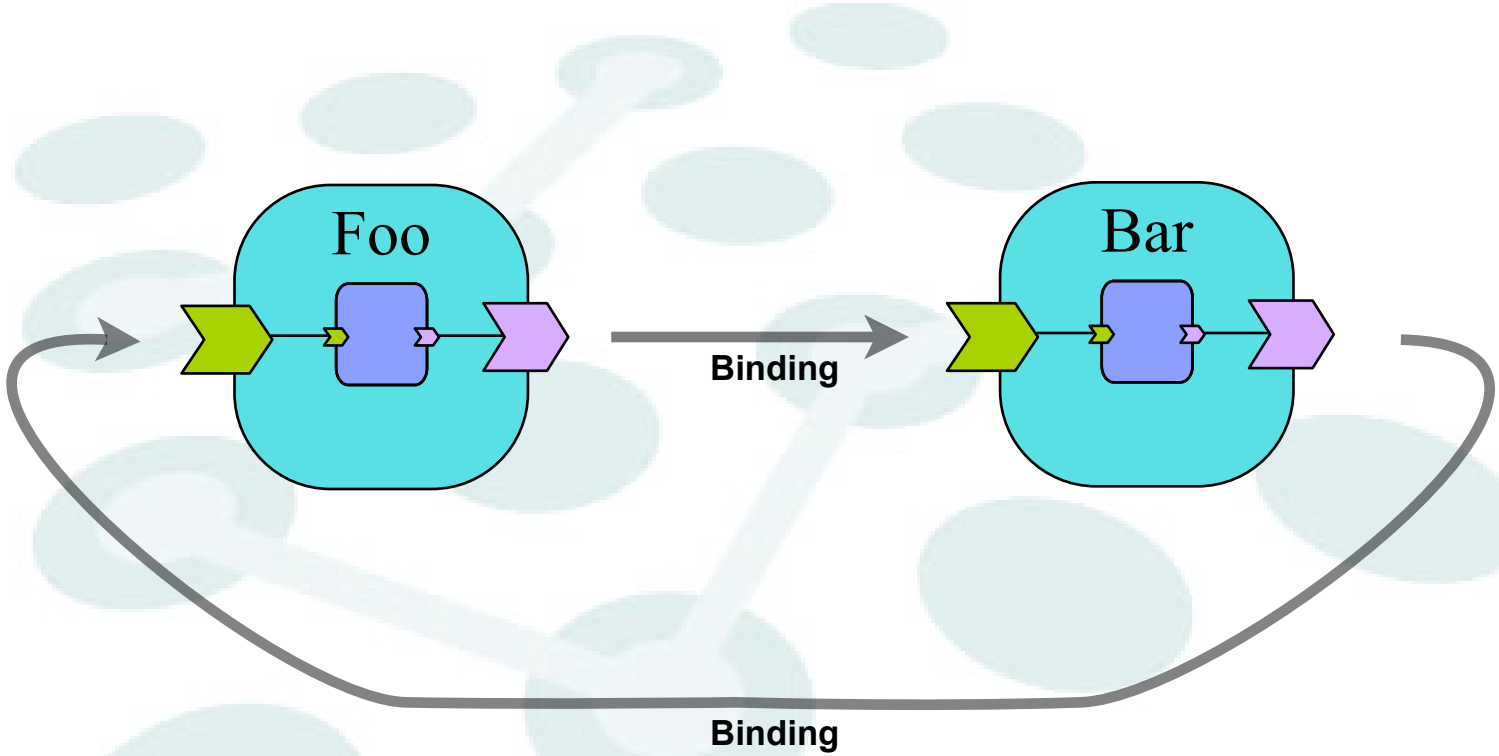

Paremus provide a set of bindings for jini and osgi as part of the Newton Framework

# Composite Building Blocks

```
<composite name="aggregate">
    <include name="foo">
    <component name="bar">
        <implementation.composite name="bar" />
    </component>
</composite>
```



# Binding Composites



# Example Composite

```
package com.example;

public interface FooService {
    void callBar();
    void call();
}

package com.example;

public interface BarService {
    void callFoo();
    void call();
}
```

# Example Composite

foo.call(); // results in sys out “foo”

foo.callBar(); // results in sys out “bar”

# Example Composite

```
package com.example;

public class FooImpl implements FooService {

    private BarService bar;

    public void addBar(BarService bar) {
        this.bar = bar;
    }

    public void removeBar(BarService bar) {
        this.bar = null;
    }

    public void callBar() {
        bar.call();
    }

    public void call() {
        System.out.println( “foo” );
    }
}
```

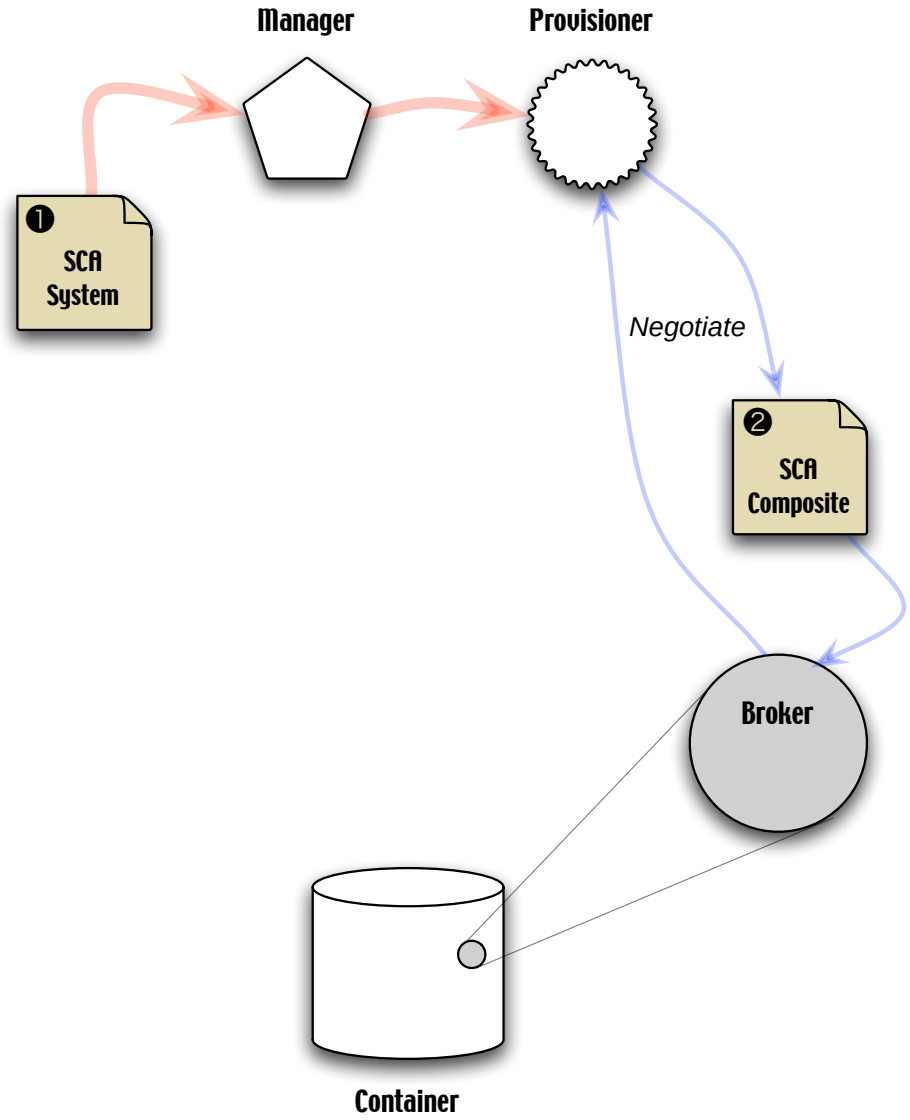
# Example Composite

```
<composite name="foo.composite">
    <reference name="bar.service">
        <interface.java interface="com.example.BarService" />
        <binding.jini />
    </reference>

    <service name="foo.service">
        <interface.java interface="com.example.FooService" />
        <binding.jini />
    </service>

    <component name="foo.impl">
        <reference name="bar"/>
        <implementation.java implementation="com.example.FooImpl" />
    </component>
    <wire>
        <source.uri>bar.service</source.uri>
        </target.uri>foo.impl/bar</target.uri>
    </wire>

    <wire>
        <source.uri>foo.impl</source.uri>
        <target.uri>foo.service</target.uri>
    </wire>
</composite>
```

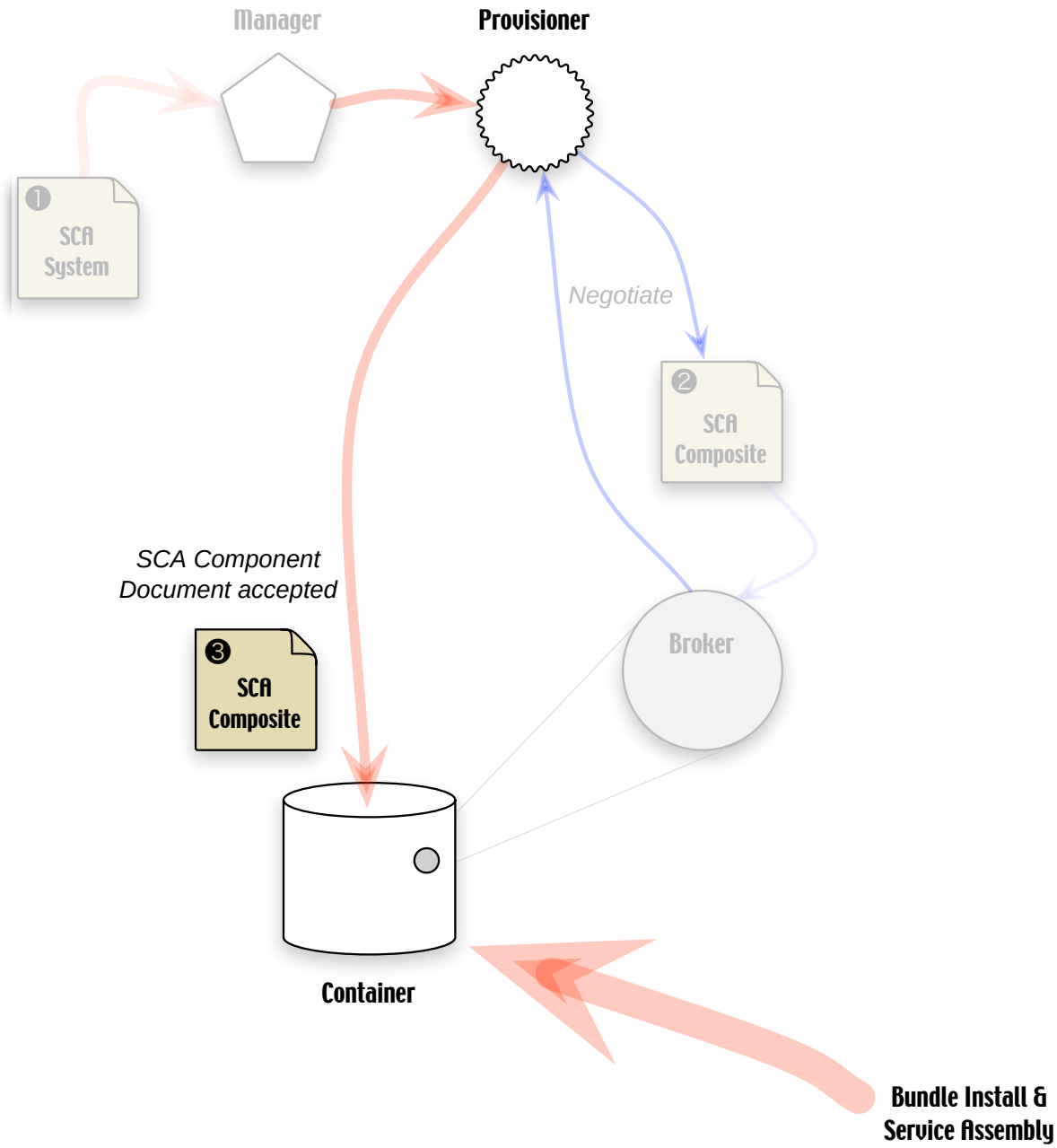
# Towards SCA – The Newton Approach

- Acronym city, a brief guided tour
- The Newton Component Framework
  - Concepts
  - Core Components
- Building a Service Orientated Architecture using Newton
- Demo
- Questions

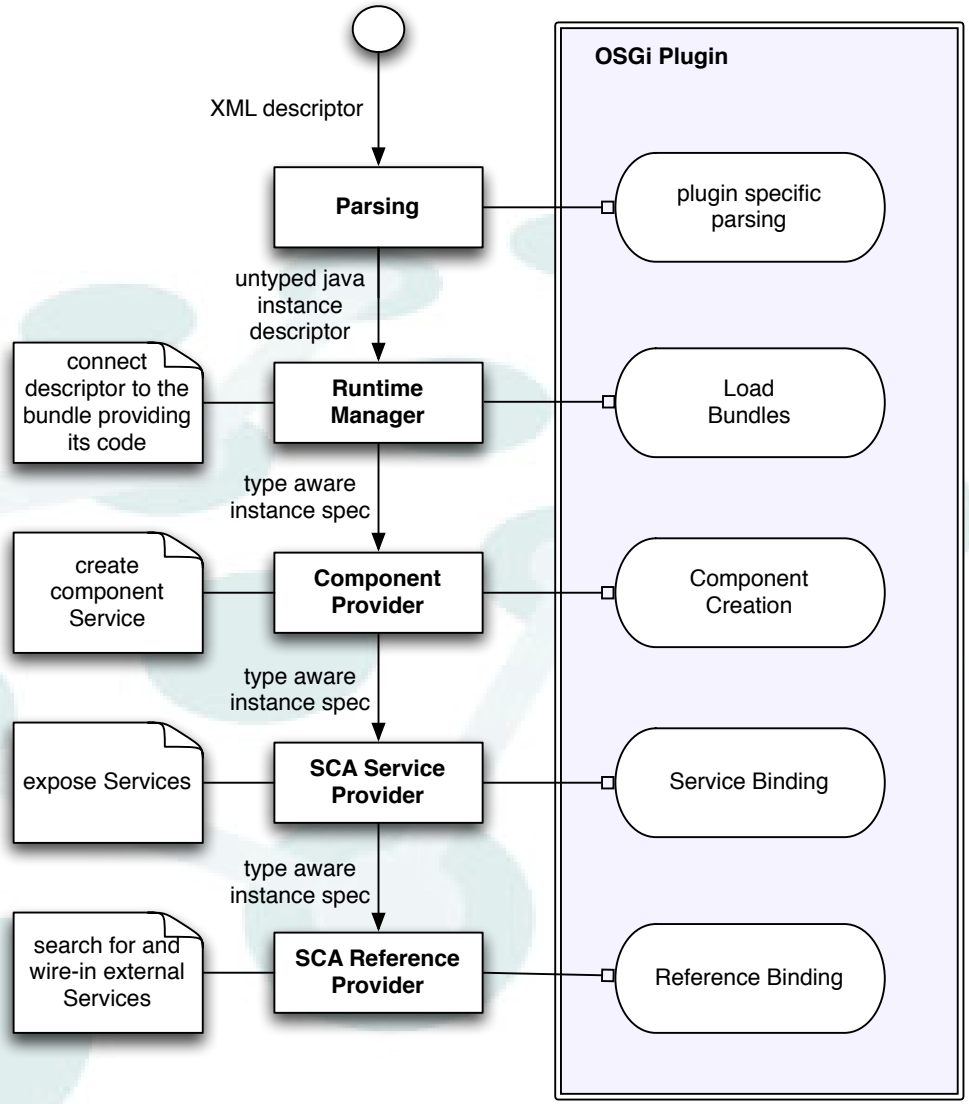
# Deployment Process



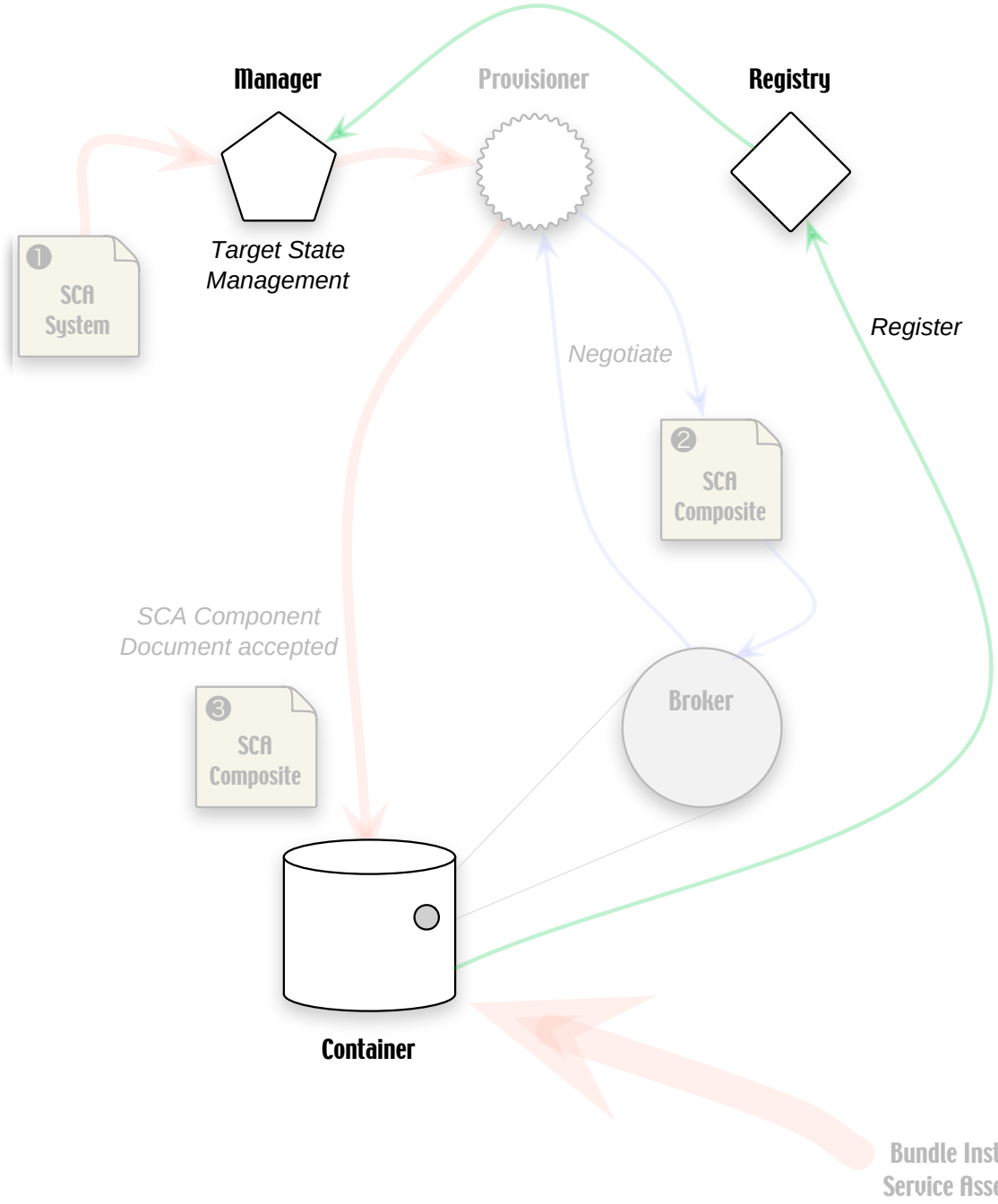
# Deployment Process



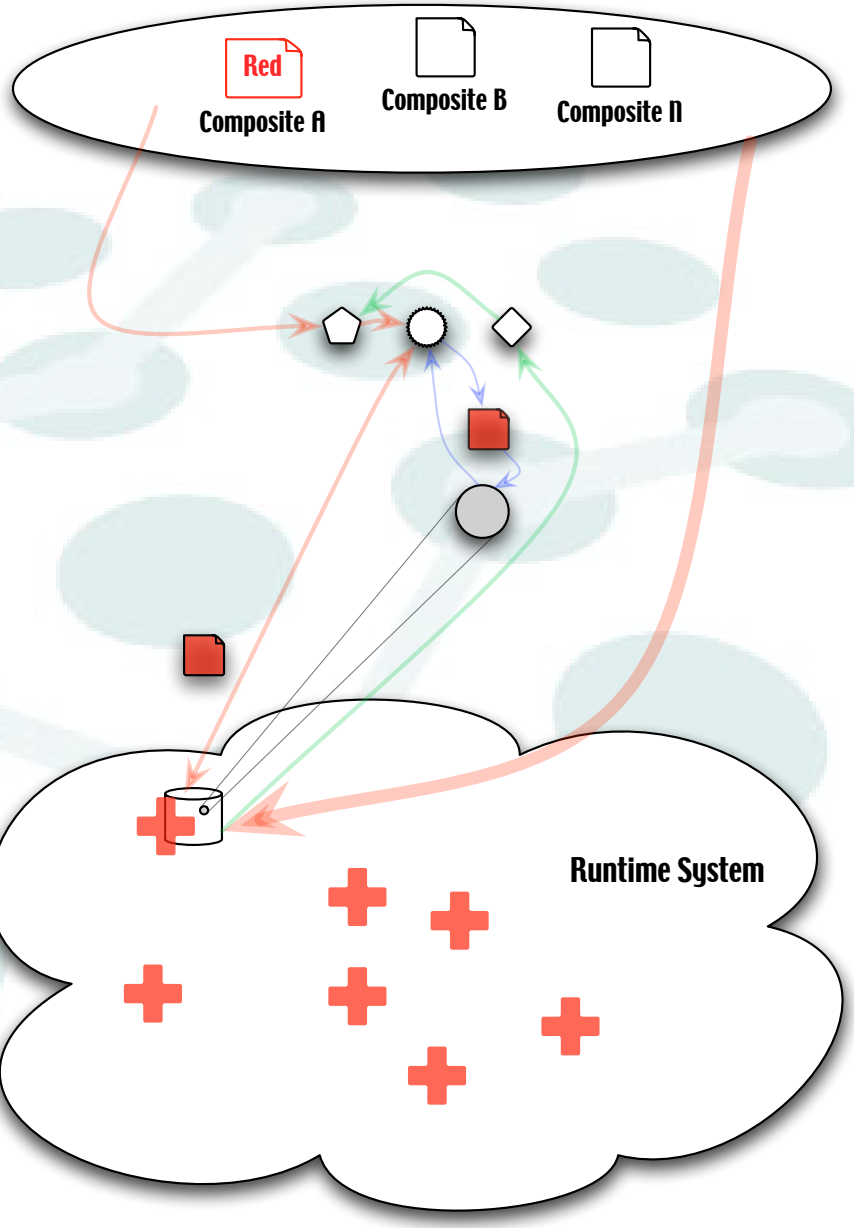
# Installation Pipeline



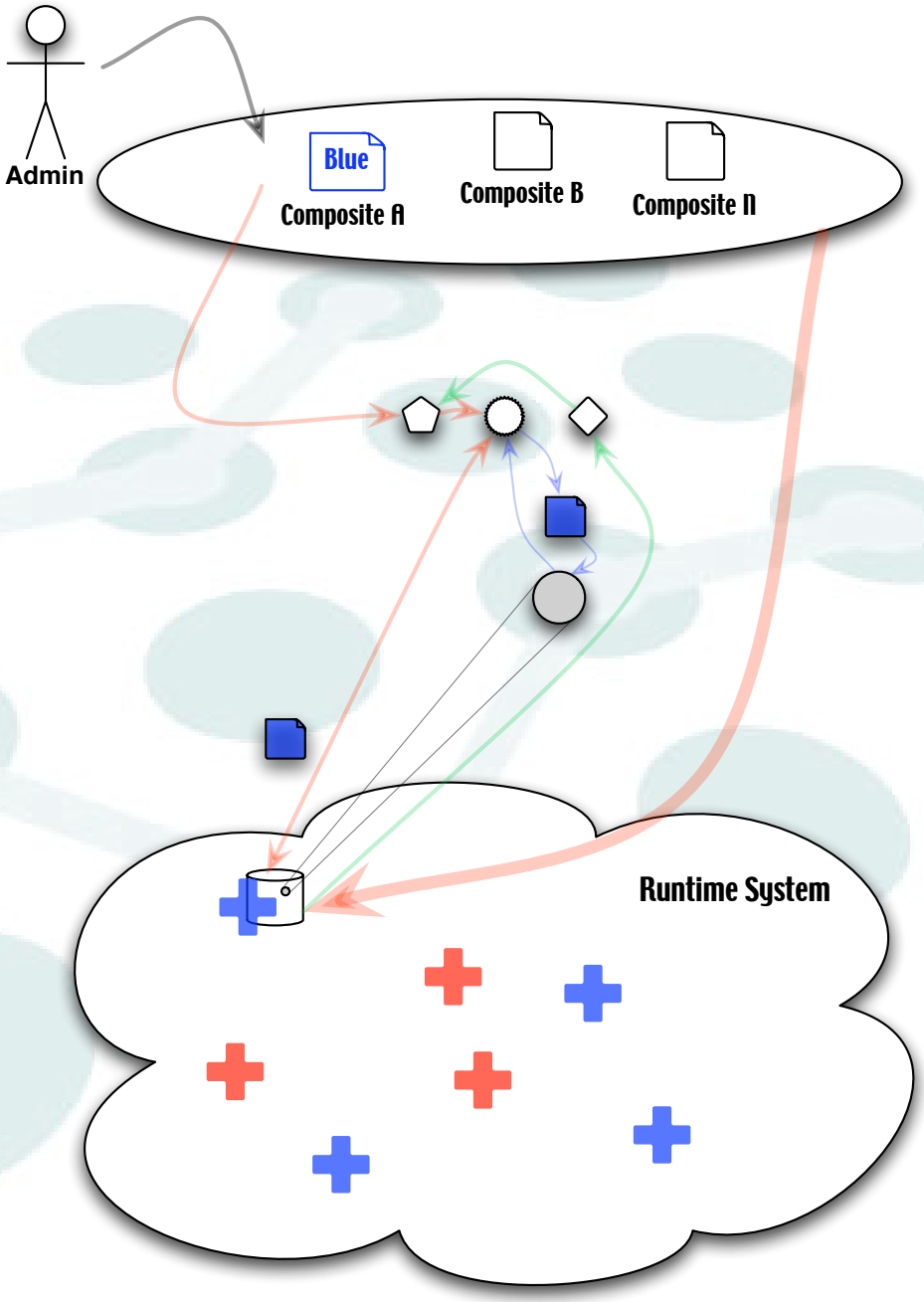
# Deployment Process



# System Management



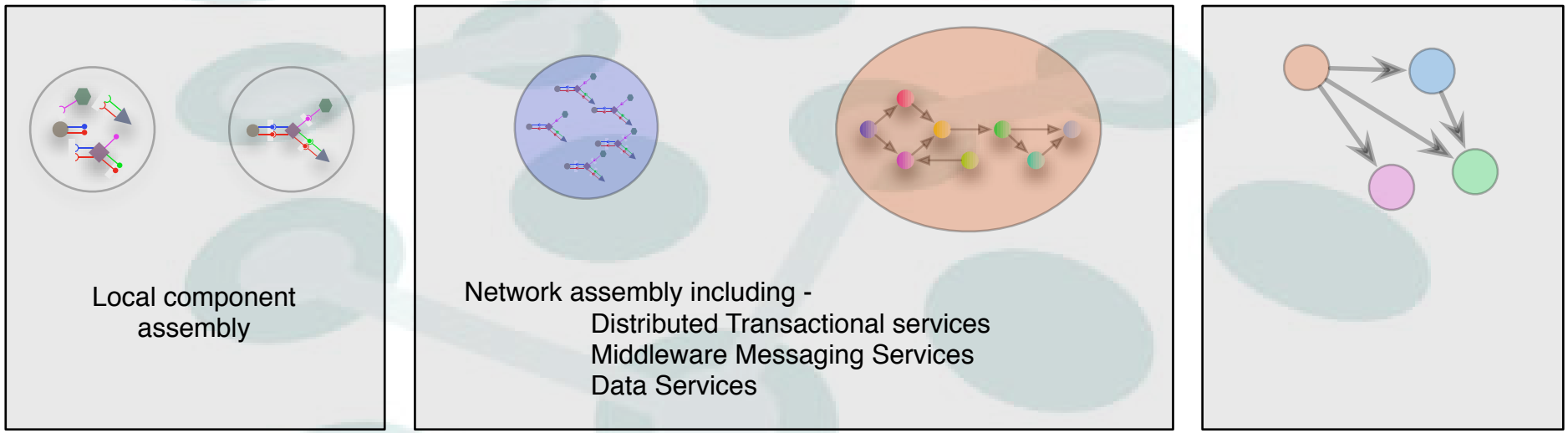
# System Management



# Newton Concepts

Emergent Assembly, Monitoring and Management Hierarchy

Declarative SCA




Physical Machine or Virtual Container / Partition.

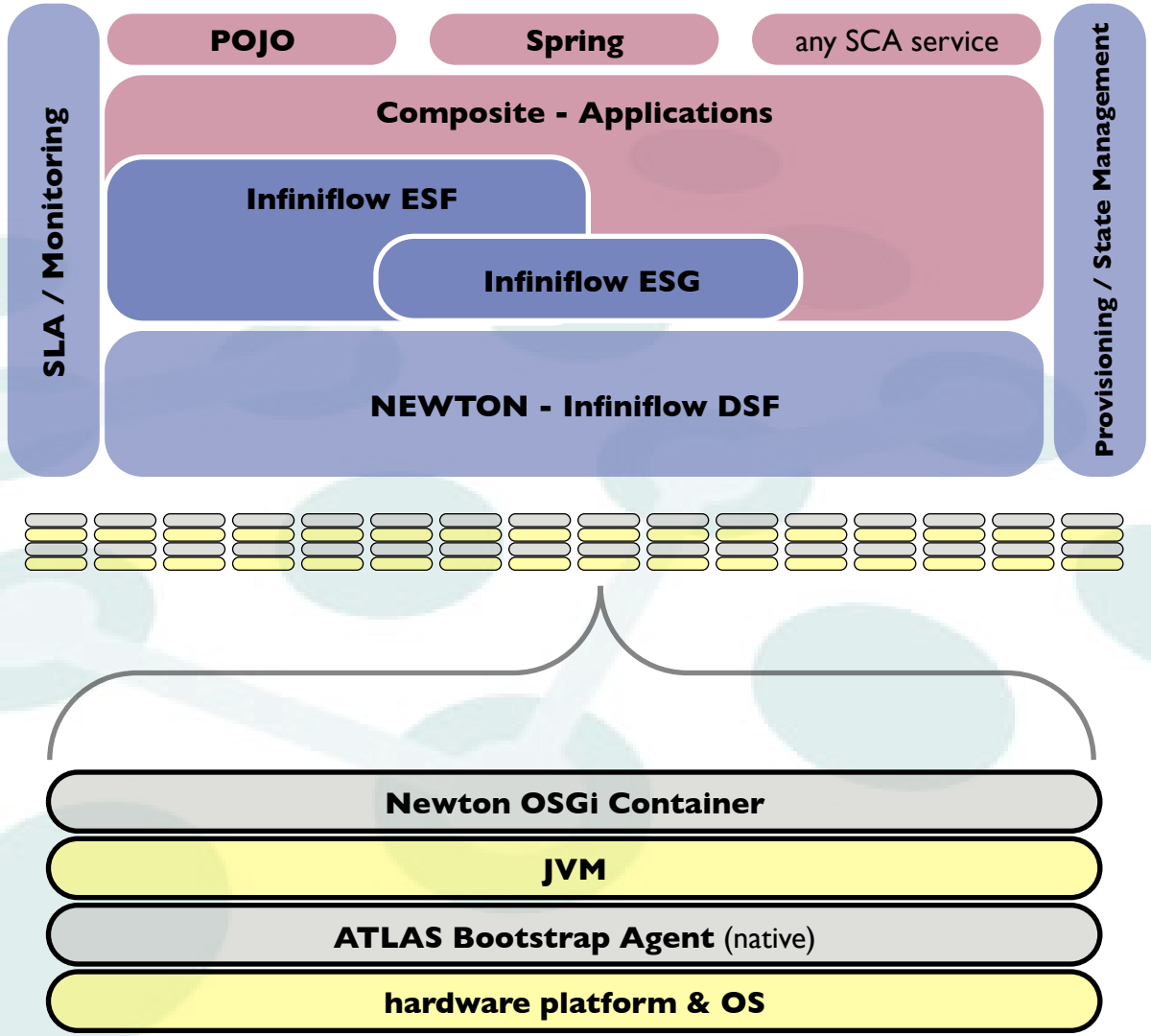
TRUSTED

UNTRUSTED

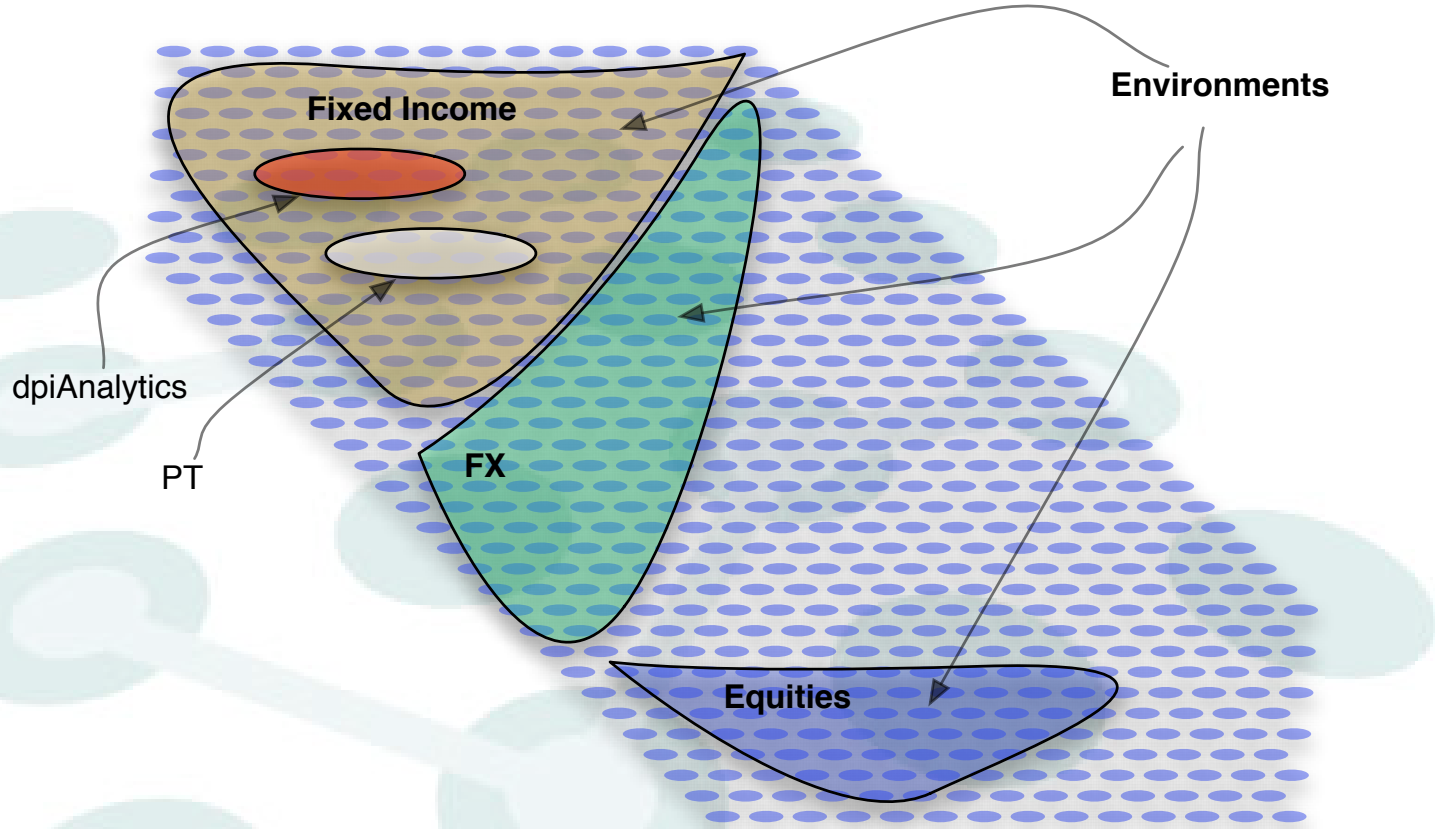
OSGi — Jini — Web Services →

Jini — Legacy Applications (EJB / Corba)

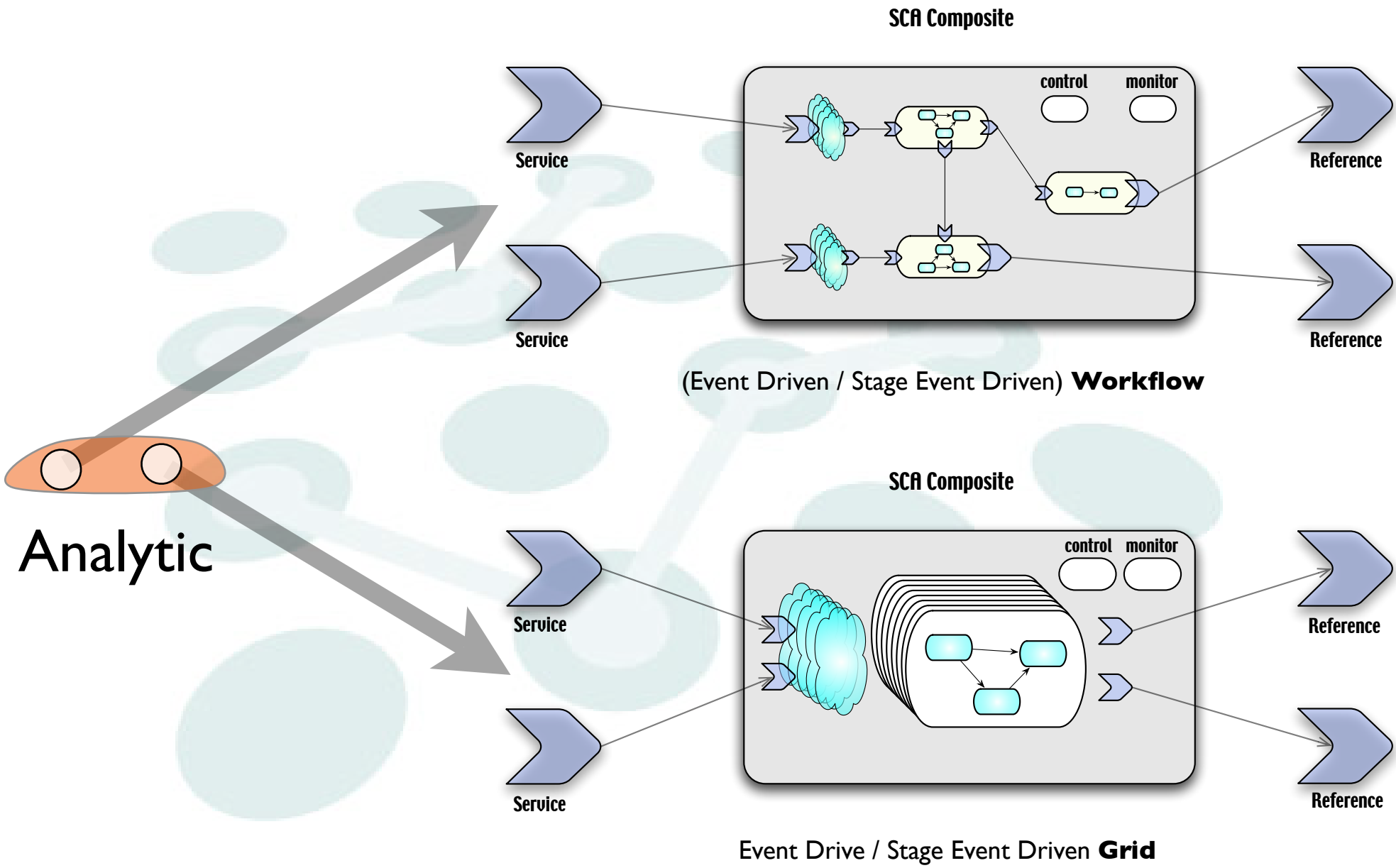
# Infiniflow



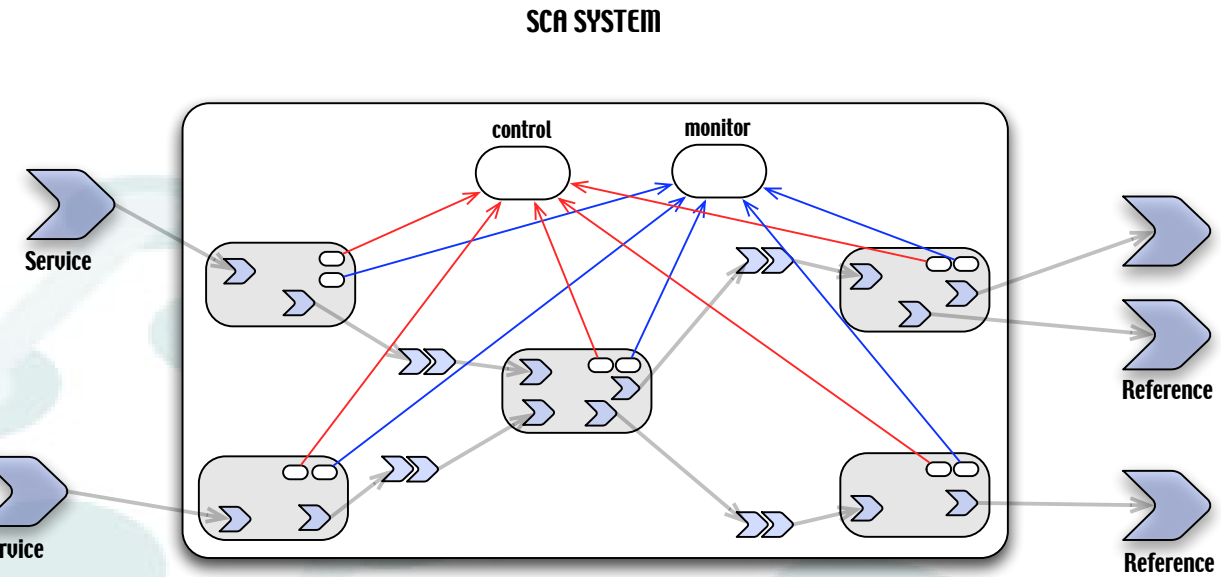
# The Fabric



# Infiniflow Composites



# Infiniflow System




A SCA "*SYSTEM*" comprising of many dynamically instantiated *SCA composites*.

Analytics Resource

# Towards SCA – The Newton Approach

- Acronym city, a brief guided tour
- The Newton Component Framework
  - Concepts
  - Comparison
- Infiniflow - A Service Orientated Architecture using Newton
- Demo - Fractal Scatter / Gather
- Summary

# Summary

- Newton overview
  - Distributed component framework
  - Code and Component dependency wire-up, garbage collection, multi-level
- Why OSGi?
  - Classloading, bundle lifecycles
- Why SCA?
  - Industry standard defining interdependencies between services
- Component based system management
  - A framework built using the SCA model
  - Self similar, scalable, extendable
- Demo
  - Simple mechanism to build large scale distributed applications

# Interested in knowing more?


See - http://www.codecauldron.org

David Savage
dave.savage@paremus.com