

Vision for the Future



Semantic Service Oriented Architecture

Case Study for

OMG SOA/MDA/WS Workshop

March 30, 2006



Lack of Awareness and Sharing of Available Capabilities (Services)

- Sheer Volumes of Data and Services Compounds the Problem
 - Word of Mouth “Discovery” Typical
- Available XML Web Services Solutions are ‘Pervasive,’ but...
 - Lack Ability to Easily Discover Services
 - Are Location Dependent; “Stale” References Possible
 - Are Protocol Dependent
 - Have Weak, or No, Semantics
 - Results in (at best) a Centralized Distributed Architecture
 - Include Ever-growing Multitude of Largely Unimplemented Standards (re: WS-*)
- When Found, Services...
 - Not Readily Interoperable
 - Not Described by “What They Provide”
 - Result in Human-Centric, Ad-hoc and Intermittently Repeated Processes
- Fragmented, Sub-Optimal Operations
 - Long Standing Problem – Analysts Aren’t Able to Focus on Core Competencies

SSOA Provides Significant Value

SRC Sensitive

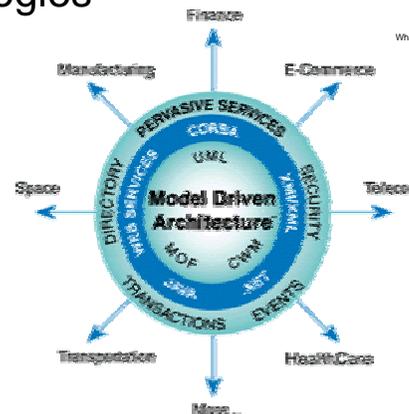
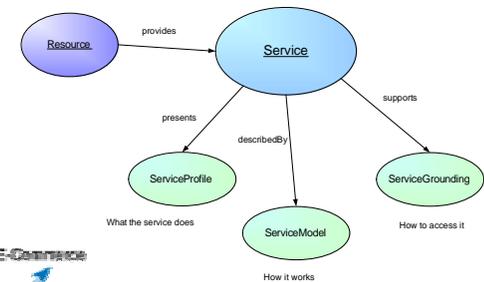
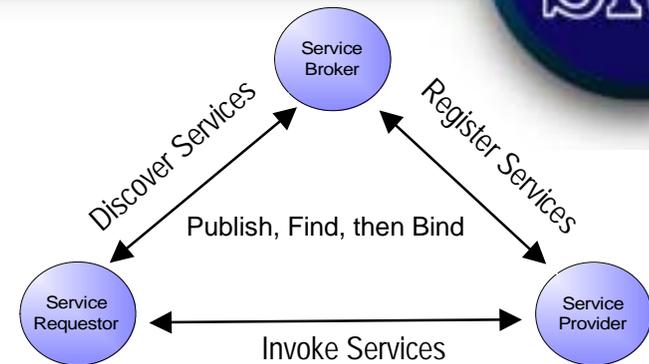


SSOA Compliments XML WS by Supporting a SOA that is:

- **Semantically Enabled**
 - Powering Efficient Publishing, Discovery, and Execution of all Available Services
 - Recommending Appropriate Services when New Services Come Online
 - Allowing Software Agents to Dynamically Construct Adaptive Workflows
 - Designed and Implemented Based on Current and Emerging Standards
 - Providing the Ability to Compose “Virtual Applications”
- **Powered By Sun Microsystems’ Jini™ Distributed Computing Model**
 - Location Independent, Protocol Agnostic Services
 - Autonomic (e.g. Spontaneous Networking, Self Healing, Self Synchronizing)
 - Allows Near-Real Time Collaboration and Capabilities Sharing
 - Relevant Services *Presented to* End Users
 - Distributed Event Model
 - Decentralized Distributed Computing Capability
- **Demonstrating Ability to Share Resources Across the Enterprise**
 - Enhancing Current SOA Projects by Acting as Risk Reduction/Complimentary Task



- Service Oriented Architecture (SOA)
 - Separation of Concerns
 - Functionality Discovered, Used, Re-used
 - Standard Interfaces Abstract Impl'n Details
- Semantics Based Computing
 - Machine Interpretable Content
 - Structure + Epistemology + Logic
- Standards Based Design (SBD)
 - Presupposes Pervasive Heterogeneity
 - Integrate Existing Apps w/ New/Future Technologies
- Standards Involved w/ Prototype
 - *ISO 11179 Metadata Registries*
 - ISO 19763 Meta-Model Framework
 - ISO 24707 Common Logic
 - OMG Ontology Definition Meta-Model
 - W3C Semantic Web Services Framework



Powerful Jini™-Based Model

SRC Sensitive



integrates at the
Business Process Layer

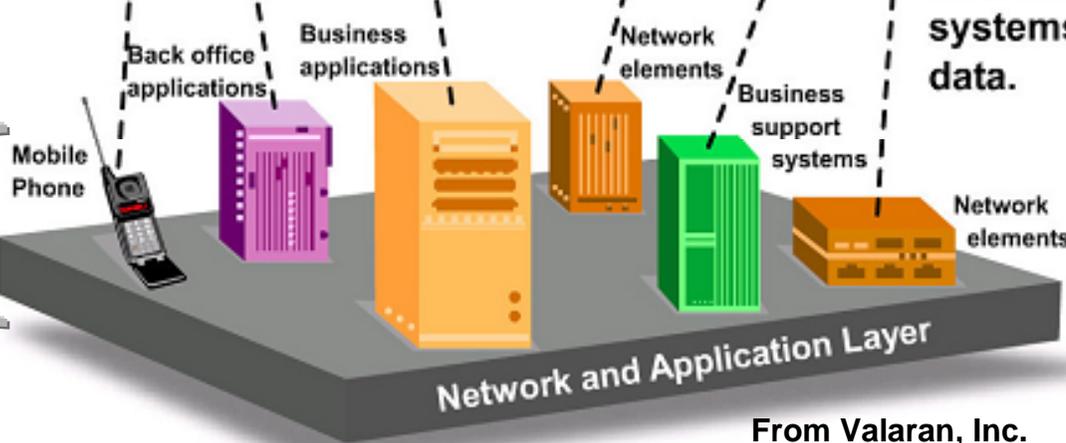
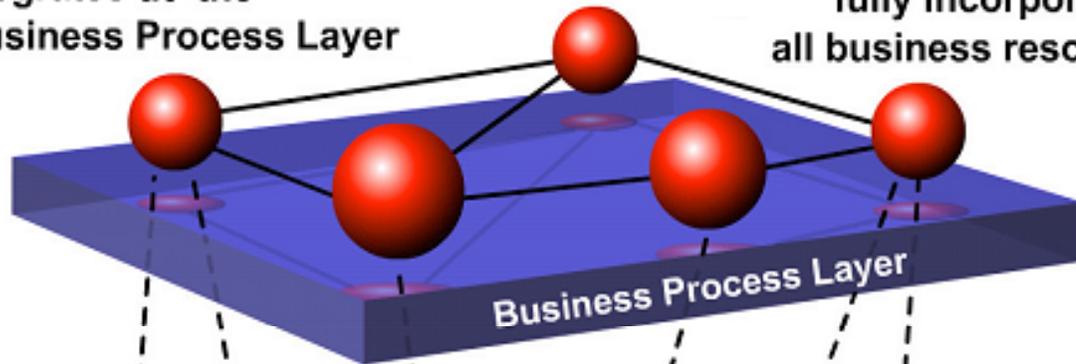
fully incorporates
all business resources

Abstraction,
Location
Independence

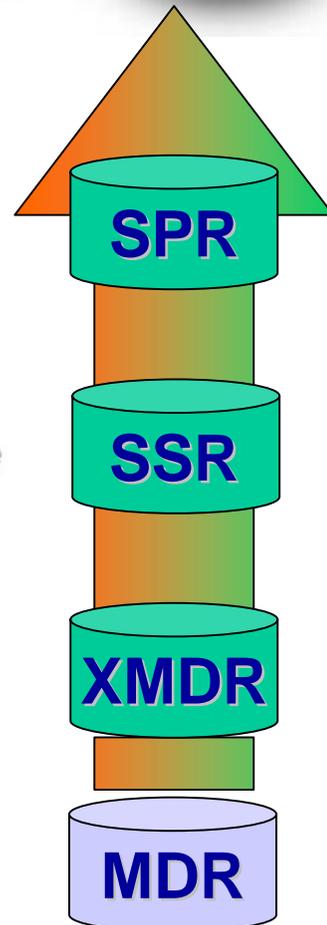
Protocol
Agnostic

Heterogeneity

- Logical (s/w)
- Physical (h/w)
- S3



unifies
all enterprise
systems and
data.



From Valaran, Inc.

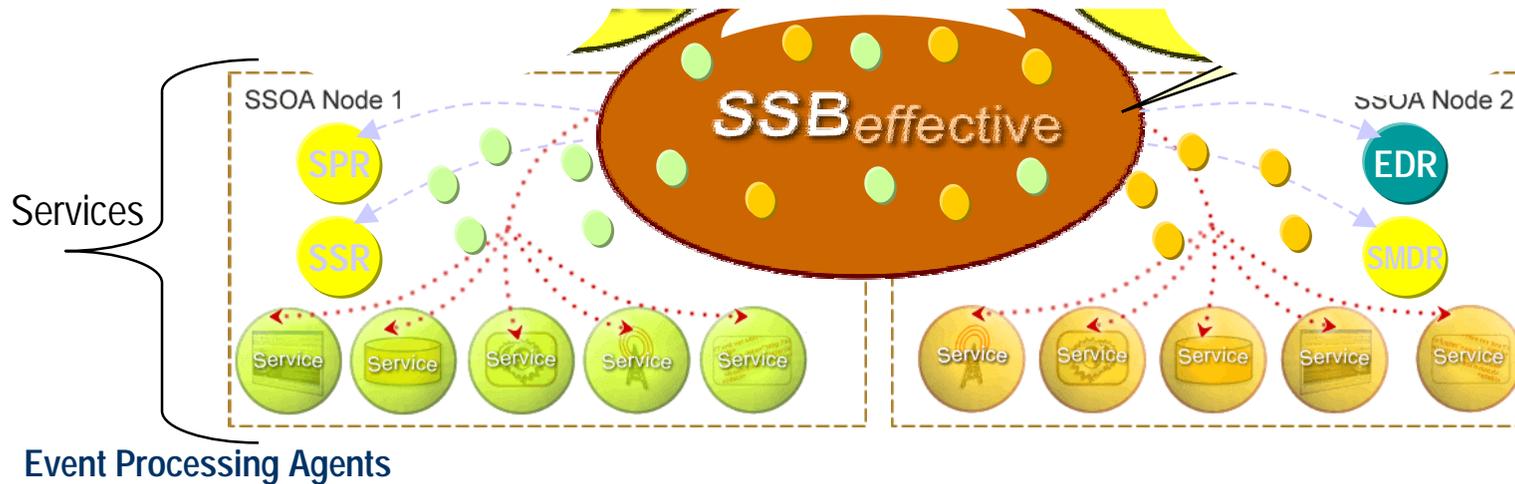
Prototype SSOA System View

SRC Sensitive



Semantic & Agent Components

Services
Locally Exchanged
(e.g., Awareness)





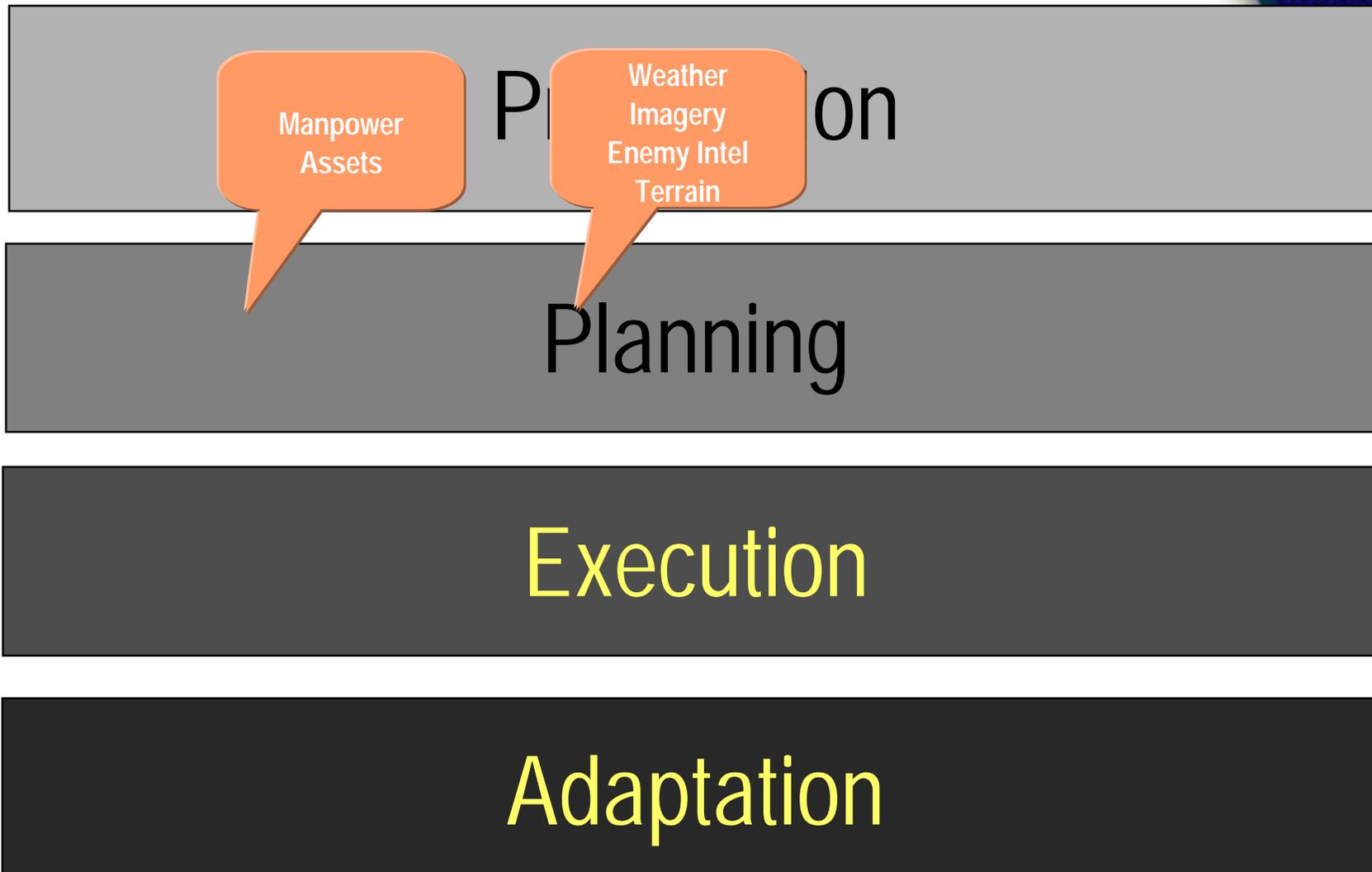
Combat Search and Rescue (CSAR)

CSAR Process (Greatly) Distilled

SRC Sensitive

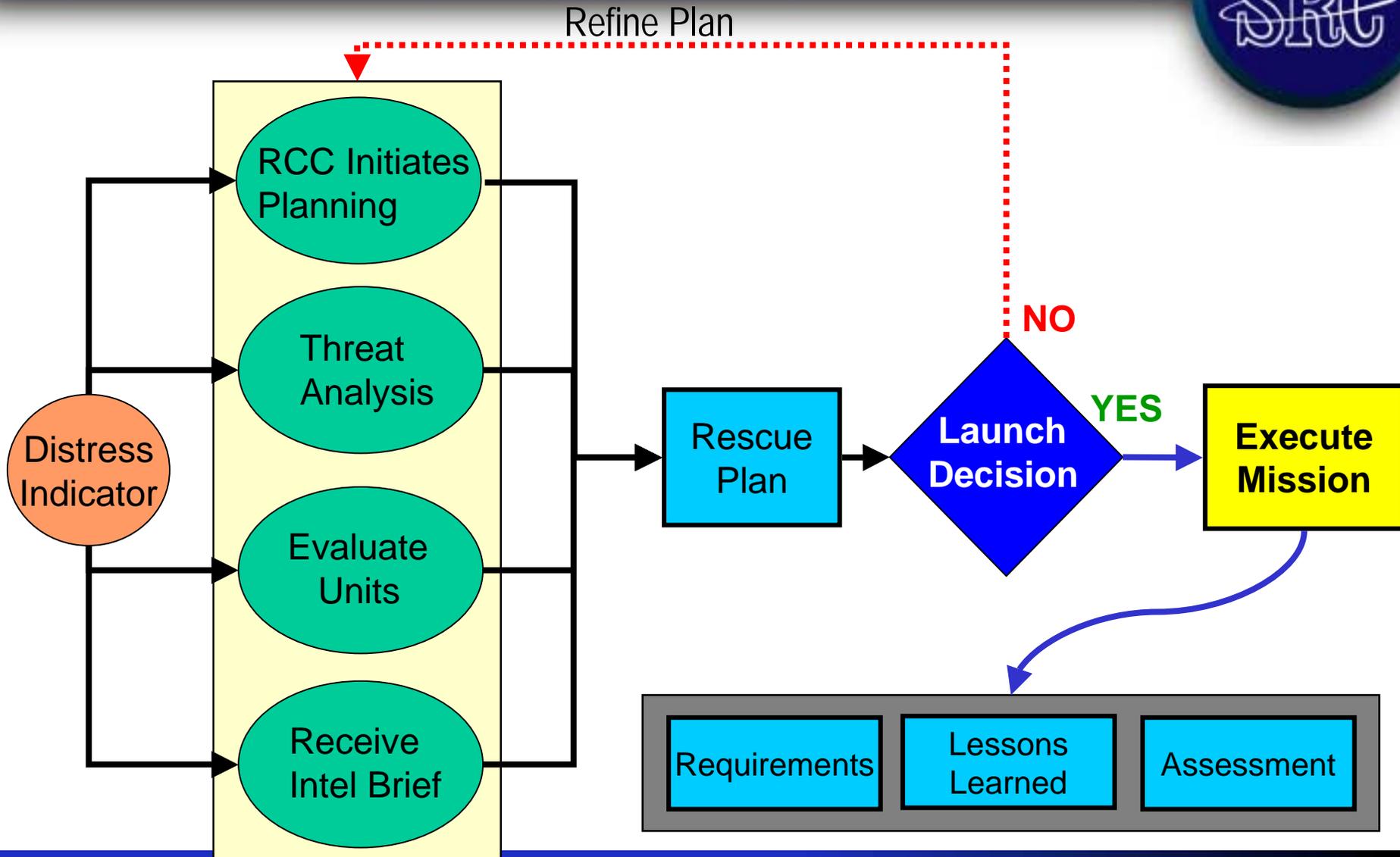


IAW Doctrine JP 3.50-2



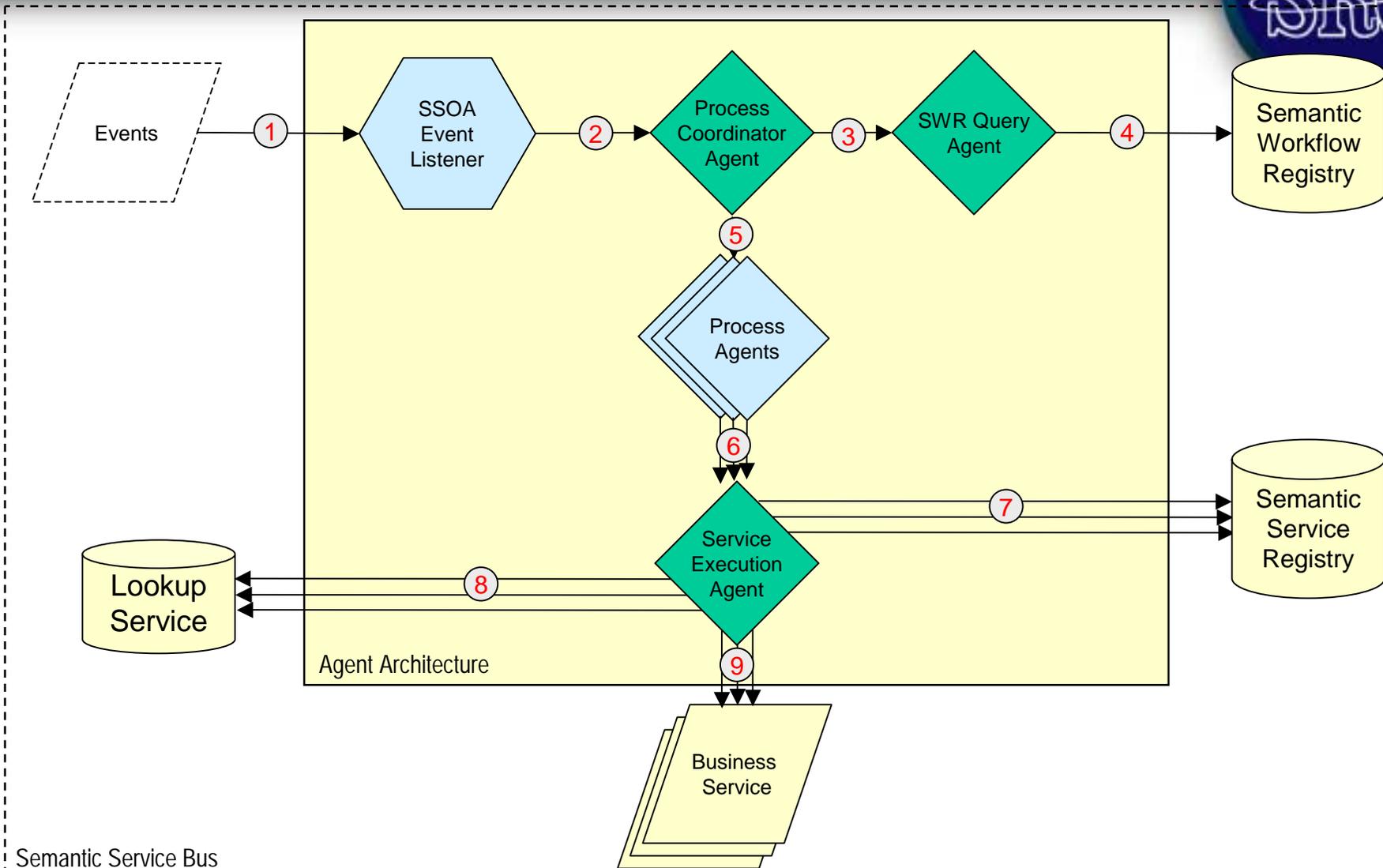
CSAR with SSOA

SRC Sensitive



SSOA Event Flow

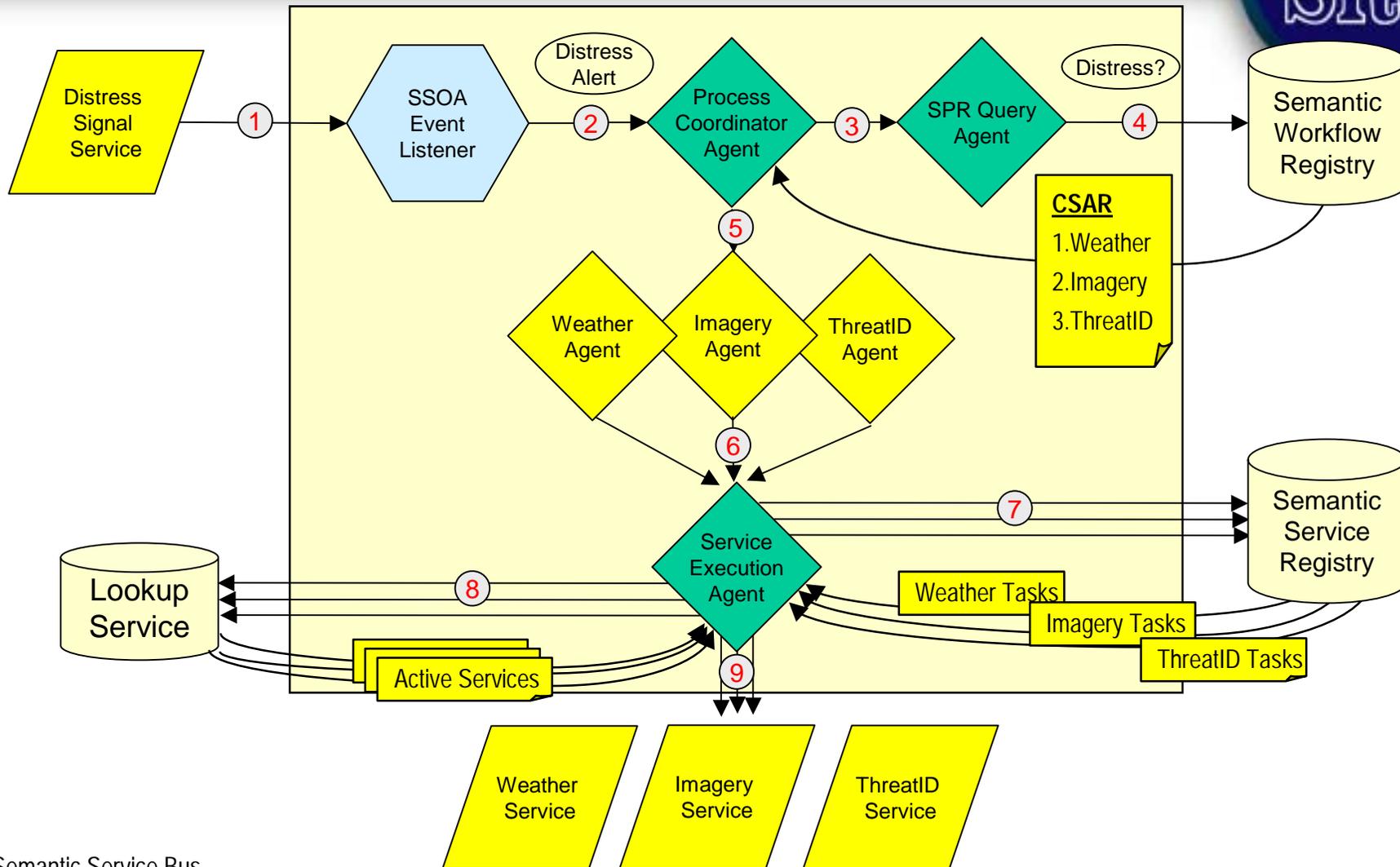
SRC Sensitive



Semantic Service Bus

SSOA Event Flow – CSAR Demo

SRC Sensitive



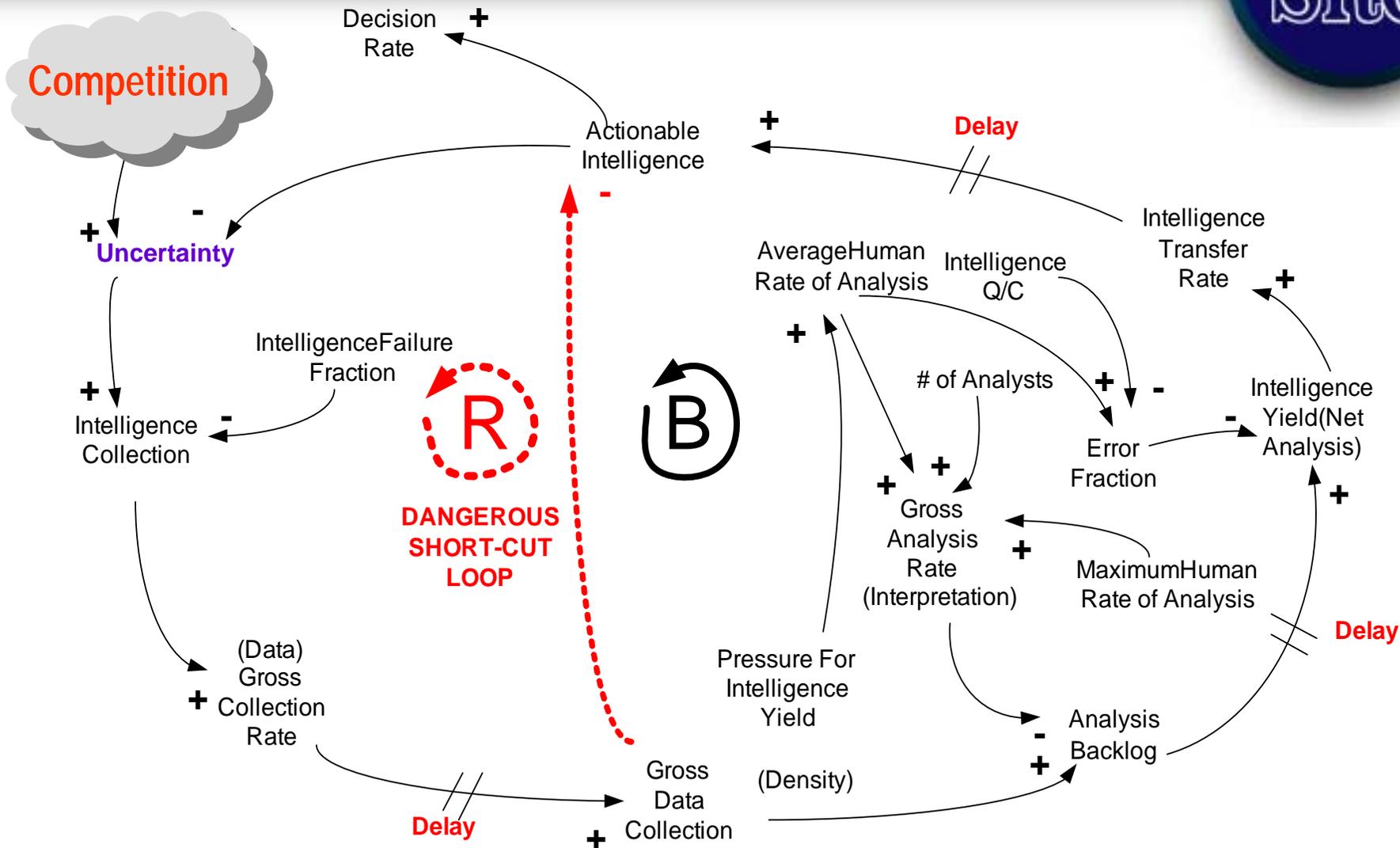


Demonstration



- Key Concept: Every Service Type is a Collection of Tasks
 - Each Task Semantically Corresponds to a Specific Operation or Action
- Every Service is a Running Instance of a *Service Type*
 - Multiple Instances of a Specific Service Type May be Deployed to:
 - Build in Redundancy & Provide Load Balancing
- Task Selection Depends on a Semantic Description, Comprised of:
 - Input, Output & Action Types
- Given a Set of these Input, Output & Action Types, the SSR will:
 - Return the Candidate Tasks and Associated Service Types,
 - Provide Necessary Information to Discover and Execute Any of the Tasks Within the SSB
- Weather Service Example:
 - Inputs are: { Location, TimeStamp }
 - Outputs are: { TemperatureC }
 - SSR Matches 3 Potential Tasks:
 - TemperatureC getTemp(Location,TimeStamp)
 - TemperatureC temp(Location,TimeStamp)
 - TemperatureC getTempCelsius(TimeStamp,Location)
 - But Not:
 - TemperatureC getTempCelsius(TimeStamp,Location,Altitude)

Conclusions

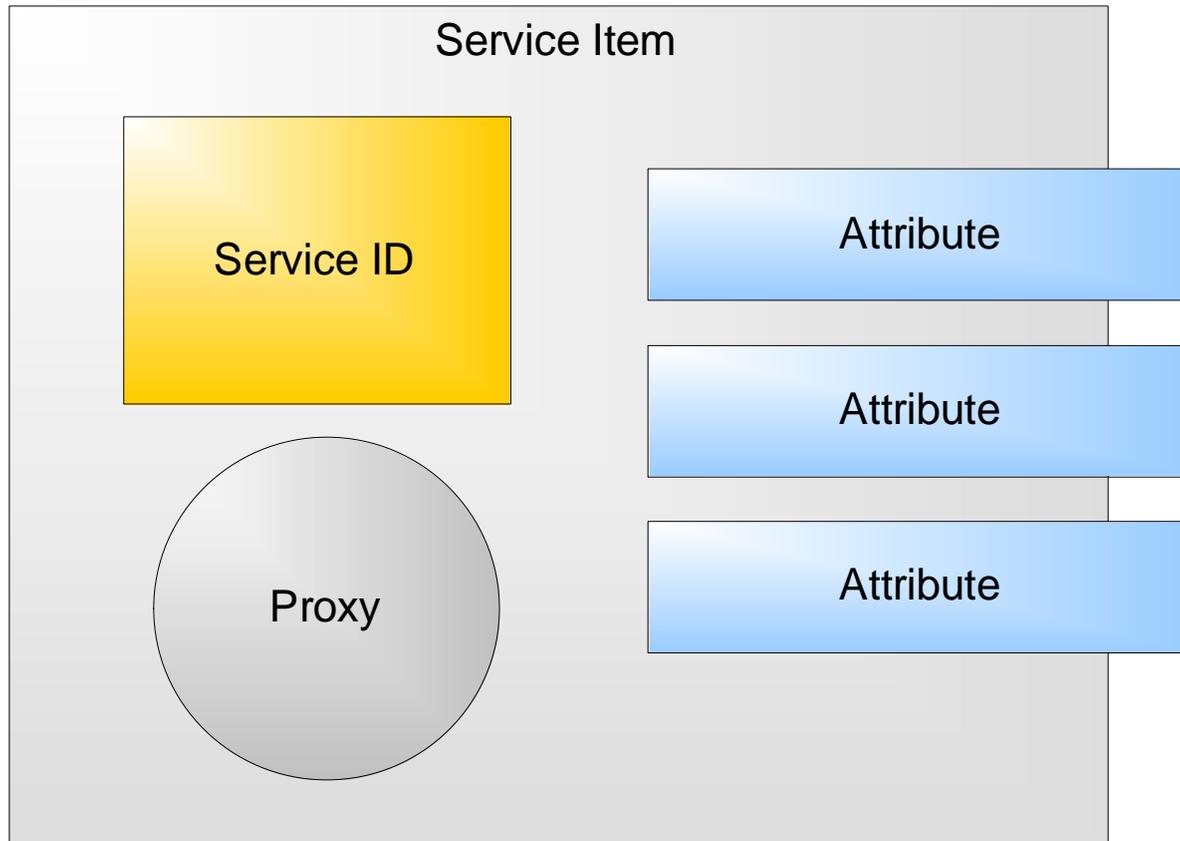




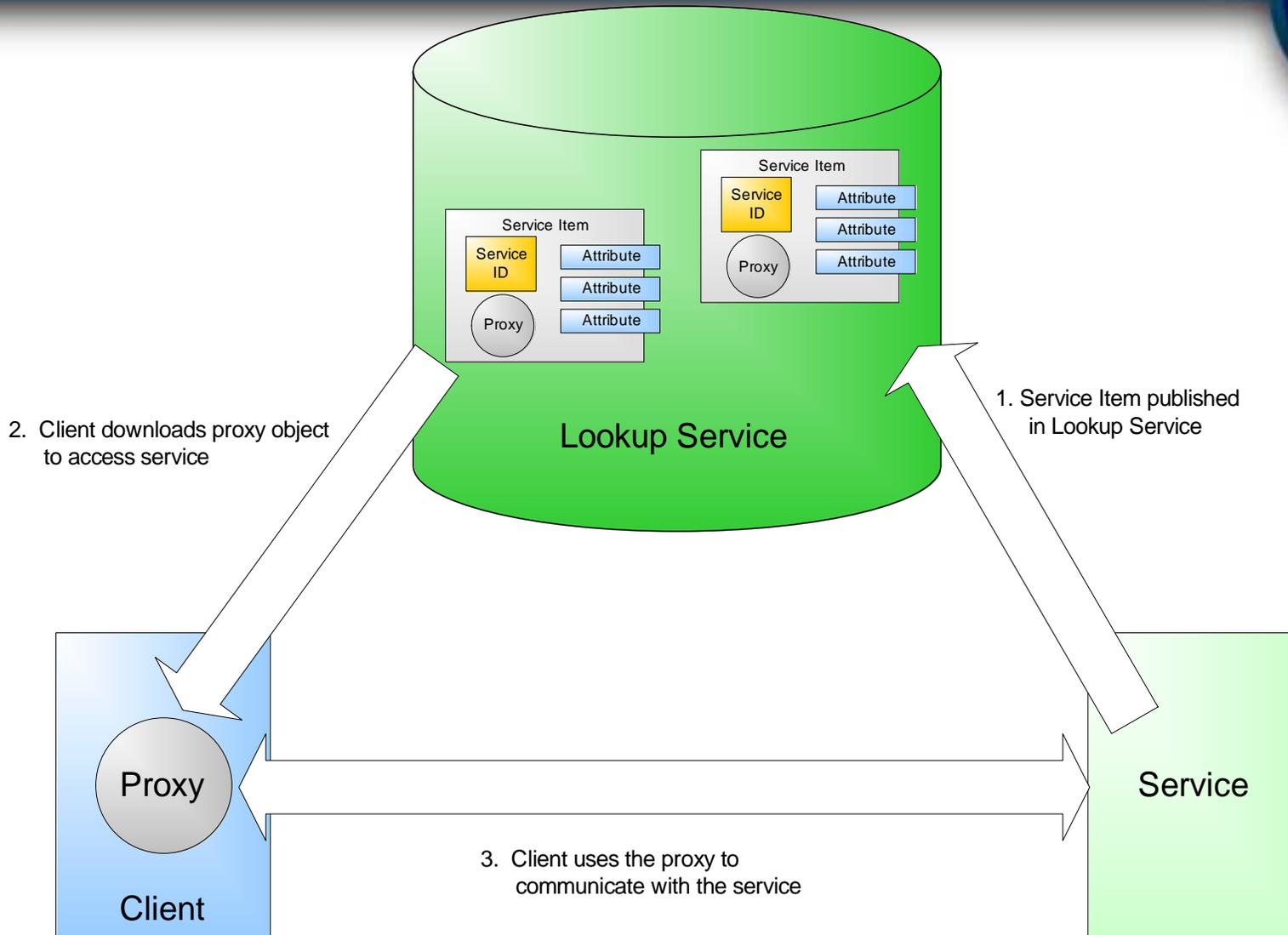
Back ups

Jini™ Service Item

SRC Sensitive



Publish, Find, then Bind



Publish, Find then Bind

SRC Sensitive

