Everware-CBDI

Independent Insight for
Service Oriented Practice

Identifying and Planning Services within a Policy Framework

John C. Butler

www.everware-cbdi.com

Agenda

- Concept Review
- The Policy Framework: Policy Categories
- Policy Impact on Service Identification
- Using Policies in Service Portfolio Planning
- Summary

Everware-CBDI

Everware-CBDI

Independent Insight for
Service Oriented Practice

Concepts

www.everware-cbdi.com

SOA Defined

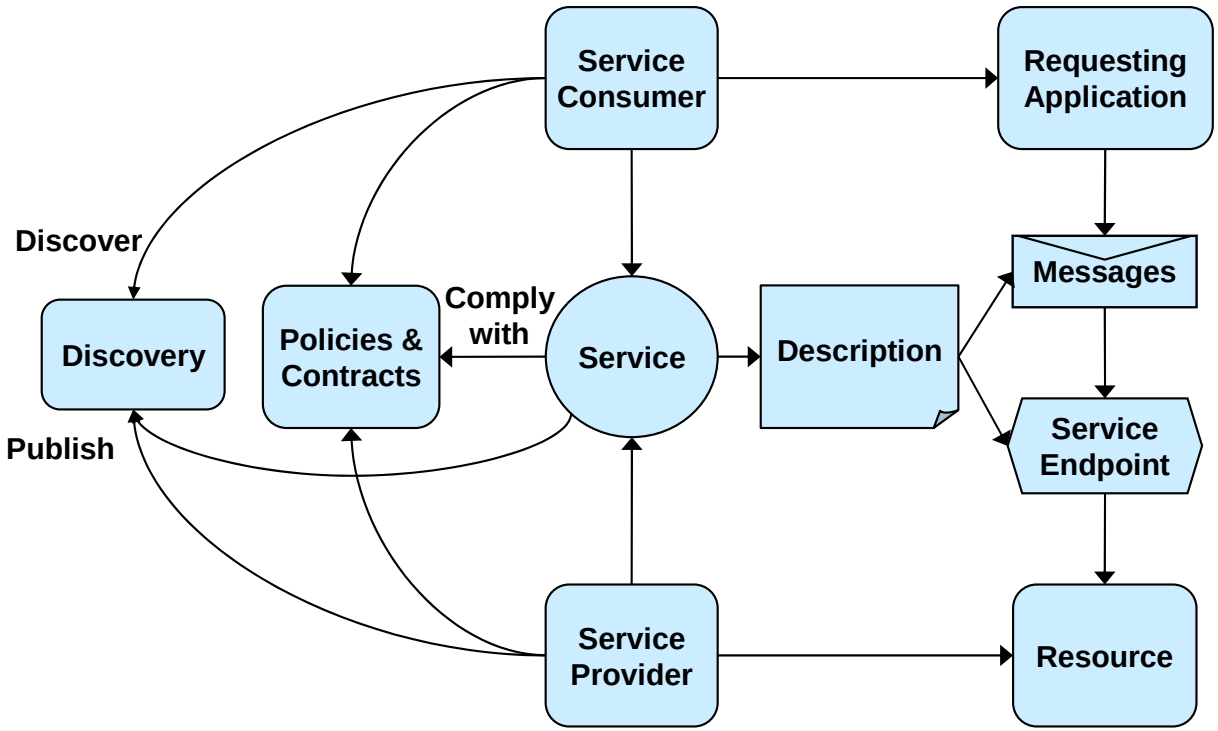
- Service Oriented Architecture (SOA) is an **architectural style** that enables the **assembly of systems from distributed, federated resources**
- SOA **is an architectural framework** for the
  - Classification of different Service Types
  - Layering of Service Types to separate different concerns
  - Identification of Services and their placement into the appropriate layer and classification
- These facilitate
  - Loose coupling of providing resources and consuming solutions
  - Sharing and reuse of Services
  - Flexibility in the provision of resources and use of Services
  - Policies to govern usage, sharing, flexibility

"Service Oriented Architecture (SOA) is the policies, practices and frameworks that enable application functionality to be provided and requested as sets of services published at a granularity relevant to the Service Consumer, which are abstracted away from the implementation using a single, standards based form of interface." (CBDI)

Everware-CBDI

What is a Service?

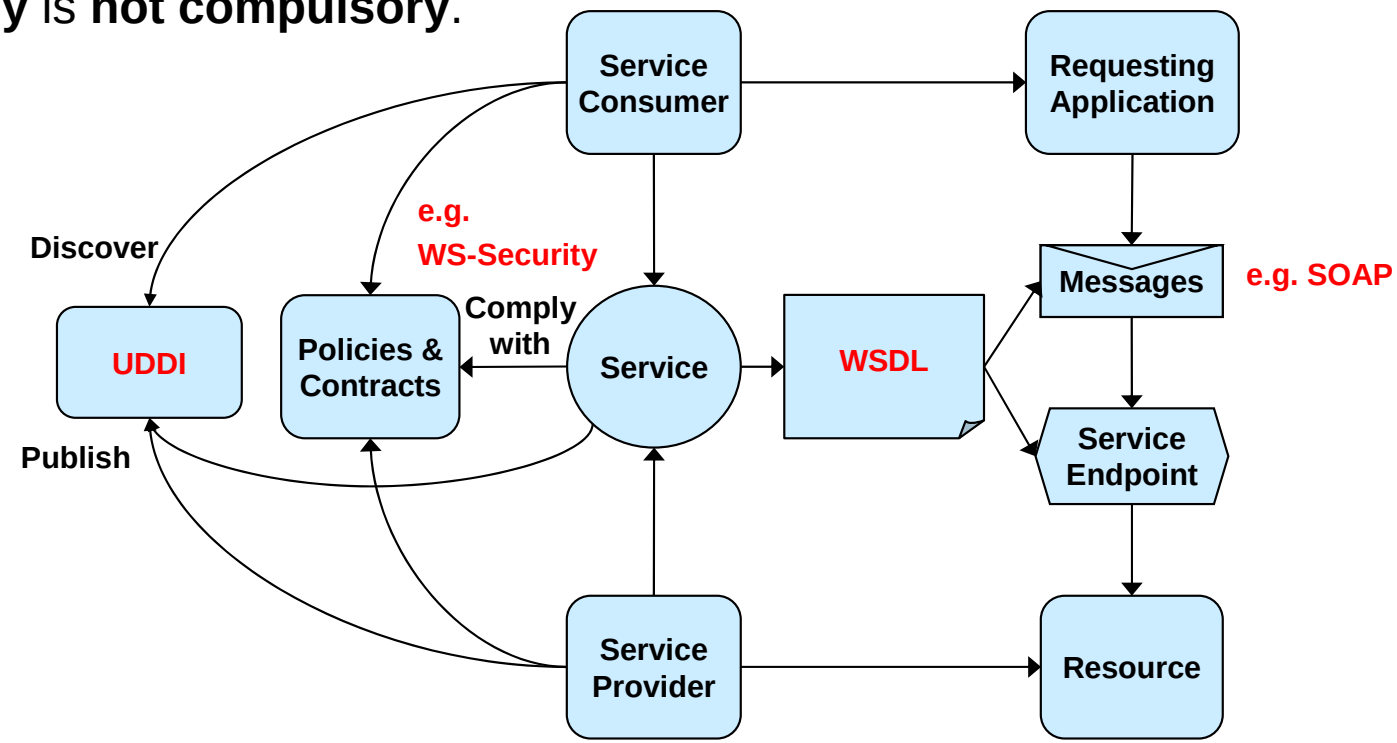
- A **Service** is a **capability** by which the **need** of the **Service Consumer** or **Requestor** is **satisfied** according to a **contract**
- Separate the **what** from the **how**, **who and where**
- Manage and govern through **policies** and **contracts**
- Communicate using **messages** that **share schema not technology**
- Are **discoverable,** at **design** and **run-time**



Everware-CBDI

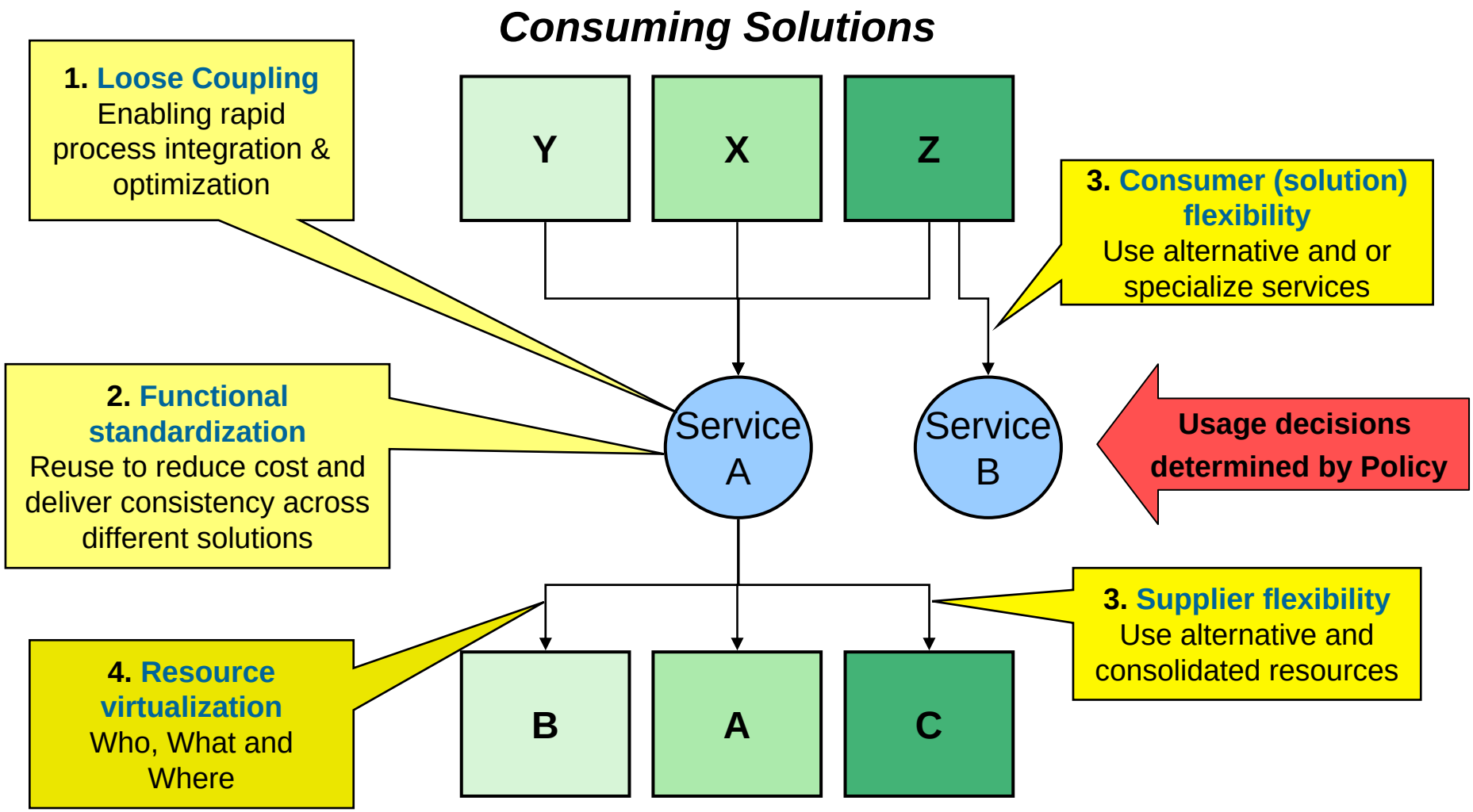
What are Web Services?

- **Web Services** is a set of **Protocols** based on the **eXtensible Markup Language (XML)** that provide a mechanism for the description and deployment of Services.
- A **Web Service** is a **programmatic interface** to a **capability** who's **behavior** is described using **Web Service Protocols,** and where **interaction** with it is conducted using those Web Service protocols.
- Whilst the term Web Services is widely accepted, the use of the **Web technology** is **not compulsory**.



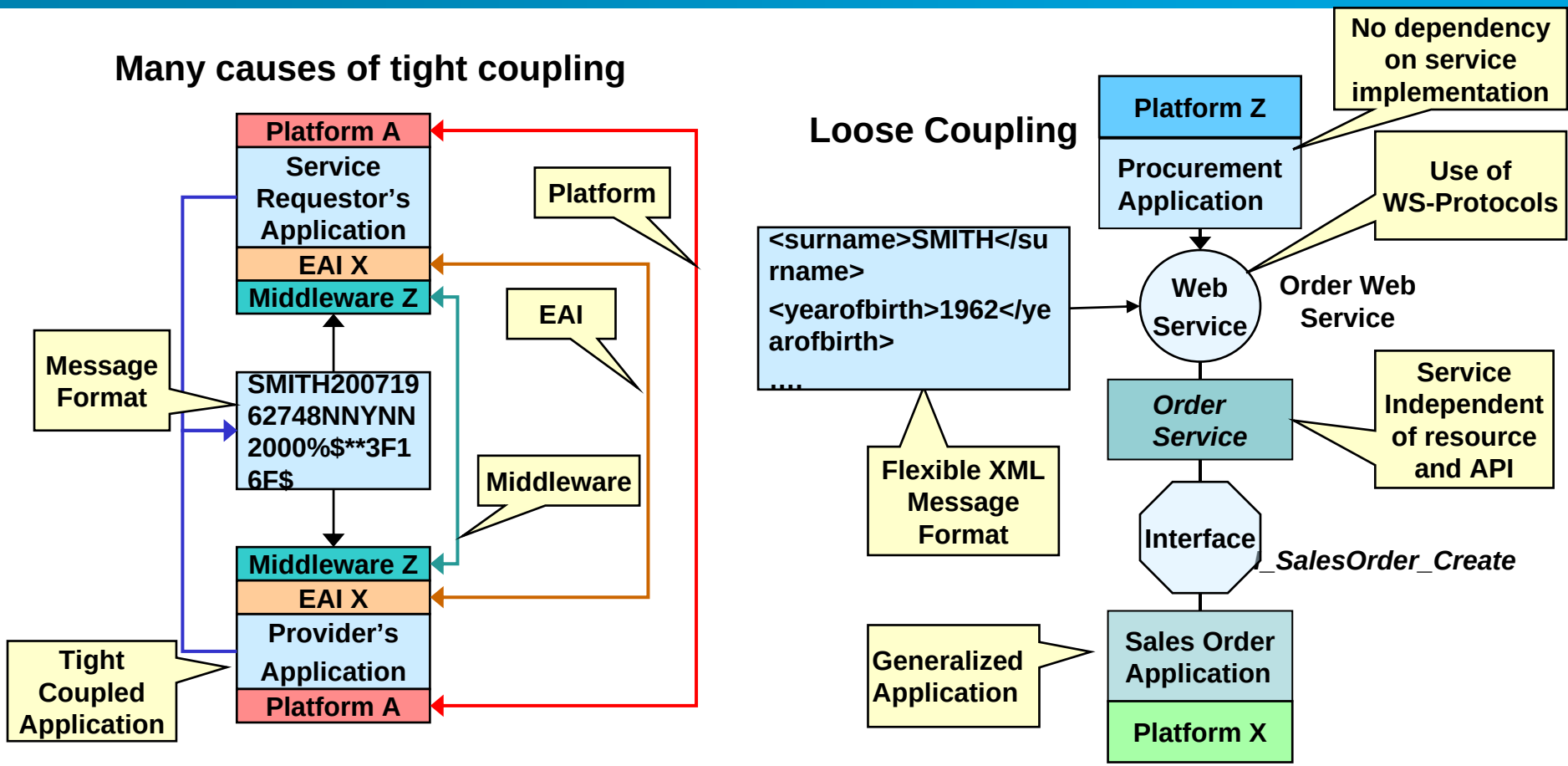
Everware-CBDI

Core SOA Characteristics



Everware-CBDI

Key Principle: Loose Coupling

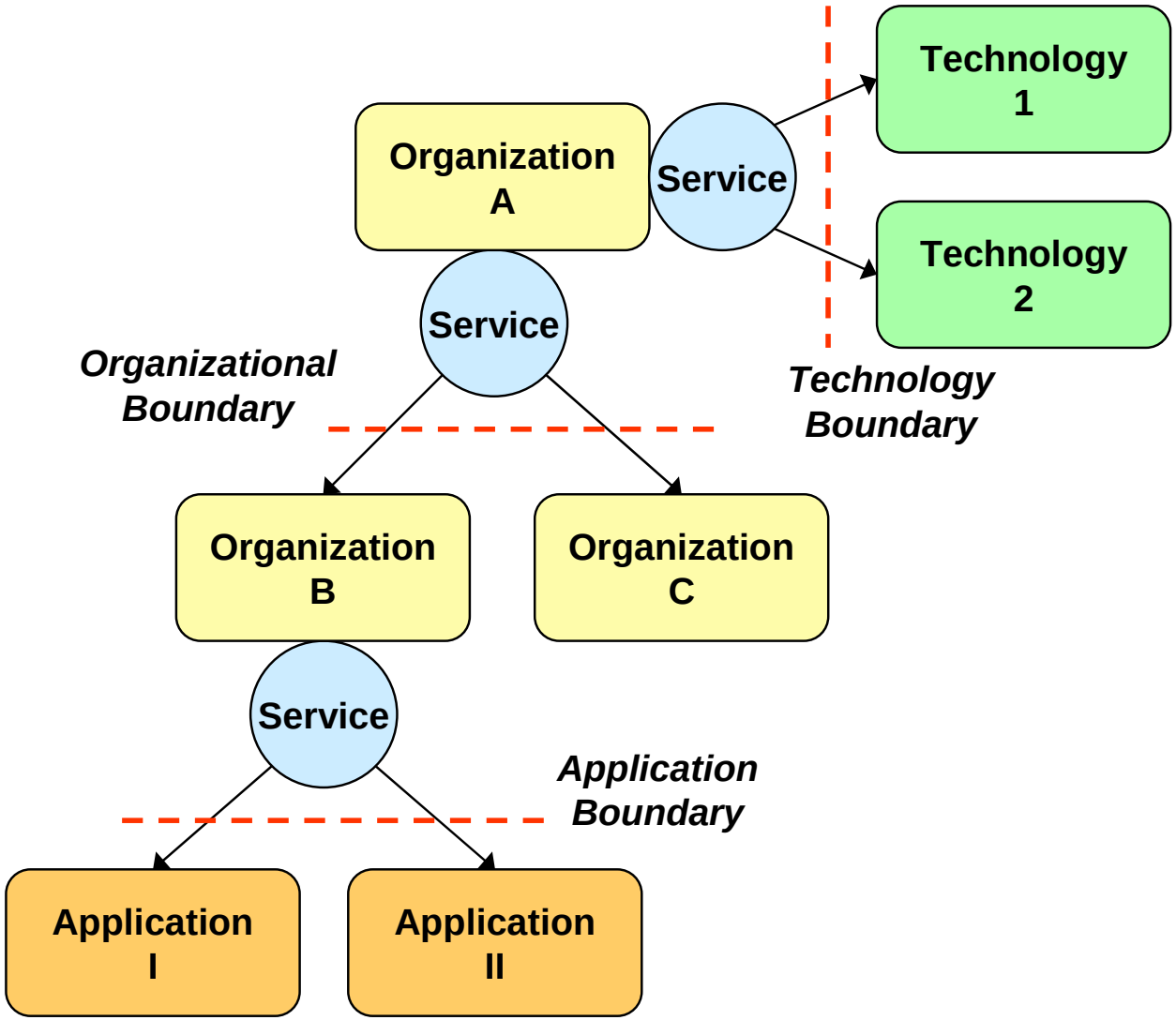


Service Goals

- Implementation replaceable without impacting consumer
- Sharable by many consuming solutions
- Consumer can select from alternate services

http://www.cbdiforum.com/secure/interact/2004-03/Principles_of_Service_Orientation.php

Service = Points of Flexibility



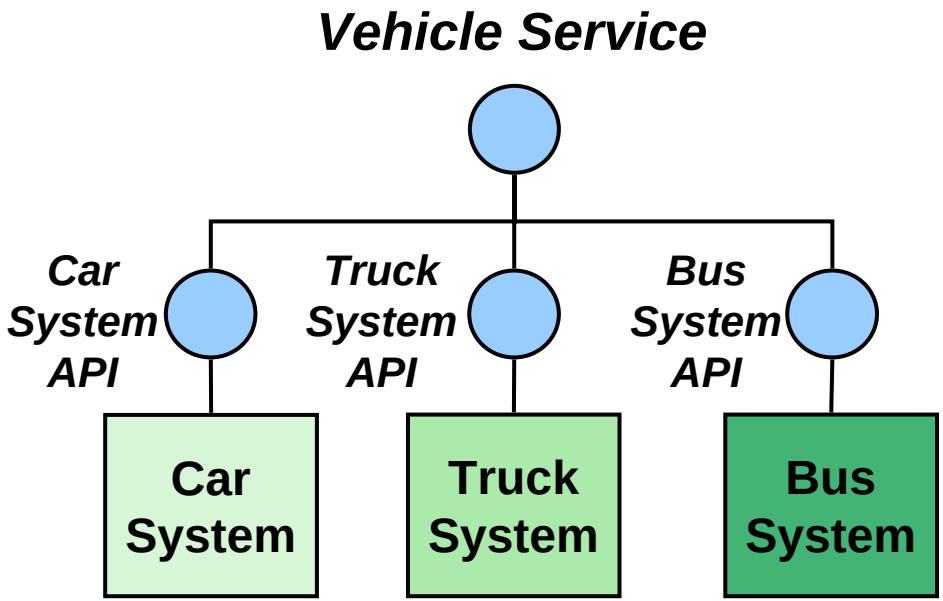
Everware-CBDI

Key Principle: Abstraction

Service Goals

Services provided to solution assemblers (internal or external) are

- **Abstracted** away from the current implementation to reduce dependency
  - Not a direct reflection of current interfaces
  - Messages should not contain implementation specific detail
- May be **Generalized** to broaden applicability
  - Service should not be solution specific
  - Service may generalize business concepts
    - e.g. Vehicle rather than Car, Truck, Bus
  - *Or are specialized from generalized Services*



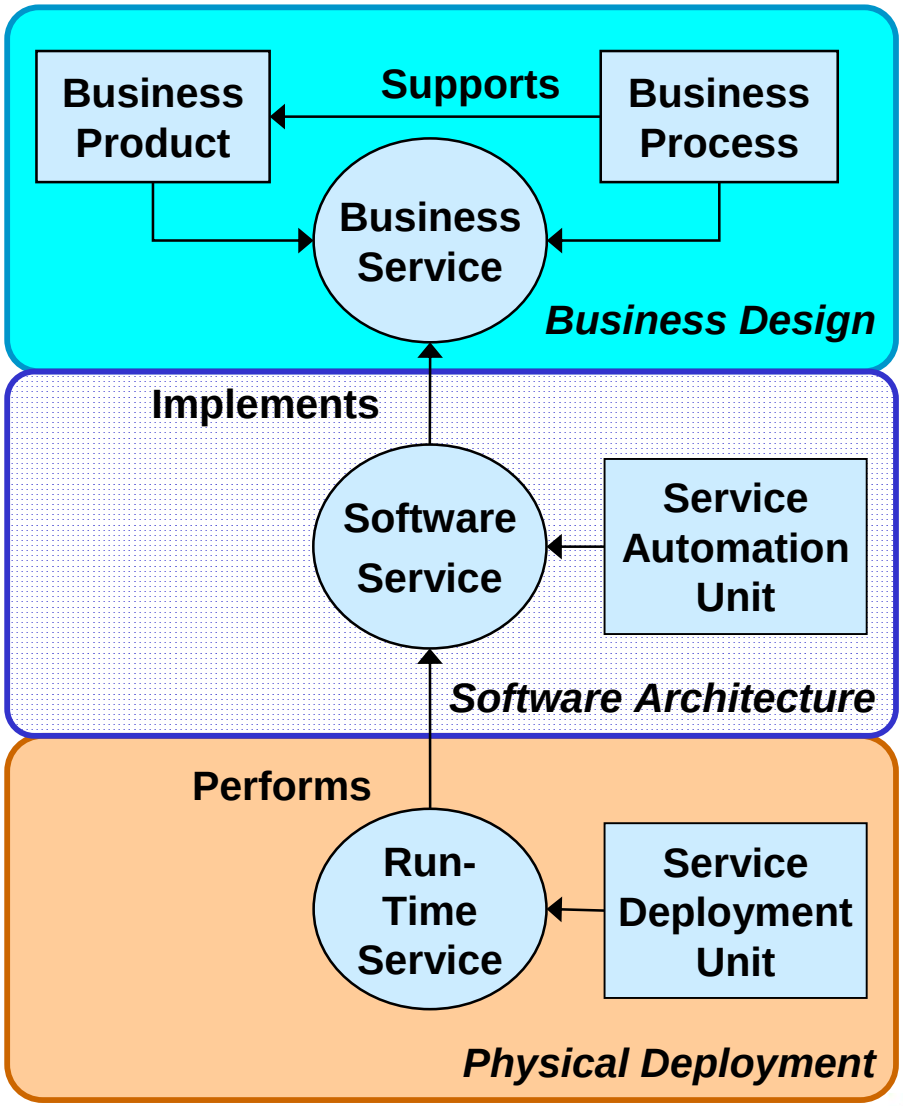
Everware-CBDI

SOA – Three Perspectives

	Focus	Interest
SOA *requires* a Management Framework	• Business and IT Resource Optimization • Business/IT Convergence • IT Process for SOA? • Provider/Consumer Supply Chain?	• Strategy and Roadmap • Organization and culture • IT Process Governance • Provisioning and Sourcing Policies
SOA *is* an Architectural Framework	• Federated Service Architectures • Service Identification and Specification • Service Lifecycle	• Enterprise Architecture Context • Architectural Constructs for SOA • Architectural Governance • Architectural and Design Policies
SOA *is* a Deployment Framework	• Run-time deployment of Services and Resources • Operational Infrastructure • Service Management	• Standards • Service Technology • Run-time Governance • Operational Policies

Everware-CBDI

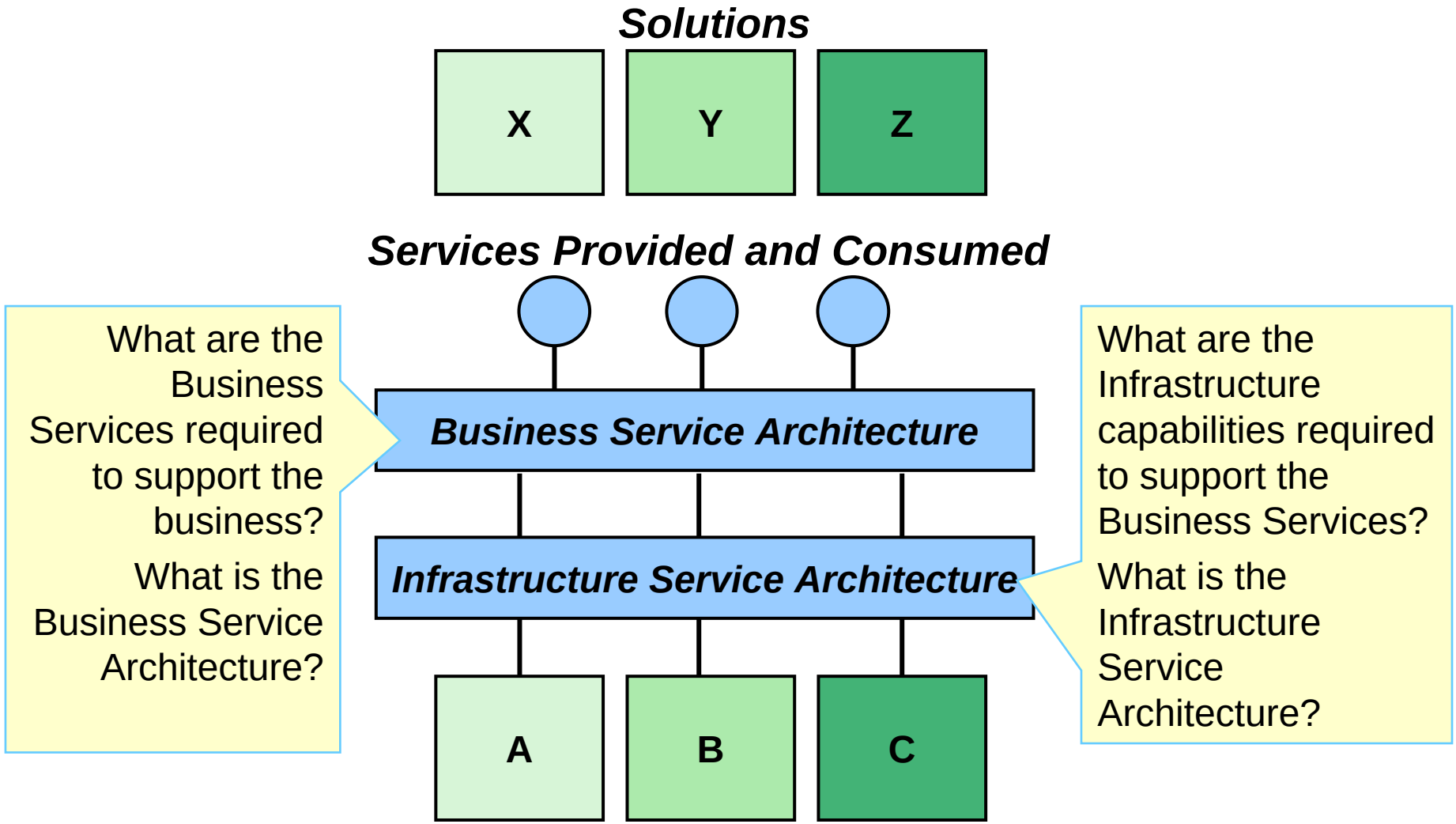
SOA – Three Levels of Realization



- Service concepts are equally applicable to the way the both business and IT each thinks about the provision and consumption of capability and resources
- SOA is as much of a business modeling approach as it is a software engineering paradigm
- Services can represent meaningful business capability
  - Recognizable by the business
  - Enable Business/IT convergence
- Run-time Services provide a further layer of flexibility over the software architecture
  - E.g. Many Run-time Service instances of the same Software Service – resolved by dynamic routing

Everware-CBDI

Two Service Architecture Perspectives



Everware-CBDI

Concepts Summary

- **Services** focus on the **What**, not the **How**
- **SOA** provides a
  - **Structural approach** to deliver **loosely coupled, abstracted** architectures
  - **Reference architecture** for
    - Service classification/taxonomy
    - Policy implementation and governance
    - Contracts
    - Determining sharing and generalization at many levels
  - Basis for a **repeatable engineering process**
- SOA is not confined to use of Web Services, but there are benefits of using them

Everware-CBDI

Everware-CBDI

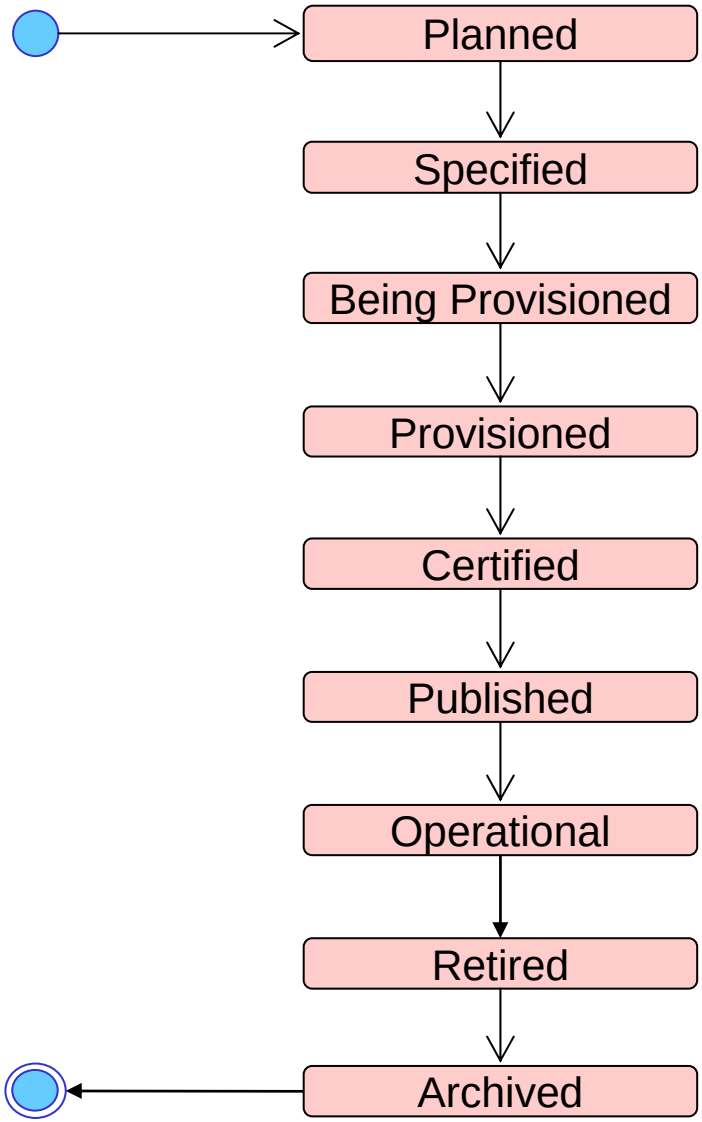
Independent Insight for
Service Oriented Practice

The Policy Framework

Policy Categories

www.everware-cbdi.com

The Service Life Cycle – Defines Service State



/include proposed service in portfolio plan

/prepare service specification and WSDL

demand for operations arises / …

/handover tested service

/confirm service offers required quality

/publicize service, catalog and subject to change control

/deploy service

/withdraw obsolete service

/archive service artifacts

Everware-CBDI

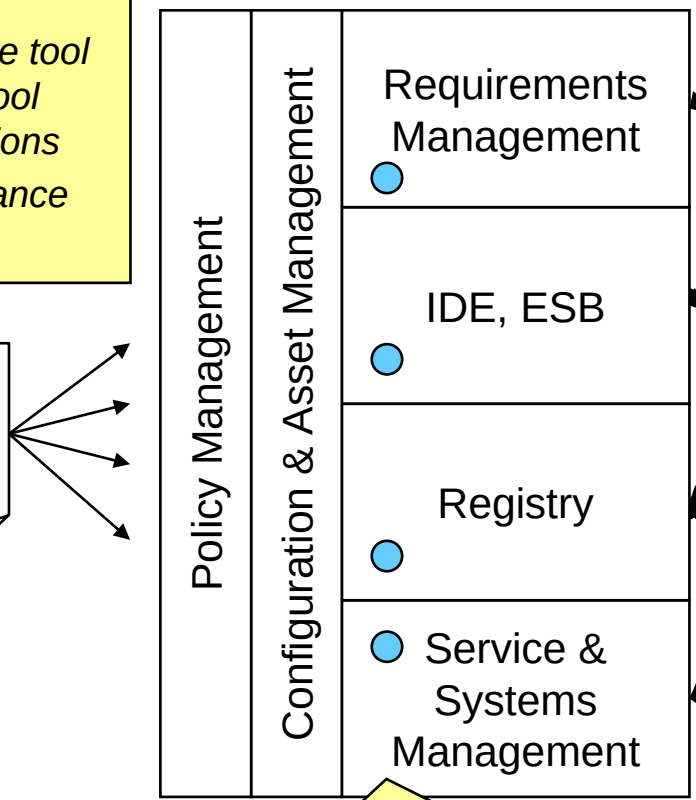
Service Lifecycle Challenges

How can Policies be applied across different tools?

Policies may be tool specific, with tool specific definitions

How is compliance checked?

Policies



Planned
Specified
Being Provisioned
Provisioned
Certified
Published
Operational
Retired
Archived

Changing State may mean

- *Moving from tool to tool*
- *Changing Level of Abstraction*

OMG RAS – Reusable Asset Specification

OMG UML 2 – Models used to document service and the SOA

WS-protocols –

even if the Service is not a WS

Use of WS-Policy

Service is defined in many different tools

How is consistency maintained?

How is the compliance with the specification checked?

Standards that may help share Service artefacts or information across the lifecycle

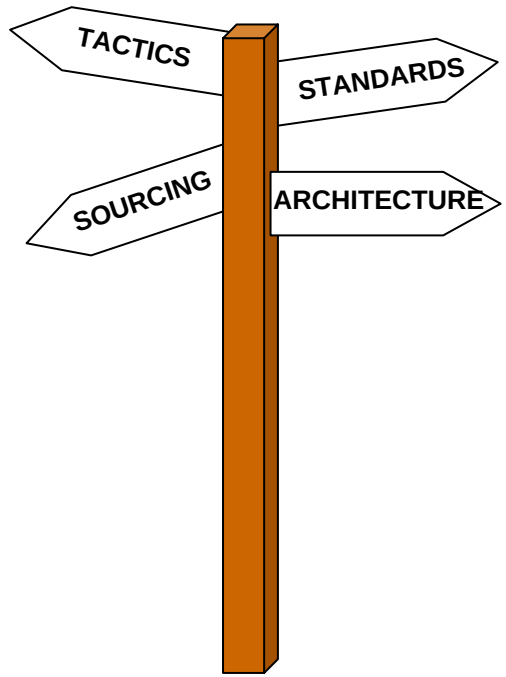
Introduction to Policy Definition

- Policies direct and constrain the service planning and provisioning work, and provide greater transparency
- Ideally, all these policies get decided before service planning commences, but probably need two or three iterations to complete and stabilize the policies
- Beware of procrastination – policy definition is important to the overall consistency outcome of the planning task.

Everware-CBDI

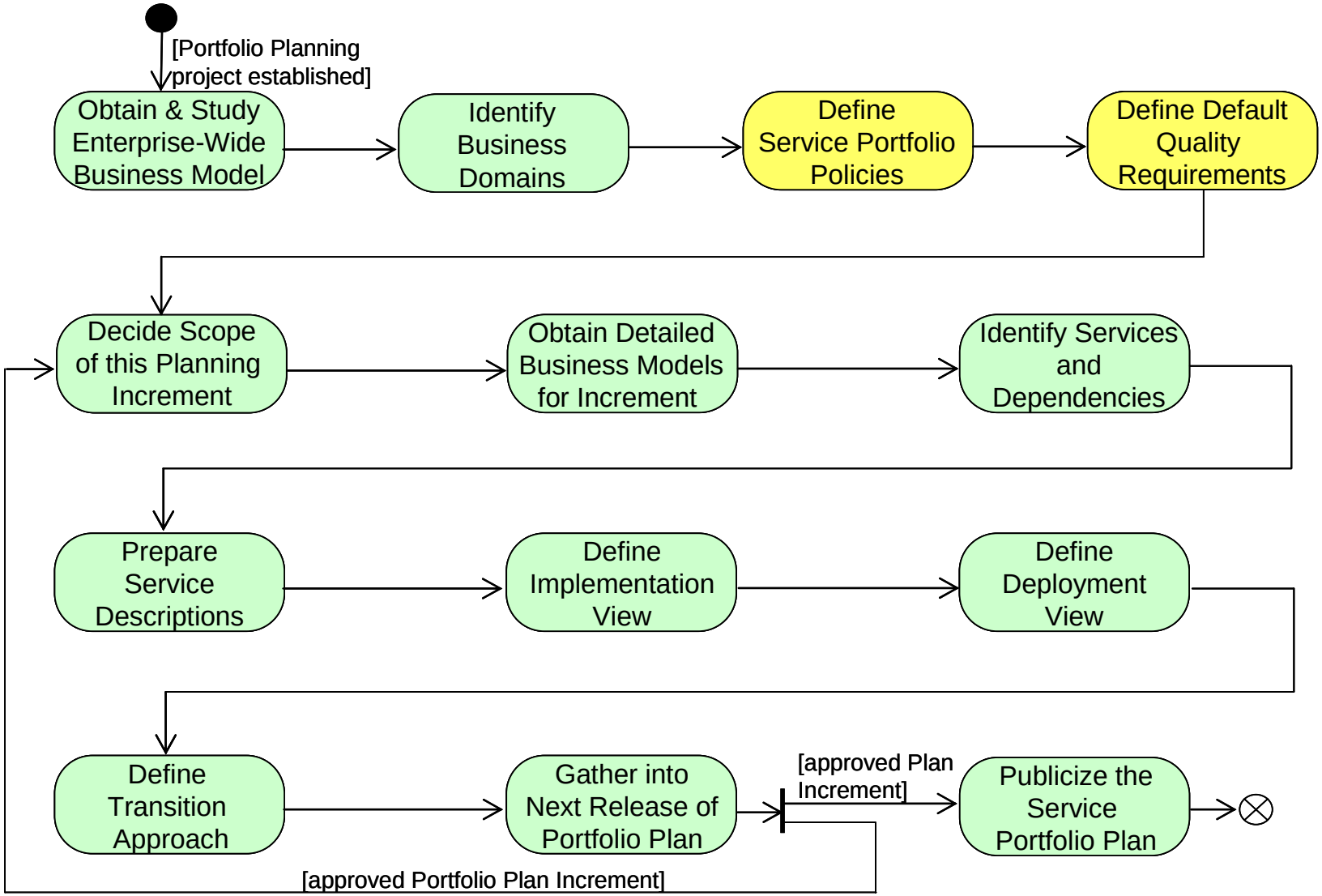
Meaning of “Policy” in Portfolio Planning

- “A strategy or directive defined independently from how it is carried out – that governs service portfolio planning or provisioning ”
- In service portfolio planning (SPP), some policies are inviolable (must comply) while others are guidelines (recommendations)
- Usually expressed in narrative text, that should make it clear if it is “must” or “recommended”.



Everware-CBDI

Policy Development within SPP



Everware-CBDI

Service Policy Categories

- Tactics
  - Broad strategies which influence process & other policies
  - Triage approach
- Sourcing Policies
  - Permitted sources of supply for each class of service
- Design Policies
  - Layering, dependency rules allowed
  - Implementation approaches
- Commercial Policies – govern buying or selling from 3rd parties
- Default Standards to be used
  - WS-* standards, industry standards
- Service Life Cycle Policies
  - Policies for Publishing, Certification, and Asset Management and Change Control
  - Operating and Management Policies (for production services)
  - Portfolio Plan and Service Specification Approval
- Default Quality-of-Service expectation
  - Extended or possibly overridden for particular domains, services or operations

Tactics Drive Policies & Planning (1)

- These are the broad strategies and objectives for SOA, which influence the more detailed policies and the way work proceeds
- These need to be called out — selected by stakeholders and made known to SOA participants

1. Principal Role: Consumer Only| Consumer + build for own use| Service Supplier for profit | Consumer & Supplier
2. Consumption Scope: In-house services only| + from trusted partners| + external services, read-only| + external services, updating
3. Supply Scope: Own use only | + Partners | + Customers | + Public | For Sale, you host | For Sale, buyer hosts
4. Planning Scope: Full Enterprise | Full with Exclusions | Autonomous Division | Department | Selected Areas only | Development Project Focused
5. Planning Sequence: Full Scope, in Advance of Provisioning | By Domain, In Advance | Provision & Plan concurrently

Tactics Drive Policies & Planning (2)

6. Dominant Provisioning Sequence: Services in Advance of Solutions| Services as Solutions Require them
7. Dominant Provisioning Source: Build From Scratch| Adapt Legacy| Acquire from (ERP) Vendors| Use external sources
8. Current Commitment is from: Business | Mainly IT | Mainly a Project. Who is sponsoring SOA?
9. Committed to What?: to SOA | to Web Services| to some other technology (which ? ……..) ; to software reuse | to systems integration | to faster delivery & modification | lower costs

Everware-CBDI

Tactics Drive Policies & Planning (3)

10. Realization technology: Web Services | XML messages | message-ware | CORBA | component technology | <some other> | combination of
11. ESB Design Sequence: Before Portfolio Planning | After Planning | in parallel with SPP
12. Solution Delivery Practice: Rollout new methodology |expand existing methods| no new methods needed
13. Development Organization: Reorganize for SOA (new teams)| Modify for SOA (new roles)| Use Existing Structure, same roles
14. Triage (priority & selection) Approach: Existing Dev. Plan Driven| Critical-Business Issue Driven | Core- focus, context-standardized

Everware-CBDI

Sourcing Choices

- Different ways to acquire a service:
  - Pre-existing, commercially available
  - Pre-existing offered by partners
  - Built for you by (trusted) partner
  - Acquire by assembling existing services
  - Legacy system functions — wrap as service
  - Software component offers required service
  - Software Package offers required services
  - Adopt industry spec, outsource building
  - Specify in-house, outsource build
  - Specify in-house, outsource build and deploy
  - Specify & build in-house
  - Specify & build for partner usage
  - Specify & build for 3rd party usage

'buy'

'borrow'

'base-on'

'build'

Everware-CBDI

Sourcing Policy & Commoditization Level (1)

NON-DIFFERENTIATING FUNCTIONS DIFFERENTIATING FUNCTIONS

- ***Services classified by these commoditization levels:***
- **Innovative**
  - Functions which our company has invented, which is expected to give it significant competitive advantage, helping to raise its stock price
  - Need the ability to change these functions, and deploy variants which fit new products, markets, channels, etc
- **Specialty**
  - Functions which our company performs in a unique or highly competitive manner, that differentiate our company, and clearly contribute to its stock price
  - Relatively stable, but need full control and the ability to change
  - Not available from package vendors, though development might be outsourced
- **Standardized**
  - Non-unique business functions which our company must perform as efficiently as competitors, but no better. Aim to standardize these functions across our business. Not confident enough to outsource however – since no trusted suppliers known, or internal control remains highly desirable.
  - A “cost of market participation”— a.k.a. “treadmill service”.
  - Might be available in packages from enterprise application vendors
- **Commodity**
  - Well-known business functions which our company must perform as efficiently as competitors, but no better. Possible to pay other companies to perform this on our behalf, without detriment to our business. In fact, the suppliers can probably perform it for lower cost or more efficiently than our own company.

Everware-CBDI

Sourcing Policy & Commoditization Level (2)

SERVICE SOURCE:	innovative	specialty	standard	commodity
Pre-existing, commercially available				
Pre-existing offered by partners				
Built for you by (trusted) partners				
Acquire by Assembling Existing Services				
Legacy system functions, wrapped as services				
Software component offering required service				
Software Package offering required service				
Adopt industry spec, outsource building				
Specify in-house, outsource build				
Specify in-house, outsource build and deployment				
Specify & build in-house				
Specify & build for partner usage				
Specify & build for 3rd party usage				

Architecture / Design Policies

Skip these for the moment

Commercial Policies: Govern …

- Purchase of services from third parties or trusted partners
- Use of services supplied (free) by trusted partners or third parties
- Sales of services to third parties or partners
- Supply of (free) services to partners or third parties

These policies may govern the definition of specific non-functional requirements

Everware-CBDI

Decide on Default Standards to be Used

- Portfolio Planning needs to be undertaken in context of relevant standards, with a defined position for each

- Define a position on these categories of standard:
  - **Web Service protocols** from OASIS and W3C, including SOAP, WSDL and UDDI, and interoperability Profiles from WS-I

    Liaise with ESB designers
  - **Generalized industry standards** such as ebSOA

    Generalized process patterns for eBusiness potentially providing templates for process and utility services
  - **Vertical industry standards** such as RosettaNET

    Business dictionary

    Industry process models
  - **Chosen supplier models**

    E.g. SAP's Enterprise Service Architecture

- All services should normally comply with a minimum set of "default standards"
- There may be additions & variations by service domain, service or operation

Service Lifecycle Policies

- All the stages of service planning, development and production need to be well managed and coordinated
- Designed procedures are needed, to take the service securely and efficiently through its life cycle
- The procedures may be automated or manual, formal or informal, processes or standards
- In SPP, the planners define policies that the lifecycle procedures must accommodate
- Further projects need to be initiated, to create the lifecycle procedures. These must comply with the policies defined by planners

- ✓ Service Specification Approval Policy
- ✓ Service Certification Policy
- ✓ Service Publishing and Advertising Policy
- ✓ Asset Management for Services
- ✓ Change Control for Services
- ✓ Production Operations Policies
- ✓ Production Monitoring Policies

Everware-CBDI

Certification Policies

- The services to be deployed as a part of the software service portfolio should be “certified” by the service provisioners
- Certification principles, rules, constraints & guidelines may be defined by planners
- For example: certification requires
  - Service tests results to be examined by certifier
  - Service to be exercised by the certifier
    - for example, a simple execution of each operation
  - Delivery documentation is complete and appears accurate
  - Certification applies to all services, whether 3rd party external, or built & deployed in house
- Certification procedures must be subsequently devised, which conform to these policies

Everware-CBDI

Publishing & Asset Management Policy

- The services to be deployed as a part of the Portfolio should be “cataloged” by the service provisioner
  - UDDI standard for this: especially for externally-available services
- Cataloguing principles, rules & guidelines are defined in the Policy
- Cataloguing procedures must be subsequently devised, and tools acquired, which support these policies
- For example: the Service Catalog must contain:
  - Overview of all services for a domain
  - High-level business-oriented “advert” for each service
  - Service properties
  - Rich service specification
  - Agreed QoS
  - and may contain

    Example usage
    Test results
    SLA
- The automation units and service specs. are valuable company assets
  - Need to be managed by an asset management system
  - Consider adopting Reusable Asset Specification standard & buying tool
- Services available to external consumers need to publish commercial terms

Everware-CBDI

Other Lifecycle Policies

- Change Control Policy
  - Change control procedures are needed, to properly manage the correction and improvement of production services
  - The policy document defines general rules and requirements that must be covered by the change control procedures
- Operating Policy
  - High-level Requirements for Management/Monitoring Services
  - High-level Requirements for Production Operators
- Portfolio Plan Approval Policy
  - E.g. Each new release must be “signed off” by SOA Sponsor, relevant Domain Owner, the IT director
- Service Specification Approval Policy
  - E.g. Each new release of specification should be approved by Domain Owner and Planning team

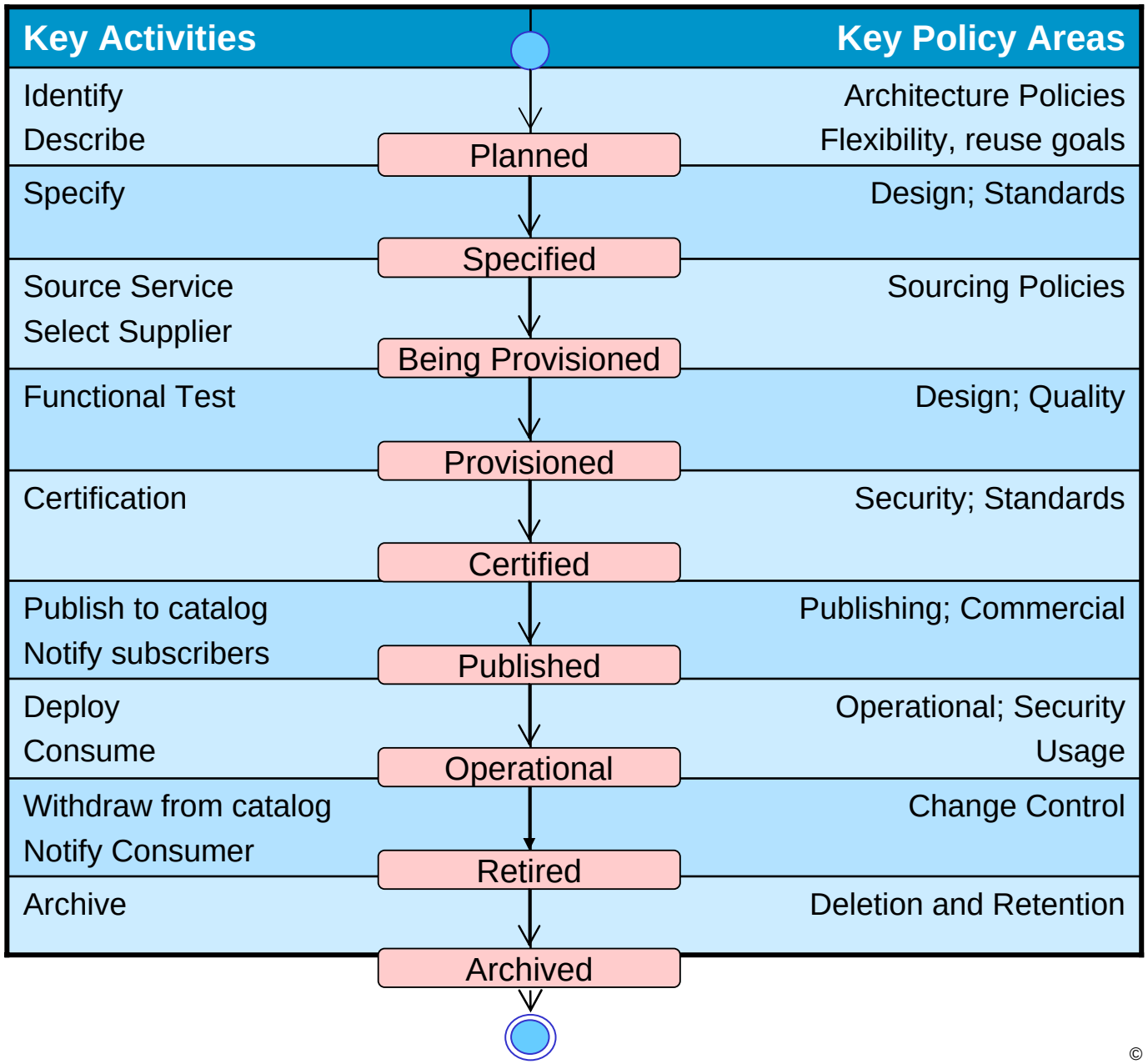
Quality-of-Service Policies

- The idea is:
  - Define set of ‘default’ quality targets that can reasonably apply to all services in the Service Portfolio
  - For a specific service domain, these targets may be made more stringent
  - The planners may authorize:
    - an individual service to override a target quality
    - an individual operation override a target quality
- Hence the planners define common quality targets that should work much of the time
  - Don’t over-demand
  - Only set essential requirements
- In some organizations, might get decided by ESB architects

Establish Default Targets for Selected Quality Category

CATEGORY ————————- ref nr.	Requirement
RELIABILITY	
Default.1	Service must have 98% availability per 24 hour period, averaged over one month
Default.2	Service downtime must never exceed 1 hour
Default.3	A service must not crash more than twice in one calendar year
Default.4	Services must leave all data storage with full integrity, and consistent with one another: compensation services must be supplied in cases where platform cannot deliver full integrity.
Default.5	Alternative endpoint for service must be available for services designated as “business critical” by domain owner.
PERFORMANCE	
Default.10	Synchronous operations must respond in < 0.4 seconds
SECURITY	
Default.20	Passwords of applications using this web service must be changed monthly
REGULATORY	
Default.91	Online credit card sales may only be sent to card billing address.

Policies and the Service Life Cycle



Everware-CBDI

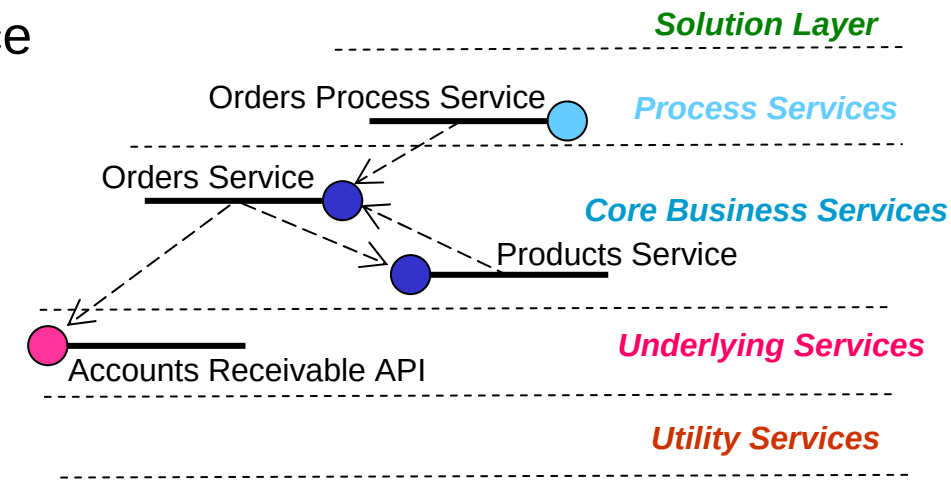
Independent Insight for
Service Oriented Practice

Architecture & Design Policies

www.everware-cbdi.com

Portfolio Architecture / Design Policies

- Design Policies are the directives and guidelines which govern:
  - Structure of the Portfolio content
  - Rules & Guidelines for Service Design

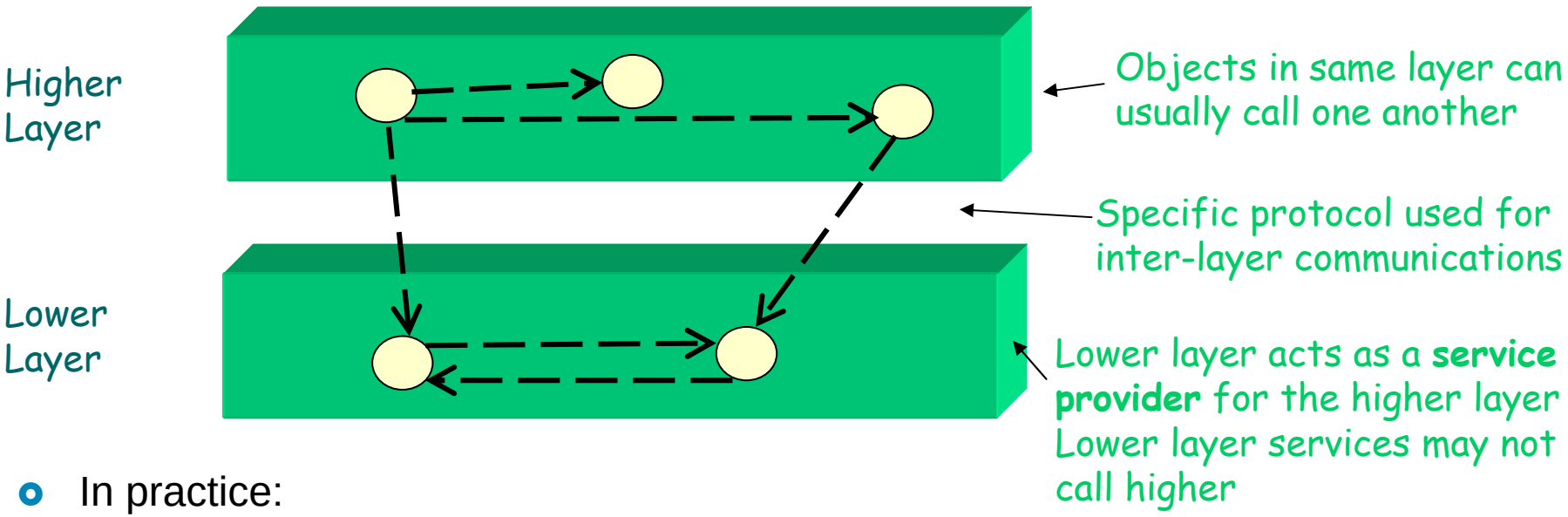


- Design Policy Categories:
  - Layering Policy
  - Dependency Rules
  - Service Filter
  - Encapsulation Policy
  - Service Style Policy
  - Deletion, Identification and RI rules
  - Service Usage Patterns
  - Documentation

Everware-CBDI

Layering

- "Subdivision of a system into ordered layers of processing
  - where objects in one layer can only communicate with their peers, or with the next layer down"
- Building software in layers makes it easier to maintain
  - Breaks up monoliths
  - Role of each software object is limited, but clearer
  - Lasagna instead of Spaghetti



- In practice:
  - the intra-later and inter-layer rules might vary by layer
  - 'upward' calls allowed for special circumstances

Service Layers

- CBDI proposes six basic layers:
  - Solution layer (consumes services, does not offer any services)
  - Process Service layer (value chain rules, coordinates other services)
  - Core Business Service layer (reusable enterprise services, maintain data)
  - Underlying Service layer (services that need a façade)
  - Utility Service layer (commonly needed logic, highly shareable)
  - Infrastructure Service layer (technical services, outside our scope)

- A service that has operations for >1 layer must be broken up

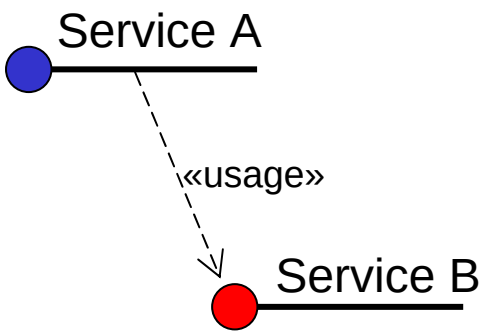
- Why do this?
  - A useful classification scheme
  - Dependency rules may vary by layer
  - A way to achieve
    - improved maintainability
    - higher degrees of reuse / sharing
    - specialization of labor
    - standardized lower layers
    - customizable higher layers

Service Layers Defined

Layer: main role	Operations are:	Dependencies allowed:	More rules:	Data Storage:
Solution Logic: UI, channel and dialog-specific processing	Non-existent - this layer does not contain services	May call Process, Core Business & Utility services directly	Supplies user interface, validation, user messages, session management	Not normally, except for temporary session data
Process Services: orchestrate other services; apply business process-specific rules	UI-independent, but designed for a specific business process	May call Core Business & Utility services directly	May be called by solutions that support other business processes	Only where not stored by Core Business Services. Stored data likely more transient than for core services.
Core Business Services: apply enterprise-wide business rules	UI- and business process-independent, so can be used in different contexts	May directly call other Core Business, Underlying and Utility Services	Cyclic dependencies not normally permitted, except for “call-back”. May not call Process Services	Normally maintains principal data stores, unless this delegated to Underlying Services
Underlying Services: apply the business rules, but not exposed as well-formed service	Highly generic or implementation-based, so its interface not ideal for exposing to solution developers	May call Utility Services, but normally would not	May not call Core Business or Process Services	Often maintains significant data stores, and core services call underlying services to access this data & its processing rules.
Utility Services: shared by Core Business Services	UI-, business process- and often domain-independent	May call other Utility Services directly. Some may use Underlying Services	Cyclic dependencies not normally permitted.	Often required — for directories, look up tables and logs for example.

Service Dependency Defined

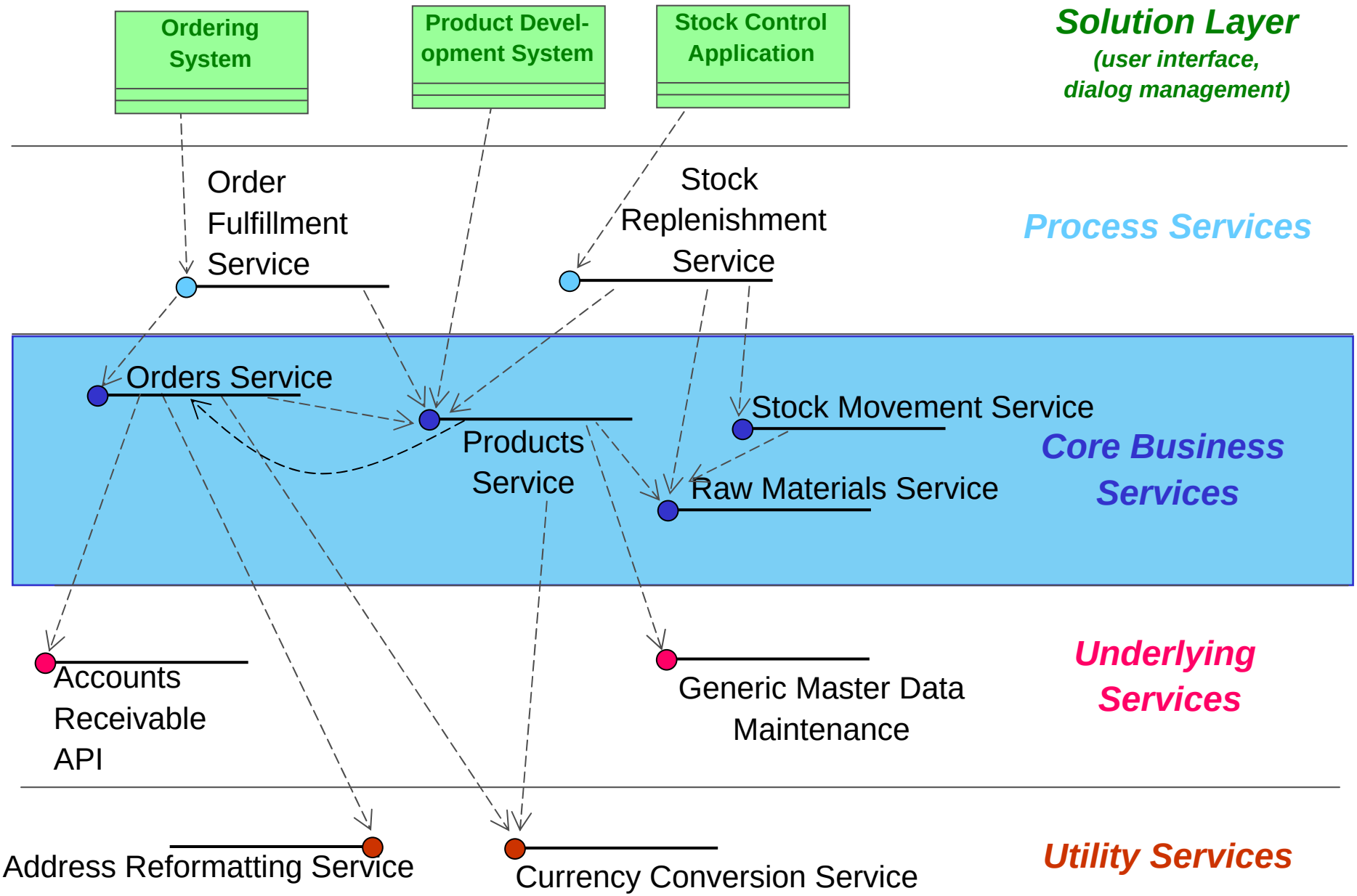
- Service A depends on Service B if service A cannot fully function without Service B being available
- We are primarily concerned with “usage” dependencies



- Means service A requests service B to perform one of its operations
- Other kinds of non-usage dependency, not usually depicted:
  - A creates business type instances which must be validated against B (invariant dependency)
  - A cannot function without B creating business type instances first (for A, not for itself) (creation dependency)
  - A cannot function without directly executing some code, or accessing some data, belonging to service B (implementation dependency)

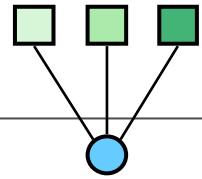
Everware-CBDI

Example of a Service View



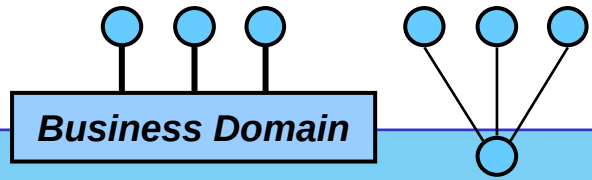
Basis for Single and Shared Service Policy

Solution Layer
(presentation and dialog)



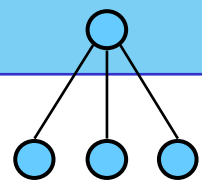
Orchestrate operations from many core business operations
Support process unique processing
Store process level information

Process Services
(orchestration layer)



Single Service provides consistent view of corporate data and business rules
Provides a 360° view of the resource
Stores a record of each instance of each business type
Applies common validation and business rules

Core Business Services
("backbone" layer)



Exploit pre-existing functionality for wider reuse
Aggregate functionality from pre-existing Services and systems

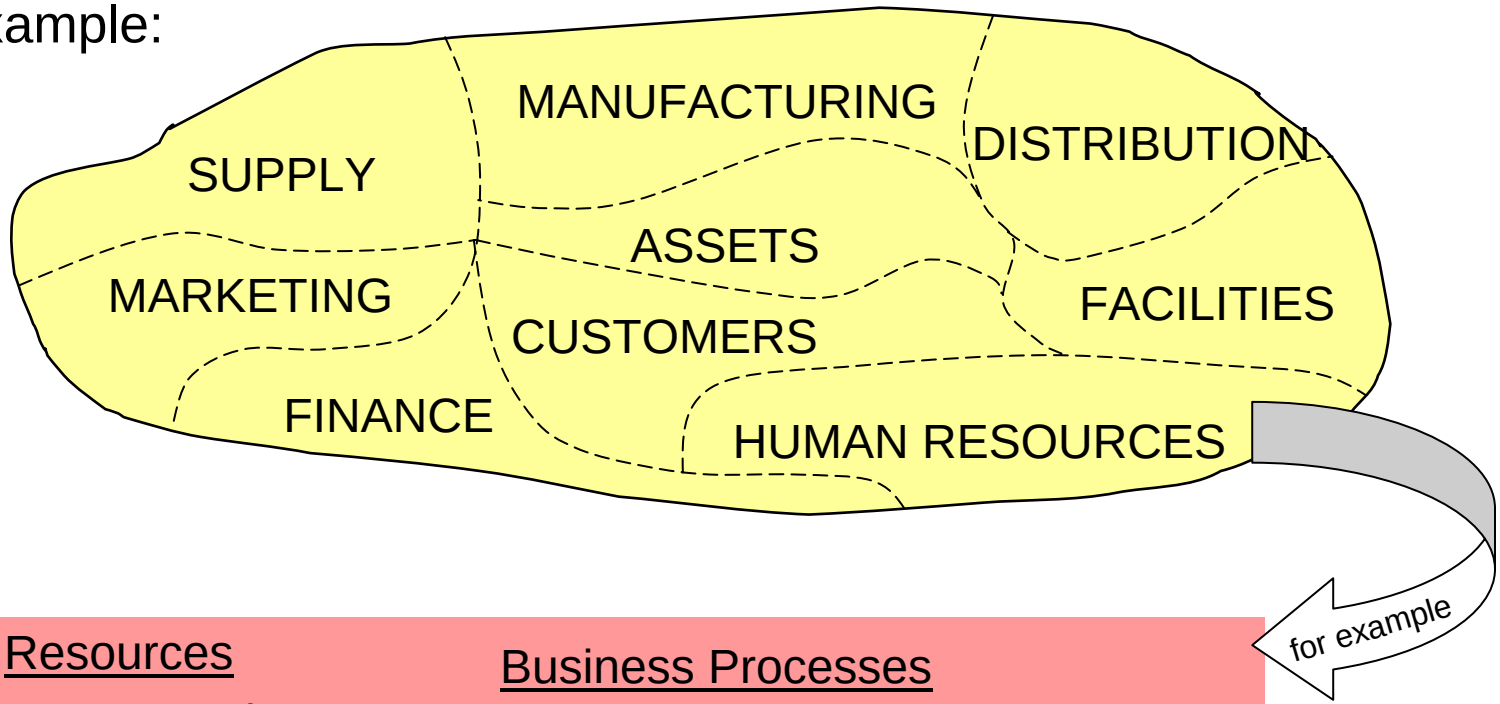
Underlying Services
(that need a facade)

The most widely reused ***Shared Services***
Services that perform widely used sub-routines, operations

Utility Services
(high reuse layer)

Business Domain

- *A major logical partition of an enterprise consisting of a set of associated business resources plus the business processes which act upon those resources*
- Example:



Resources

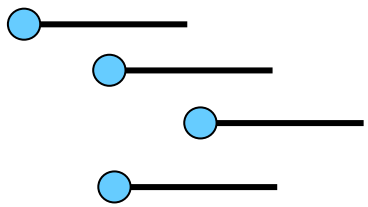
- Employee
- Job
- Organization Unit
- Pension Fund

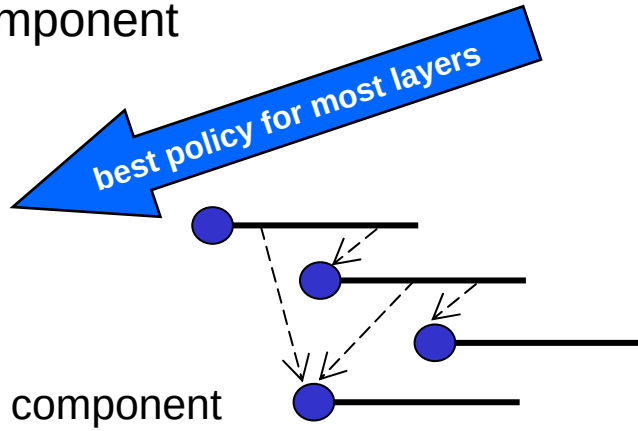
Business Processes

- Recruit Employee
- Retain Employee
- Change Organization
- Administer Pension Fund

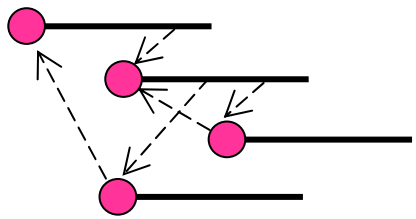
Service Dependency Rules: May vary by Layer

- Independent Services
  - No access to one another's code or data
  - Do not call one another
  - Reusable in other contexts, without one another
  - Each can be implemented by an encapsulated component
  - Consuming logic has to tie data together





- Acyclic Dependent Services (1-way dependency)
  - Services call one another's operations:
    - but not so cyclic dependencies occur
    - Services can be implemented by an encapsulated component
    - Operations can be more powerful than for independent services



- Cyclic Dependent Services (2-way & circular dependency)
  - least code, most power
  - harder to test and maintain
  - some say "don't do it"

Everware-CBDI

A Service "Filter"

- Define "tests" which help you decide whether functionality should be service-ized, or not. For example:

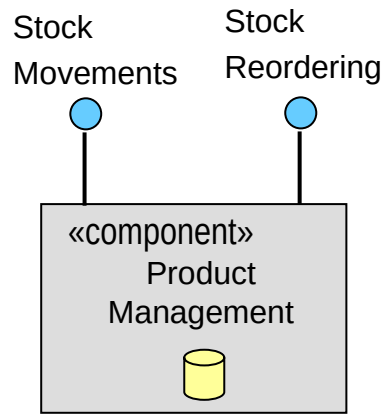
- The service must support at least 2 of the following, else it is not provisioned:

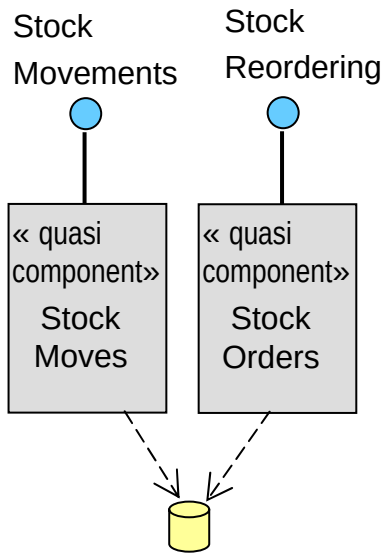
  - it offers operations that can be usefully shared within more than business process or application or UI device
  - it offers operations that can be usefully assembled into higher level services
  - it provides programmatic access to operations that we want customers or trading partners to use (B2B, B2C support)
  - it brings together information from several sources providing a 360° view of a major business resource
  - it enables existing system functionality to be accessed by other systems, enabling applications to be integrated
  - it enforces standardized processing, across the enterprise
  - it provides access to high-value third party functions not available in-house (commodity services, or specialized functions)

- On the other hand, your policy may be that all application software except the "solution layer" is organized into services

Encapsulation Policy

- ***Encapsulated*** **is a property of the service's automation unit**
- **One or several services could be realized by one automation unit that is an encapsulated component**
  - **Means**
    - **logic and data only accessible through service interfaces**
    - **logic and data storage can be changed with no (functional) impact to consumer, nor any other service**
    - **cyclic dependencies might be permitted between services realized by one component**
    - **fragmented data storage is hard work**
- **One or several services could be realized by an automation unit that is a quasi-component**
  - **Means**
    - **logic only accessible through service interface; data accessed by other services and/or non-service software**
    - **automation unit could be replaced by a different implementation**
    - **data storage replacement could impact several services or non-service software**
    - **could aim for an encapsulation boundary round a cluster of quasi-components**
    - **cyclic dependencies might be permitted within a cluster**





Deletion, Identification, and RI Policies

- Instance Deletion Policies
  - Physical deletion vs. logical?
  - Clearing "logical deletes"
  - Clearing accidental creates

- Instance Identification Policies
  - Instances that one service is responsible for, may need to maintain references to the instances that another service is responsible for
  - What is the content and format of these references?
    - Use existing Business Identifiers, as known to business?
      - Not always immutable, and may have structure
      - May not be useable when extra instances to be referenced (mergers, in other packaged products)
      - Hinder provision of generalized services (which can reference any type)
    - Use artificial internal identifiers for references?
      - Decide if required
      - Decide format
      - May not be suitable for references maintained within commodity or external services

- Referential Integrity Policy
  - How is referential integrity maintained for references to instances which are the responsibility of another service?
    - Not an issue when all services share a monolithic database
    - But is when an instance is (to be) deleted, and it is referenced from elsewhere (& not integrated data storage)

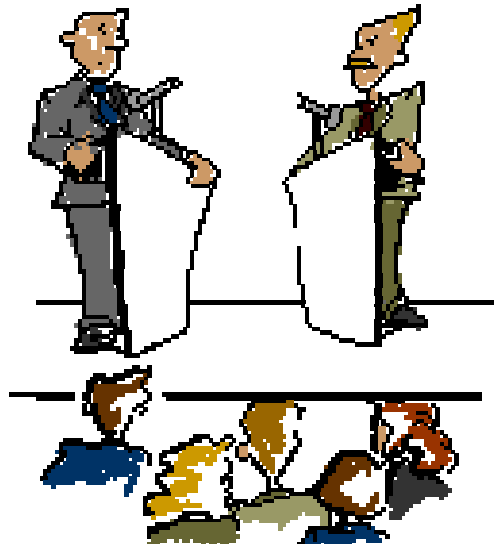
Other Design Policies

- Service Style Policies
  - Synchronous | Asynchronous | both styles,
  - RPC (Parameter) or Document Style,
  - Transactional Behavior,
  - Etc.
  - *These design policies probably get decided by the SOA platform architects, and may already form part of the ESB definition.*
  - Note: Some topics covered by other policies
    - Service Security (session 4 and ESB Design)
    - WS-Standards to be used (session 4)
    - Runtime Service Management
- Service Usage Patterns
  - Define patterns of recommended multi-operation sequences
  - Express using UML sequence diagrams
  - May require further WS-Standards to be adopted
  - Example 1: Asynchronous Operation “callback”
    - Consumer must provide the service & operation dictated by provider, to receive feedback from asynch message (Alternative: call again for result)
    - Adopt WS-Addressing to synchronize callback message?
- Service Documentation Policies

  Standards typically needed for:
    - Naming Services, Operations and Automation Units
    - Specifying Services
    - Entries in Service Catalogue or Directory
    - Etc.

Audience for Architecture / Design Policies

- Service Portfolio Planners (definitely)
- Service Provisioners (definitely)
- Service Implementers (definitely)
- Service Testers (possibly)
- Serviced Consumers (possibly)



- Documentation standards decided will impact all these roles, since they all need to use the service documentation

Everware-CBDI

Architecture and Design Policies Summary

- Architectural Policies govern Portfolio Content and Structure
- The main policies in this category concern:
  - Service Layering
  - Service Dependency Rules
  - Filtering proposed services

- Design Policies standardize certain service design features:
  - Deletion, Identifiers and Referential Integrity
  - Automation Unit Design
  - Service Usage Patterns
  - Service Style
  - Documentation Standards

- We gave recommendations for some policies
  - Each company will need to establish its preferred policies
  - Policies can be adjusted as further SOA experience gained

Everware-CBDI

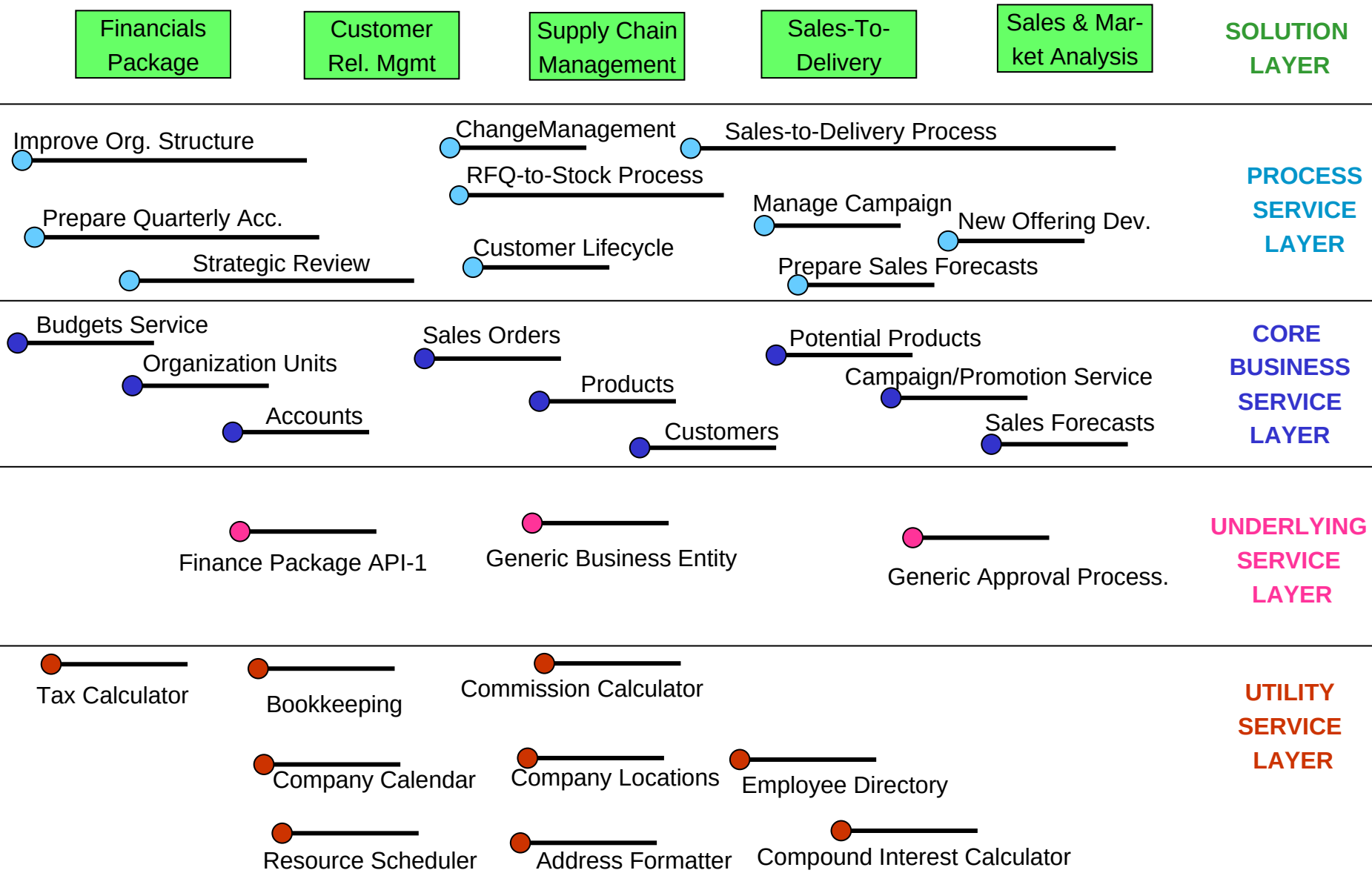
Everware-CBDI

Independent Insight for
Service Oriented Practice

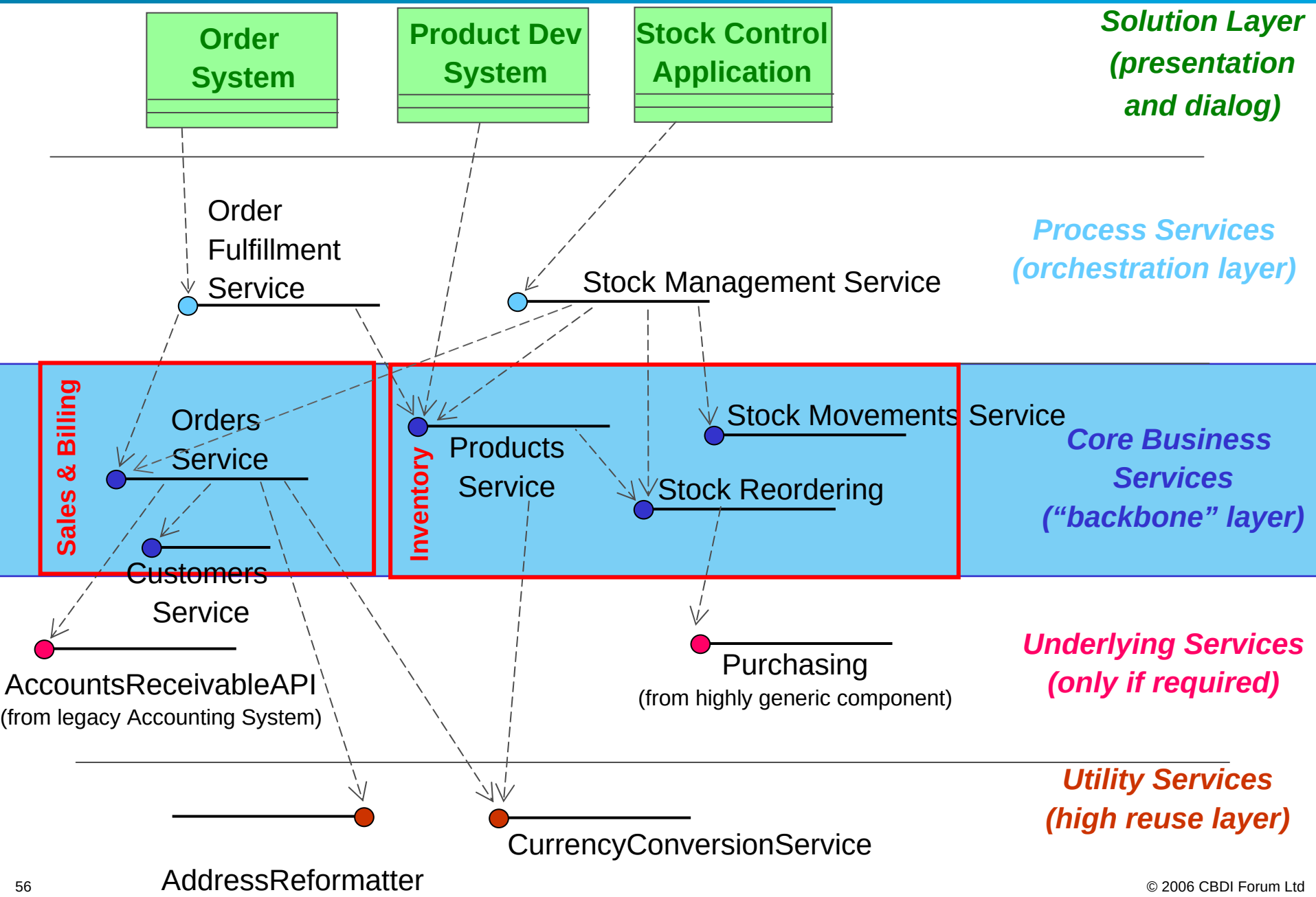
Policy Impact on Service Identification

www.everware-cbdi.com

Layering Affects Identification of Services

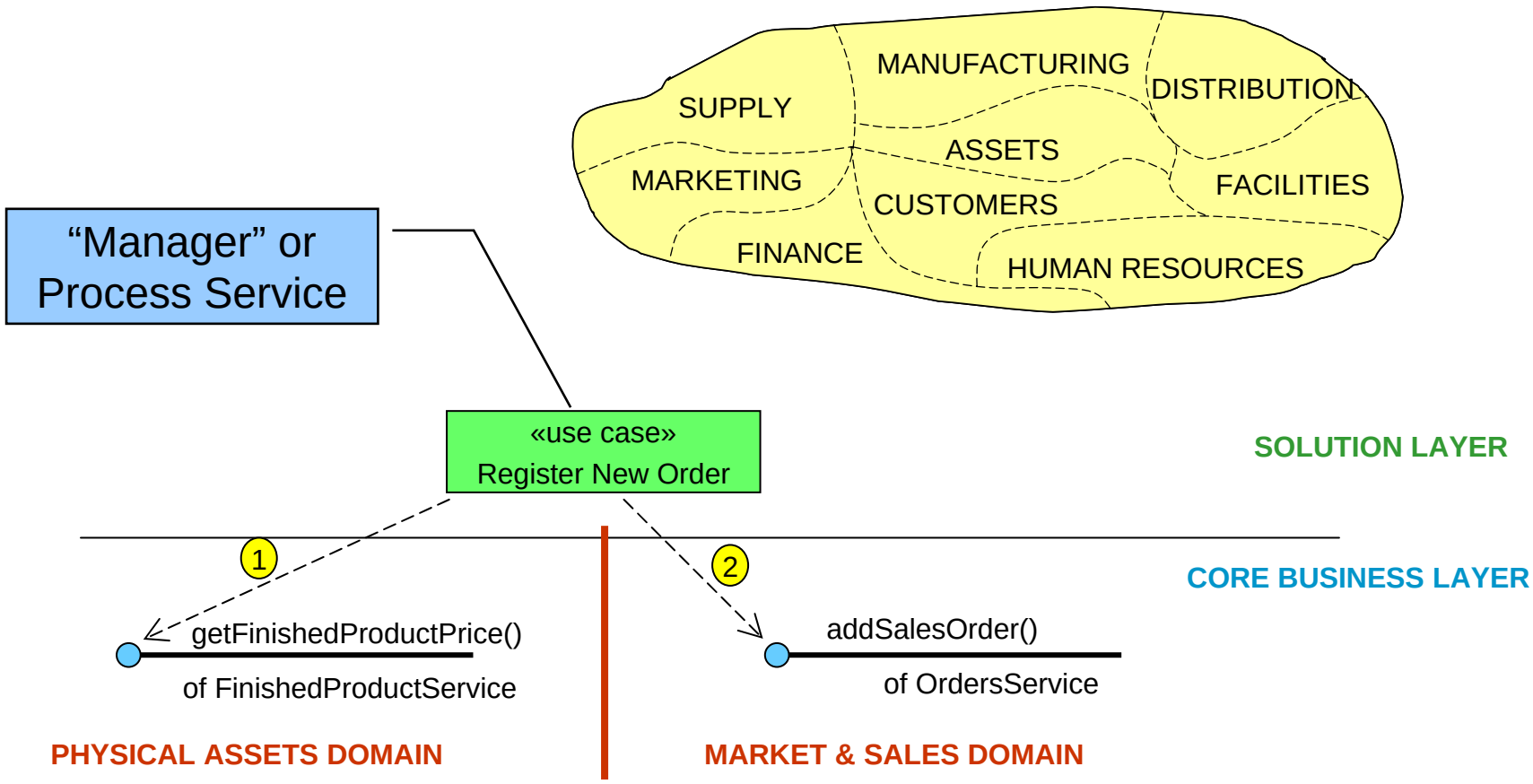


Services Organized into Layers and Domains



Inter-Domain Interactions

- May need to create new services to manage interaction between Domains



A Service "Filter"

- Define "tests" which help you decide whether functionality should be service-ized, or not. For example:

- The service must support at least 2 of the following, else it is not provisioned:

  - it offers operations that can be usefully shared within more than business process or application or UI device
  - it offers operations that can be usefully assembled into higher level services
  - it provides programmatic access to operations that we want customers or trading partners to use (B2B, B2C support)
  - it brings together information from several sources providing a 360° view of a major business resource
  - it enables existing system functionality to be accessed by other systems, enabling applications to be integrated
  - it enforces standardized processing, across the enterprise
  - it provides access to high-value third party functions not available in-house (commodity services, or specialized functions)

- On the other hand, your policy may be that all application software except the "solution layer" is organized into services

Other Policy Impacts on Identification

- Encapsulation
  - Some services will be hidden within a domain or layer
- Customization/Generalization
  - Are customized services offered?
  - Or is one coarse-grained service offered?
- Other Service Reuse
  - COTS Services
  - Outsourced Services
  - Etc.

Everware-CBDI

Everware-CBDI

Independent Insight for
Service Oriented Practice

Using Policies in Service Portfolio Planning

www.everware-cbdi.com

City Planning Metaphor

- Town planners set Policies
  - Rules and Regulations
- Policies establish standards
  - How individual buildings interface to the city's infrastructure
  - How parts interoperate and fit together
- Balance of compliance with regulations and individual freedom
  - Optional vs Mandatory vs Advisory

Everware-CBDI

City Planning Analogy

		City Planning	SOA
Town Planning Policies	How is the problem carved up?	Zoning	Domain Identification and ownership; RAEW
Infrastructure Policies	What infrastructure should be shared?	Roads, Utilities	ESB, Security, Management infrastructure
Architectural Policies	What styles apply?	Conformance to “style” Componentization	Standardization/Differentiation Layering; Flexibility
Standards Policies	How do the parts fit together?	Interfaces to infrastructure Standard Building parts	Interfaces to infrastructure Service Standards
Quality Policies	How do we ensure it is fit for purpose?	Building Regulations	Specification quality Operational quality
Safety Policies	How do protect users?	Firewalls	Security
Commercial Policies	How is it paid for?	Rates, Taxes	Metered usage
Policy Setting	Who sets policies? What is the policy setting process?	Town Planners	Roadmap Managers Service Portfolio Planners Enterprise Architects
Compliance Monitoring	Who monitors compliance?	Building Inspectors Fire Chief	Run-time Infrastructure Enterprise Architects
Accountability	Who are the planners responsible to?	Tax Payer	Business

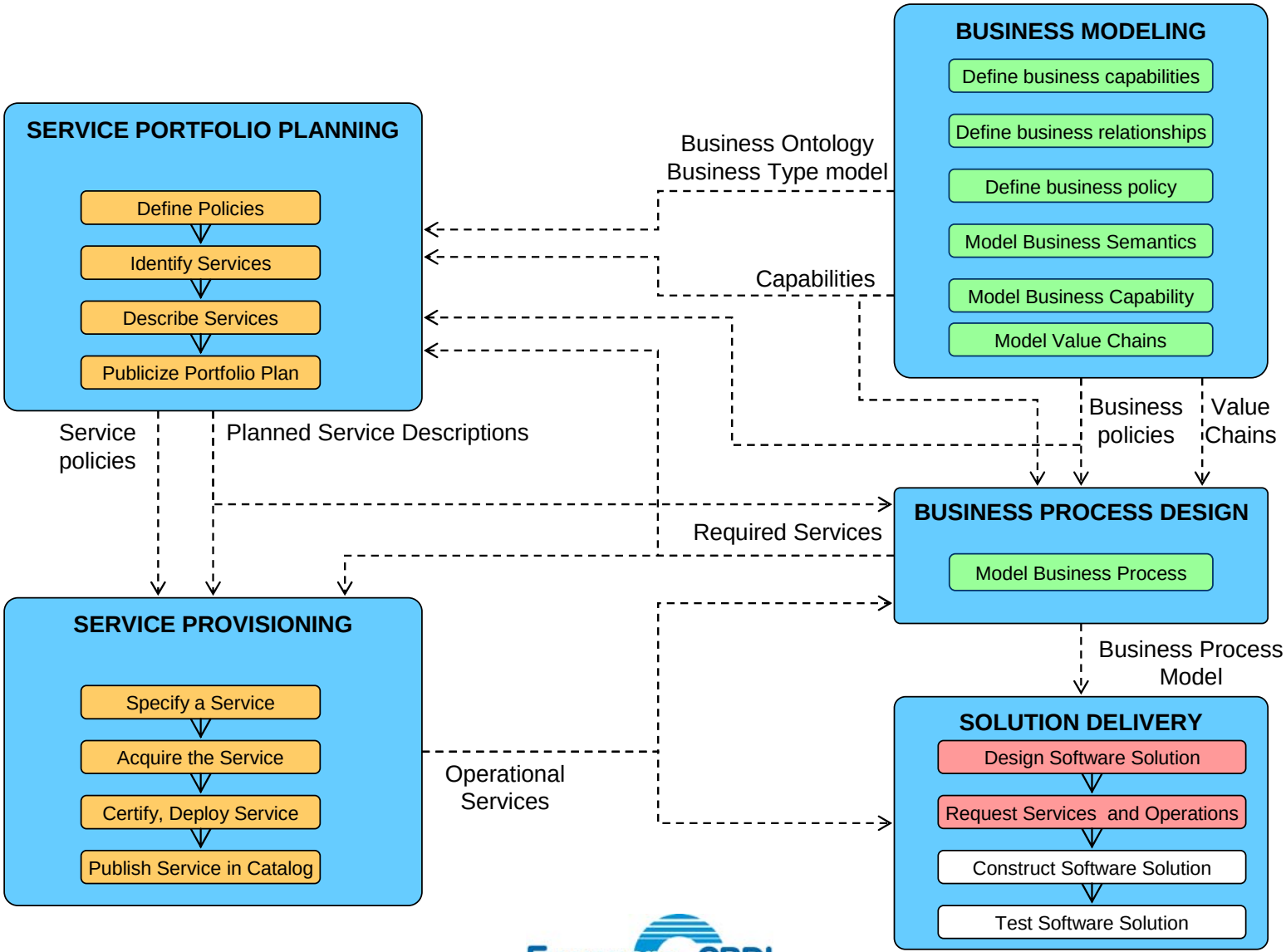
SOA Planning Challenges

		Challenge
Town Planning Policies	How is the problem carved up?	Agreeing and delegating ownership of domains and Services that span business units and current organizational structure
Infrastructure Policies	What infrastructure should be shared?	Ensuring loose coupling of the infrastructure
Architectural Policies	What styles apply?	Allowing differentiation and adaptation whist retaining control
Standards policies	How do the parts fit together?	Choosing standards Enforcing compliance
Quality Policies	How do we ensure it is fit for purpose?	Clear statement of purpose
Safety Policies	How do protect users?	
Commercial Policies	How is it paid for?	How is ROI calculated (e.g. flexibility) funding decisions – e.g, Funding of long-term investments
Policy Setting	Who sets policies?	Business unit autonomy
Compliance Monitoring	Who monitors compliance?	Man vs Machine
Accountability	Who are the planners responsible to?	Ownership Different goals of individual stakeholders

SOA Governance and Policy Areas

Type	Establishes and enforces policies for	Example
Architecture /Design	The design of Services and Service-oriented solutions	
	Use of architectural constructs in the SOA	Layering
	Flexibility	Mediation
Standards	What standards should be complied with	Permitted usage of WS-Protocols
Sourcing	How Services and associated resources are sourced	Sourcing of commodity capability
Asset Lifecycle	Lifecycle policies including both design/implementation lifecycle and runtime/execution lifecycle.	
	Change in state - Service lifecycle	Certification
Operational	Governs Services and Service-oriented solutions at run-time. Enforces run-time policies.	
	Run-time policies	Monitoring; SLA
Quality	Sets Quality of Service requirements and expectations	Reliability
Security	Security levels; Permissions	Authentication
Commercial	How a Service is paid for	Pricing
Relationship	Relationships between different participants in the SOA, and the SOA delivery process	Provider/Consumer
Programme	Organizational; SOA Delivery process; Policy Setting	RAEW

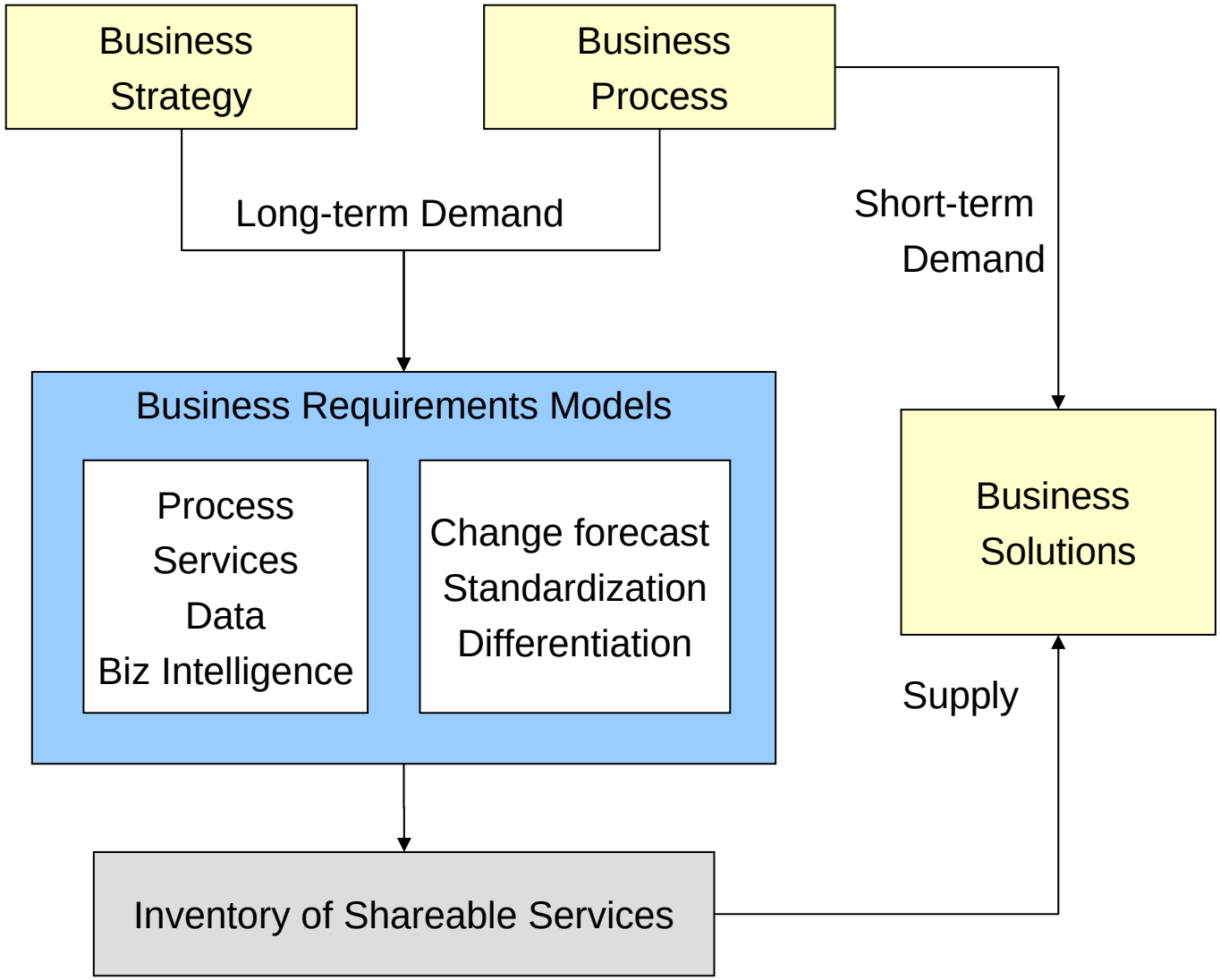
CBDI Service Engineering Process



Everware-CBDI

Supply/Demand

- Investment to create inventory of reusable assets
- Long term and short term processes
- Extended requirements model
  - Change forecast
  - Core
  - Commodity
  - Context
  - Differentiation
- Placing new responsibilities on business



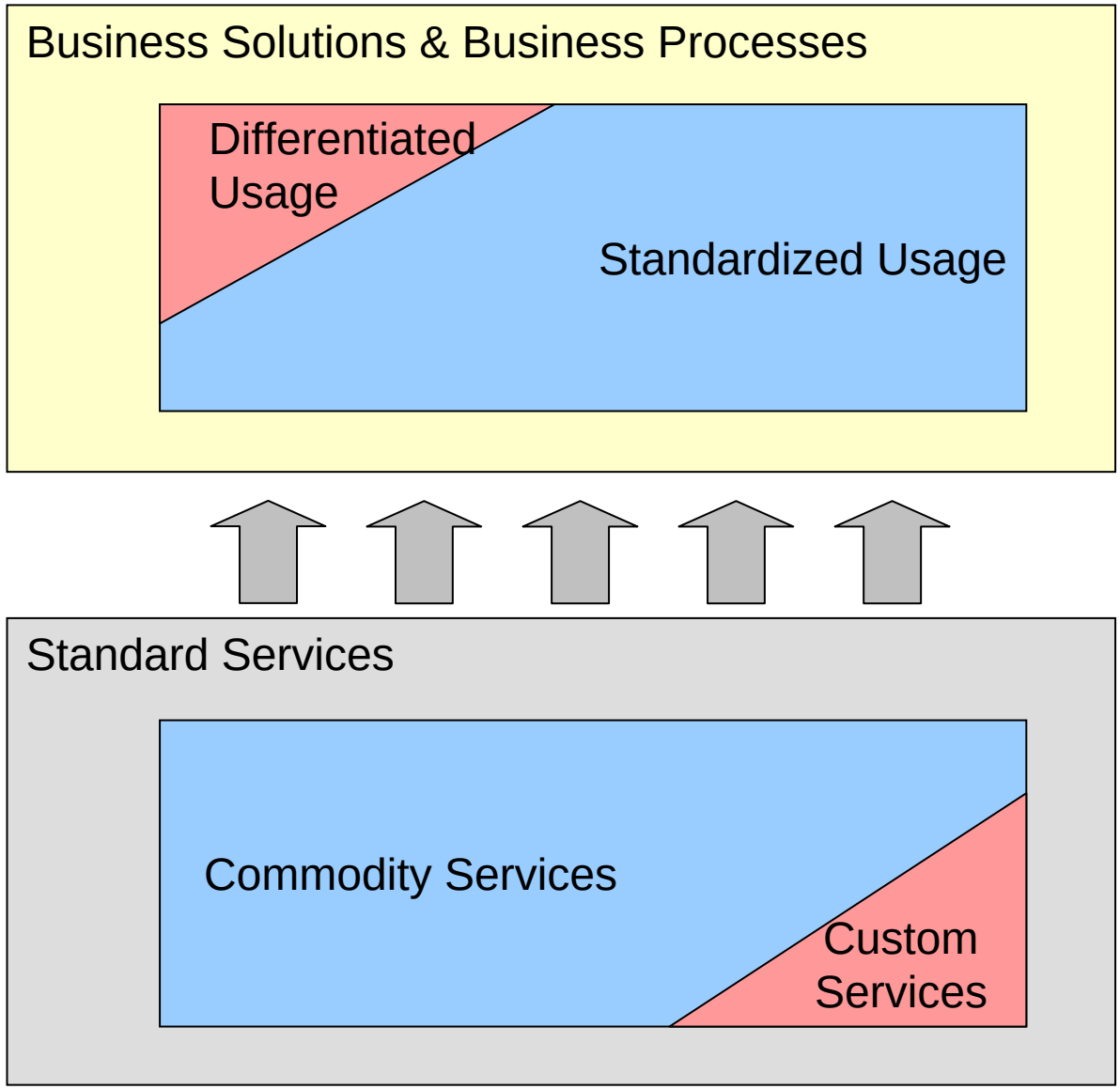
Policy Development

- Establish a Service Portfolio Planning Team, as soon as management give go-ahead to SOA approach
- Ideally, before the planners identify any services & automation units
  - Decide on Business Domains
  - Devise and document Policies that guide the design of the Service Portfolio Plan
  - Establish a set of “default quality requirements”
- In practice, can’t hold up the Portfolio Plan Fragment needed by a current Software Solution project, while waiting for all policies to be finalized
  - Formulate the Policies in parallel with Service Domain Planning
  - Policies will be more practical when based on some real experience
- But, you don’t want to end up with lots of design features and tactics that break policy
  - So policies need to be complete after (say) three service portfolio planning increments

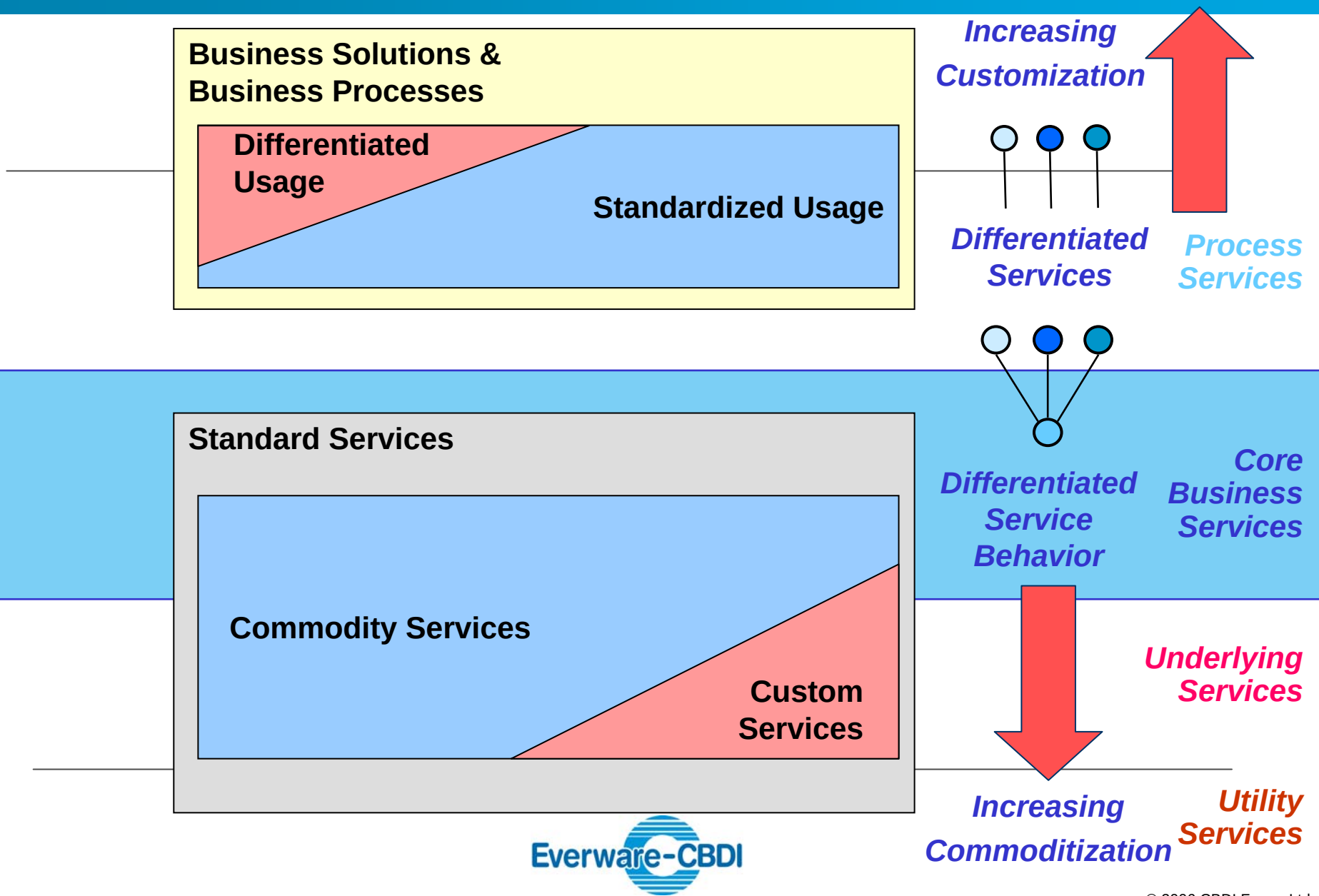
- Planning team delegate service specification & acquisition to Service Provisioning team
- Planning Team should
  - Approve any Portfolio Fragments originated by Solution developers or Provisioners
  - Monitor progress of service specification and acquisition, so plan can be updated

Standardization and Differentiation

- **Critical policy area**
- Determines economics, flexibility, competitive differentiation and standardization
- Determines sets of standard services based on economics and feasibility
- Manage solution usage on basis of competitive differentiation
  - Core/Context
  - Core/Non Core
- Manage sourcing on basis of economics



Basis for Standardization/Customization Policy



Everware-CBDI

Service Portfolio Plan Content

1. Introduction
- Background
- Business Goals for SOA
- Objective & Scope of Plan
- Audience for Plan
- Responsibilities
- Progress To Date

2. Plan Structure
- Overview of Domains
- Domain Definitions

3. Policies
- Tactics
- Sourcing Policy
- Commercial Policies
- Design Policies
- Default Standards Adopted
- Service Life Cycle Policies

Layering, Dependency, Filter
Deletion, Identification and RI
Usage Patterns
Customization Techniques
Implementation Techniques

4. Default Quality Requirements
- Security
- Performance
- Reliability

5.1 Portfolio Plan for <name1> Domain

5.2 Portfolio Plan for <name2> Domain

5.3 Portfolio Plan for <name3> Domain
- Current Progress
- Service View
- Implementation View
- Deployment View

Dependency Diagram
Service Descriptions

Dependency Diagram
Automation Unit Descriptions

6. Development Schedules
- Schedule for Portfolio Plan Releases
- Schedule For Service Provisioning
- Schedule for Solution Development
- Transition Approach & Schedule (switching old software & data to SOA)

Everware-CBDI

Portfolio Planning Summary

- For Portfolio Planning, the Policies provide:
  - transparent decisions
  - consistent standards and service quality
  - better alignment to business objectives and management expectations
  - basis for asset management
  - basis for good governance
- We **did discuss**
  - tactics
  - sourcing choice — may depend upon ‘commoditization level’
  - default standards set
  - default quality targets
  - service lifecycle policies
  - service architecture / design policies
- We have **not discussed**
  - triage (prioritizing what to work on)

Everware-CBDI

Policies and Service Planning Summary

- Transformation of Principles into Practice
- Policy instantiation as practice guidelines, clusters, permitted relationships, classifications and usage requirements
- Policy impacts how services are identified and grouped
- Appropriate Planning Methodology – short/long, stable/dynamic
- Policies provide governance over entire Service Lifecycle

Everware-CBDI

www.everware.com

www.cbdiforum.com

CBDI on SOA Basic Concepts

- SOA Fundamentals
  http://roadmap.cbdiforum.com/reports/fundamentals/
- A consolidated set of CBDI Reports covering basic SOA principles:
  - Understanding SOA
  - Principles of Service Orientation
  - The Business Case for SOA
  - Towards the Service Oriented Organization
  - Enterprise Framework for SOA
  - Establishing a Service Lifecycle
  - Service Supply - A Fundamental Shift in Development Practice?
  - Business Modeling for SOA
  - Composite Applications
  - Save Our Assets
- NB. These materials are public domain

Everware-CBDI