IBM Software Group

Changing Business and Software

James Rumbaugh, IBM Distinguished Engineer

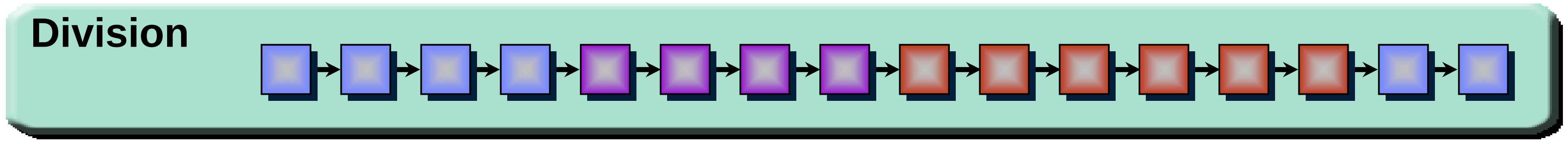
ON DEMAND BUSINESS™

Business is Changing

- Monolithic → distributed
- Tightly controlled → open
- Stable → constantly changing

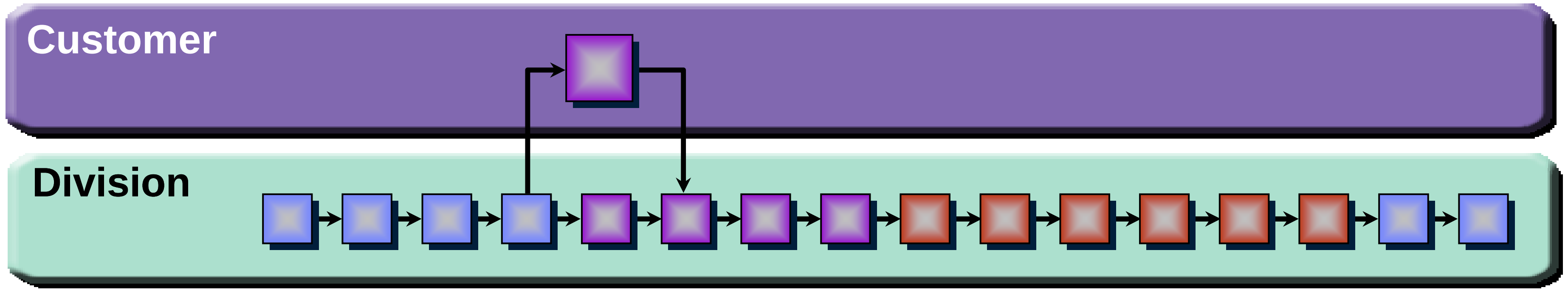
Where We Are Heading

Case Study: Procure to Pay Process



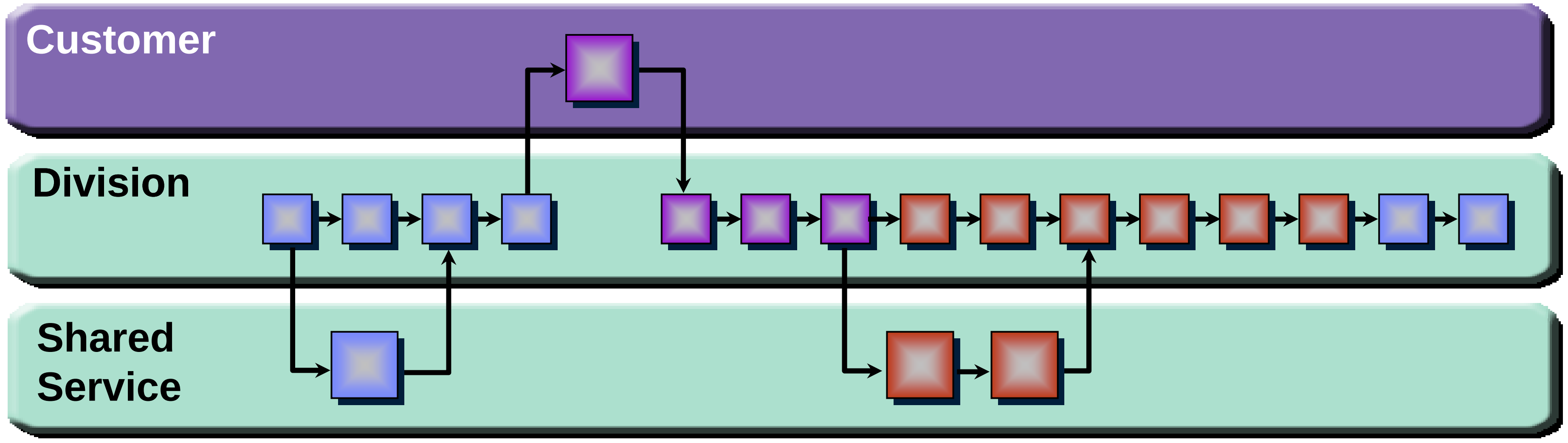
Where We Are Heading

Case Study: Procure to Pay Process



Change: Customer Order Entry

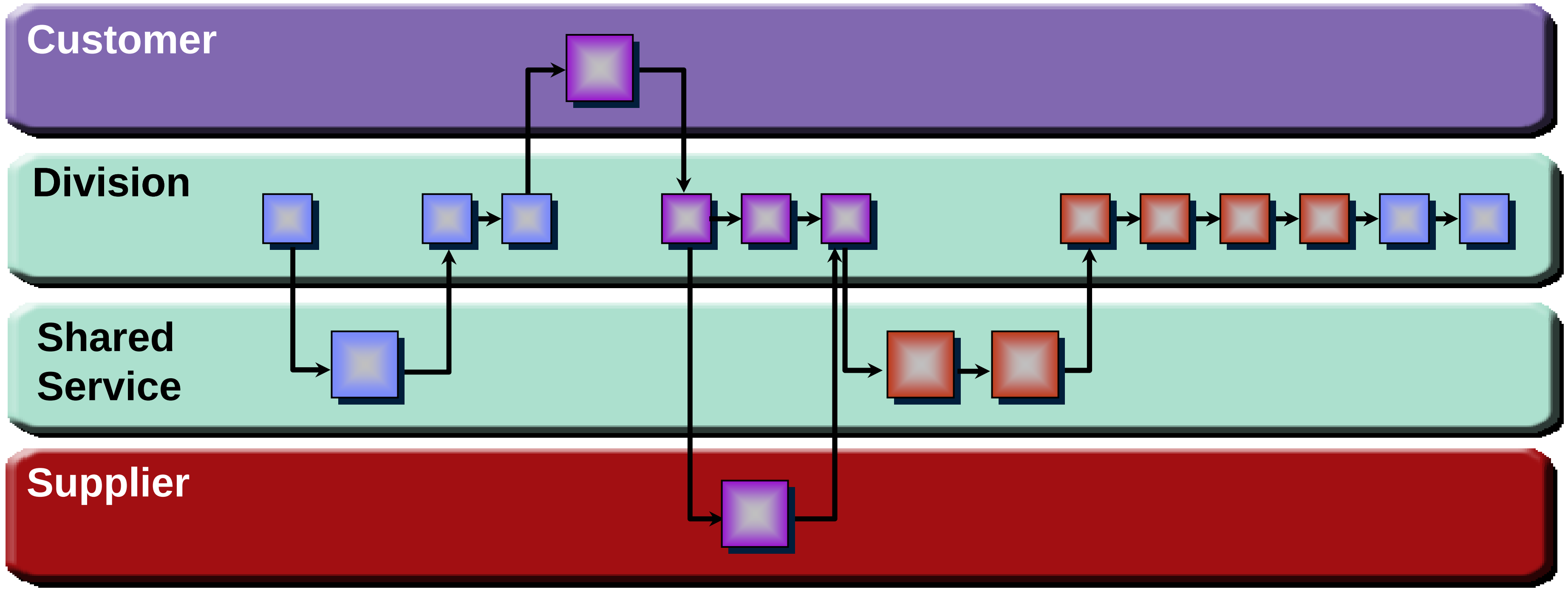
Where We Are Heading
Case Study: Procure to Pay Process



Change: Shared Service – Marketing, Billing, Receivables

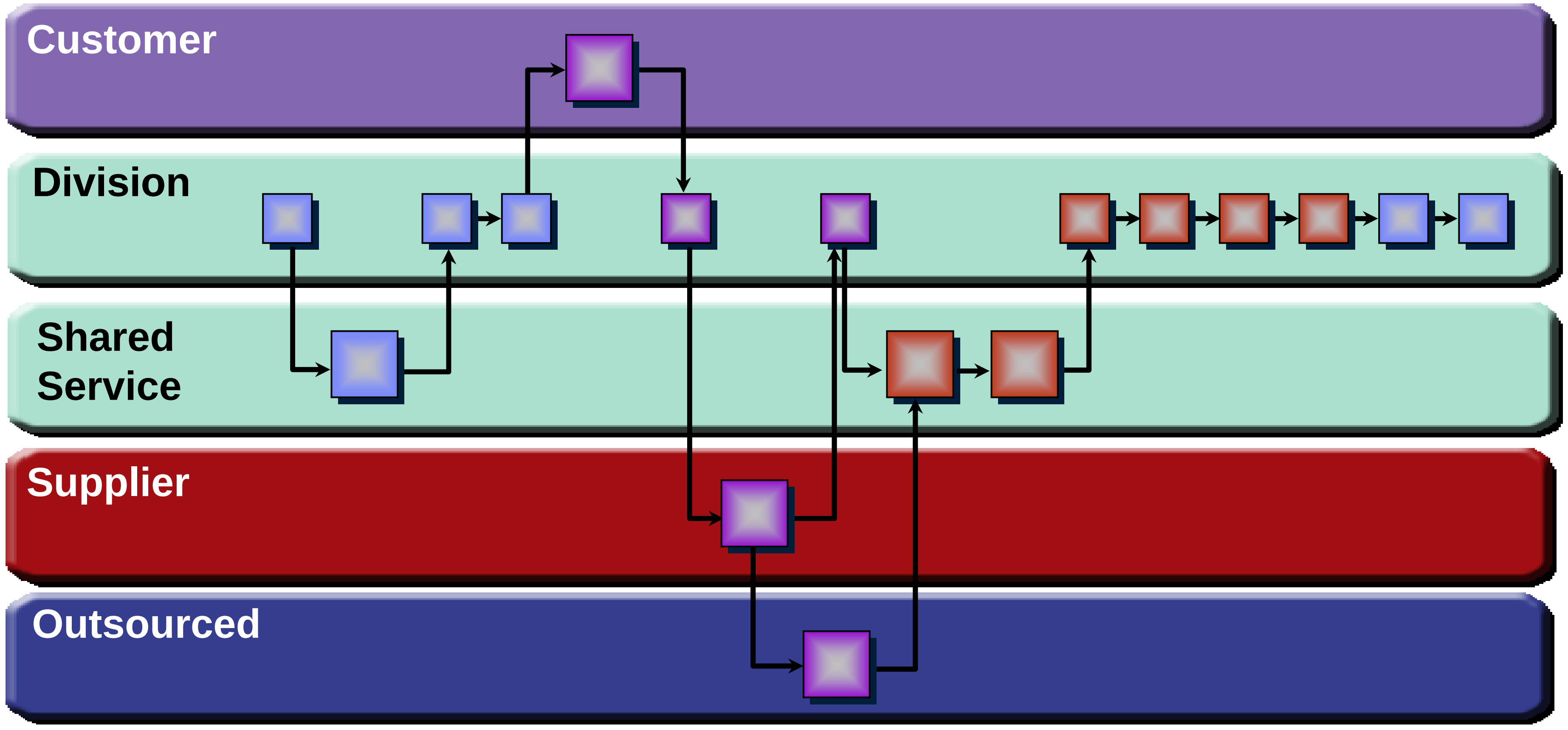
Where We Are Heading

Case Study: Procure to Pay Process



Change: Supplier Handles Inventory

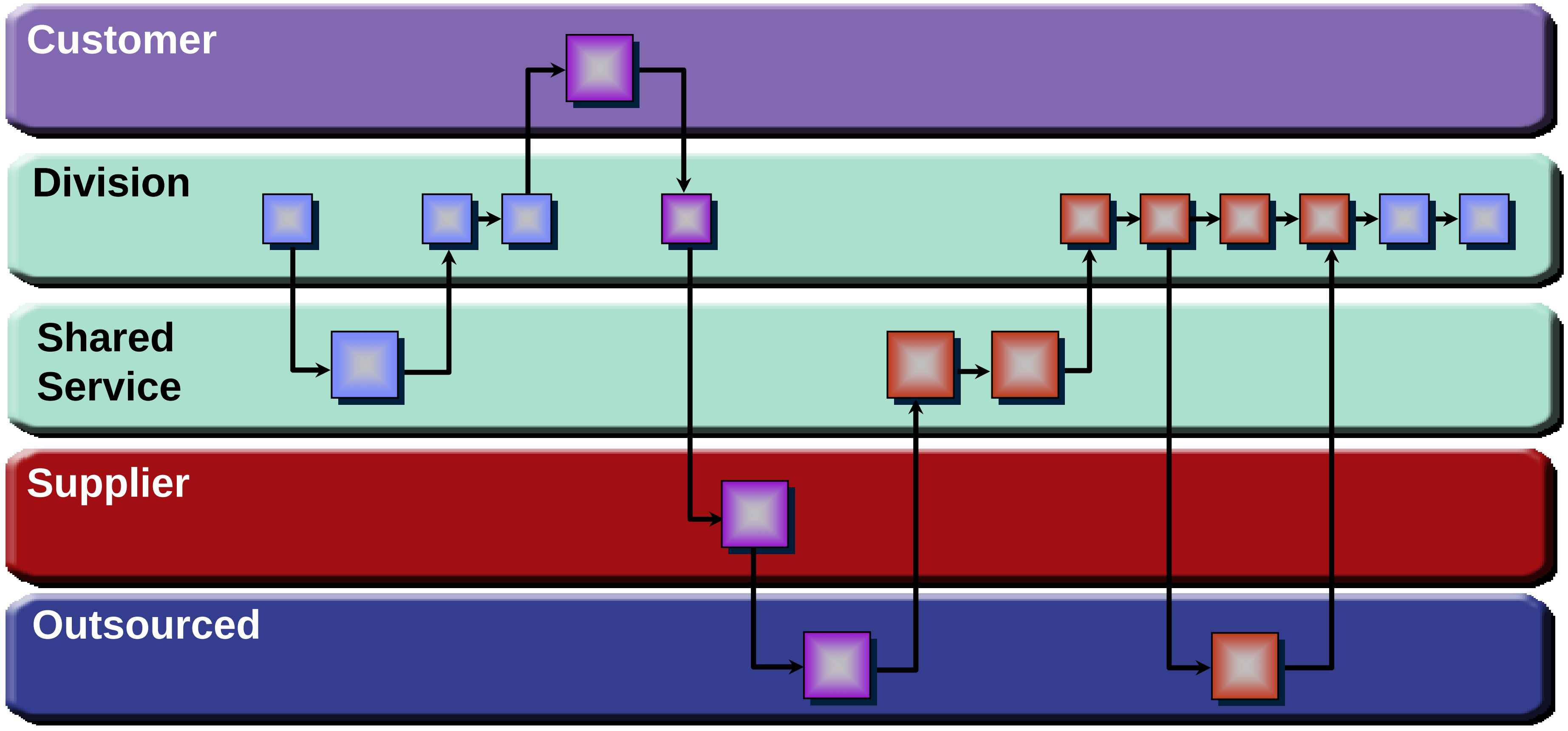
Where We Are Heading
Case Study: Procure to Pay Process



Change: Shipping by FedEx, DHL or UPS

Where We Are Heading

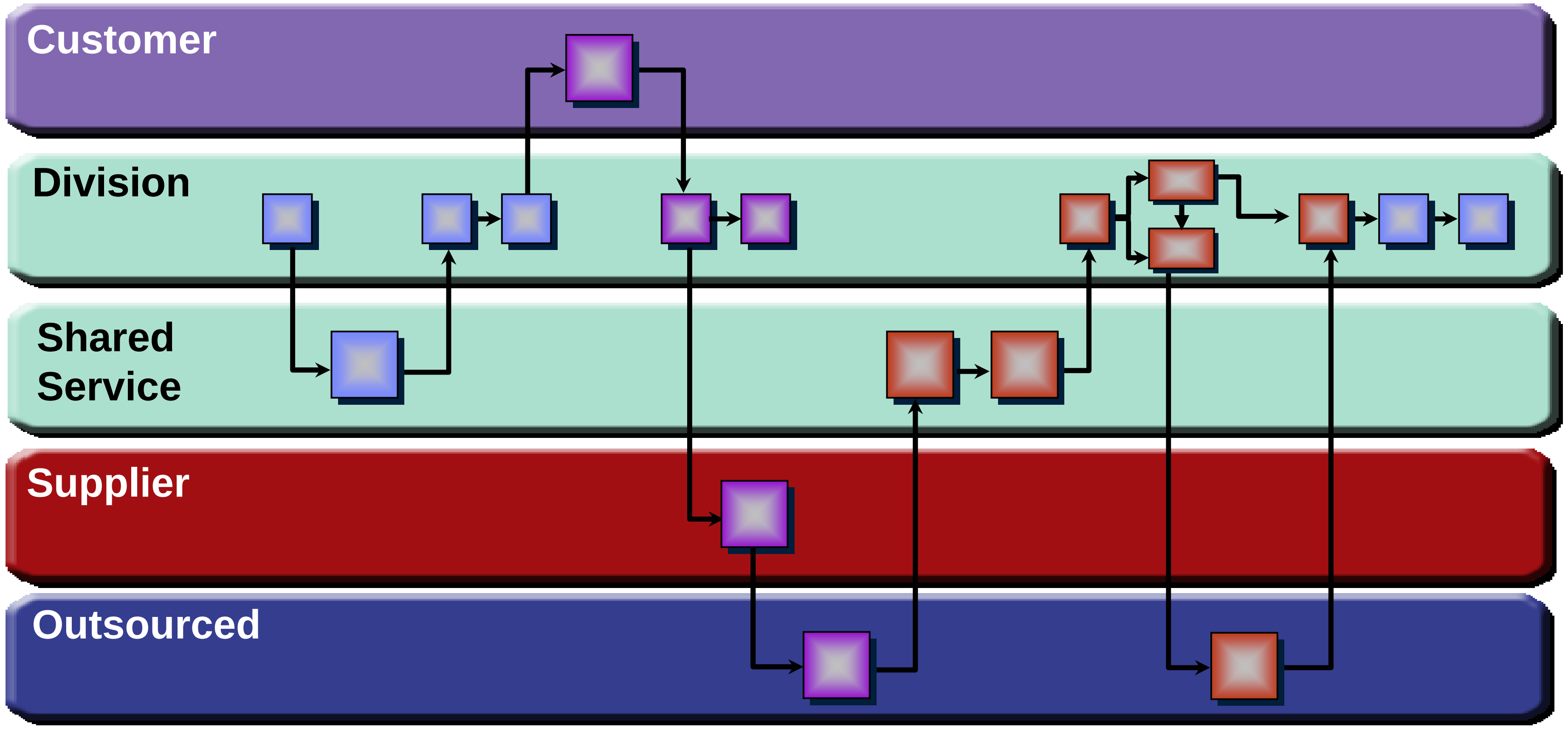
Case Study: Procure to Pay Process



Change: Collections Outsourced

Where We Are Heading

Case Study: Procure to Pay Process



Change: Process Optimization

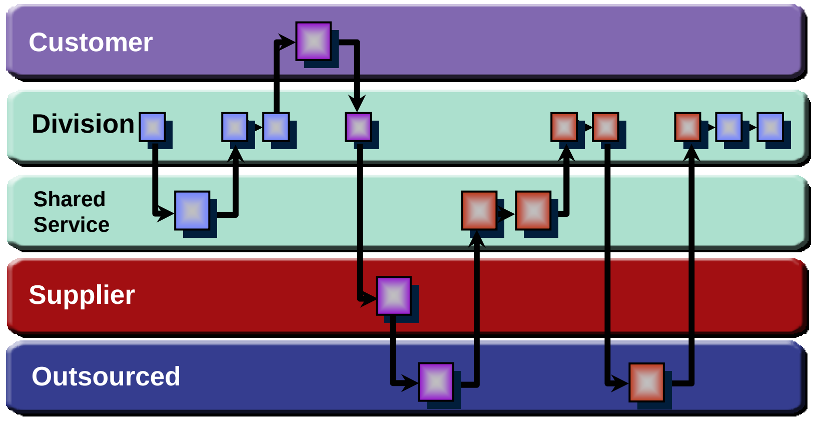
Why is this Scenario Desirable?

- More pragmatic approach to re-engineering efforts
  - Incremental deployments
  - Tied to tangible business benefits
- Each function where it best meets business needs:
  - Within the organization
  - From a business partner
  - Completely outsourced
- Tune ongoing business without widespread disruption: optimize in isolation

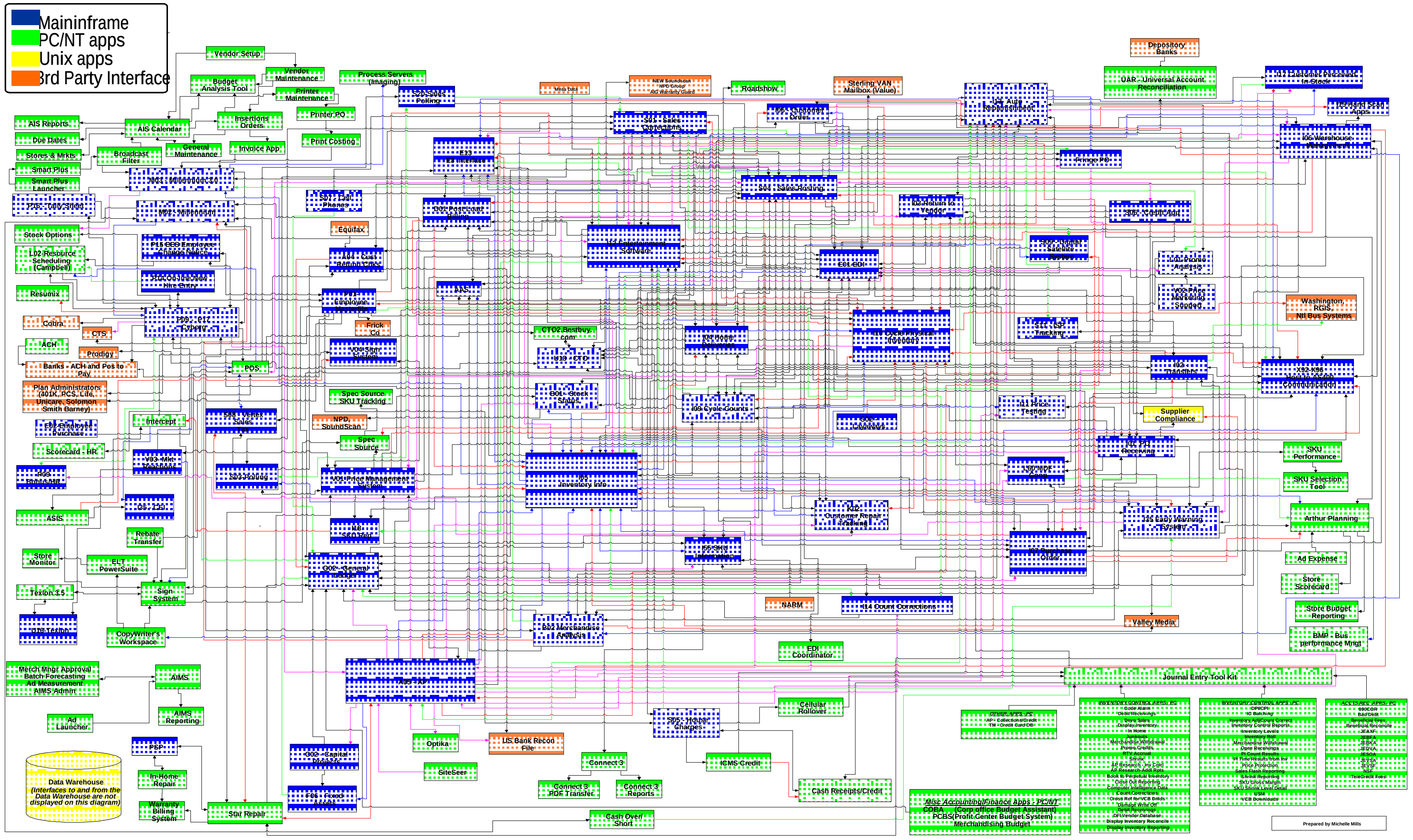
IT Architecture is a Choke Point for Business

- Monolithic systems and applications can't be reused
- Ad hoc integration is difficult to change/maintain
- Lack of standards limits interoperability
- Rigidity makes small improvements hard to justify

Case Study: Procure to Pay Process



Complexity is Forcing Change

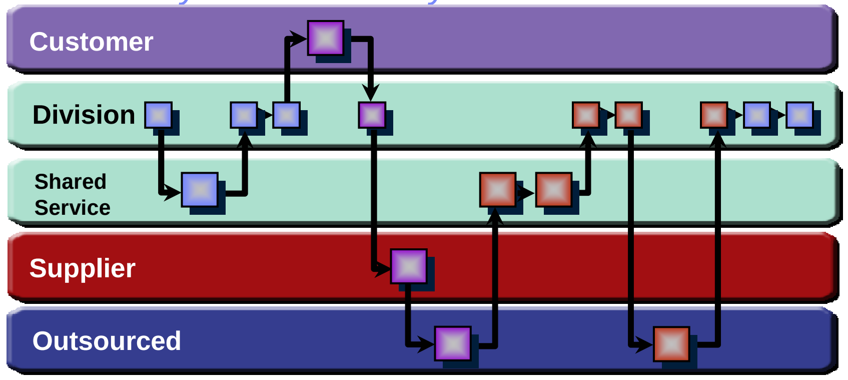


Actual Application Architecture for Consumer Electronics Company

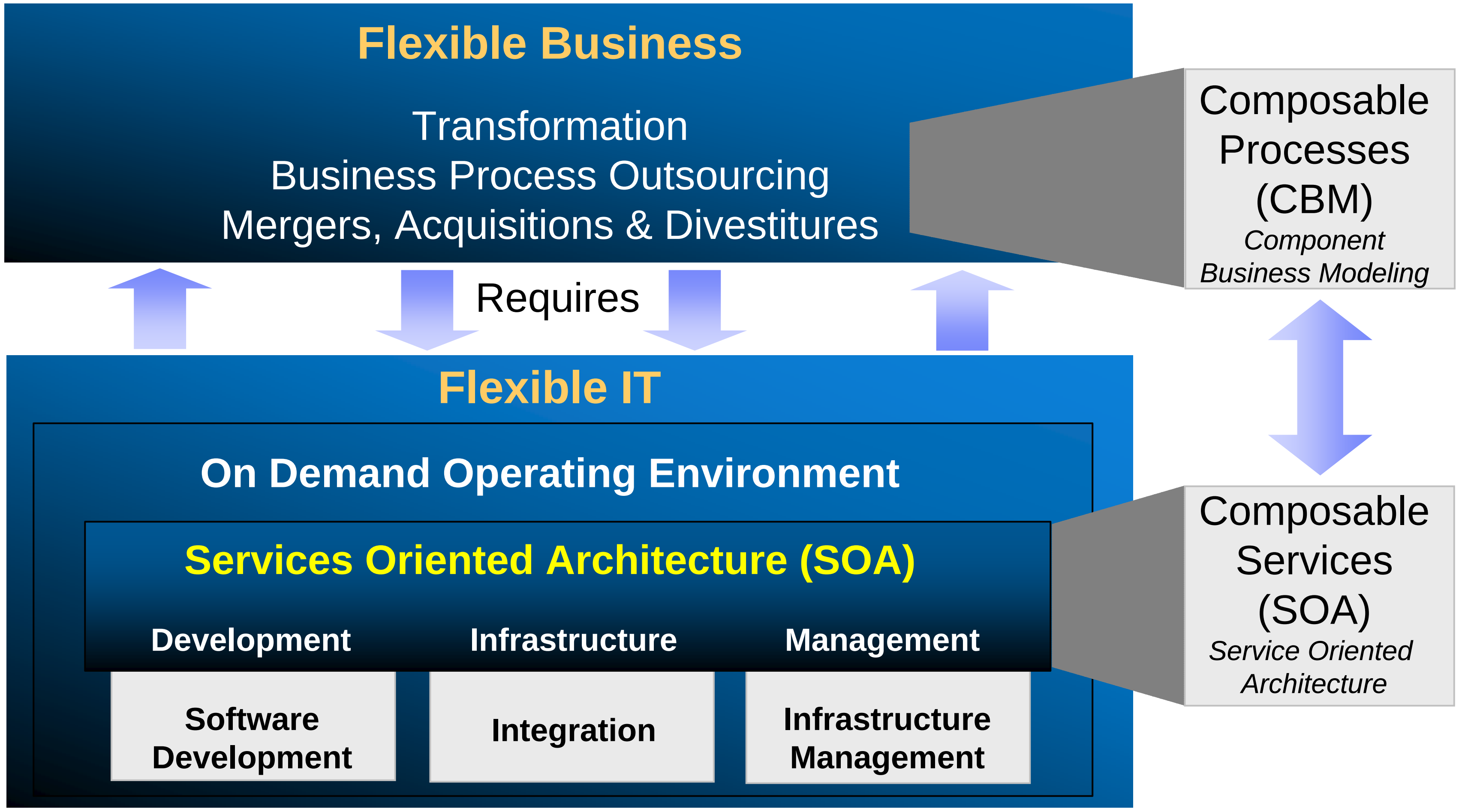
But … Technology Applied Correctly can Pave the Way for successful Business Innovation

- Standards (including open source) for interoperability
- Self-defined, loosely coupled interfaces
- Tools to visualize and integrate existing assets
- Model Driven Architecture (MDA)
- Declarative specifications and languages
- Compositional techniques like Aspect Oriented Programming

Case Study: Procure to Pay Process



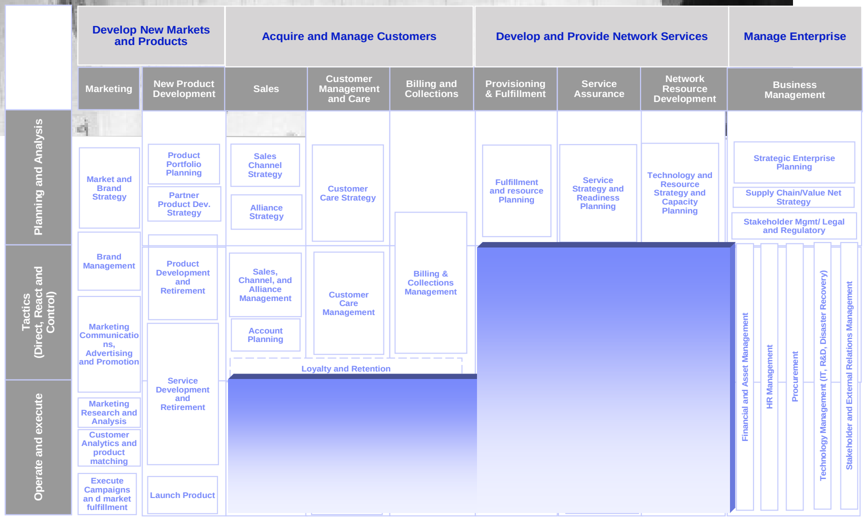
Tying Business Models to the Supporting IT Architecture


Flexible Business
Transformation
Business Process Outsourcing
Mergers, Acquisitions & Divestitures
Requires
Flexible IT
On Demand Operating Environment
Services Oriented Architecture (SOA)
Development
Infrastructure
Management
Software Development
Integration
Infrastructure Management
Composable Processes (CBM)
Component Business Modeling
Composable Services (SOA)
Service Oriented Architecture

Three Key Concepts

- Component Business Modeling (CBM)
- Service Oriented Architecture (SOA)
- Model Driven Architecture (MDA)

Component Business Models Permit Change


Develop New Markets and Products
Acquire and Manage Customers
Develop and Provide Network Services
Manage Enterprise
Marketing
New Product Development
Sales
Customer Management and Care
Billing and Collections
Provisioning & Fulfillment
Service Assurance
Network Resource Development
Business Management
Planning and Analysis
Tactics (Direct, React and Control)
Operate and execute
Market and Brand Strategy
Product Portfolio Planning
Partner Product Dev. Strategy
Sales Channel Strategy
Alliance Strategy
Customer Care Strategy
Fulfillment and resource Planning
Service Strategy and Readiness Planning
Technology and Resource Strategy and Capacity Planning
Strategic Enterprise Planning
Supply Chain/Value Net Strategy
Stakeholder Mgmt/ Legal and Regulatory
Brand Management
Product Development and Retirement
Sales, Channel, and Alliance Management
Customer Care Management
Billing & Collections Management
Marketing Communications, Advertising and Promotion
Account Planning
Loyalty and Retention
Service Development and Retirement
Marketing Research and Analysis
Customer Analytics and product matching
Execute Campaigns an d market fulfillment
Launch Product
Financial and Asset Management
HR Management
Procurement
Technology Management (IT, R&D, Disaster Recovery)
Stakeholder and External Relations Management

- Identify business transformation opportunities at the business process level
- Make new business process composable and flexible
- Model the business as components with defined *services that the business provides or consumes*
- Services are provided by IT systems, sourced from providers or supported manually

Business Components

- Business performs *activities*
  - Ex: Rent vehicle
- Activities use *resources*
  - Ex: Rental car, parking spot, agent
- A *business component* is a set of activities and resources
  - Ex: Rental location owns cars and rents and checks in cars
  - It is a virtual business unit that can operate independently

Dependencies among Components

- Activities depend on other dependencies
  - Resource usage
  - Control and information flows
- Goal: restrict dependencies among components
  - Regroup activities and resources
  - Channel them through services

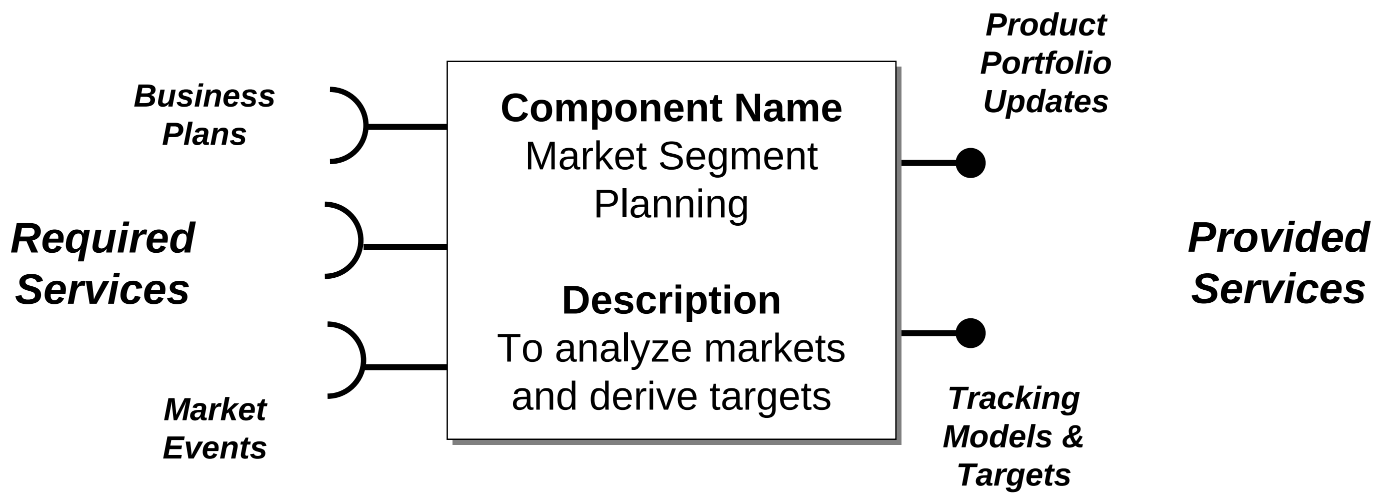
Services

- A service is an explicit interface with no “back doors”
- Service has a defined contract
  - Inputs and outputs
  - Interaction protocol
  - Performance
- Components linked by services are substitutable

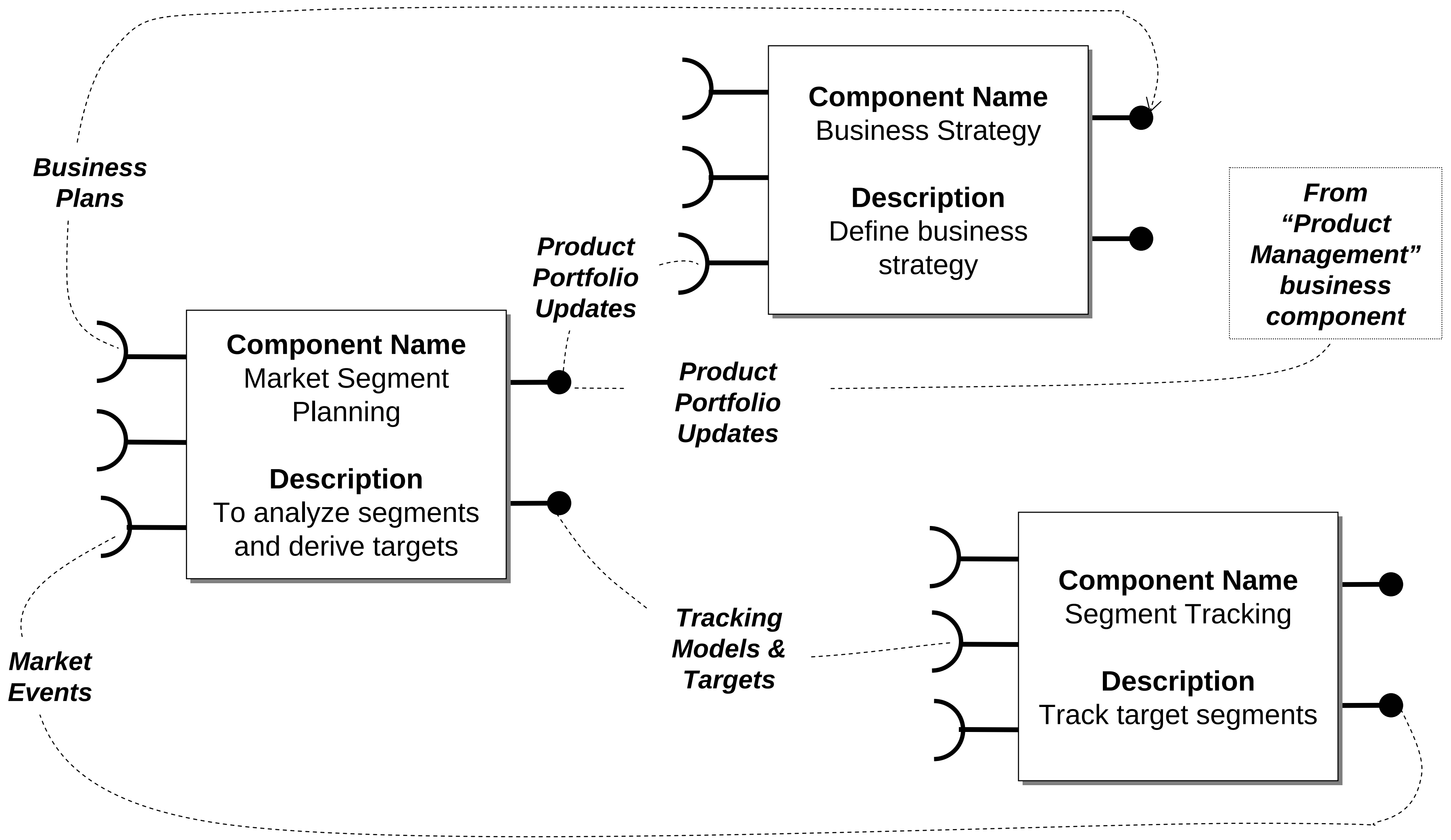
A Business Component Map is a tabular view of the business components in scope.

	Business Administration	New Business Development	Relationship Management	Servicing and Sales	Product Fulfillment	Financial Control and Accounting
Directing	Business Planning	Sector Planning	Account Planning	Sales Planning	Fulfillment Planning	Portfolio Planning
Controlling	Business Unit Tracking	Sector Management	Relationship Management	Sales Management	Fulfillment Planning	Compliance Reconciliation
	Staff Appraisals	Product Management	Credit Assessment			
Executing	Staff Administration	Product Directory	Credit Administration	Sales	Product Fulfillment	Customer Accounts
	Product Administration	Marketing Campaigns		Customer Dialog	Document Management	General Ledger
				Contact Routing		

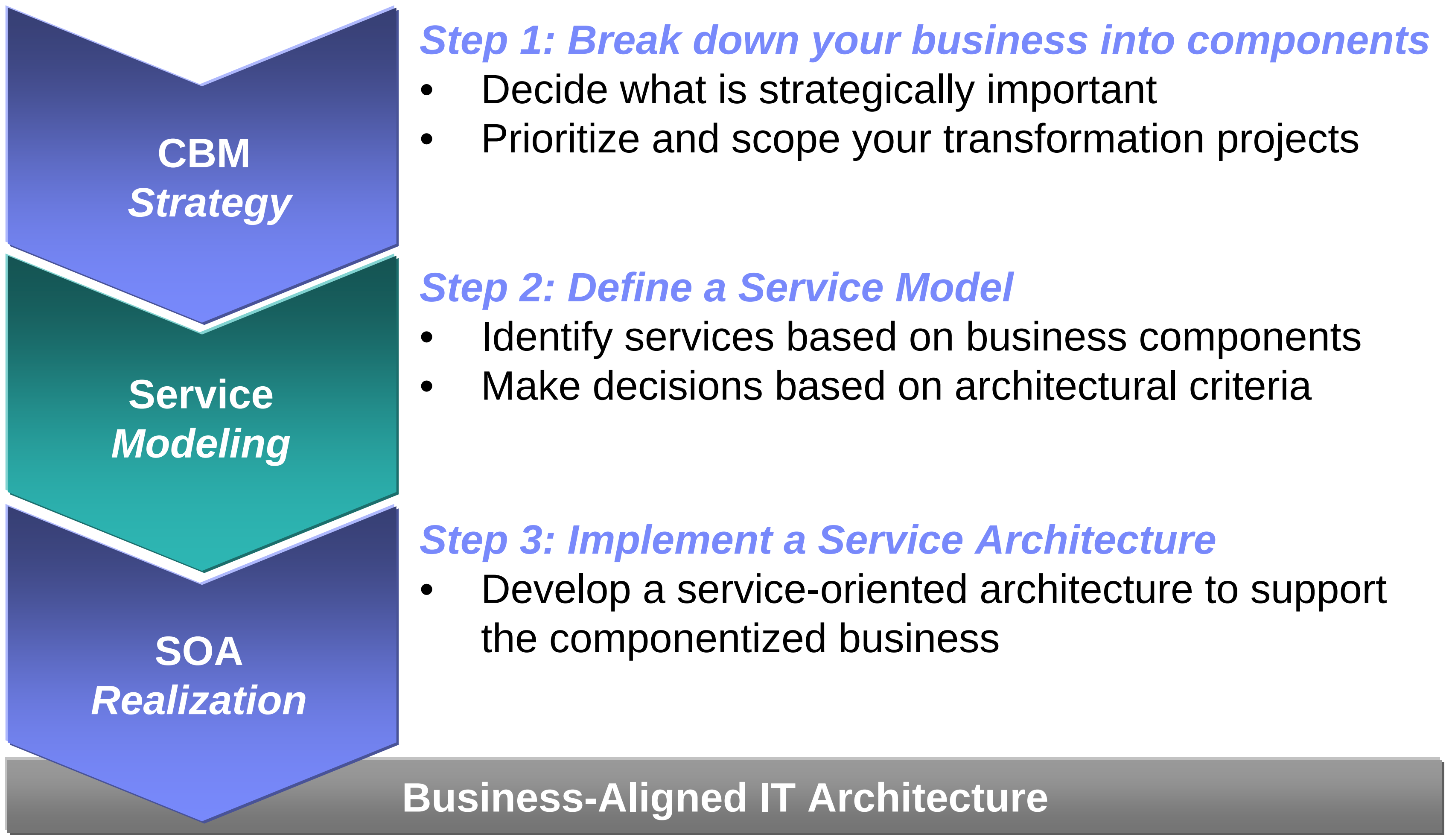
A Business Component offers services to other components



Business components use each other’s services.



Building an Architecture



Step 1: Break down your business into components

- Decide what is strategically important
- Prioritize and scope your transformation projects

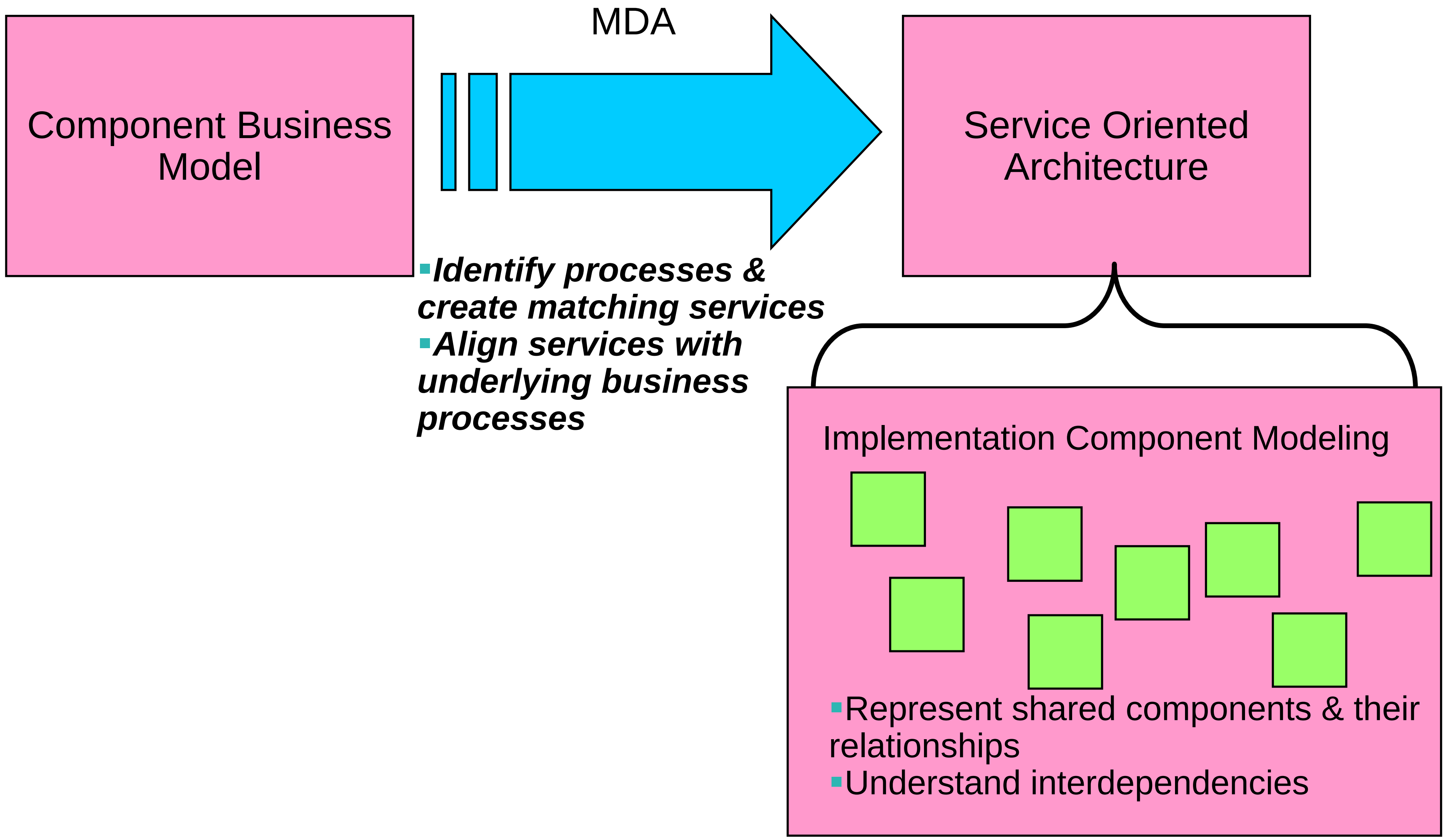
Step 2: Define a Service Model

- Identify services based on business components
- Make decisions based on architectural criteria

Step 3: Implement a Service Architecture

- Develop a service-oriented architecture to support the componentized business

Component Models & SOA


MDA
Component Business Model
Service Oriented Architecture
Implementation Component Modeling

- *Identify processes & create matching services*
- *Align services with underlying business processes*

- Represent shared components & their relationships
- Understand interdependencies

Service Oriented Architecture (SOA)

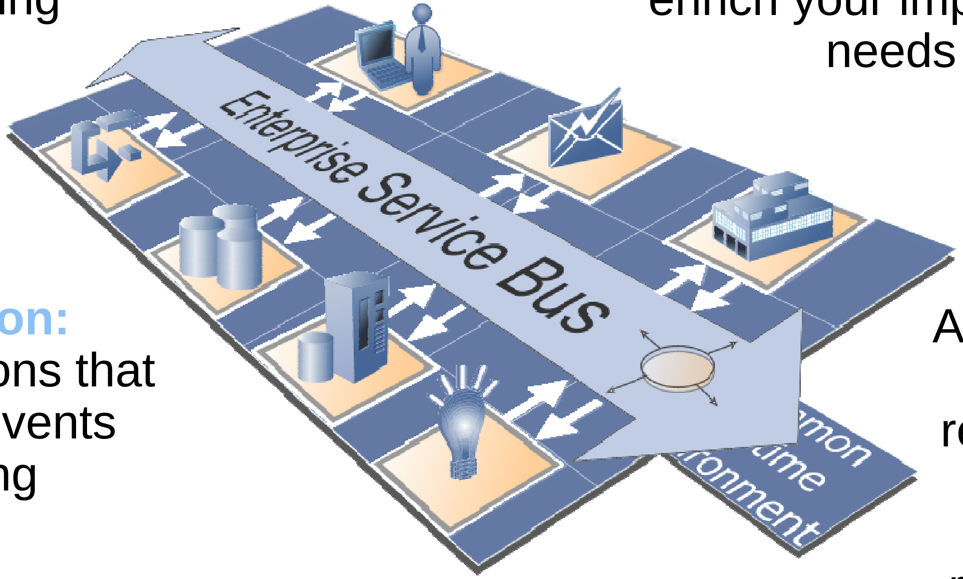
- Flexible connectivity:
  - Represent application or resource as a **service** with a **standardized interface**
  - Enable **exchange of structured information** (messages, documents, 'business objects')
  - Mediate message exchange through a connectivity layer (**enterprise service bus)**
  - Provide **on-ramps** to the bus for legacy application environments
- Combines new and existing applications to address changing business needs
- Facilitates the management of business performance, quality of service, and dynamic monitoring

SOA Defining Concepts

Universal Connectivity: Integrates most diverse environments, bridging protocols, languages, platforms, APIs and messaging paradigms – providing scale and scope of integration required by today's extended enterprise

Service Orientation: Facilitates loose coupling between software components

Incremental Integration: Start small and plug in capability to enrich your implementation as needs dictate

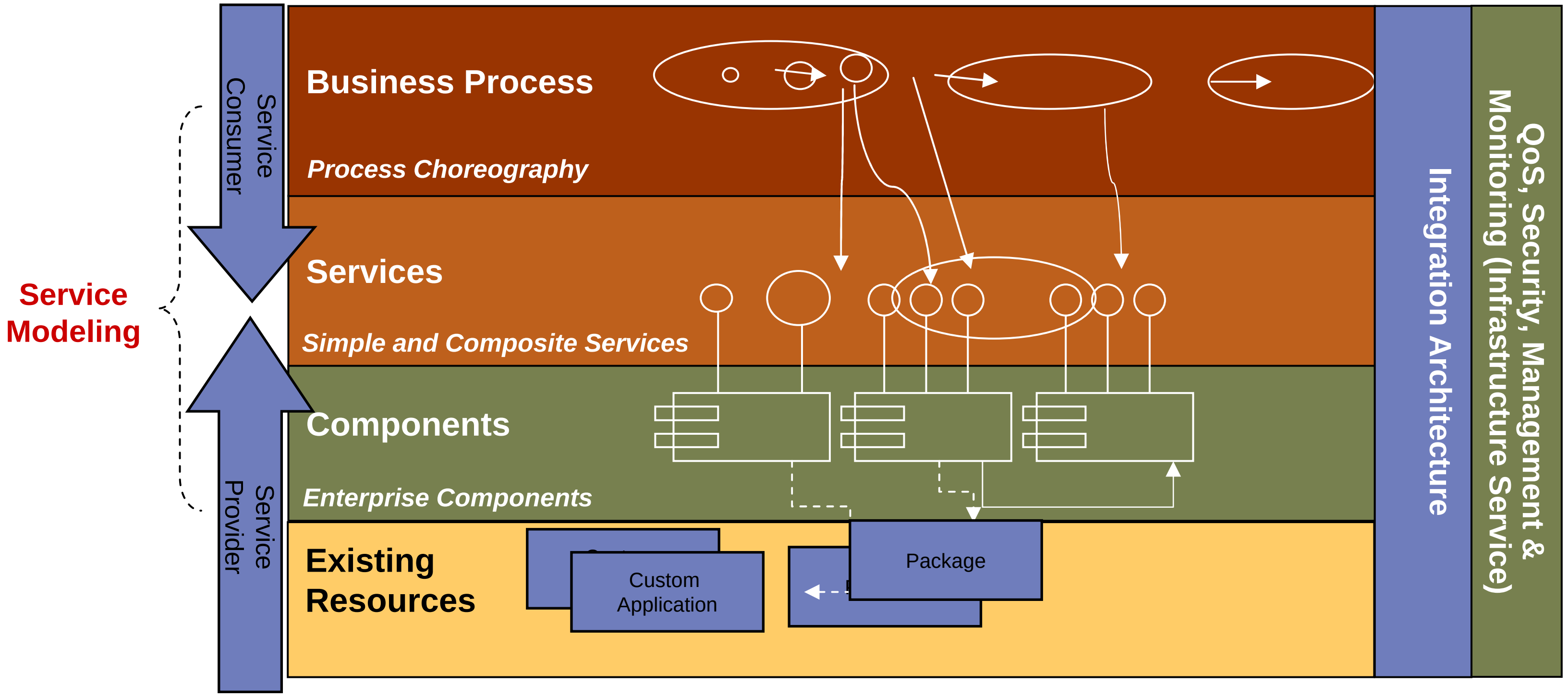


Event Orientation: Decouples applications that publish business events from subscribing applications

Flexibility: A variety of options for persistence, reliability, security, availability... Deploy where required, manage centrally

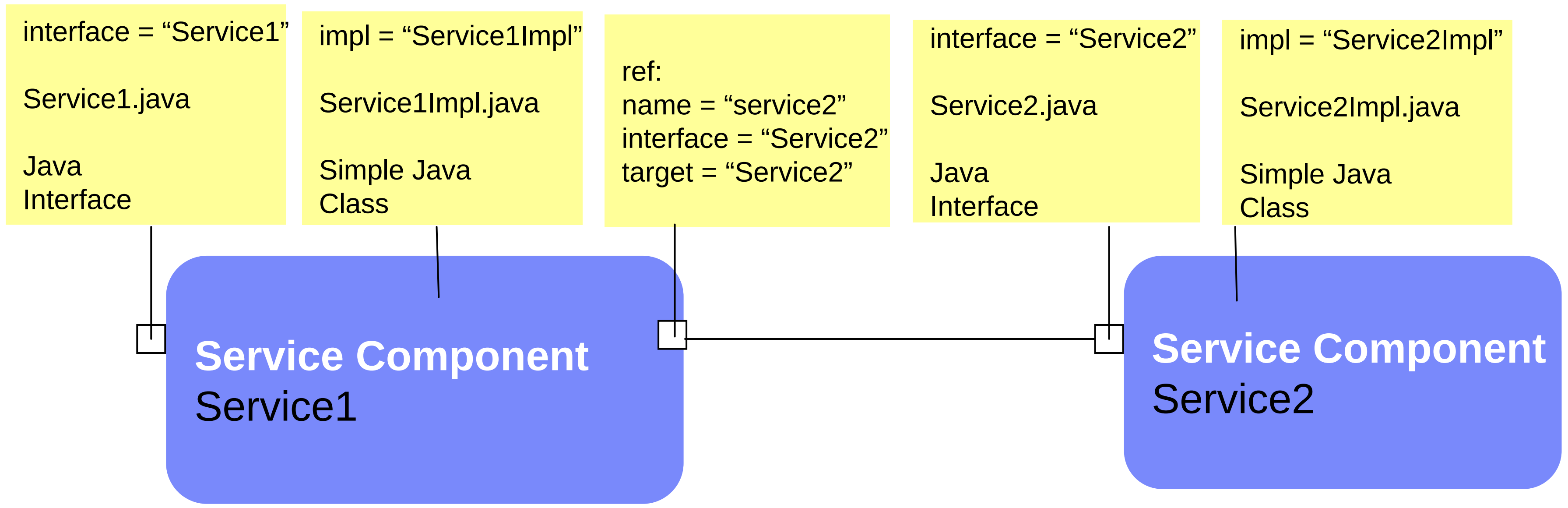
Open, standards-based: Open APIs and protocols support the interoperability and substitution of middleware from multiple vendors

A SOA has Multiple Layers



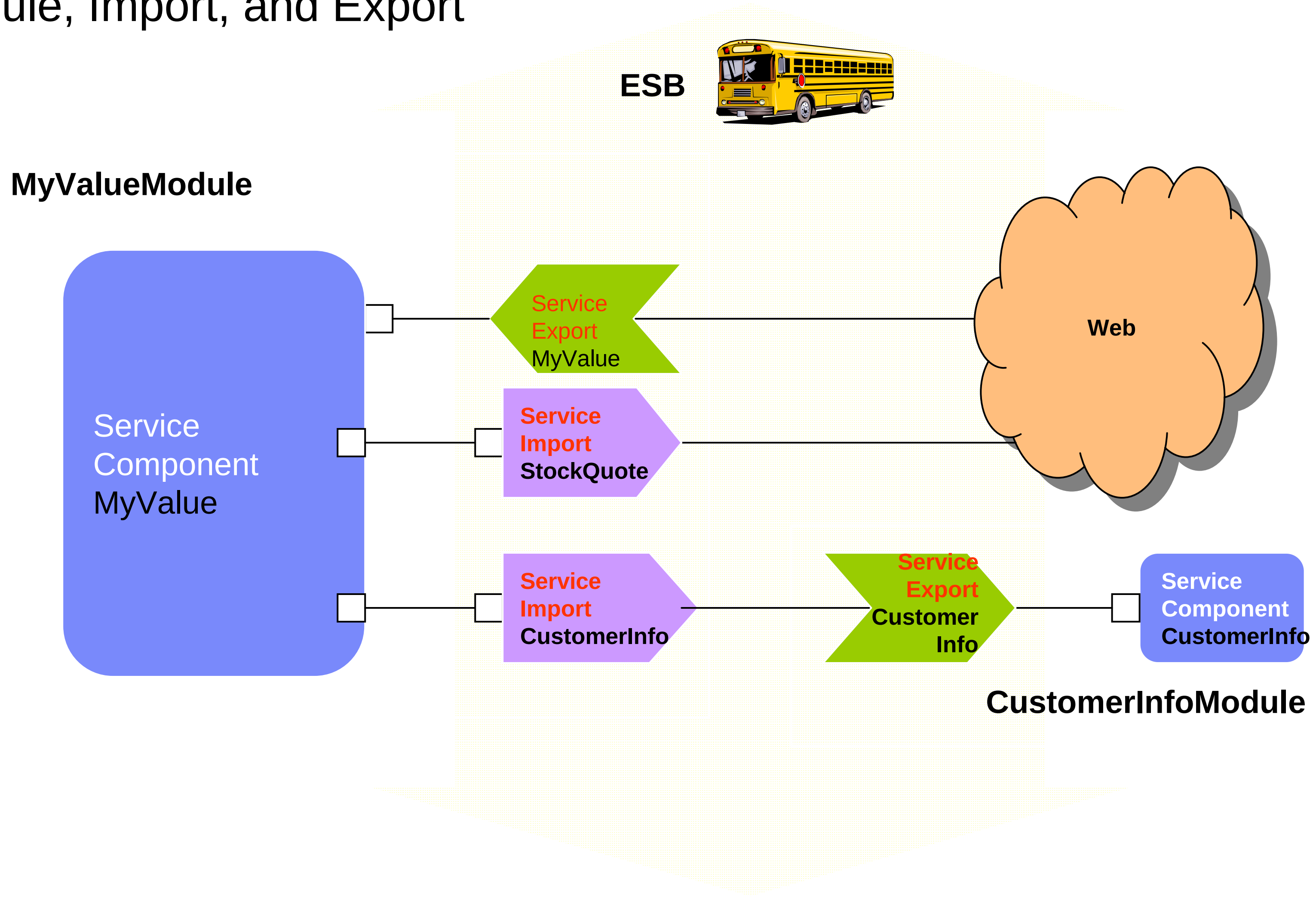
Service Component Conceptual Model

Component, Interface, Implementation, Reference and Wire



Service Component Conceptual Model

Module, Import, and Export

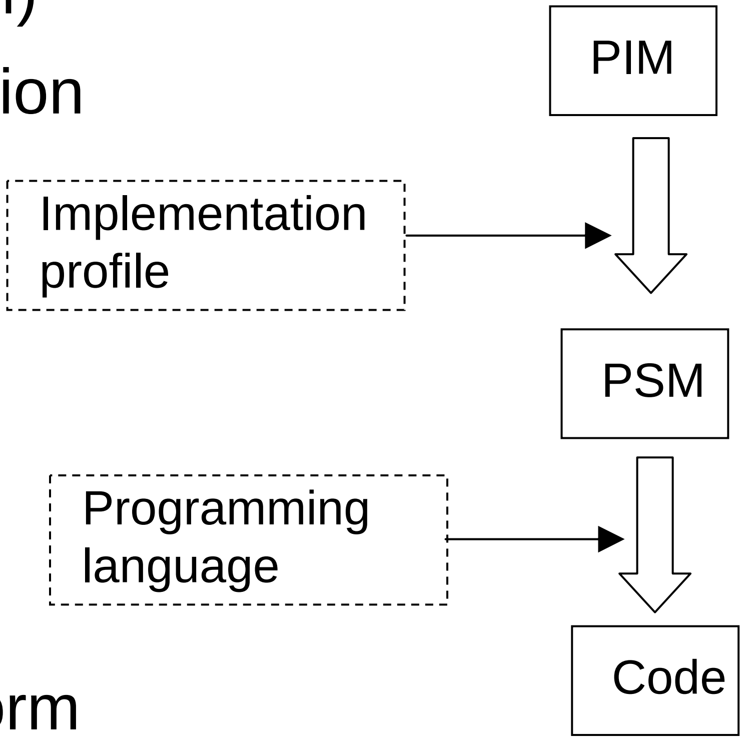

ESB
MyValueModule
Service Component MyValue
Service Export MyValue
Service Import StockQuote
Web
Service Import CustomerInfo
Service Export Customer Info
Service Component CustomerInfo
CustomerInfoModule

Model Driven Architecture (MDA)

- Build high-level models
  - Expressed in domain concepts
- Generate platform-specific code
  - Into specified architectures
- Using automated tools
  - Based on standards

MDA Approach

- Platform Independent Model (PIM)
  - Captures logic of the application
- Implementation profile
  - Platform
  - Technology decisions
- Platform Specific Models (PSM)
  - Target multiple platforms
  - Translator incorporates platform knowledge



MDA Requires

- Problem domain models
  - UML or domain-specific language (DSL)
- Architecture frameworks
  - Make assumptions about infrastructure
  - Use them to remove details from models
- Automated tools
  - Modeling tools
  - Translators

Problem Domain Models

- Syntax and semantics for a particular purpose
  - Capture business-level content
  - More intuitive syntax
- Reduce mindless repetition
  - Predefined control and data patterns
  - Eliminate repetitious control specification
- Two approaches:
  - UML Profile
  - Domain specific language (DSL)

Architecture

- Decomposition into subsystems
- Topology
- Interaction rules
- Data formats
- Resource management
- Hooks for future extensions
- Scaffolding for testing

Architecture Framework

- Framework = reusable architecture pattern
- Includes:
  - Skeleton execution environment
  - Predefined subsystems and topology
  - Interaction and data rules, formats, interfaces
  - Attachment points for plug-ins
  - Component libraries of useful functionality
  - Sample applications
- Integrated development environments (IDE)
  - Eclipse, .NET, J2EE

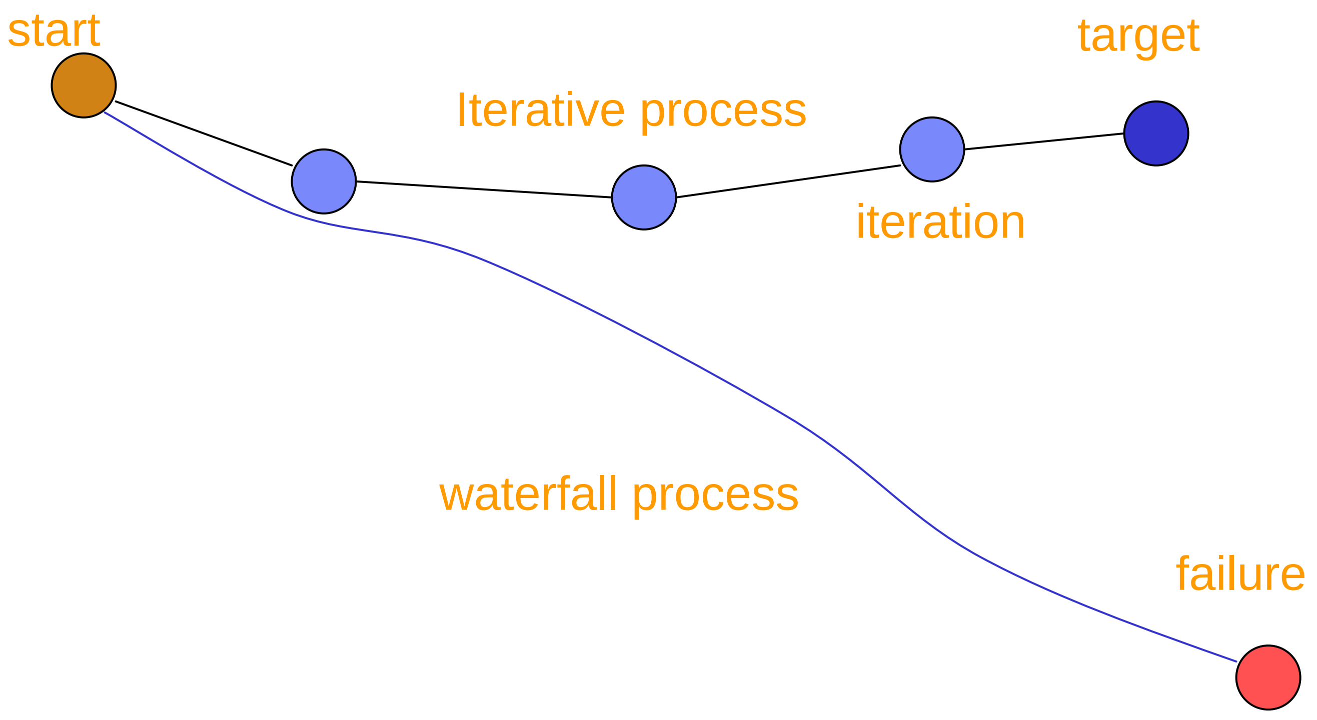
Framework Advantages

- Enable domain specific languages (DSL) to optimize repeated patterns
- Ensure consistency
- Excellent for incremental design
- Enforce design trade-offs
- Supports static and dynamic analysis

Automated Tools

- Modeling tools
  - Organize large models
  - Check correctness
  - Enable teamwork
- Translators
  - Apply patterns
  - Generate repetitious code
  - Key to MDA

Iterative Development



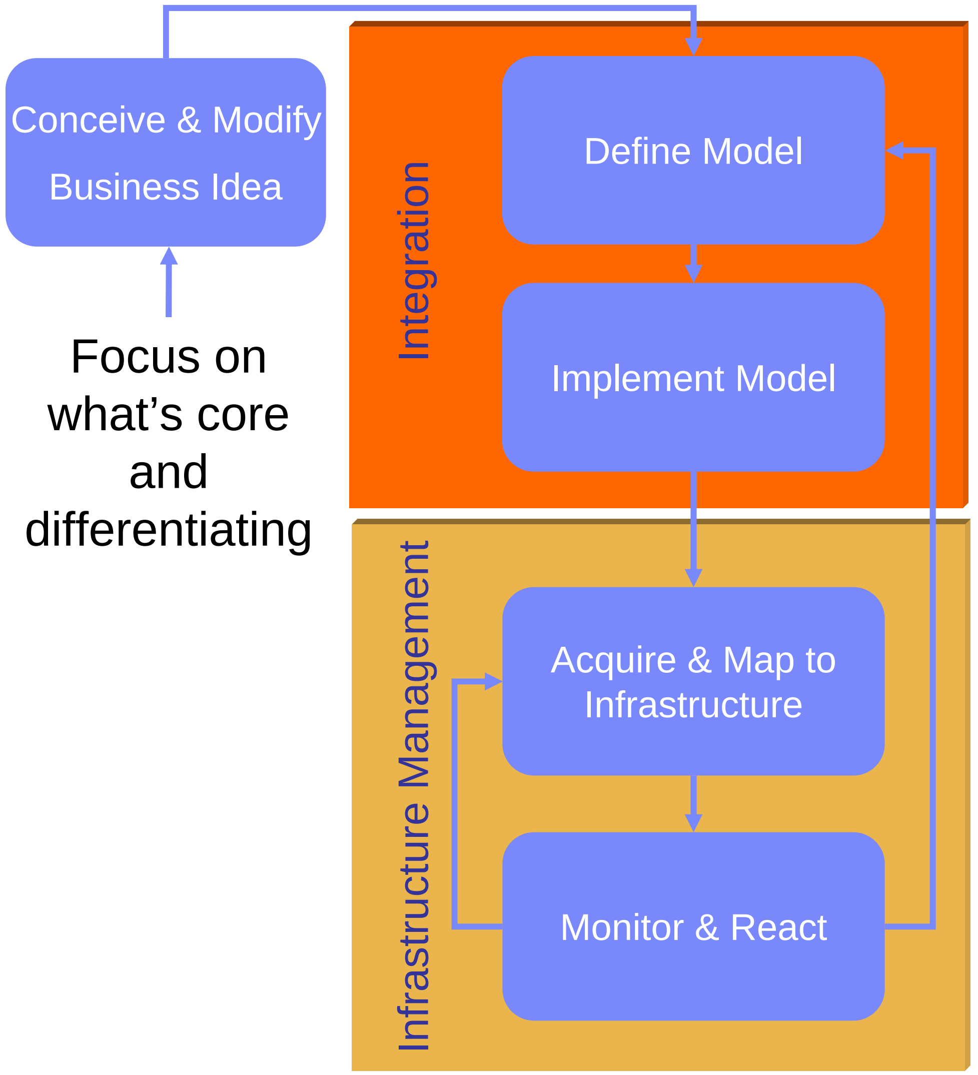
Iteration

- Common baseline
  - No divergent versions or schedules
- Specific goals
  - Set priorities by adjusting goals
- Specific deadline
  - All iterations about the same length
- At the end of an iteration:
  - Check in everything
  - Build an executable system
  - Test and examine it
  - Reevaluate priorities

Iterative Development Process

- Series of iterations allows mid-course corrections
- Executable releases permit constant evaluation
- Attacking problems early reduces risk

Life Cycle of an On Demand Business Service


Conceive & Modify
Business Idea
Focus on what's core and differentiating
Integration
Define Model
Implement Model
Infrastructure Management
Acquire & Map to Infrastructure
Monitor & React

- Business modeling
- Component creation, customization, and assembly
- Deploy internally and externally
- Monitor and control process for business and IT status

Summary

- Business is becoming distributed
  - Component Business Model
- IT architecture must support business models
  - Service Oriented Architecture
- Need to automate business-to-architecture mapping
  - Model Driven Architecture
- Must avoid monolithic development
  - Iterative development process