Everware-CBDI

Specialists in Service Oriented Application Modernization

Achieving Architectural Agility

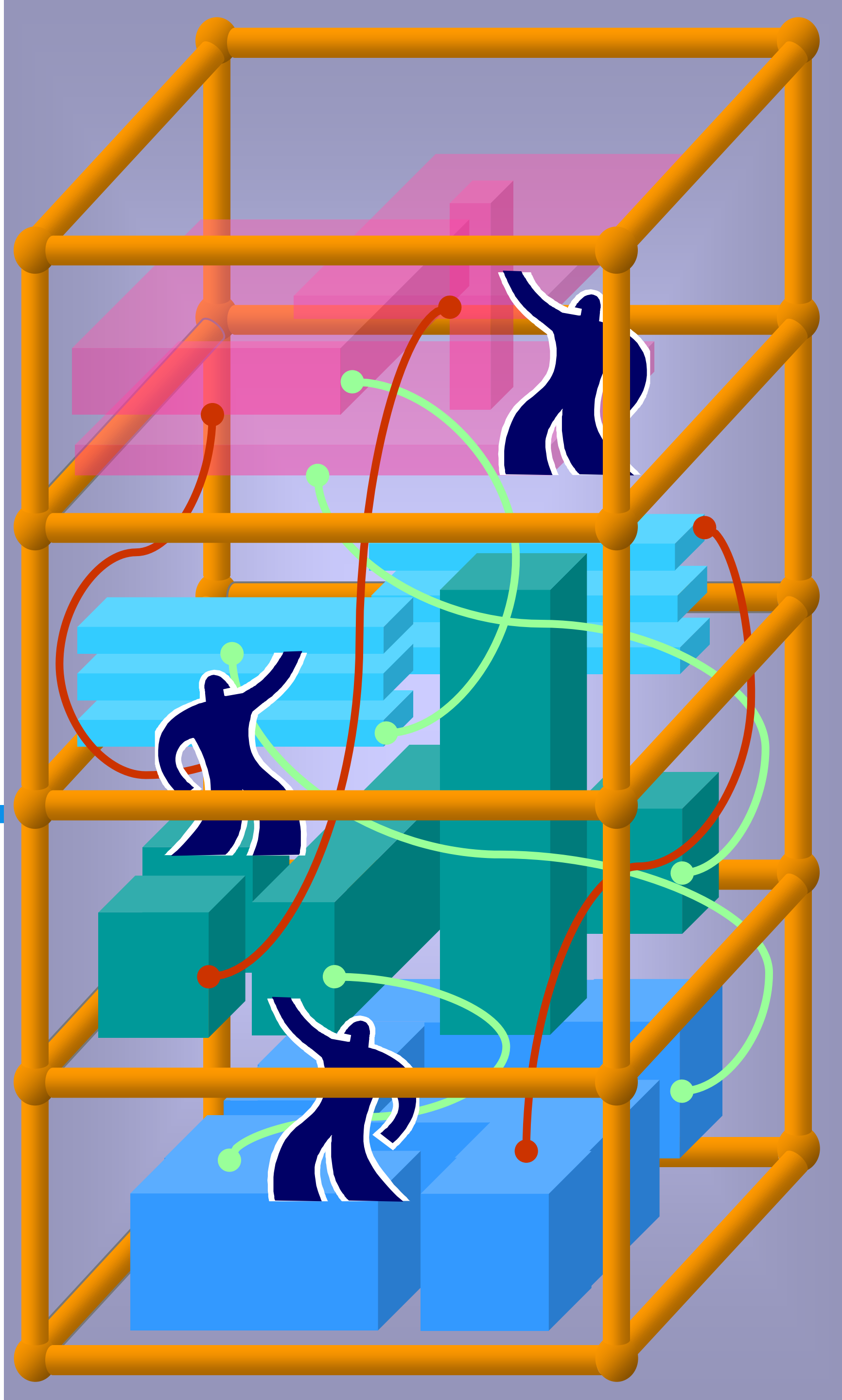
Dave Mayo

Agile in Government Summit 2017

Association for Enterprise Information (NDIA)

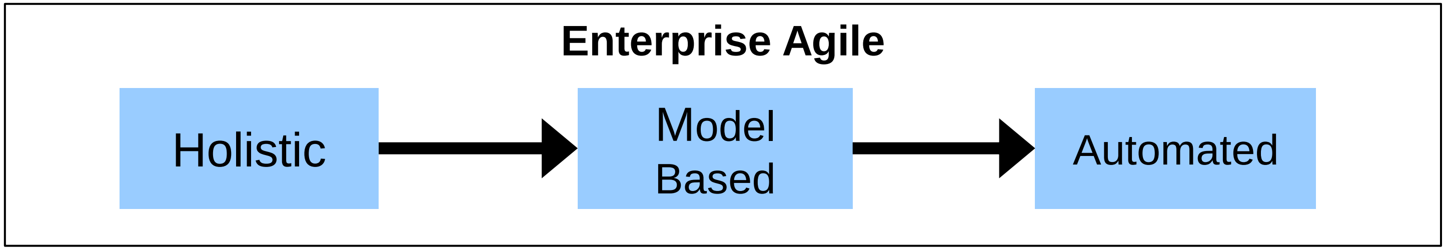
April 19, 2017

dmayo@everware-cbdi.com



Introduction

- ***Who am I?***
  - President of Everware-CBDI
  - Enterprise & Solution Architect
  - IT Strategy & Agile Program Advisor
- ***Who is Everware-CBDI?***
  - 20 years in July
  - Advanced practitioners of Agile, Lean & SAFe
  - IT modernization using Service Architecture & Model Driven Development (MBSE)
    - Agile Service Factory™



Enterprise Agile

Holistic → Model Based → Automated

Mission: Assist organizations to achieve business and IT agility.

Agile development is not the goal.

Not even DevOps.

Agility is the goal.

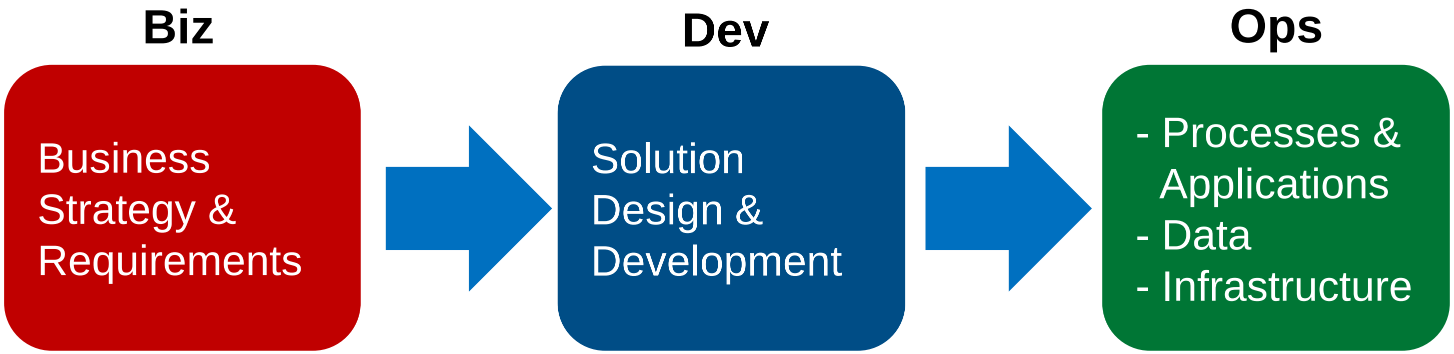
Agility: Ability to sense environment changes, make business decisions, and implement them quickly.

Delivering software using agile methods doesn’t guarantee that the resulting software solution is ***agile***.

Derived from Simon Brown, 2013.

Agility doesn’t just happen. It must be architected and engineered.

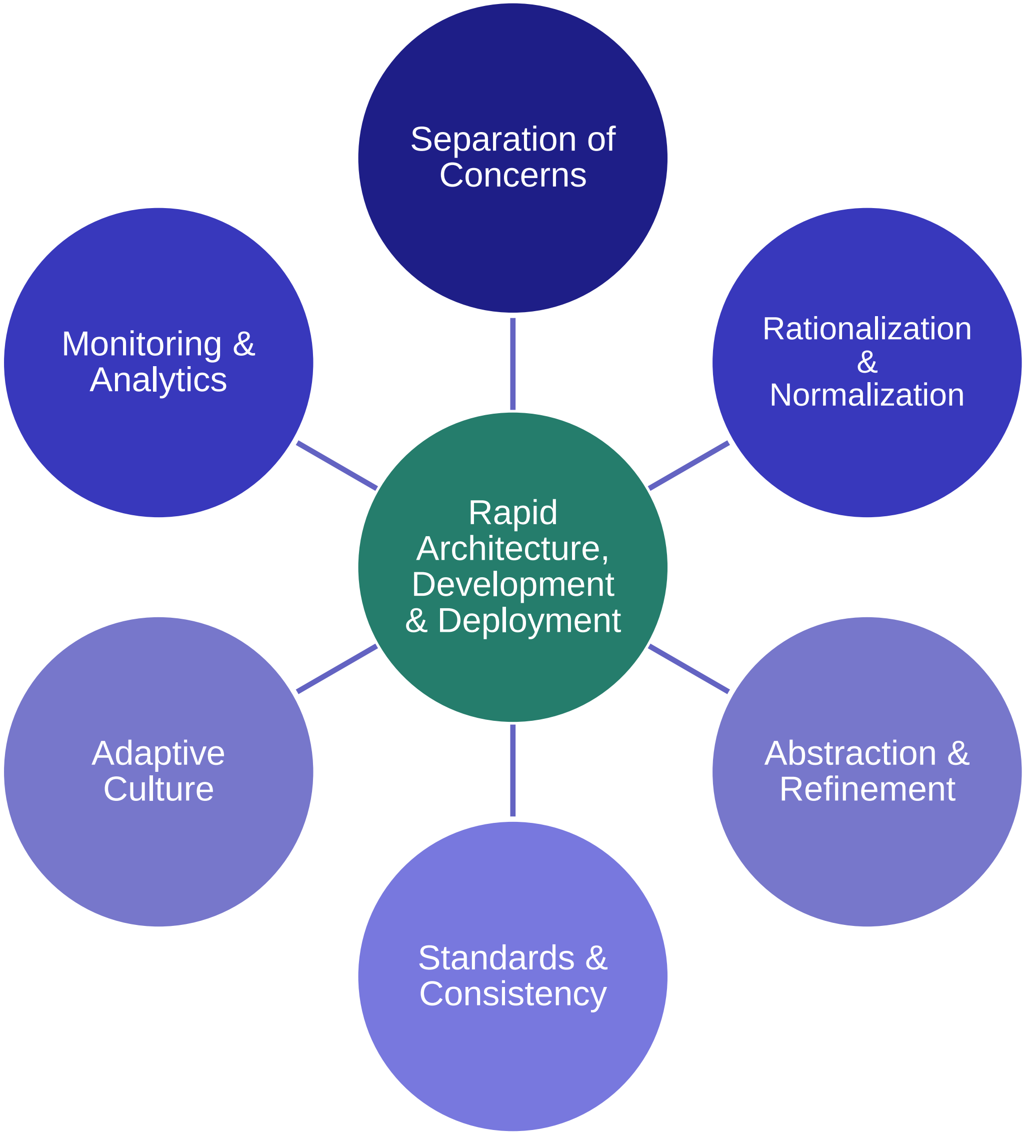
Agility: Minimize Time from Need to Solution


Biz
Business Strategy & Requirements
Dev
Solution Design & Development
Ops
- Processes & Applications
- Data
- Infrastructure

Derived from:
P. Weill,
MIT-CISR

- **DevOps**
  - Collaboration among Dev, Ops, QA and Security
  - Apply best practices and patterns to accelerate delivery
  - Apply automation (testing and deployment)
- ***BizOps* –** Incorporate business analysts/architects in the collaboration to reduce cycle time and improve communication
- Requires a common language to introduce requirements and design (models/specifications)

Foundations of Agility

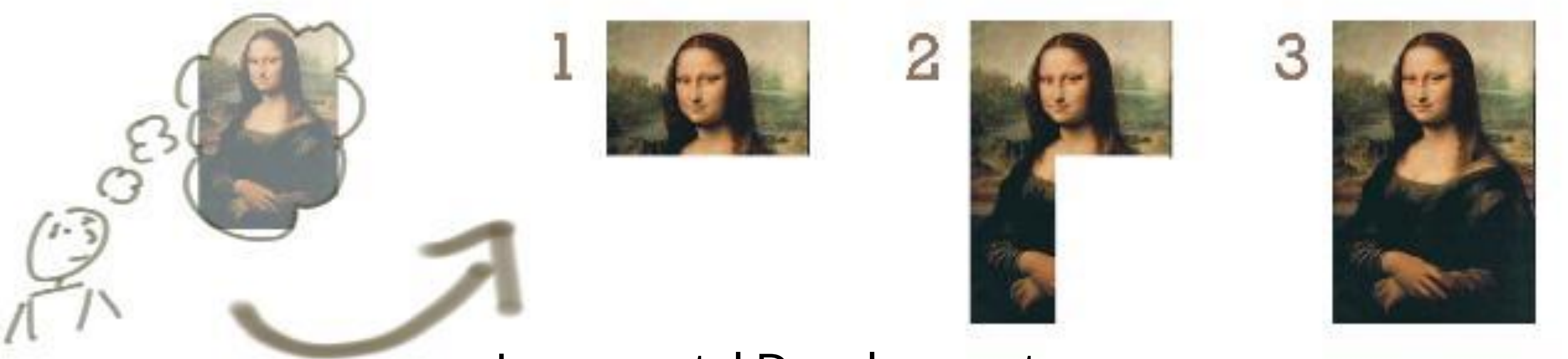


Architecture is Mandatory at Large Scale

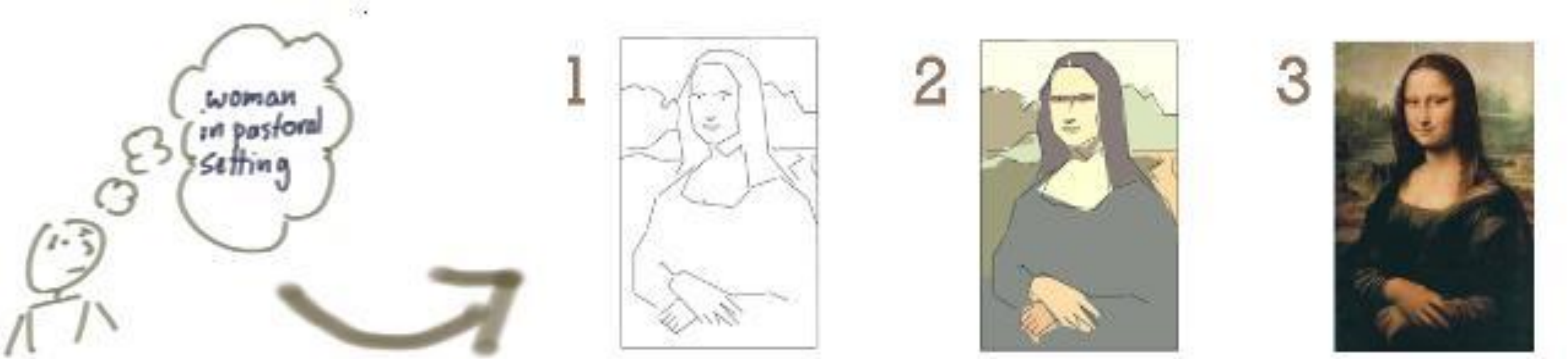
- Purpose: Manage complexity; establish boundaries & relationships.
- Separation of Concerns means modular architecture (services).
- But Architecture doesn’t mean Big Design Up Front (BDUF).
- Provide sufficient architecture to support next increments (Architecture Runway).

Architecture should be an enabler, not a constraint.

Agile Architecture: Incremental & Iterative



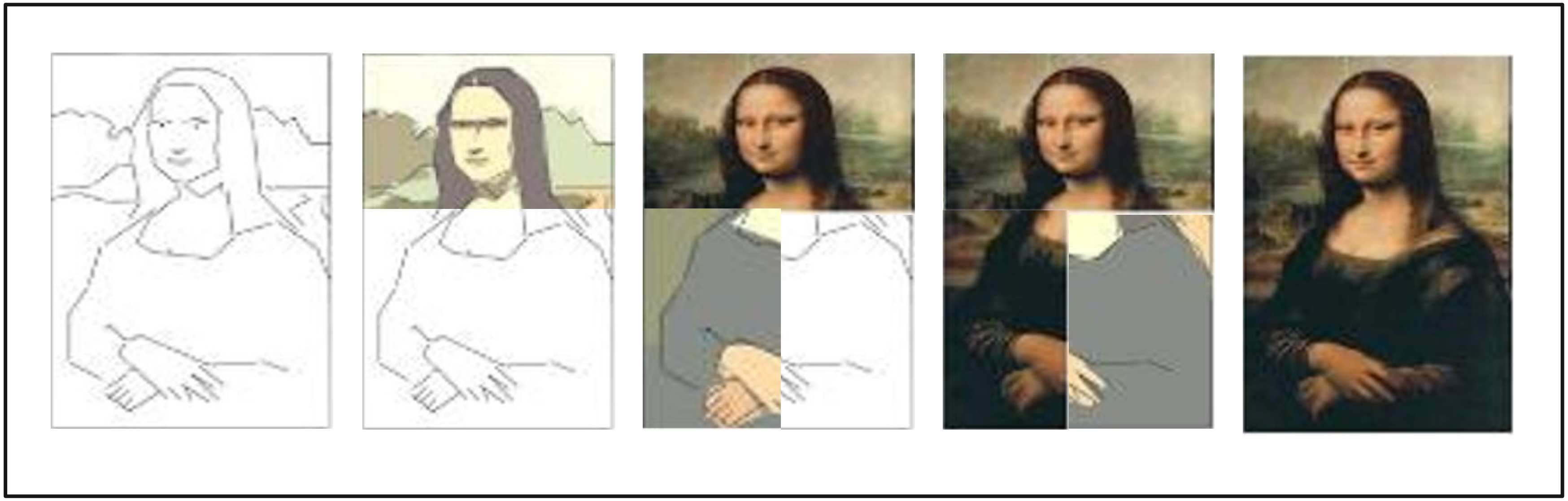
Incremental Development



Iterative Development

From: Jeff Patton

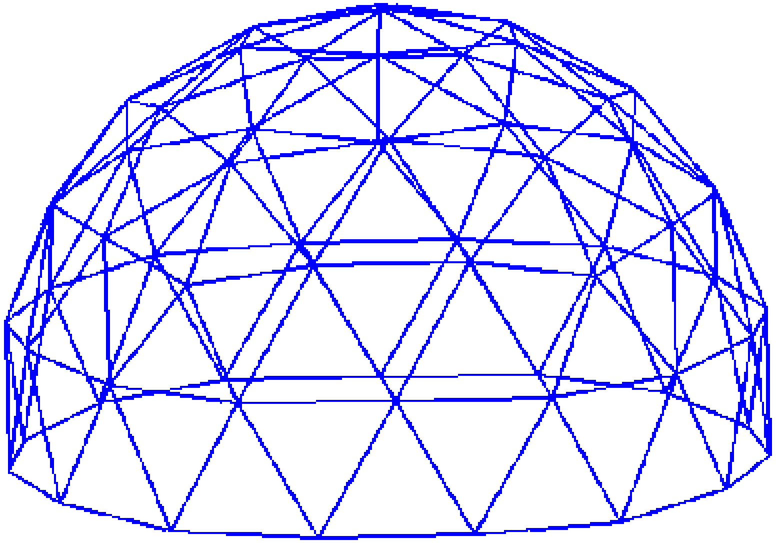
Agile Architecture: *Combining* Incremental & Iterative



Derived from: Jeff Patton

Agile Architecture

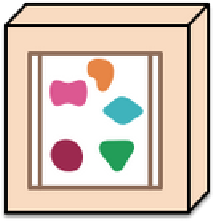
- Apply ***Rolling Wave*** concept to develop the architecture
  - Overall structure of components and how they fit with no detail
  - Detail for each component as needed for next increments
- Agile Architectural Pattern
  - SOA & Microservices



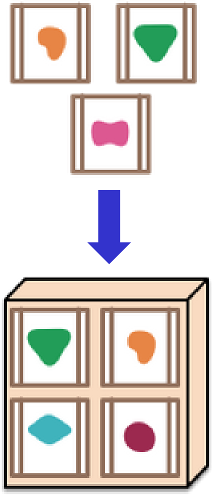
Think big. Start small. Scale fast.

Separation of Concerns: Microservices

- **Develop applications by assembling suites of software services.**
- **Each service:**
  - Is loosely coupled and runs in its own process
  - Is built around a single resource, business capability or domain
  - Is self-contained & has well defined boundaries & interfaces
  - Communicates with lightweight mechanisms
  - May be written in a different language and use a different data storage technology
  - Is independently deployable in an automated manner



Monolithic Application

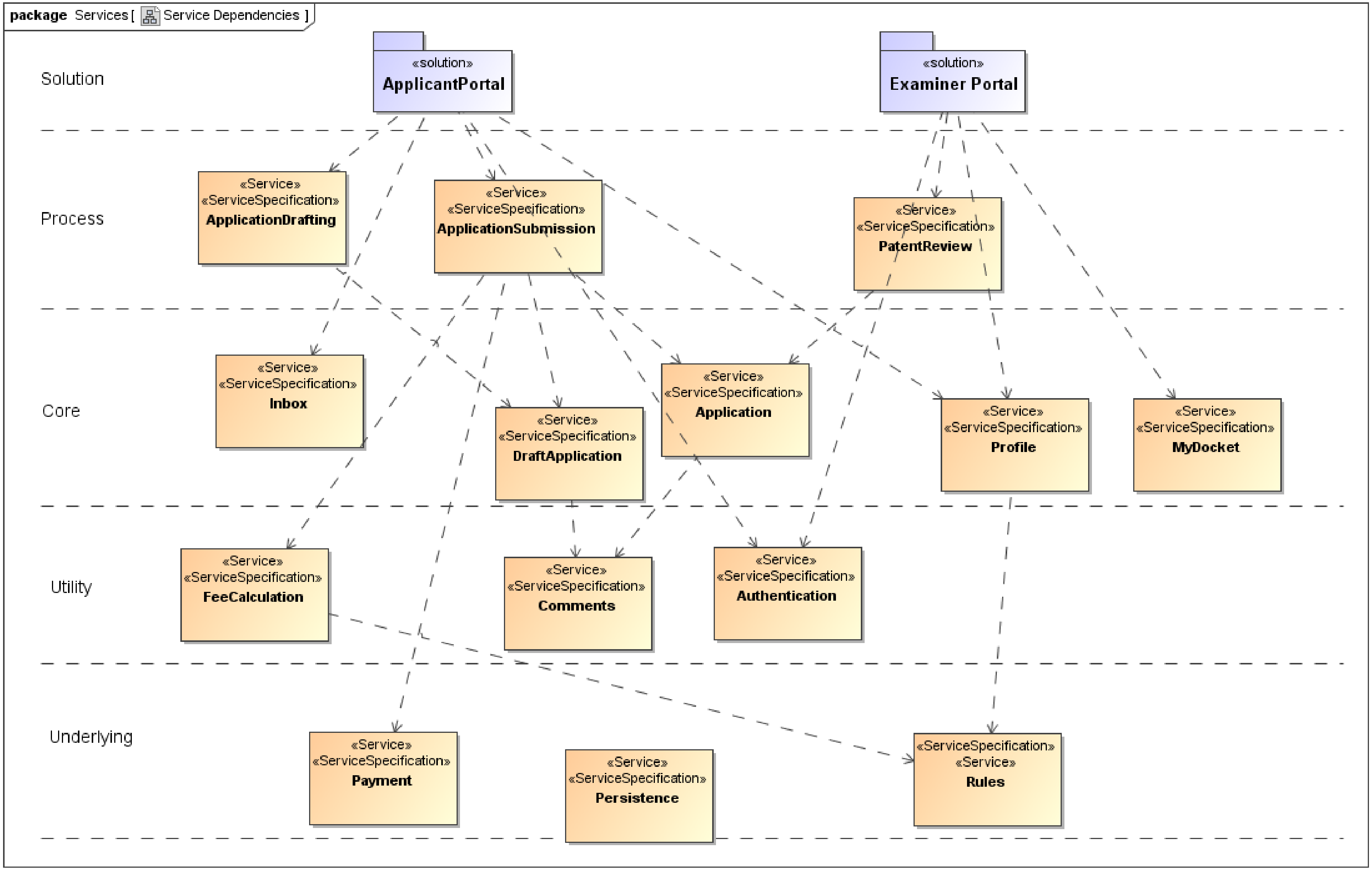


Microservice Application

Microservices at scale require *architecture*.

Derived from: Martin Fowler, et. al.

Layered Service Architecture – Drives Consistency & Sharing


package Services [Service Dependencies]
Solution
Process
Core
Utility
Underlying
«solution» ApplicantPortal
«solution» Examiner Portal
«Service» «ServiceSpecification» ApplicationDrafting
«Service» «ServiceSpecification» ApplicationSubmission
«Service» «ServiceSpecification» PatentReview
«Service» «ServiceSpecification» Inbox
«Service» «ServiceSpecification» DraftApplication
«Service» «ServiceSpecification» Application
«Service» «ServiceSpecification» Profile
«Service» «ServiceSpecification» MyDocket
«Service» «ServiceSpecification» FeeCalculation
«Service» «ServiceSpecification» Comments
«Service» «ServiceSpecification» Authentication
«Service» «ServiceSpecification» Payment
«Service» «ServiceSpecification» Persistence
«ServiceSpecification» «Service» Rules

Separation of Concerns: *Business Platform* Abstracts Business Functionality from Technology Platform

Application Layer Contains the Custom Assets

Business Logic

Business Rules

Page Layout

Multi-Channel Access

IT Organization Owns the Business Platform

Presentation Frameworks

Services Frameworks (business, data)

Data Access Layer,

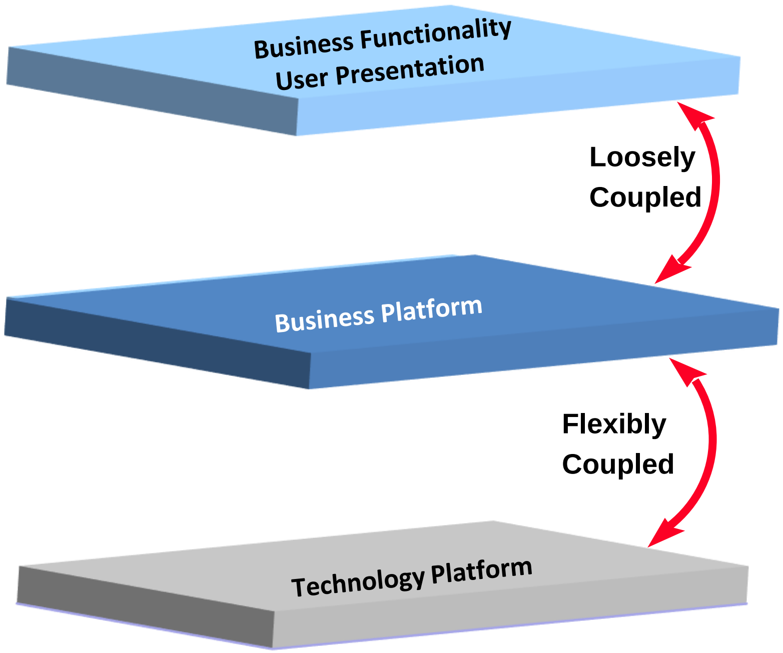
Security, Logging, Auditing,

Pattern Items & Starters

Utilities

Technology Platform Provides the Infrastructure

(Middleware, DBMS, Frameworks, Security, COTS, 3rd Party Utilities, Vendor Services, etc.)

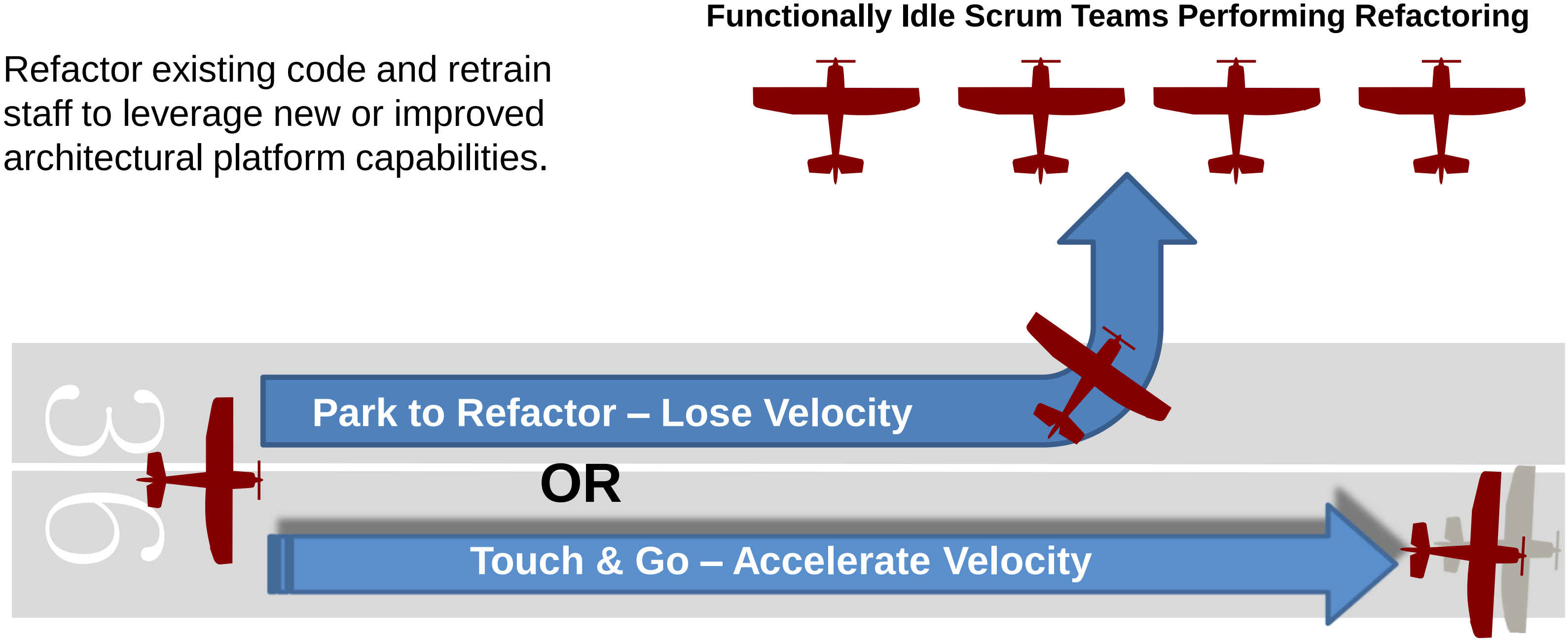


Development teams should focus on business issues only. Everything else should be part of the Platform.

The Business Platform minimizes Technical Debt.

Architectural Runway

Architectural Runway – Technical platform to support the implementation of Features and Stories by Feature Teams. The Architecture Runway is delivered on a *just-enough* basis and extended as Feature Teams require additional technical capabilities. Requires enterprise investment in refactoring and extending the platform. Ref. Scaled Agile Inc.



Functionally Idle Scrum Teams Performing Refactoring

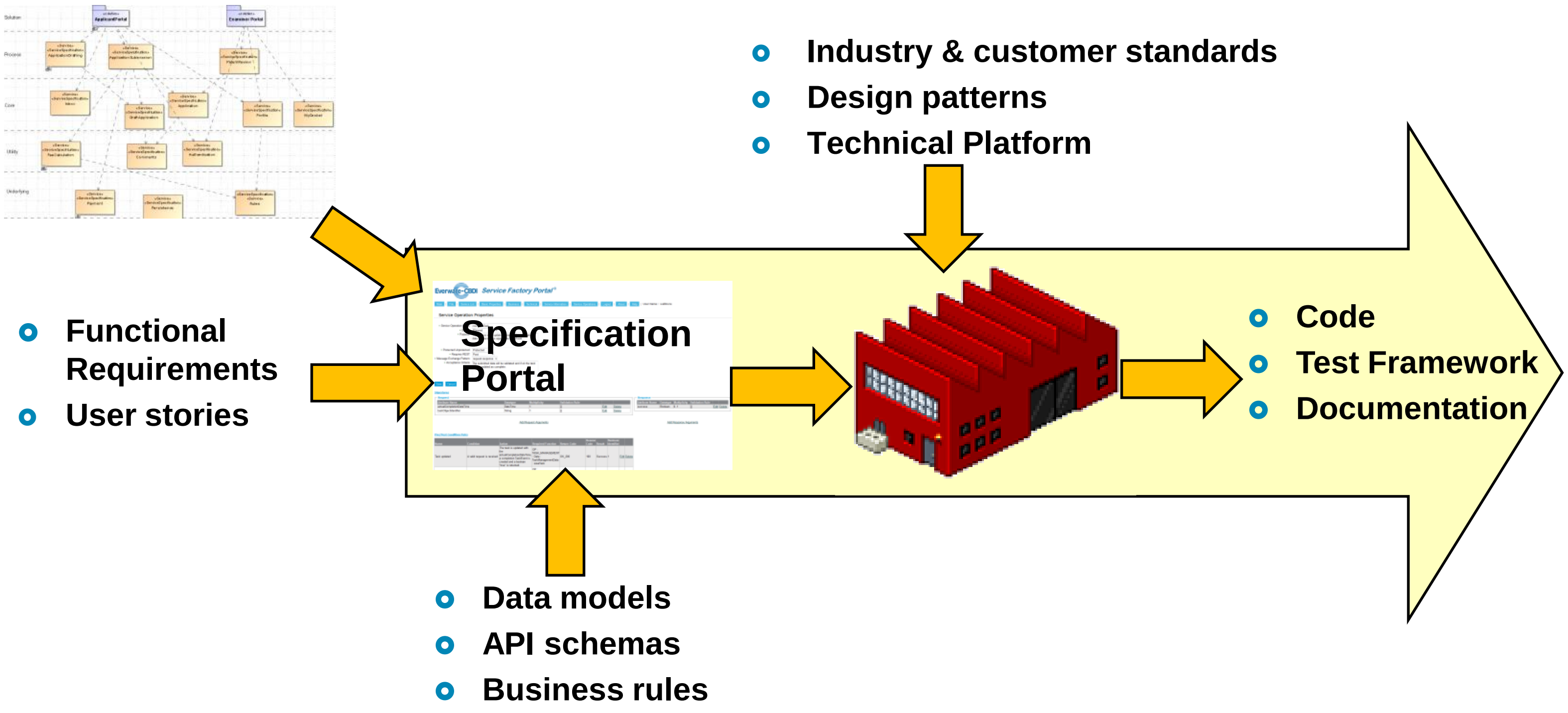
Refactor existing code and retrain staff to leverage new or improved architectural platform capabilities.

36
Park to Refactor – Lose Velocity
OR
Touch & Go – Accelerate Velocity

With the Business Platform, Feature Teams automatically “pick-up” architectural (platform) changes in the produced codebase and immediately “take off” to code remaining functional logic.

Enabling IT Agility: Model Driven Development (MDD/MBSE)

- **Agile Service Factory™**
  - Specification driven (contract/test driven)
  - Provides basis for agile collaboration with business analysts
  - Automated production of code, test framework, documentation, etc.
  - Embodies standards, industry patterns, platform specifics



- **Industry & customer standards**
- **Design patterns**
- **Technical Platform**

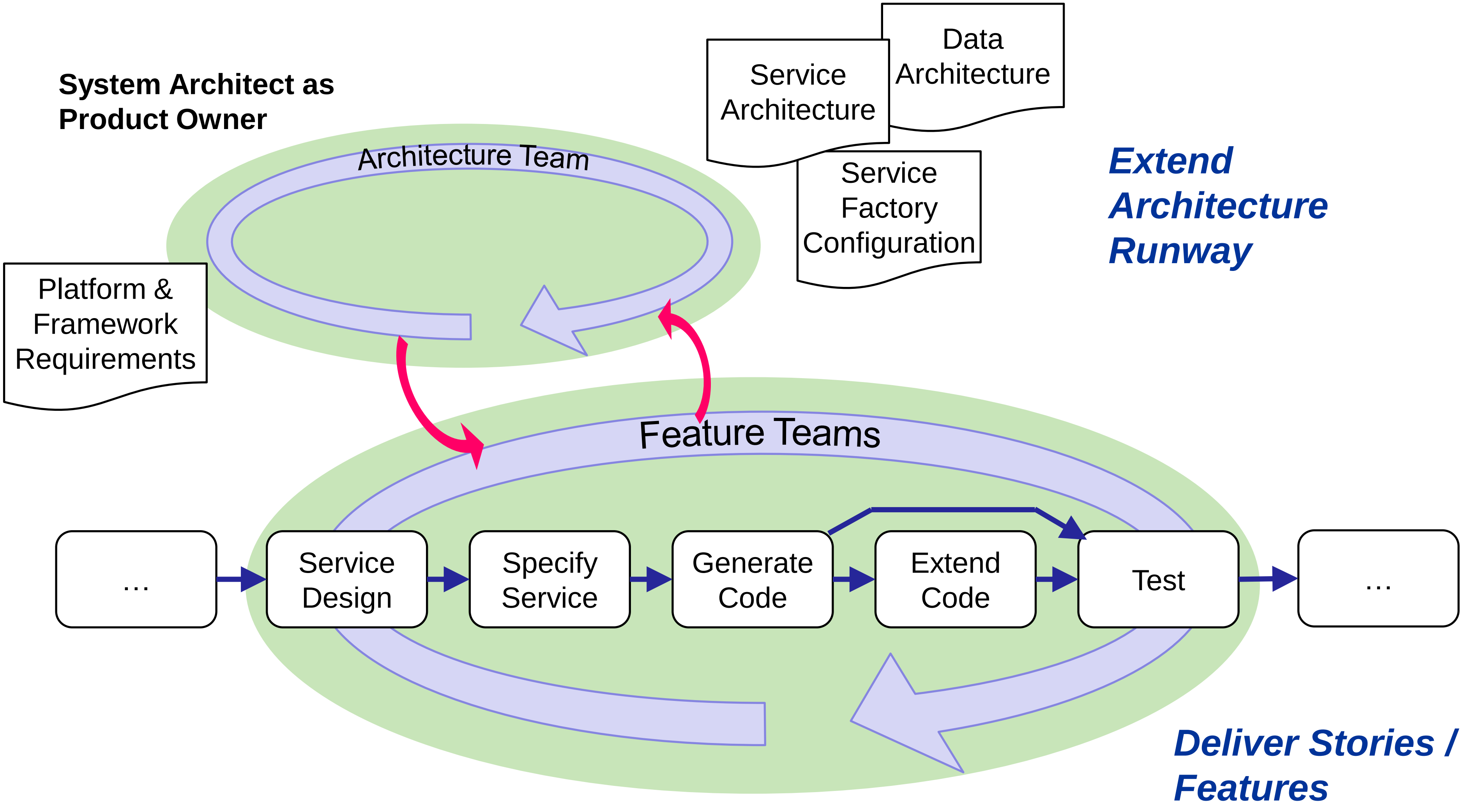
- **Functional Requirements**
- **User stories**

Specification Portal

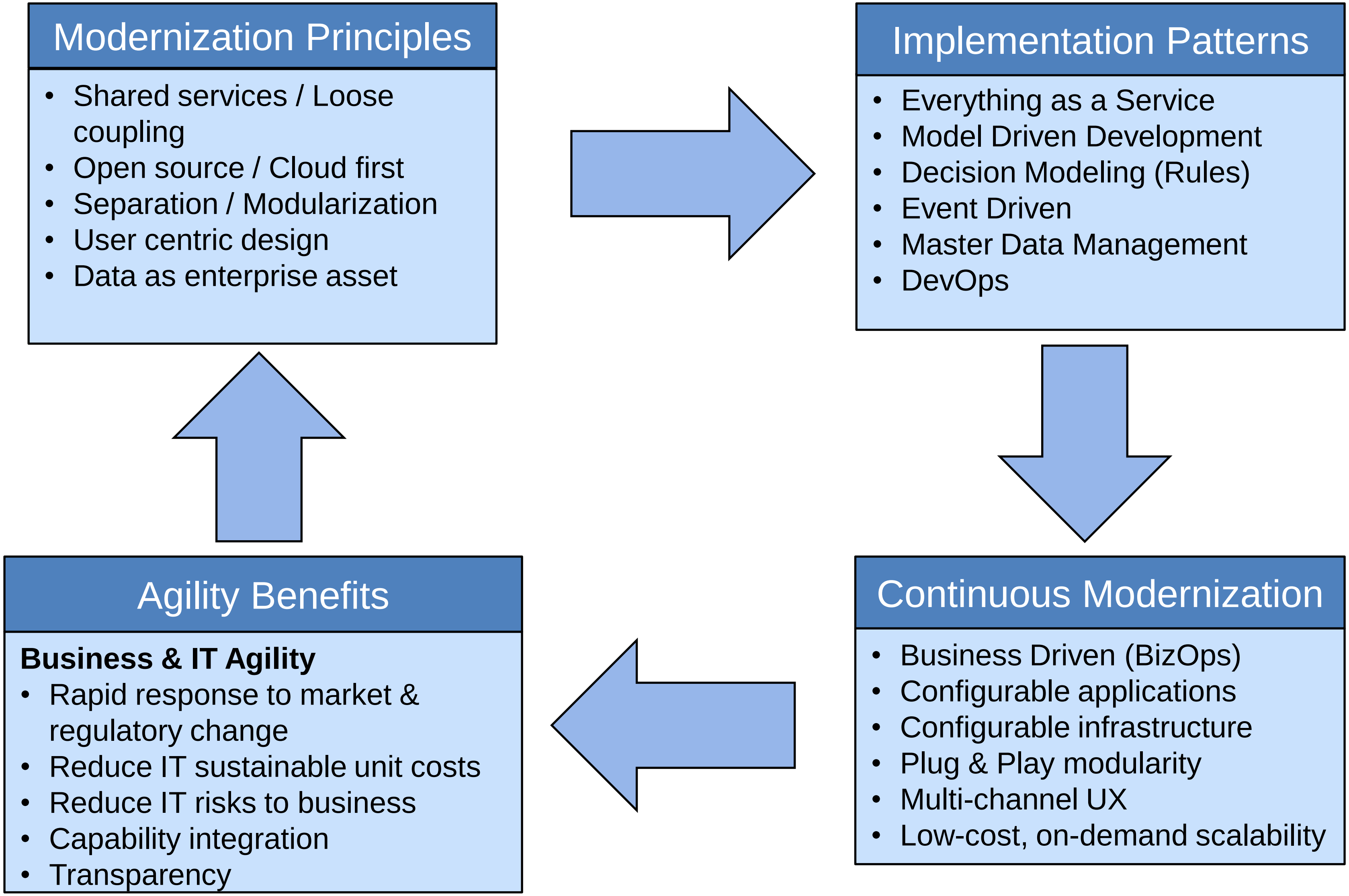
- **Code**
- **Test Framework**
- **Documentation**

- **Data models**
- **API schemas**
- **Business rules**

Agile Service Factory Process


System Architect as Product Owner
Architecture Team
Service Architecture
Data Architecture
Service Factory Configuration
Extend Architecture Runway
Platform & Framework Requirements
Feature Teams
…
Service Design
Specify Service
Generate Code
Extend Code
Test
…
Deliver Stories / Features

Continuous Modernization



Modernization Principles

- Shared services / Loose coupling
- Open source / Cloud first
- Separation / Modularization
- User centric design
- Data as enterprise asset

Implementation Patterns

- Everything as a Service
- Model Driven Development
- Decision Modeling (Rules)
- Event Driven
- Master Data Management
- DevOps

Continuous Modernization

- Business Driven (BizOps)
- Configurable applications
- Configurable infrastructure
- Plug & Play modularity
- Multi-channel UX
- Low-cost, on-demand scalability

Agility Benefits

Business & IT Agility

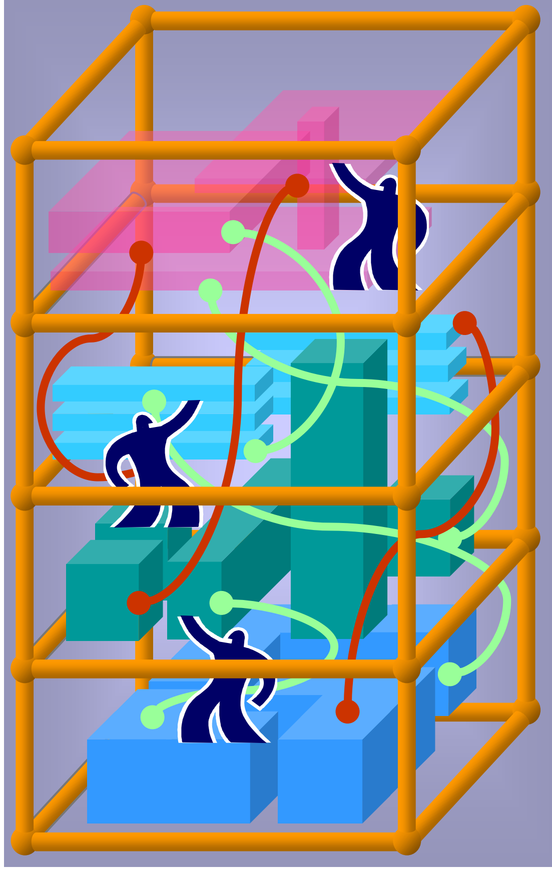
- Rapid response to market & regulatory change
- Reduce IT sustainable unit costs
- Reduce IT risks to business
- Capability integration
- Transparency

No more legacy.

Everware-CBDI

Thank You.

Questions?



www.everware-cbdi.com