Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Microservice Architecture For Integrations

2017

People matter, results count.

Purpose & Scope

- The purpose of the document is to provide a Point-of-View on Microservice Architecture for Service Oriented Architecture (SOA) & Enterprise Application Integration (EAI).
- It states some of the key architecture principles and its implications for SOA and EAI.
- These guiding principles will be applied for designing an **API-led Microservice Architecture** solution for our clients.
- It also provides mapping of Capgemini's Integration Reference Architecture and accelerators to microservice architecture strategy
- This document is not a primer for Microservice Architecture

Table of Content

Microservice Architecture (MSA)

Microservice architecture (MSA) is an architecture pattern for building and delivering service-oriented applications with two primary objectives: **agility of delivery** and **flexibility of deployment**.

- Source Gartner

MSA combines **SOA best practices** with **modern application delivery tooling** and **organizational disciplines**.

- Source Gartner

Microservices are **small**, **autonomous services** that **work together**.

— Sam Newman, Thoughtworks

Microservice Architecture Characteristics

- Componentization via services
- Organized around business capabilities
- Products not projects
- Smart endpoints and dumb pipes
- Decentralized governance
- Decentralized data management
- Infrastructure automation
- Design for failure
- Evolutionary design

— *Martin Fowler and James Levis, "Microservices", March 2014*

"SOA Done Right -

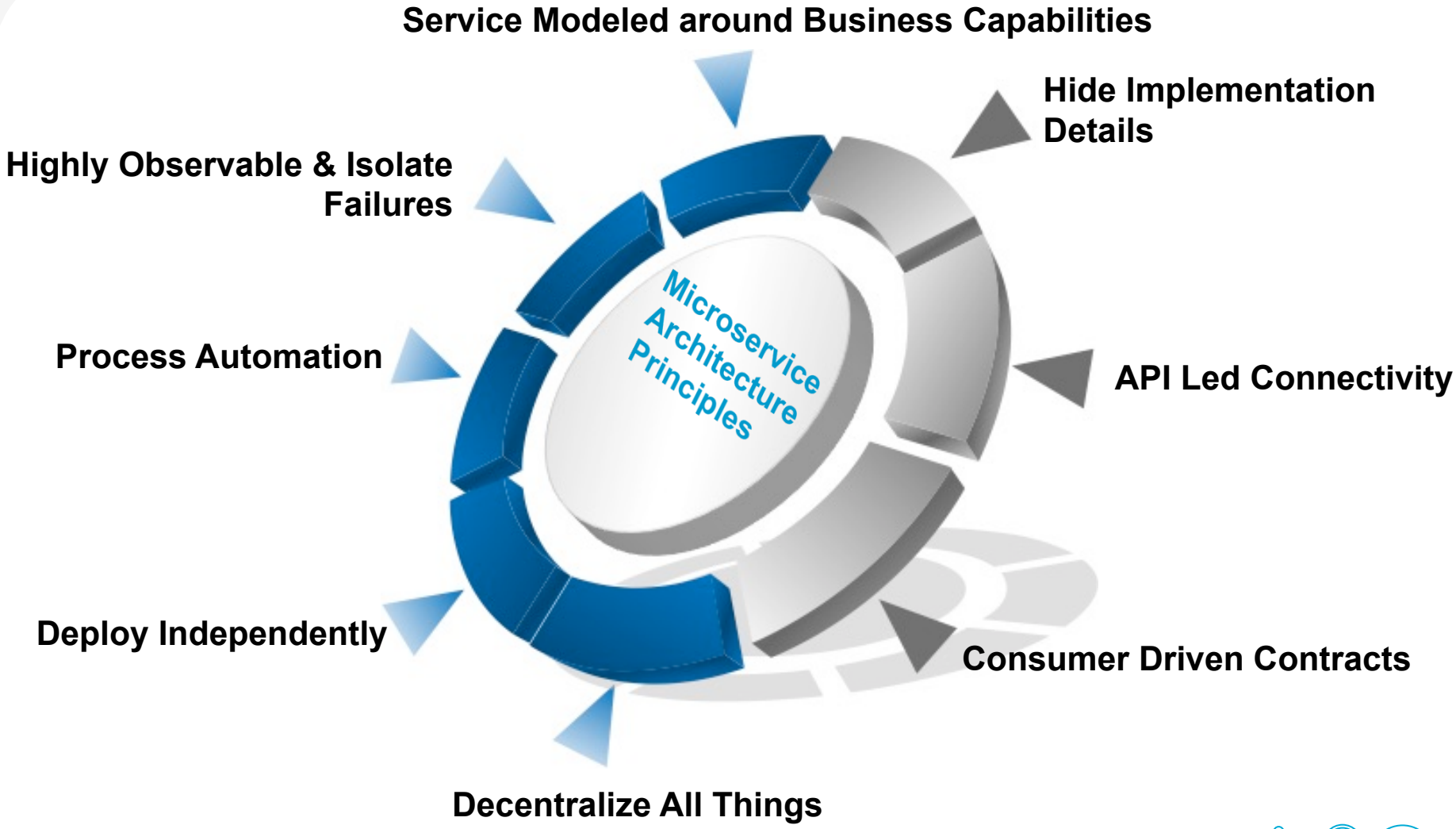
A flexible systems built as a collection of services that are built around business capabilities*, that can adapt to its complex environments, make changes without rigid dependencies "*

Key Benefits

Business Agility	▪ Enable organization to deliver business features and functions more quickly
Greater Efficiency	▪ Reduce cost in design, implementation & support ▪ Independent manageability of systems
Improved Visibility	▪ Well structured monitoring of application and platform
Improved Resilience	▪ High Availability of systems ▪ Better Scalability that allows systems to scale up/down based on needs

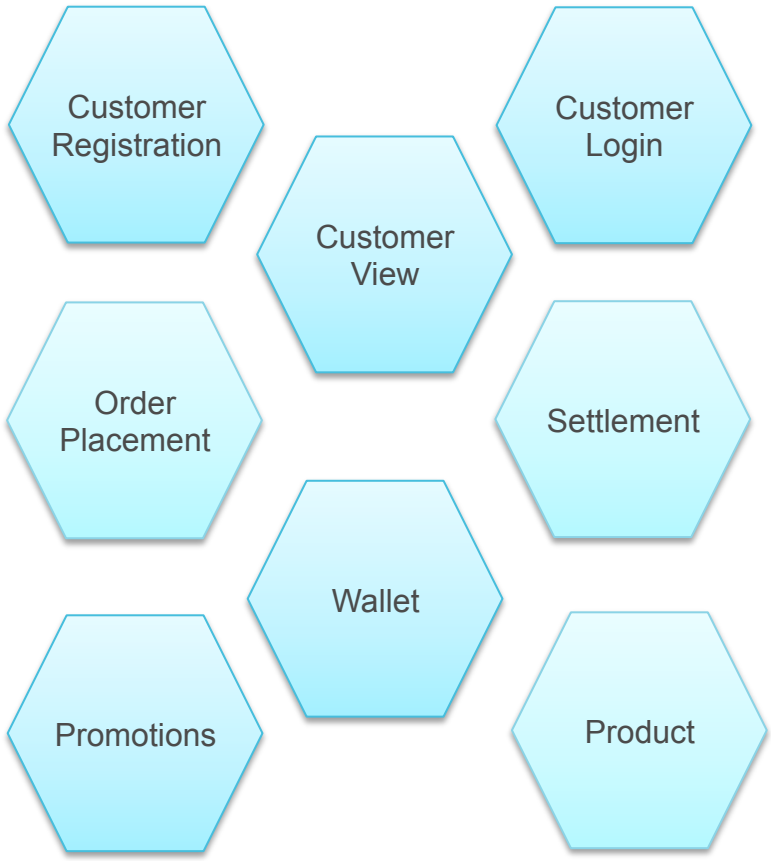
Microservice Architecture Principles For Service-Oriented Integration

Microservice Architecture Principles Overview



MSA Principle: Service Modeled around Business Capabilities

Use **Domain Driven design** to structure around **business-bounded context**



- Functional system decomposition into services organized around **business capability** that are **independently deployable** & **manageable** components
- Line of separation is along **functional boundaries**, not along tiers *i.e. vertical slicing in contrast to horizontal slicing through layers*
- Each microservice is functionally complete with **resource representation** & its **data management**

MSA Principle: Hide Implementation Details

Each Microservice is primarily an **encapsulation** of a coarse-grained business capability

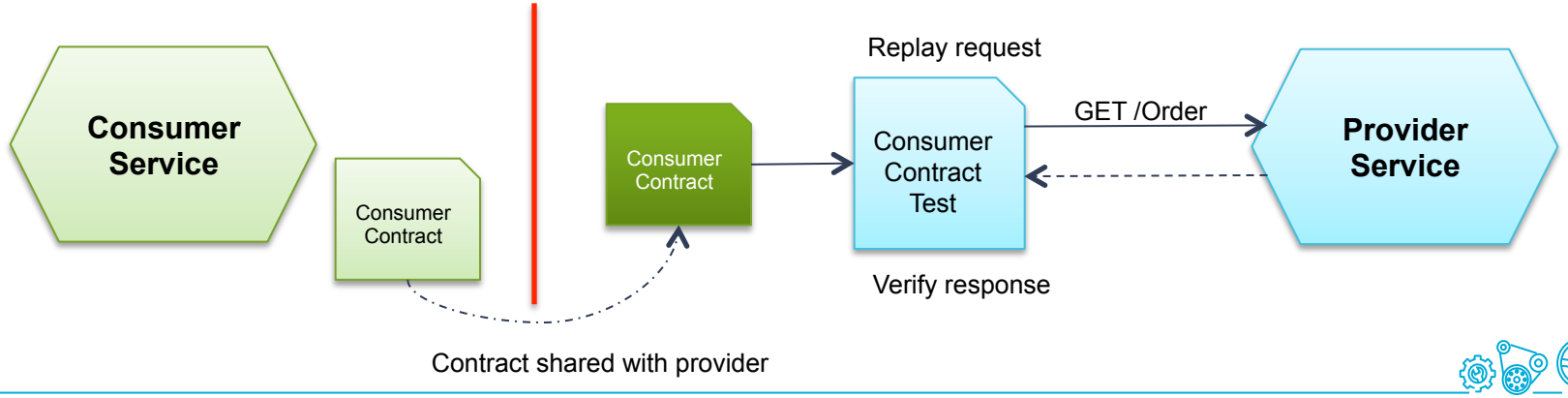
- Each Microservice should be designed with enough abstraction to hide information about technology, implementation logic, data source, physical location and **underlying systems of record**
- **Services contracts and interfaces** must not leak implementation details e.g. expose implementation types, details of technical implementation, details on data source physical location
- Each Microservice should have a well documented contracts i.e. API document or service contract documents

MSA Principle: API Led Connectivity

- Each microservice should expose a **well-defined API** that is consumed by other microservices or by application's clients.
- Communication between microservices should be through standardized communication protocol - HTTP(s), REST, JSON
  - ***HTTP*** *offers a rich set of standardized interaction mechanisms that still allow for scaling*
  - ***JSON*** *offers a simple data format that can be (partially) validated*
  - ***REST*** *provides principles and ideas for leveraging* ***HTTP and JSON to build evolvable microservice interfaces***
- For **pub-sub model**, it is recommended to have messaging over a lightweight messaging bus that offers reliable asynchronous fabric. The endpoints producing and consuming messages should have smart processing logic i.e. services
- Use tools such as Swagger or RAML for API documentation

MSA Principle: Consumer Driven Contracts

- **Consumer-Driven Contract** pattern helps Microservice provider create service APIs that reflect client needs & evolve services without breaking existing clients.
- Service owners should have consumer's integration tests from each client and incorporate these tests into the service's test suite.
- The set of integration tests received from all existing clients represents the service's aggregate obligations with respect to its client base.
- The service owner is then free to change and evolve the service just so long as the existing integration tests continue to pass.



MSA Principle: Decentralize All Things

Each microservice should be implemented on its own technology stacks that fits the task best. This implies:

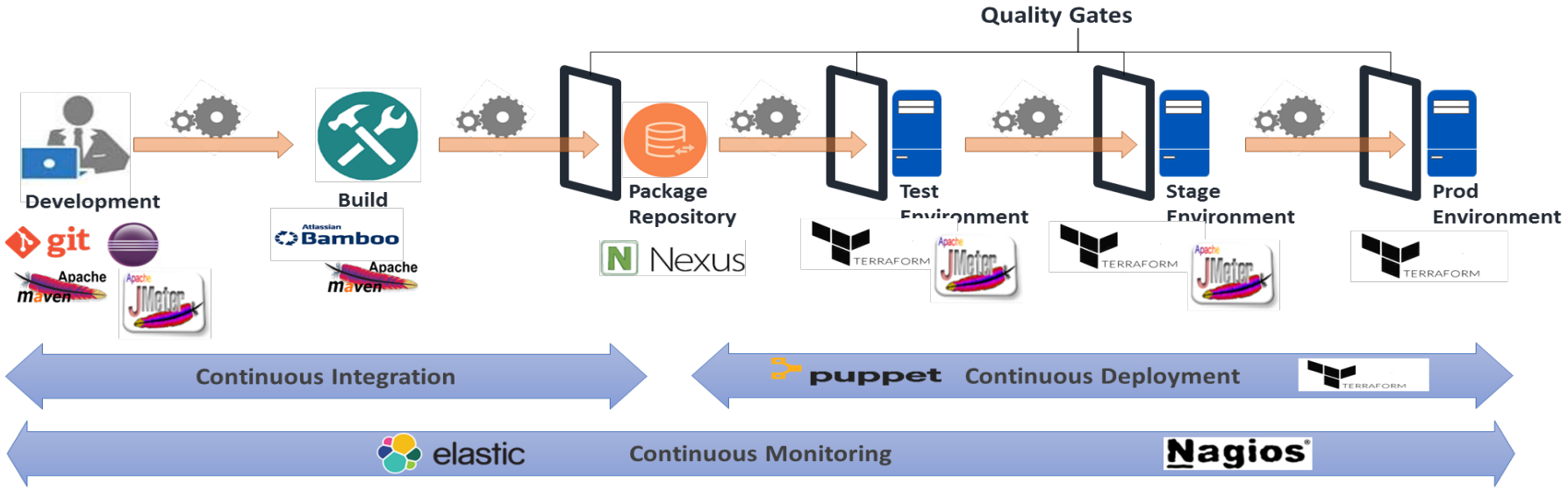
- **Different Programming language** that fits the need
  - No accidental cross-dependencies between code bases
- **Decentralize technology Governance** –
  - Focus on standardizing the relevant parts (communication protocol, message format and communication pattern) and leverage best suited technology for implementing business capability
- **Decentralize Data management** - Each service can choose the persistence solution that fits best
  - Data access patterns
  - Scaling and data sharding requirements

MSA Principle: Deploy Independently

- **Independent Code base** for each microservice that can be deployed independently without impacting other microservice
- **Location Independence** for each microservice i.e. cloud /on-premises or in hybrid model
- **Independent scaling** (horizontal & vertical) of each microservice to fit business needs
- **Automate deployment** of applications inside independent portable **containers**
- **Service Versioning**
  - **Co-exist of service endpoints** with different versions.
  - APIs should be backward compatible in order to enable change

MSA Principle: Process Automation

- **Continuous integration -** automation in Build and deployment process.
- **Automated testing** – unit and regression testing
- **Monitoring** – automated failure detection
- **Support for Containerization**
- **Support for DevOps automation**

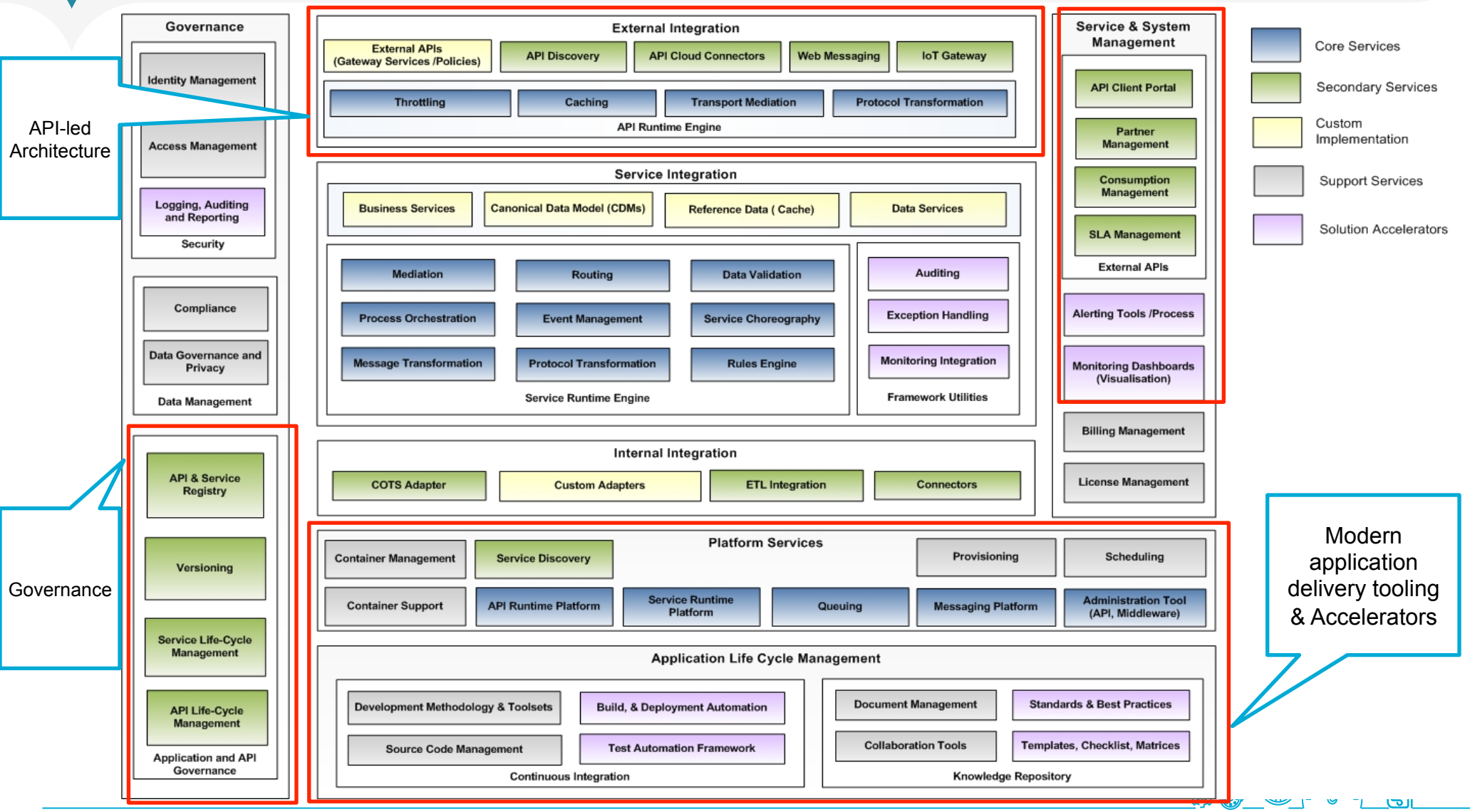


MSA Principle: Highly Observable & Isolate Failures

- **Use of centralised monitoring solution** that works for all technologies within Microservice platform
- **Monitoring Solution** should provide real-time statistics for key business services, feeds and business processes
- Monitoring Solution should provide **end-to-end business activity monitoring** for key business process
- Provision to **analyze performance** – message throughput and latency & reliability of middleware IT platform based on service level agreements and proactively take corrective actions ensuring better compliance
- Use of **correlation id** for tracking overall flow across microservices
- Individual services should be able to isolate its failure from other services and consumers

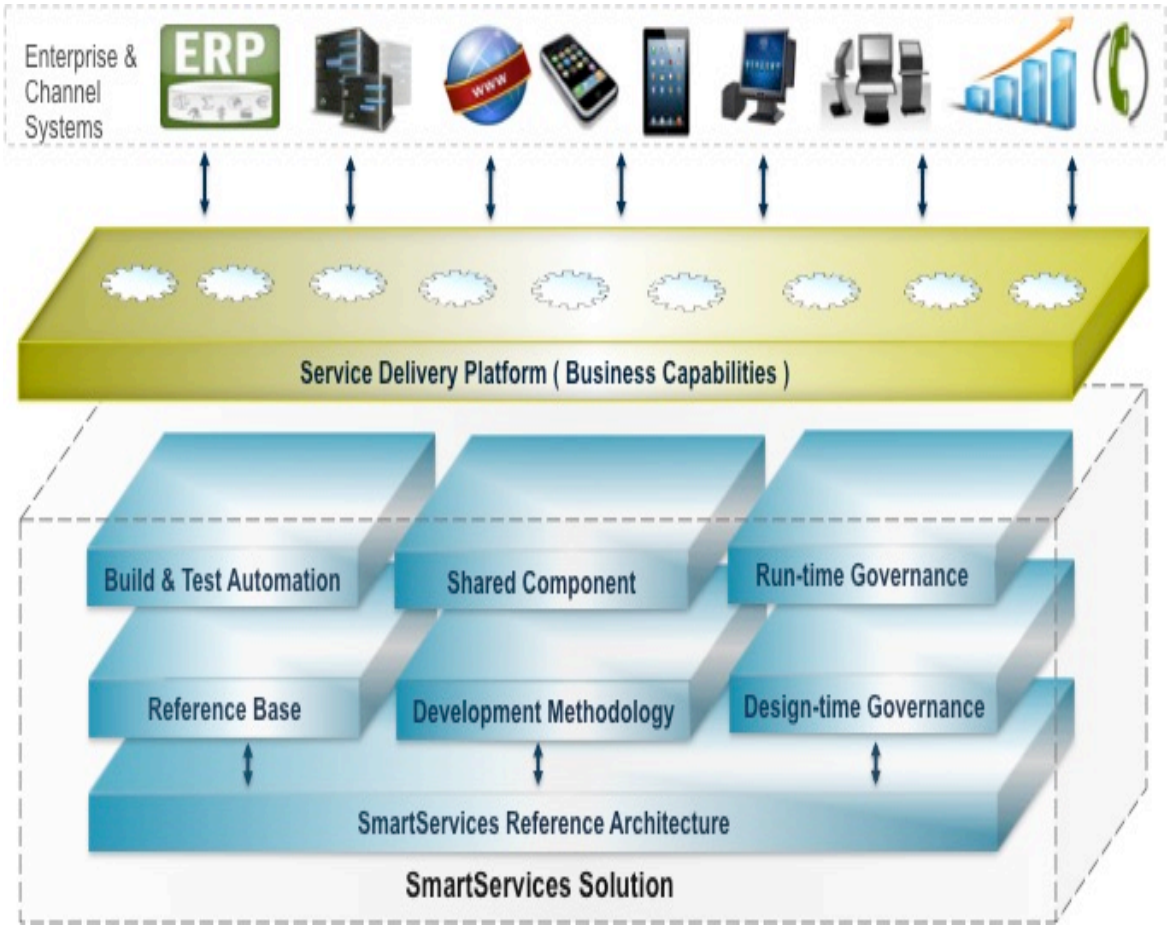
Capgemini’s Integrations Solution mapping to MSA Strategy

Capgemini's Integration Reference Architecture – Comprehensive Integration Strategy aligned to API-based MSA



Integrations Accelerators – Capgemini SmartServices Framework

***SmartServices** is our Integration foundation framework for API and microservices based Enterprise Integration Service Delivery platform*



SOA with DevOps Model:

- Information **Reference Base**
- Technology-specific **Integration Shared Component**
- **Build & Test Automation Framework**
- **Monitoring Solution**
- **Design-time Governance**
- **Integration Development Methodology**

Case Study

Ladbrokes - Integration Competency Centre using Microservice Strategy

Requirements

- ***Migration of a point-to-point legacy integration solution on to a strategic enterprise integration platform***
- IT shared service to cater for business change demands
- Development of SOA-aligned Integration services for their key business programs
- Enterprise Service Management
- Application Support & Environment Management

Solution Approach

- Migrated legacy integration solution to new Strategic enterprise middleware platform
- ***Delivered API-led Microservices based Middleware platform***
- Implemented Ladbrokes core Integration framework
- Providing end-to-end Enterprise Integration services
- Integration DevOps model
- ***Application Life Cycle & Platform Management using Atlassian and opensource toolset***
- Enterprise Agile Methodology

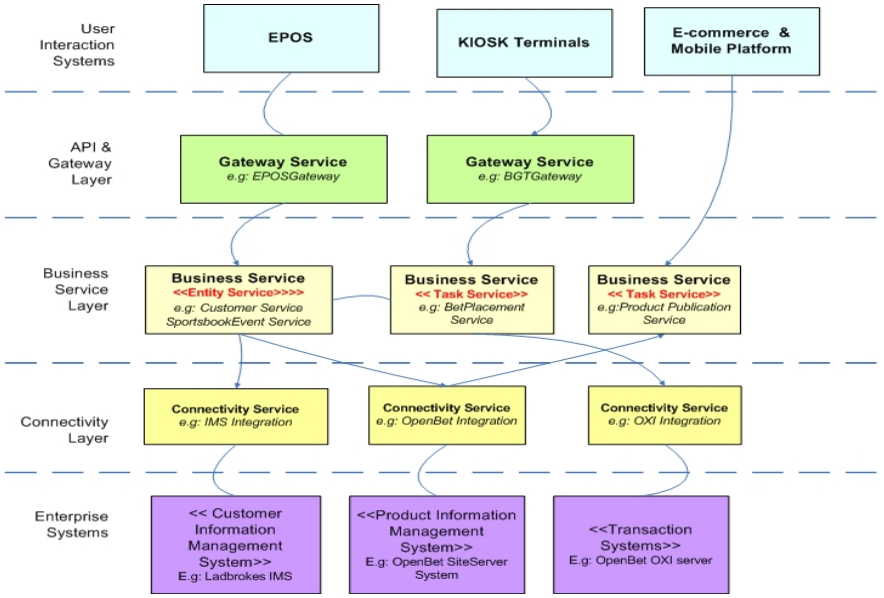
Benefits

- ***Reduced costs of business change function & maintenance by 20% - 30%***
- 40% service reusability
- Improved ROI through creation and reuse of framework and services
- Average Response time for processing <500ms with 50% YoY growth in traffic
- Peak load of 34M transaction /day
- ***100% Availability for the last 2 years***

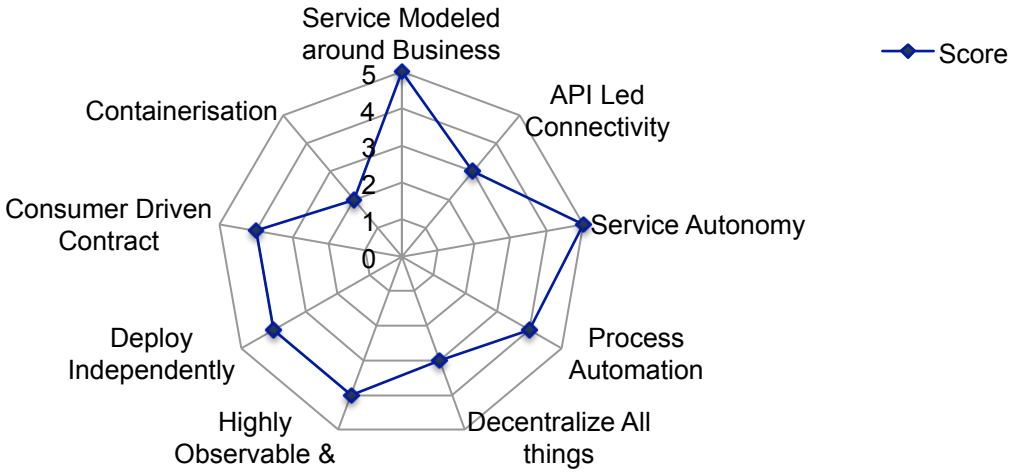
Ladbrokes - Integration Competency Centre using Microservice Strategy

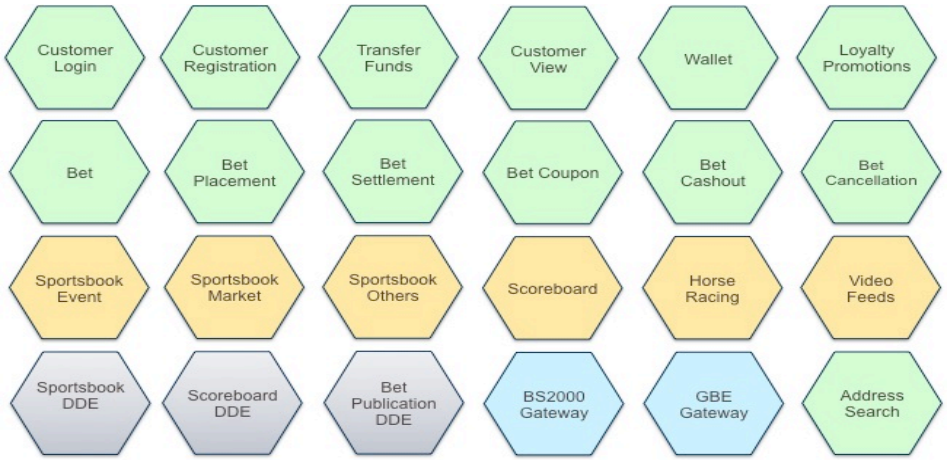
Microservice Highlights

- Domain driven Business-aligned services
- Services autonomous
- API driven architecture
- Process Automation in-place
- Well-defined monitoring solution and isolated failure



Microservice Readines





Capgemini

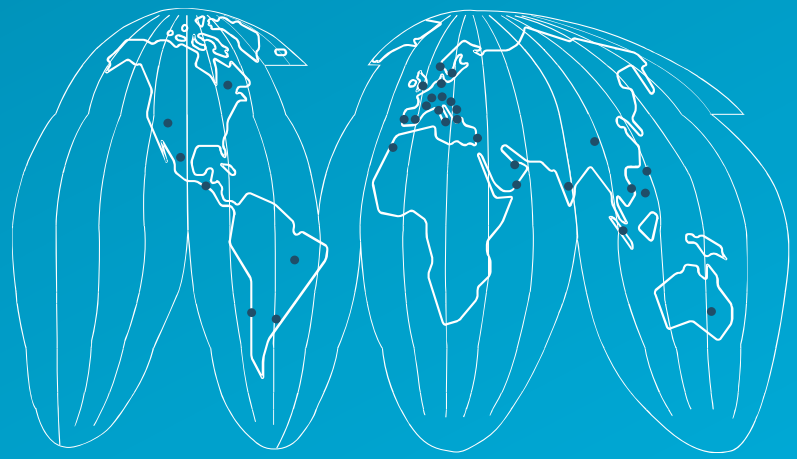
CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.

About Capgemini

With more than 180,000 people in over 40 countries, we are one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2015 global revenues of EUR 11.9 billion.
Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organisation, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.
Learn more about us at www.capgemini.com.

Rightshore® is a trademark belonging to Capgemini.



www.capgemini.com