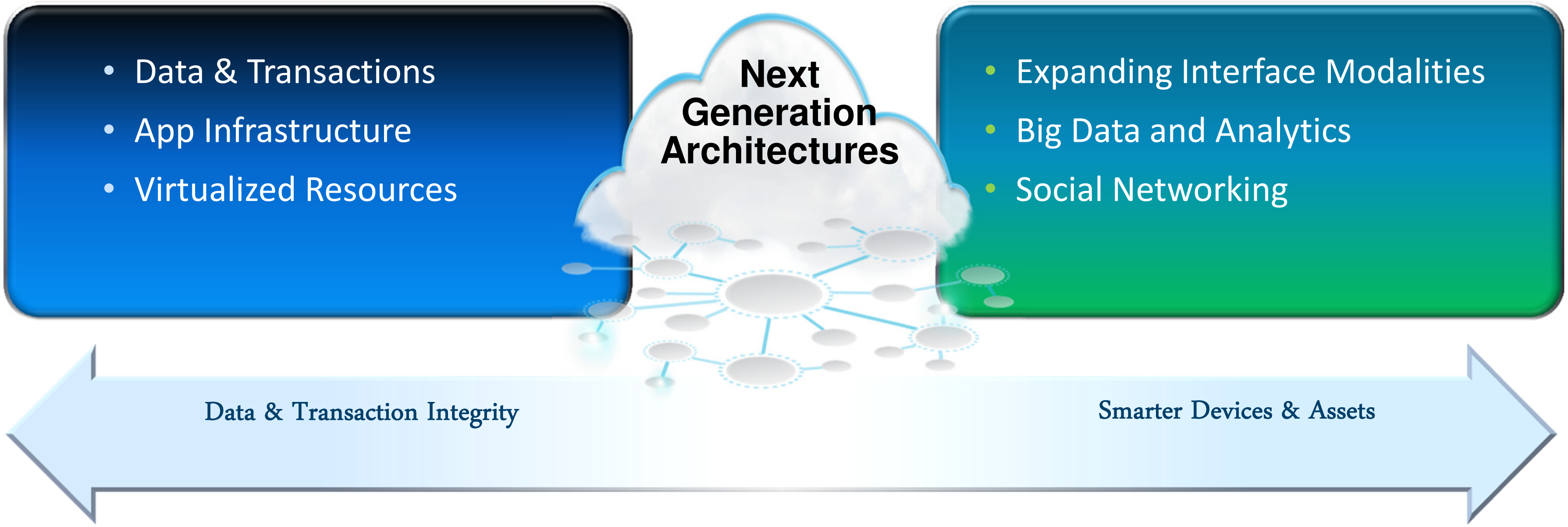
Michael Behrendt (michaelbehrendt@de.ibm.com)
Senior Technical Staff Member
Cloud & Smarter Infrastructure CTO Office

Cloud Native Applications

A new class of applications is emerging



Systems of Record

- Data & Transactions
- App Infrastructure
- Virtualized Resources

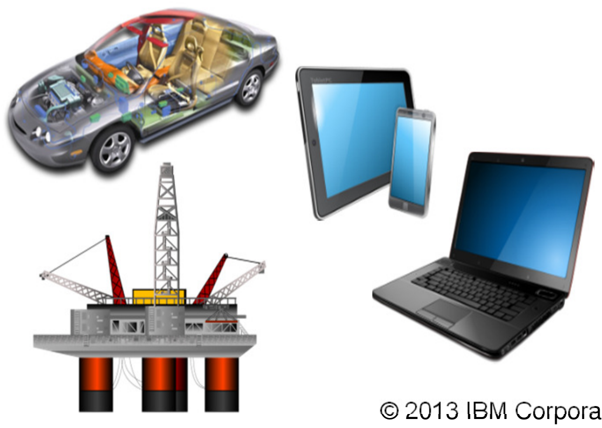
Systems of Engagement

- Expanding Interface Modalities
- Big Data and Analytics
- Social Networking

Next Generation Architectures

Data & Transaction Integrity

Smarter Devices & Assets



This transformation is impacting many industries

INSURANCE

- LOB need for new solutions to “..get closer to their customers…”
- Address millennial generation of customers and interaction models **(social, mobile)**
- Enhance current Sales System with a multi-channel integration system that provides for sales (quoting) and service of all products to Agents, Call Centers and direct to Policyholders

RETAIL

- LOB need for new solutions to engage customers in-store and over web channels
- Address customer acquisition, customer retention, customers interaction in-store (coupons, promotions) and metrics such as average revenue per user **(social, mobile, analytics)**
- Enhance current retail systems with a multi-channel interaction

GOVERNMENT & PUBLIC SECTOR

- New solutions to engage citizens driven by Smarter Cities & Government
- Address citizen interaction with local government resources **(social, mobile, analytics)**
- Integrate current systems (e.g. work order management systems) with a multi-channel interaction leveraging GPS, GIS and mobile devices

IBM Social Business

- Making the work environment for sellers & sales managers simpler, social, more integrated, and insightful…”
- Applications that utilize CRM tools and integrates IBM Sales tools to deliver an integrated solution
- Enhanced with social network mapping and expertise location (e.g. LinkedIn)
- Integrating CRM applications with **social, mobile and analytical capabilities**

Example: Web start-up – Design Philosophy and Evolution

Design Philosophy:

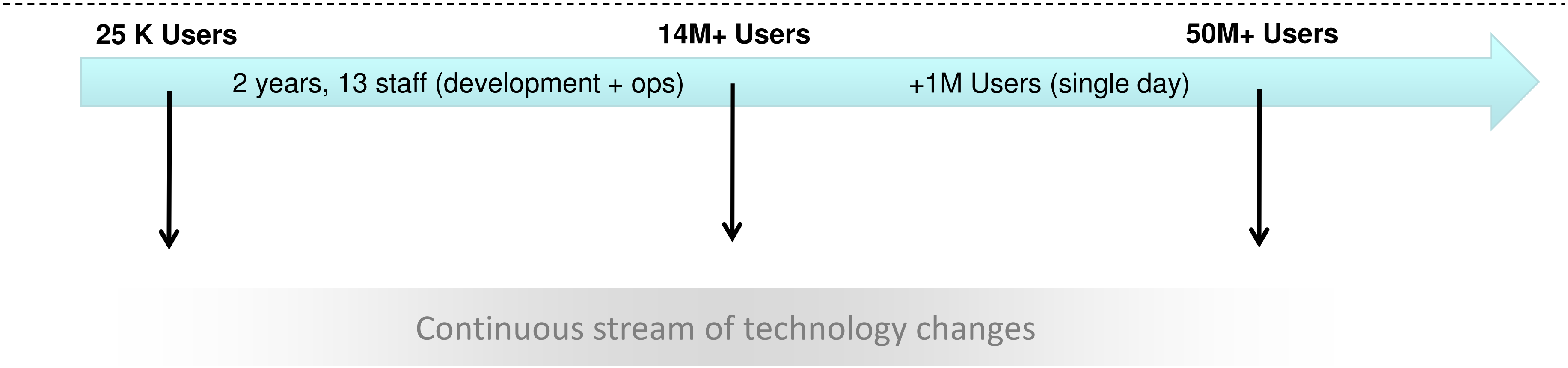
- Simplicity
- Optimize for operational burden
- Continuous updates with continuous availability
- Instrument everything

Development Philosophy:

- Extensive code reviews, unit and functional tests
- Loose coupling using notification/signals
- Do most work in Python; C when necessary
- Extensive monitoring

Business Impact

- Solution evolved & changed with the business
- Architecture re-evaluated constantly in relation to business goals
- Progressive composition of services
- Majority of development focus on creating business value



25 K Users

14M+ Users

50M+ Users

2 years, 13 staff (development + ops)

+1M Users (single day)

Continuous stream of technology changes

There will be a continuum of Systems of Engagement & Systems of Record

Client Applications

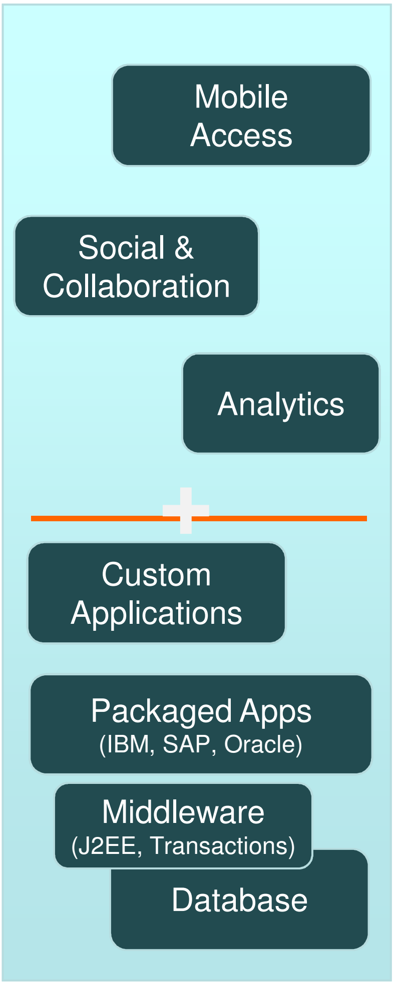
New workloads

Cloud Centric Systems of Engagement

+

Existing Workloads

Cloud Enabled Systems of Record



- Hadoop
- PHP
- Cassandra
- Nginx
- Munin
- …

- Optimized for agility and velocity of developing new apps
- Driven by market needs vs. technology
- Focuses on progressive composition

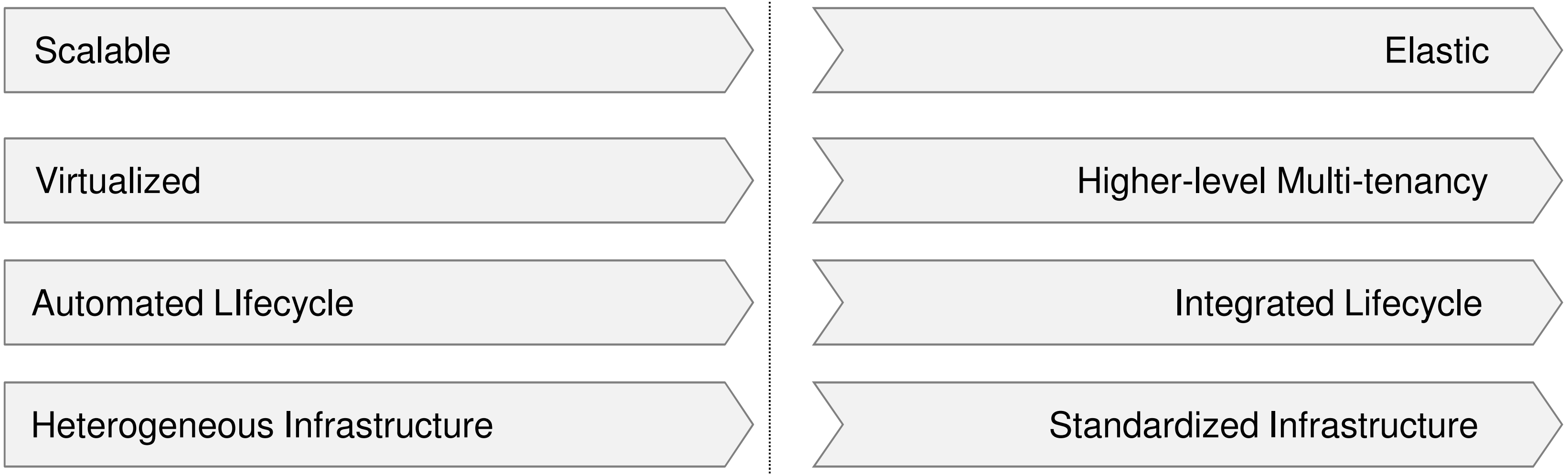
- BPM
- JEE
- CICS/IMS
- Portal
- …

- Workload optimizations
- Optimized for reducing OPEX for existing patterns
- Significant body of automation and integration content for existing workloads (SAP, Oracle etc.)

Cloud Services Spectrum

Cloud Enabled Workloads

Cloud Centric Workloads



Scalable

Virtualized

Automated LIfecycle

Heterogeneous Infrastructure

Elastic

Higher-level Multi-tenancy

Integrated Lifecycle

Standardized Infrastructure

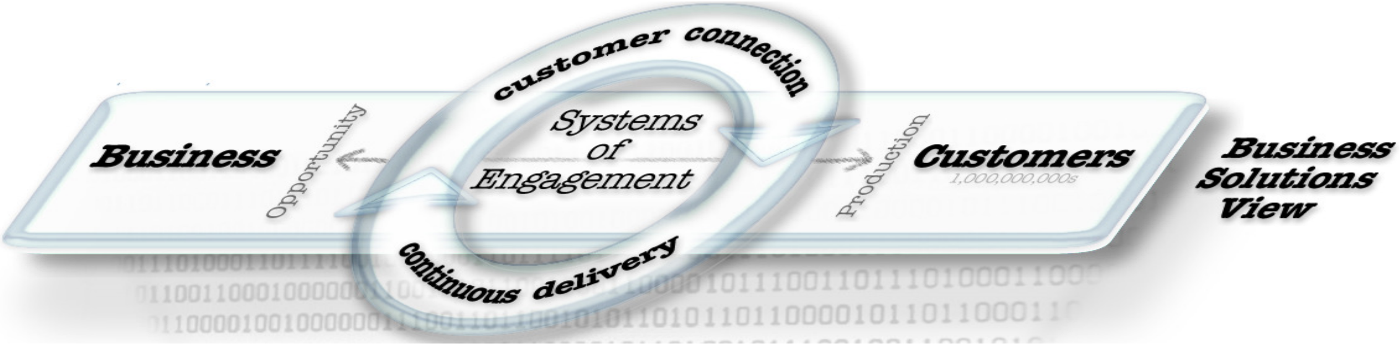
Existing
Middleware
Workloads

Emerging
Platform
Workloads

Compatibility with existing systems

Exploitation of new environments

Comparing Systems of Record and Systems of Engagement



Capabilities and User Experience	Existing	Emerging
Primary Workload Types	Systems of Record (Transactional)	Systems of Engagement (+ Record) (Big Data, Analytics, Mobile/Social Channels)
Delivery Model	Planned	Incremental (DevOps)
Development and Operations Team Sizes	100s and Costly	10s with built-in DevOps automation
Release Frequency	Months to Years	Days to Weeks, based on business opportunity
Integration Frequency	Weeks	Continuous
Infrastructure Deployment	Days	Minutes
Time to Value	Planned	Opportunistic
Operational Model	Systems Management	Built in to application, Recovery Oriented Computing, Continuous Availability
Service Sourcing	Develop	Consume and Assemble (Public and Private)

Attributes of cloud-native software (1)

Modular, non-monolithic architecture	• System consists of many small, self-contained components • Each component / service developed and operated by a self-contained team
Built-in Multi-tenancy	• The higher the level of multi-tenancy, the more efficient the app is
Horizontal Scalability	• Each component can be scaled horizontally
Elastic	• Can flexibility grow and shrink depending on load or other needs
Clean Separation of stateful from stateless components	• Stateless components can be treated more efficiently than stateful ones
Loosely coupled	• Non-blocking communication • Components can come up at any point in time • Since anything can fail anytime (and be restarted elsewhere), avoid startup dependencies

Attributes of cloud-native software (2)

Design for failure	• The larger a distributed app is, the more likely it is that a failure occurs for one of its components • Avoidance of single points of failure • Code assumes any node can fail at any point in time • All application components should be resilient to reboots
Granular failure	• In case one component of an application fails, the remaining ones should continue to be available
Accessible via idempotent APIs	• API-based accessibility for app an • Both usage and lifecycle mgmt API
Developed in a "hosted-first" model	• Development and operations covered by a single team, owning the application e2e – from dev to ops • Typically very small teams • Updates are applied very frequently (vs. big shipments every 6-9 months) • Management concerns need to be addressed by the application from the beginning
Easy-of-use / self-service	• In today's cloud-native world it's critical to allow achieving early success very quickly – without requiring the user to read lots of documentation

Considerations around attributes of cloud-native software

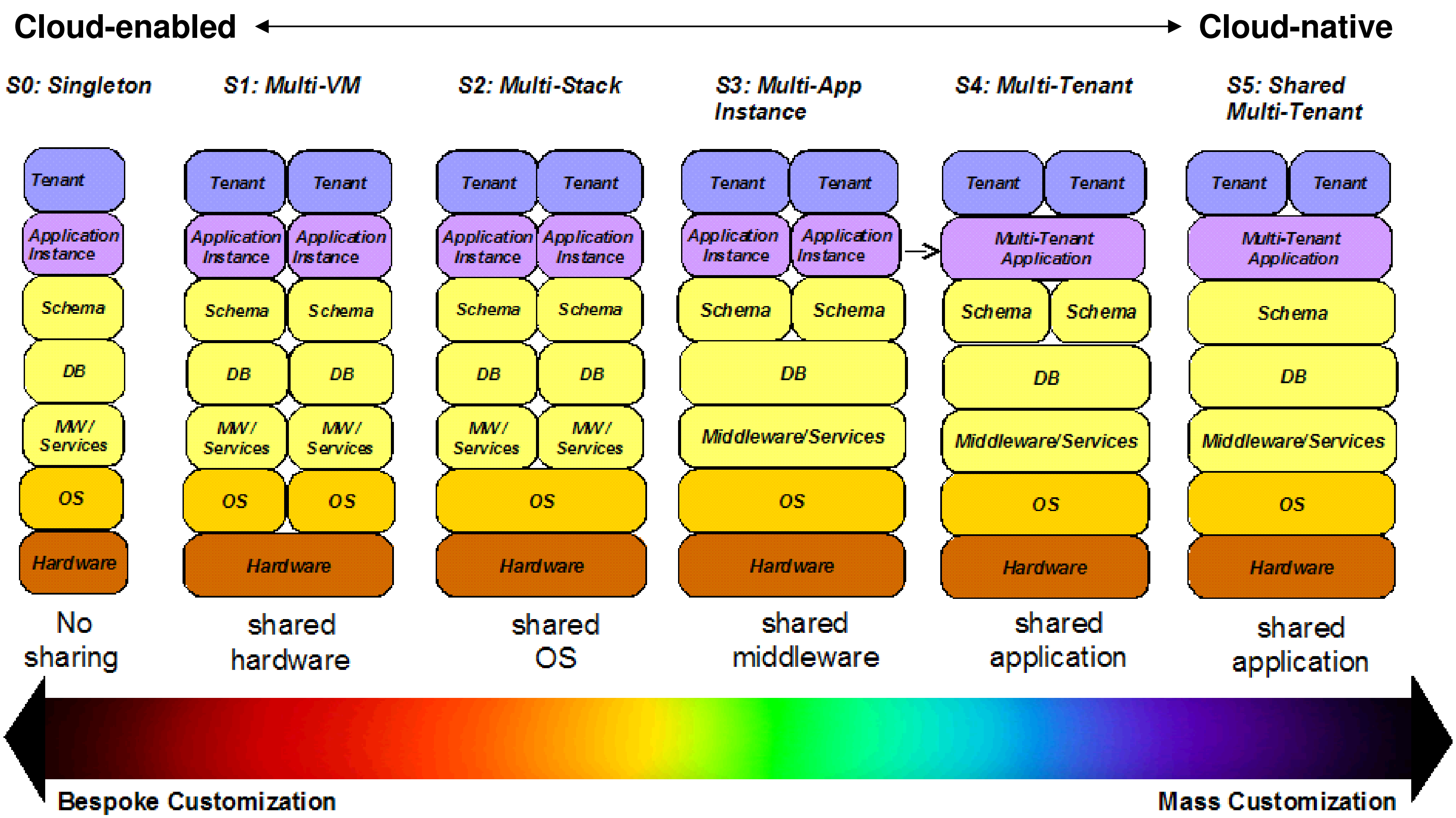
- Several cloud-native attributes are not fundamentally new (strong SOA-heritage), however they’re getting increased focus in a cloud-native world
  - Enormous scalability requirements for publically accessible applications
  - If apps run on large numbers of machines, statistically some of them are going to break on a daily basis – irregardless of the robustness of a single machine

    → Increasing need to implement QoS on the application level

- Re-architecting “legacy” SW to implement cloud-native attributes can be complex
  - Often more appropriate to apply them to new developments
  - Cloud-native attributes lends themselves between towards interaction-centric nature of systems of engagement vs. systems of record

- Implementing cloud-native attributes is not an “all-or-nothing” decision – the more cloud-native attributes a SW implements, the more “cloud-native” it is

Multi-tenancy models



Cloud-enabled ← → Cloud-native

S0: Singleton

Tenant

Application Instance

Schema

DB

MW/ Services

OS

Hardware

No sharing

S1: Multi-VM

Tenant | Tenant

Application Instance | Application Instance

Schema | Schema

DB | DB

MW/ Services | MW/ Services

OS | OS

Hardware

shared hardware

S2: Multi-Stack

Tenant | Tenant

Application Instance | Application Instance

Schema | Schema

DB | DB

MW/ Services | MW/ Services

OS

Hardware

shared OS

S3: Multi-App Instance

Tenant | Tenant

Application Instance | Application Instance

Schema | Schema

DB

Middleware/Services

OS

Hardware

shared middleware

S4: Multi-Tenant

Tenant | Tenant

Multi-Tenant Application

Schema | Schema

DB

Middleware/Services

OS

Hardware

shared application

S5: Shared Multi-Tenant

Tenant | Tenant

Multi-Tenant Application

Schema

DB

Middleware/Services

OS

Hardware

shared application

Bespoke Customization

Lower Development Cost

Greater Resource/Security Isolation

Faster Launch / Time to Market

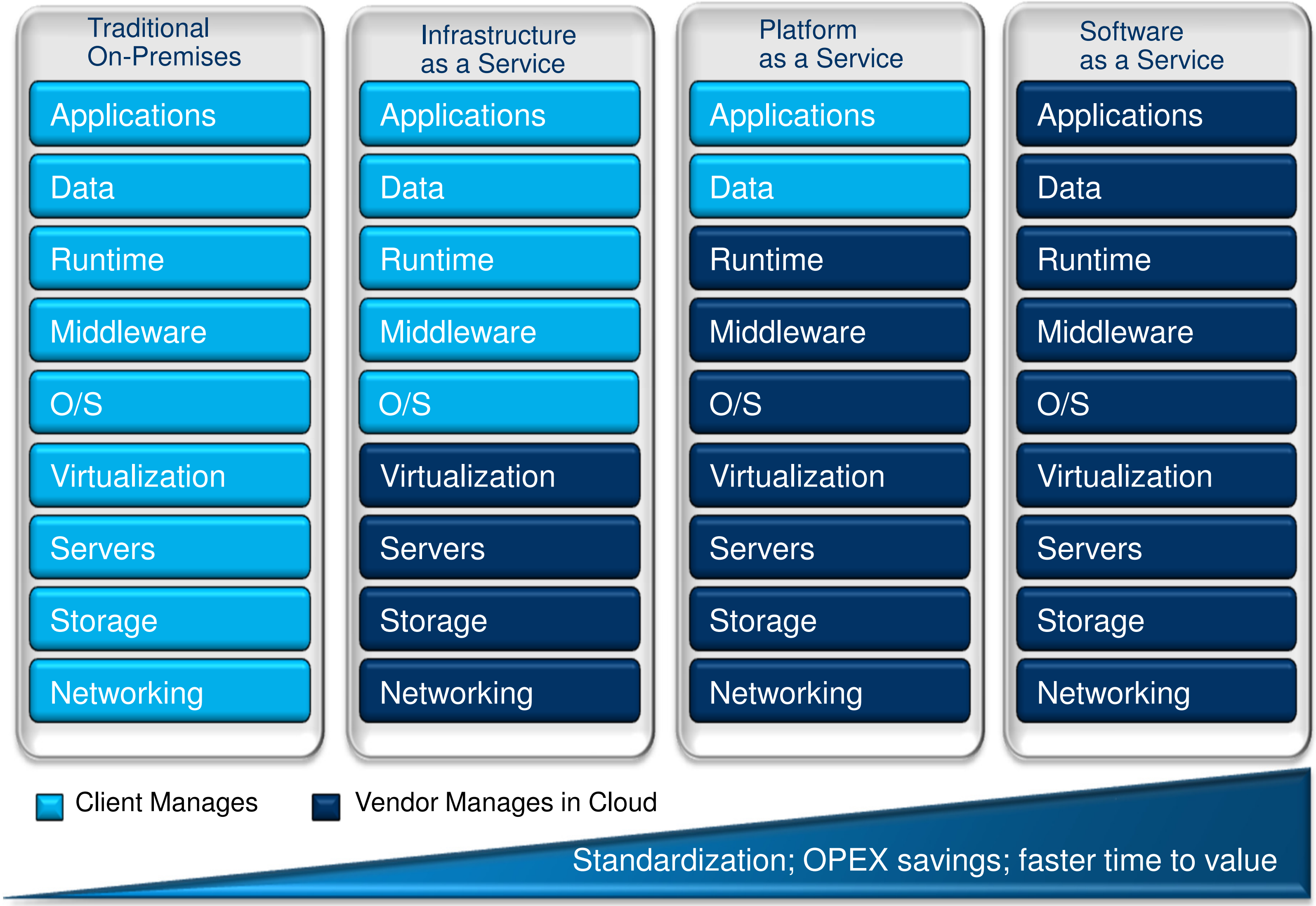
Mass Customization

Lower Operating Cost

More Sharing

Faster Iteration / Time to Value

DevOps as a key principle for cloud-native apps: Management concerns need to be factored into the app from the beginning

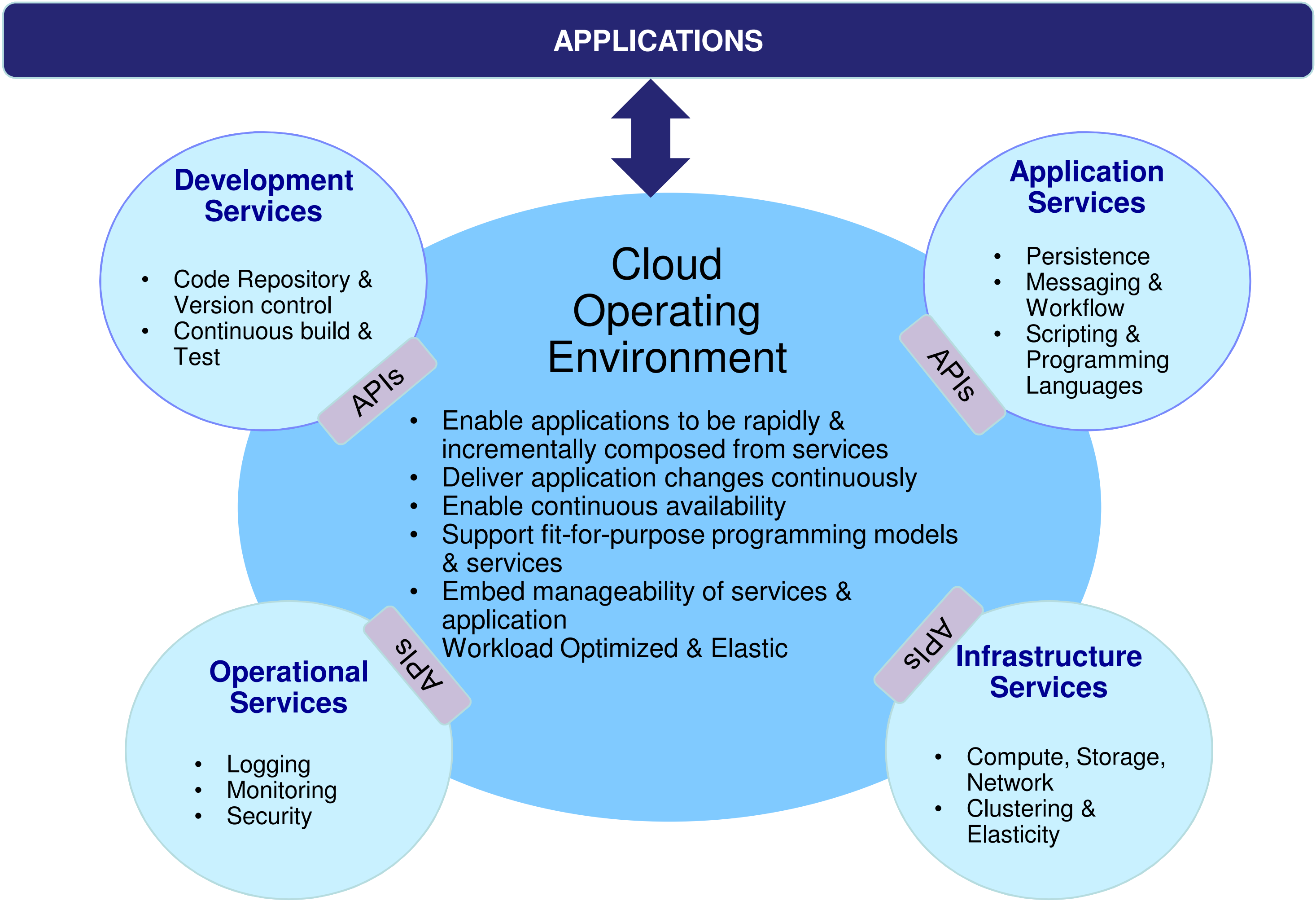


Traditional On-Premises	Infrastructure as a Service	Platform as a Service	Software as a Service
□ Applications	□ Applications	□ Applications	■ Applications
□ Data	□ Data	□ Data	■ Data
□ Runtime	□ Runtime	■ Runtime	■ Runtime
□ Middleware	□ Middleware	■ Middleware	■ Middleware
□ O/S	□ O/S	■ O/S	■ O/S
□ Virtualization	■ Virtualization	■ Virtualization	■ Virtualization
□ Servers	■ Servers	■ Servers	■ Servers
□ Storage	■ Storage	■ Storage	■ Storage
□ Networking	■ Networking	■ Networking	■ Networking

□ Client Manages ■ Vendor Manages in Cloud

Standardization; OPEX savings; faster time to value

Cloud Operating Environment – a trend towards platforms supporting the development and operations of cloud-native applications



APPLICATIONS

Development Services

- Code Repository & Version control
- Continuous build & Test

APIs

Application Services

- Persistence
- Messaging & Workflow
- Scripting & Programming Languages

APIs

Cloud Operating Environment

- Enable applications to be rapidly & incrementally composed from services
- Deliver application changes continuously
- Enable continuous availability
- Support fit-for-purpose programming models & services
- Embed manageability of services & application

Workload Optimized & Elastic

APIs

Operational Services

- Logging
- Monitoring
- Security

APIs

Infrastructure Services

- Compute, Storage, Network
- Clustering & Elasticity

Summary

- “Systems of Engagement” are emerging as a new class of interaction-centric applications
  - Complement transaction-centric “Systems of Record”

- The interaction-centric and green-field nature of Systems of engagement makes it natural to implement them as cloud-native applications

- Attributes of cloud-native applications are a mix of
  - existing SOA best practices (getting reemphasized focus) and
  - additional best practices, specifically around dealing with large distributed environments

- Cloud-native attributes cannot be applied equally well to any kind of application
  - Rearchitecture of existing applications can be costly
  - Applications with very strong focus on transactions may not be well suited

- There are platform architectures emerging, in support of implementing cloud-native software (“Cloud Operating Environment”)