Cell-based Architecture
An Emerging Architecture Pattern for Agile Integration

Asanka Abeysinghe
Deputy CTO & VP, Architecture - CTO Office
WSO2 Inc.
Objectives

#1 why: created a new pattern

#2 introduce: Cell-based architecture
Motivation
Centralized & Layered
Reality of the Enterprise
Brownfield > Greenfield

Legacy, monolithic ➝ Microservices, sprawl

Reference Implementations

picture credit: https://www.marinersmuseum.org/
Underutilization of the Technology
Gap: architecture | development | deployment
Dependency management

Architecture Patterns
Timeline
Background: Layered Architecture
A platform with an agile team
100 APIs, 60 message flows, 80 services, n DBs
Multi-tenanted, 3 active tenants
First release after 3 years
Rise of Microservices
Pragmatic Microservices

- Netflix: APIs
- Uber: Edge Gateway
- eBay: API Facade
- Gartner: Mini Services
Background: Layered Architecture with MSA
Background: Segmented Architecture
Platform of Platforms
Building the Concept
Business vs technical services
A **code** exposes through an **interface** that describes a collection of operations that are **network accessible** using a standardized messaging protocol.
Service: Business definition

Software components that can be spontaneously discovered, combined, and recombined to provide a solution to a business problem.
Microservice: Technical definition

A microservice must have a **single purpose** and be loosely coupled in design and deployed independently of other microservices. "Micro" is a concept of **scope** rather than **size**.
Microservice: Business definition

Microservices is an approach to application development in which a large application is built as a suite of modular components or services. These services are built around business capabilities.
Group of (Micro)services
The cell is the basic structural, functional, and biological unit of all known living organisms.
Cell-based Reference Architecture
Component: Atomic Units

A component represents a process or business logic running in a container, serverless environment, or an existing runtime. A component is designed based on a specific scope, which can be independently run and reused at the runtime.
Cell: Units of Enterprise Architecture

A **cell** is a collection of components, grouped from design and implementation into deployment. A cell is independently deployable, manageable, and observable.
Cell:Component
1:M
1:1
Connected Cells

picture credit: https://www.medicalnewstoday.com/
Control Plane:
- Signaling of the network
- Makes decisions about the traffic flow

Data Plane:
- Forwards traffic between hops
- Takes data packets

Management Plane:
- Configure
- Observeability, Monitor

picture credit: https://www.flickr.com/photos/teflon/
Inter and Intra Cell communication

Global-mesh

Local-mesh

DP - data plane
CP - control plane
MP - management plane
Connected Cells

Cell gateway (ingress)

Sidecar (egress)

Adaptor (egress)

Ambassador (egress)
API-centric Architecture

picture credit: https://www.flickr.com/photos/hugh_nelson/
API-centric Architecture

**Pull APIs**
- RESTful HTTP, gRPC

**Push APIs**
- Events JMS, AMQP, SMTP
- Streams Kafka, MQTT
Gateway Pattern

picture credit: https://www.flickr.com/photos/ell-r-brown/
Automated governance is made of three things:

A source of truth: Policy store/registry

Enforcement of the policy: Gateway or plugin attempting to keep the desired state

Observability: How close to the desired state are we?
Security of Cells

Security of Cells
Developer Experience (DX) of a Cell
Creating Cells

Brand new Cell

Existing (micro)services

Update an existing Cell

Create a new version
Lifecycle of a Cell
Structured Agility

Versioned Components

Versioned Cells

Dependency managed

Autowired

Reusable

Enhanced MSA & CNA
Cell-based Enterprise Architecture
## Cell Types

<table>
<thead>
<tr>
<th>Cell Type</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>Microservices, Functions, MicroGateways, lightweight storages</td>
</tr>
<tr>
<td>Integration</td>
<td>MicroESB or other integration microservices, lightweight storage and/or cache</td>
</tr>
<tr>
<td>Legacy</td>
<td>Existing systems, legacy services</td>
</tr>
<tr>
<td>External</td>
<td>SaaS and partner systems</td>
</tr>
<tr>
<td>Data</td>
<td>RDBMS, NoSQL, File, Message Broker*</td>
</tr>
<tr>
<td>Identity</td>
<td>IDP, user stores</td>
</tr>
<tr>
<td>Channel</td>
<td>Web Apps, IoT, mobile apps</td>
</tr>
</tbody>
</table>
Reference Implementation L0
Reference Implementation L1
Human-centric Architecture

picture credit: https://www.vertoanalytics.com/human-vs-data-centric-marketing/
Cells and Podular Organizations
Cell Boundaries
Defining Cell boundaries

The design of systems has always required an approach to the clustering of functionality, and it remains an open Computer Science problem - *so don't expect a definitive answer!*

The number of component-component connections within a cell should be higher than the number that crosses the cell boundary.

Other approaches such as Domain-driven Design (DDD) may help, but fundamentally the cell model is there to provide team boundaries.

Hence the size of a cell should be based on the size, responsibility, and output of a team - and the size and output of a team based on team concepts.

Cell-based architecture aims to create business focussed architectural constructs that can reuse at a higher level, so naturally organizing the teams and cells around business functions is essential.
Measure the success

picture credit: https://www.marketingdonut.co.uk/direct-marketing/running-a-direct-mail-campaign/getting-the-measure-of-your-marketing
Mean Time to Repair

https://www.tasktop.com/blog/5-best-metrics-youve-never-met/
https://dzone.com/articles/reducing-mttr
Summary: Cell-based Architecture

- Self-contained
- Deployable as a unit
- Independently elastic
- Data, control & management plane
My Contribution: walk the talk

picture credit: https://www.omgubuntu.co.uk/2018/06/microsoft-buying-github
Just a (steady) start

https://github.com/wso2/reference-architecture
Reference Methodology

Reference Methodology for Agility

https://github.com/wso2/reference-methodology

picture credit: https://www.flickr.com/photos/vasile23/
Build, run, and manage code-first composites on Kubernetes

https://cellery.io/
Cellery - simple composition and graphical view
Ballerina

A cloud-native programming language for microservices and APIs

- A compiled, transactional, statically and strongly typed programming language with textual and graphical syntaxes.
- Incorporates fundamental concepts of distributed system integration
- Offers a type safe, concurrent environment to implement microservices with distributed transactions, reliable messaging, stream processing, and workflows
- Compiles into container images for deployment on Kubernetes and Docker

https://ballerina.io/
Ballerina - code and visual syntax in sync
Invitation to consume and contribute

https://ballerina.io
https://cellery.io
https://github.com/wso2/reference-architecture
https://github.com/wso2/reference-methodology
Rate today’s session

Session page on conference website

O’Reilly Events App
digital transformation is about improving the human interaction of the business internally and externally - hence, the technical architecture should take a human-centric approach. #cellbasedarchitecture
THANK YOU

@asankama

https://www.linkedin.com/in/asankaabeysinghe/