Microkernel Architecture

Software Architecture

Richard Thomas

March 13, 2023

So far...

Simplicity – Monolith, Pipeline

Modularlity – Layered, Pipeline

Definition 1. Extensibility

Features or extensions can be easily added to the software over its lifespan.

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Question

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Answer

Monolith – Everything in one container

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Monolith – Everything in one container

Layered – Typically all layers

How easy is it to extend *Monolith*, *Layered* or *Pipeline*?

Monolith – Everything in one container

Layered – Typically all layers

Pipeline – Create a new filter

Definition 2. Interoperability

Software can easily share information and exchange data with internal components and other systems.

What about interoperability?

Question

What about interoperability?

Answer

Monolith – Everything in one container

- Internal 😃   External 😵

What about interoperability?

Monolith – Everything in one container

- Internal 😎👍   External 😰

Layered – Nearest Neighbour

- Internal 😎👍   External 😰

What about interoperability?

Monolith – Everything in one container

- Internal 😄👍   External 😷

Layered – Nearest Neighbour

- Internal 😄👍   External 😟

Pipeline – Standard Interface

- Internal 😄👍   External 😷

Question
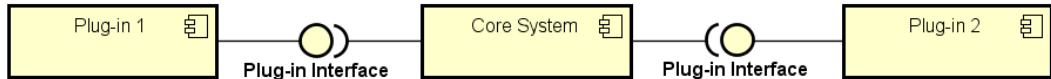
What if I want simplicity, extensibility and interoperability?

What if I want simplicity, extensibility and interoperability?

Consider *Microkernel Architecture*

> **Definition 3.** Microkernel Architecture
>
> Core system providing interfaces that allow plug-ins to extend its functionality.

Plug-in 1		Core System		Plug-in 2
	Plug-in Interface		**Plug-in Interface**	

Definition 4. Registry

Tracks which plug-ins are available to the core system and how to access them.

Loading Plug-ins

Static Loading when application starts

Dynamic Loading as needed at run-time

Registry designed for the selected strategy

Can you think of a *microkernel archiecture*?

Question

Can you think of a *microkernel archiecture*?

Answer

Web Browser?

Definition 5. Independent Plug-in Principle

Plug-ins should be independent, with no dependencies on other plug-ins. The only dependency on the core system is through the plug-in interface.

Definition 6. Standard Interface Principle

There should be a single interface that defines how the core system uses plug-ins.

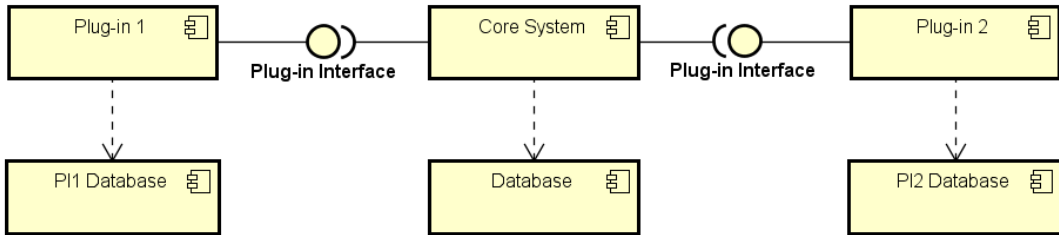Does a plug-in architecture equate to a microkernel archiecture?

Does a plug-in architecture equate to a microkernel archiecture?

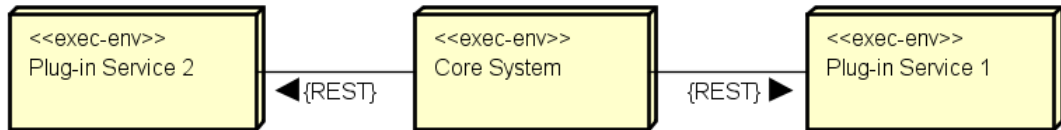What about *IntelliJ*?

Plug-ins with Separate Databases

- Plug-ins cannot access core system data
  - Core system may pass data to the plug-in
- Plug-ins may have their own persistent data

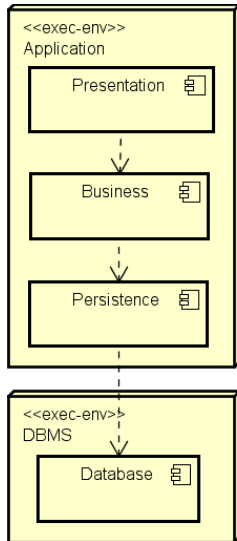# Plug-ins as External Services

- Need communication protocol
- Registry records communication contract
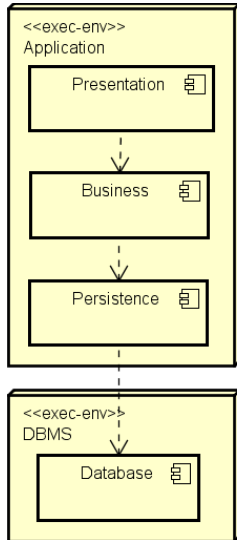 - e.g. URL of the REST endpoint & data passed to it
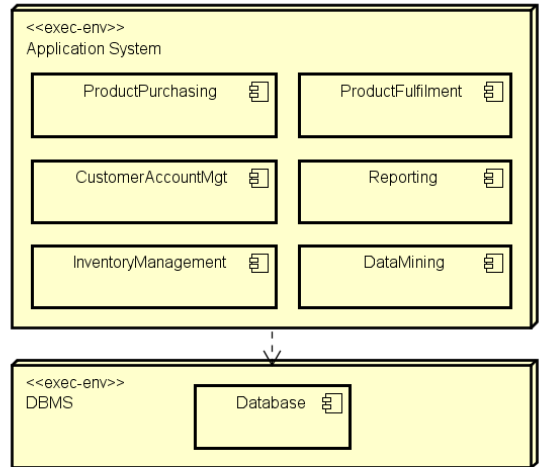
Adapting Non-Conforming Interfaces

Technical Partitioning

Technical Partitioning

<<exec-env>>
Application

- Presentation
- Business
- Persistence

<<exec-env>>
DBMS

- Database

Domain Partitioning

<<exec-env>>
Application System

- ProductPurchasing
- ProductFulfilment
- CustomerAccountMgt
- Reporting
- InventoryManagement
- DataMining

<<exec-env>>
DBMS

- Database

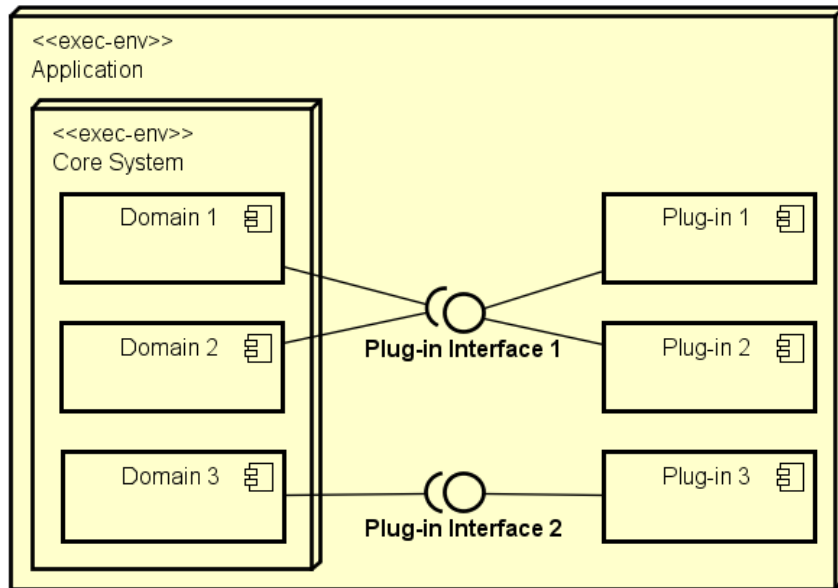Is the microkernel architecture suited to *technical* or *domain* partitioning?

Is the microkernel architecture suited to *technical* or *domain* partitioning?
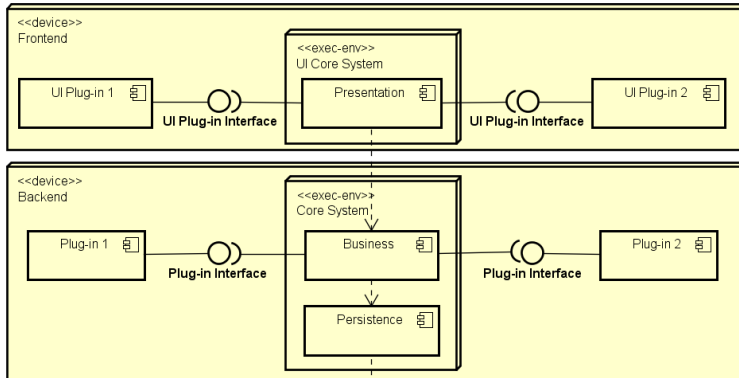
Core system can be partitioned either way.

Domain Standard Interfaces

Distributed Microkernel

- Partitions in the core system can be distributed
  - Technical or domain partitions
  - Plug-ins could also be distributed

Simplicity Core system & Plug-in interface 👍

Extensibility Plug-ins 👍

Interoperability Plug-ins 👍

Scalability ☹

Reliability ☹