

Monolith or Microservices? Designing Deploy-Time Flexibility for Modular Systems

Florin Coroș

florin@onCodeDesign.com
[linkedin.com/in/florincoros](https://www.linkedin.com/in/florincoros)



Florin Coroș

Software Architect Consultant

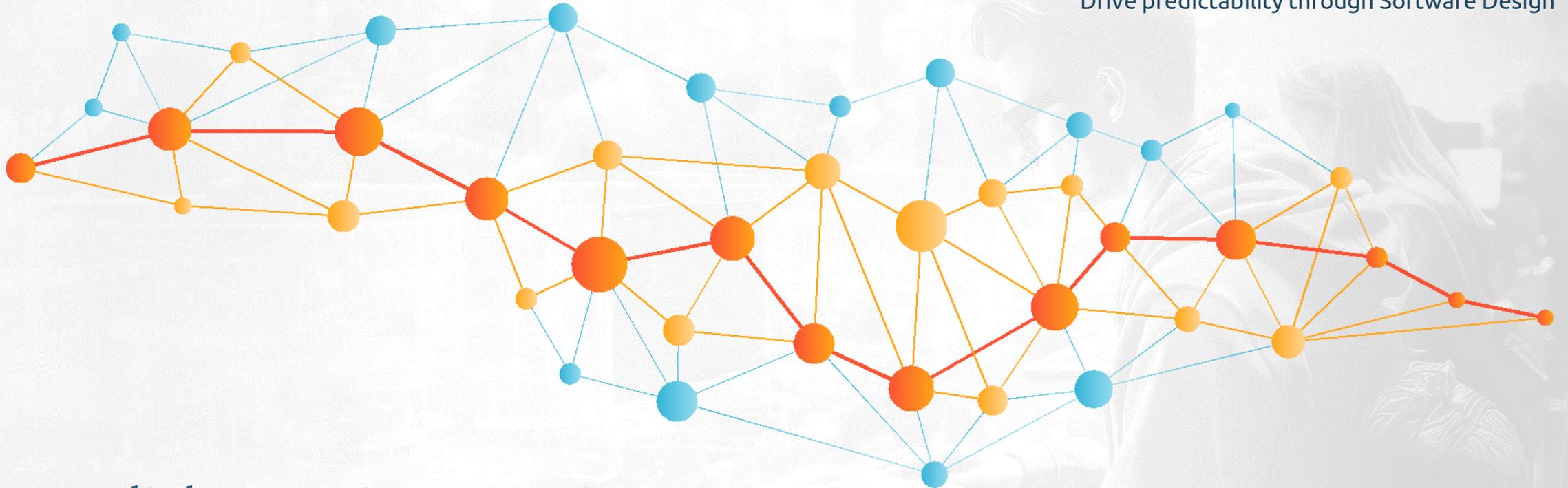
Technical Trainer

Founder of Code Design

enjoying playing GO

enjoying traveling

oncodedesign.com/monolith-or-microservices/



Monolith or Microservices? Designing Deploy-Time Flexibility for Modular Systems

Florin Coroș

florin@onCodeDesign.com
[linkedin.com/in/florincoros](https://www.linkedin.com/in/florincoros)

From Monolith to Micro-services



the MONOLITH

easy life



Not good at

Scalability

Maintainability

Reliability

Testability

Security

Availability

Modernization

High Coupling

Resilience

Extensibility

REDUNDANCY

DECOMPOSITION

From Micro-services back to a Monolith



Not good because

- Complex Communication
- Performance
- Complex Debugging & Diagnostics
- Expensive Infrastructure
- Maintainability
- Testability

**UNNECESARY
COMPLEXITY**

DECOMPOSITION

Micro-services



History Repeats Itself



the MONOLITH

easy life



NOT Enough

Micro-services

complex life



unnecessary complexity

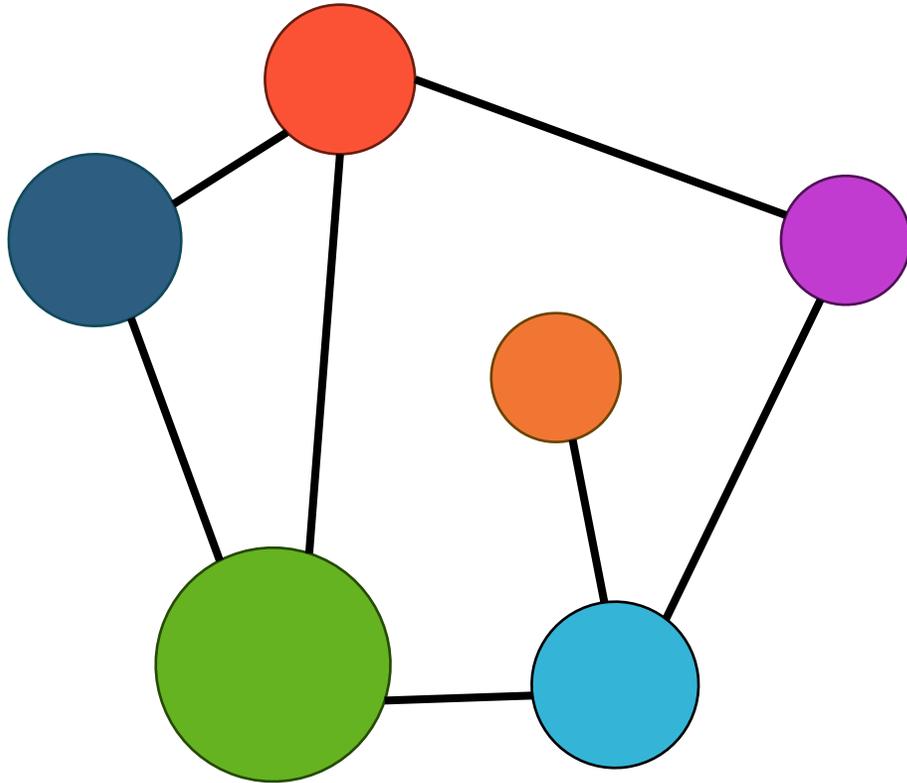
Maintainability
Testability
Modernization

Security
Performance

Scalability
Resilience
Reliability
Availability



Modular System - Concept

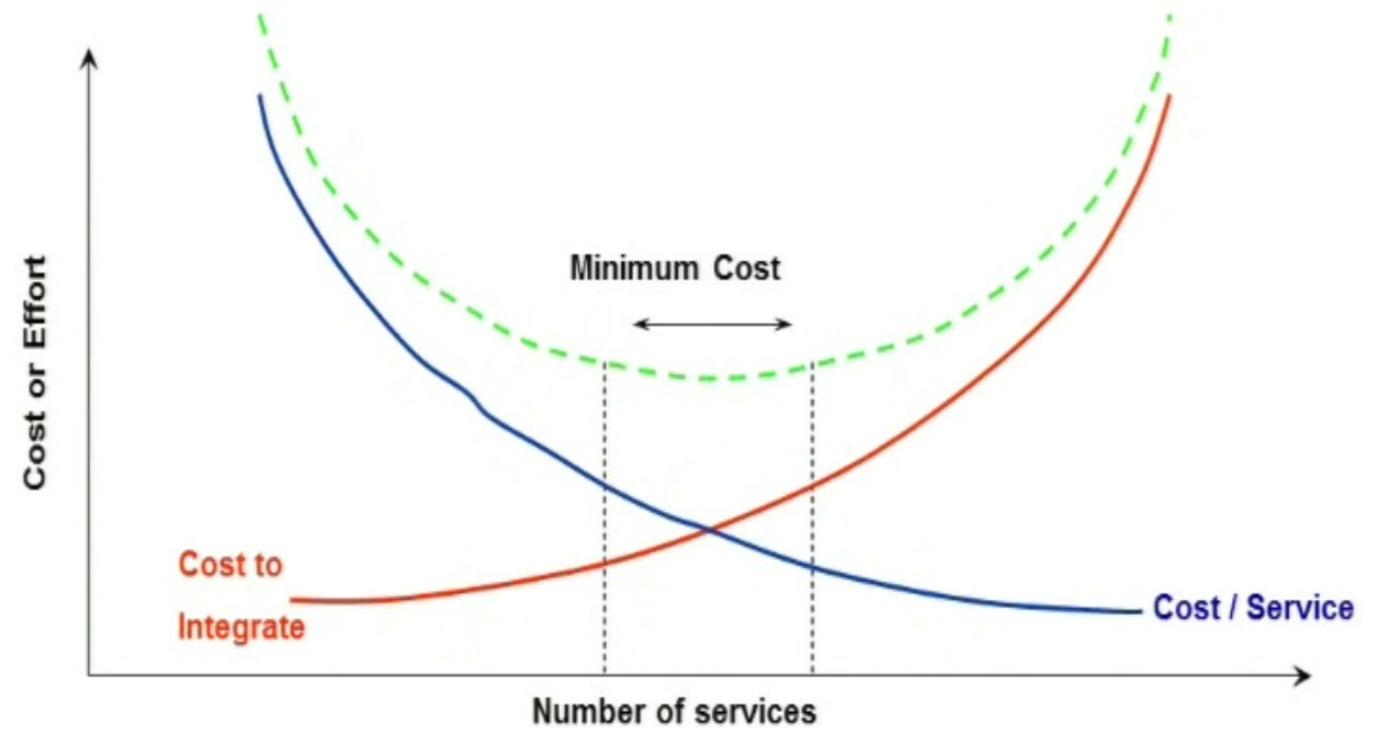
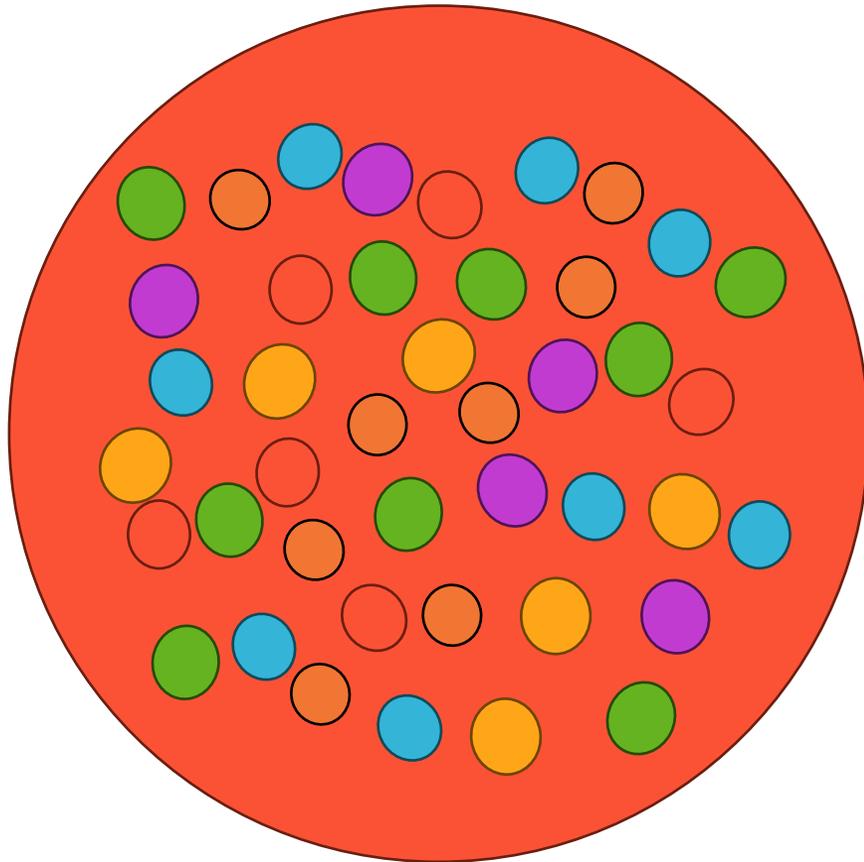


- Maintainability
- Extensibility
- Reusability



Separate the
Communication Concern

How many services?



Contracts – Are Key in Modular Systems



Services communicate through **Explicit Contracts**

- **Abstract** the functions it provides
- **Encapsulate** (hide) the implementation details



Contracts

Contracts described with language constructs:

- Operation Contracts – functions the interfaces
- Data Contracts – DTOs (the in/out params)
- Fault Contracts – Exceptions

Synchronous Communication – function calls

Asynchronous Communication – message based

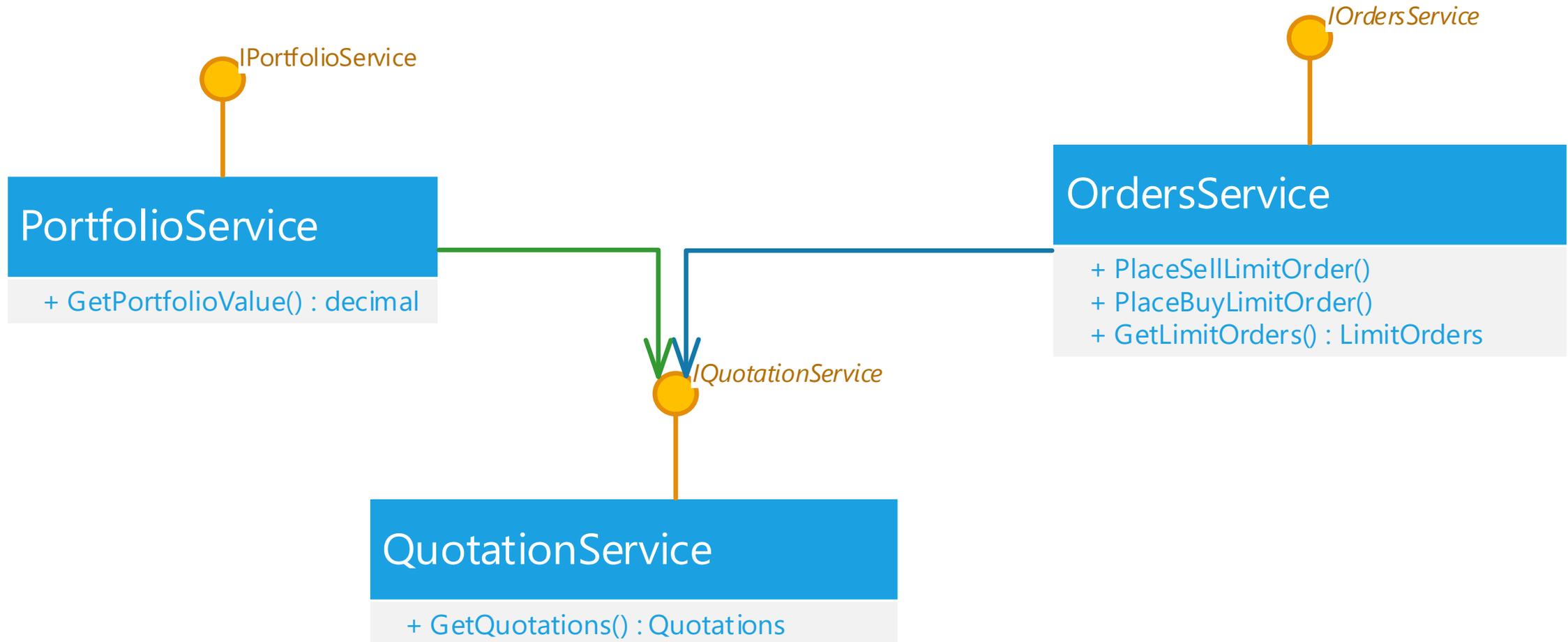
Target: Design for Deploy-Time Flexibility



***Decide **ONLY** at Deploy-Time if
deploy as a Monolith or
deploy as a Distributed System***

*without changing the code
without recompile the code*

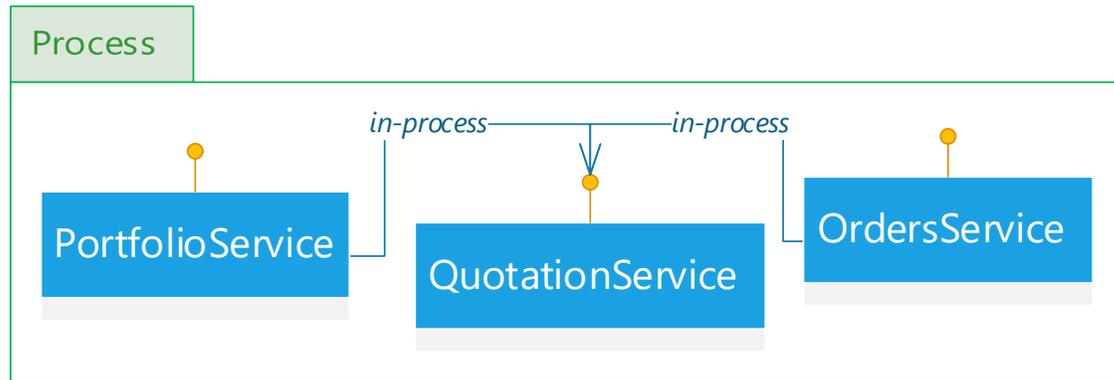
DEMO: Simplified Example of Dependent Services



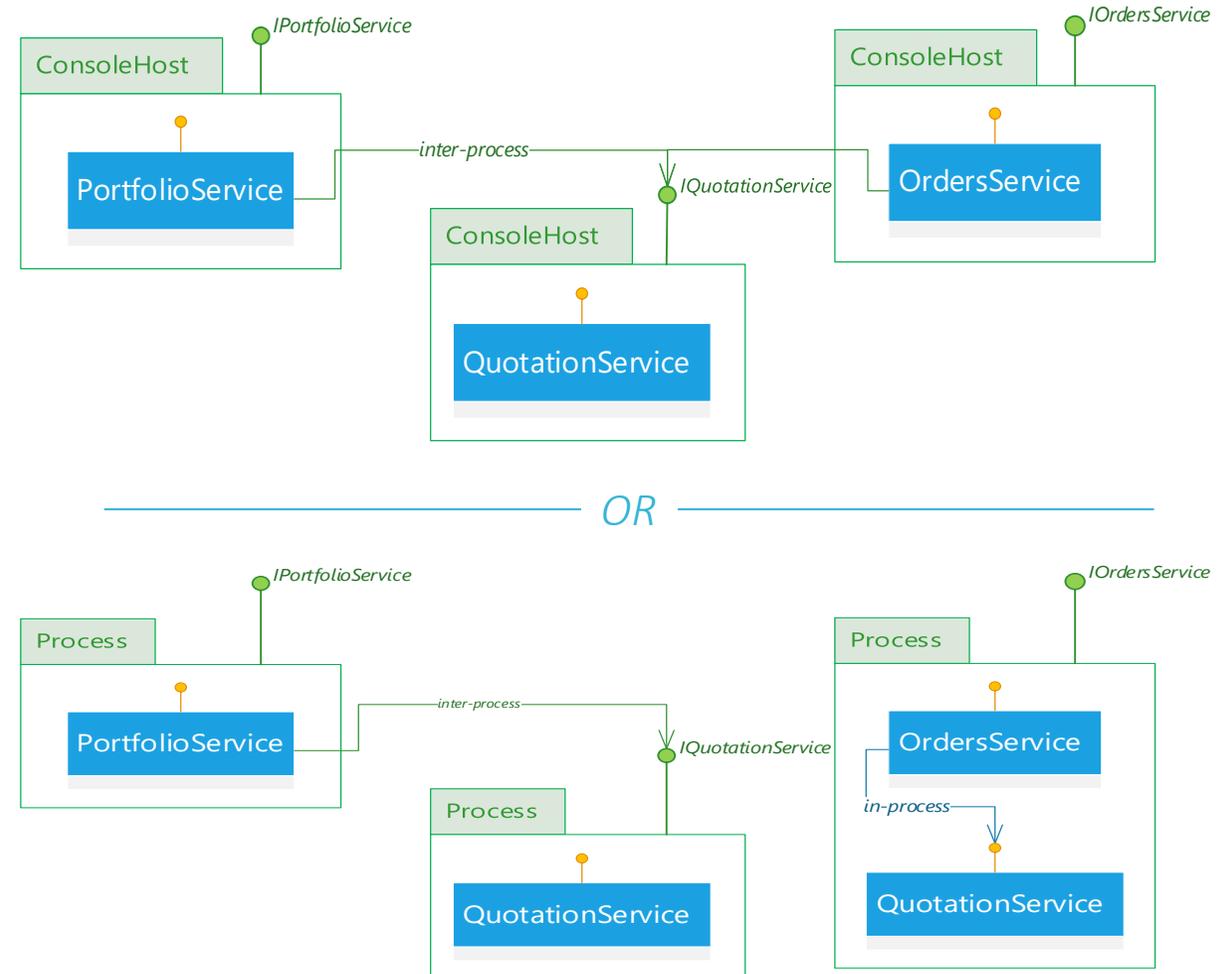
Decide at Deployment between Monolith or Micro-services



Monolith



Micro-services



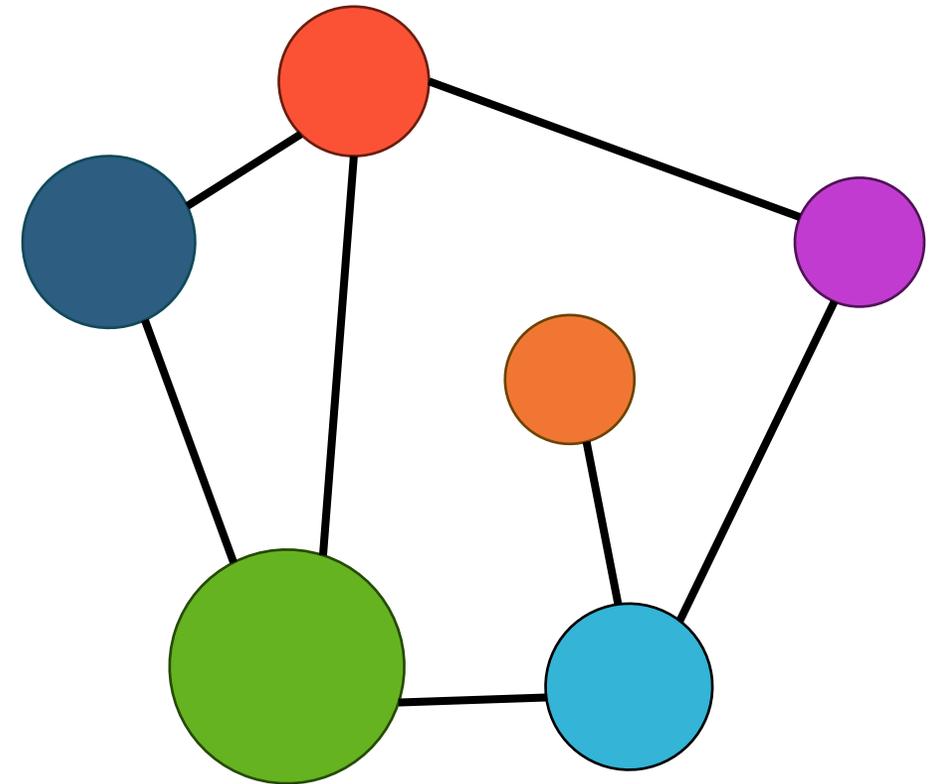
Solution for Design for Deploy-Time Flexibility



The solution stands on **Three Pillars**

1. **Depend only on CONTRACTS**
written with abstract types
2. Use **Proxies** to forward the calls to the actual implementation
3. Generic Hosts with **Type Discovery**

Modular System



Coding Demo



Github Repo:

[Code-Design-Training /
InterProcessCommunication /
TradingApp](https://github.com/OncoDesign/Code-Design-Training-InterProcessCommunication-TradingApp)

Demo build up blog posts:
oncodedesign.com/tag/communication

The screenshot displays the Visual Studio IDE interface. On the left, the Solution Explorer shows a project structure for 'TradingApp' with folders for .Notes, Infrastructure, Modules, Portfolio, Quotations, Sales, Contracts, ConsoleHost, and ConsoleUi. The main area shows an architecture diagram with boxes for Sales.Services, Quotations.Services, Portfolio.Services, Contracts, ConsoleHost, ConsoleUi, Infra.Hosts, AppBootEx, and Proxies, connected by arrows indicating dependencies. The right side shows a code editor with the file 'IQuotationService.cs' containing the following code:

```
1 using System;
2
3 namespace Contracts.Quotations.Services
4 {
5     public interface IQotationService
6     {
7         Quotation[] GetQuotations(string symbol);
8         Quotation[] GetQuotations(string symbol, string date);
9         Quotation[] GetQuotations(string symbol, string date, string time);
10    }
11 }
12
```



florin@onCodeDesign.com

[linkedin.com/in/florincoros](https://www.linkedin.com/in/florincoros)

oncodedesing.com/training

oncodedesing.com/monolith-or-microservices

calendly.com/florin-oncodedesign/short-call



Designing Deploy-Time Flexibility for Modular Systems

Florin Coroș

Software Architect Consultant
Technical Trainer