1. Layered architecture pattern



Usage:

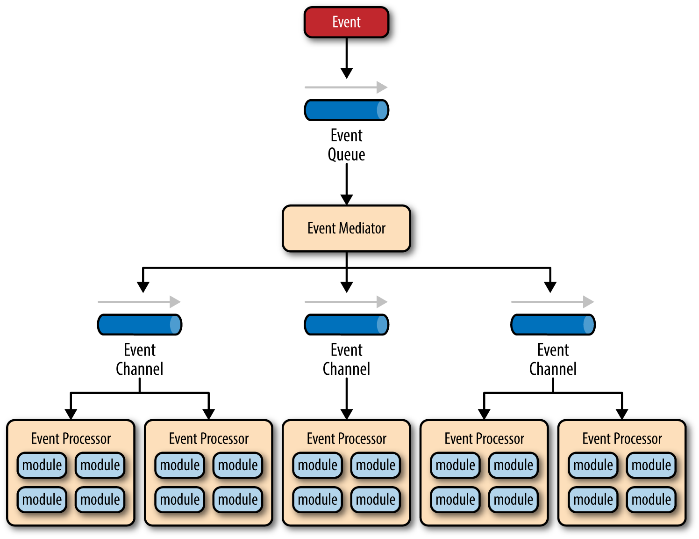
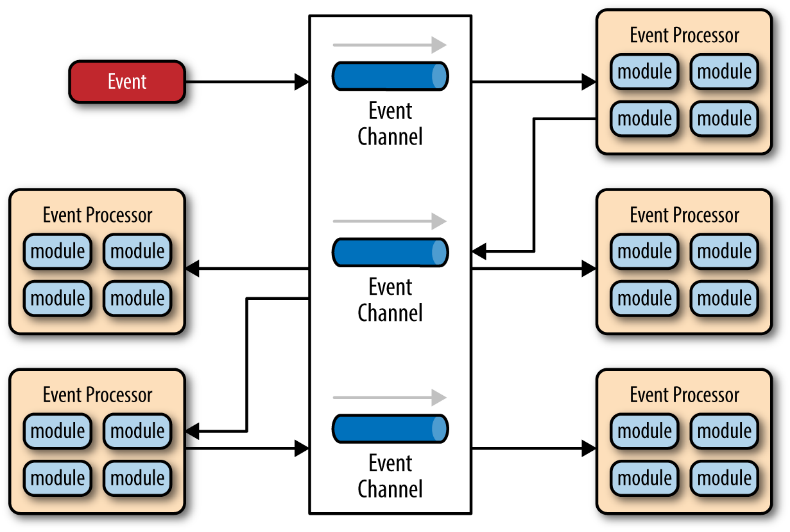
* Application that are needed to be built quickly

Cons:

* Tight coupling
* Small modifications can lead to a redeployment of the entire application

|  |  |  |
| --- | --- | --- |
| Analysis: |  |  |
| Agility | Low | Cumbersome, time consuming to make changes |
| Ease of Deployment | Low | One small change required re-deployment of the entire application |
| Testability | High | Easy, each layered can be isolate testing |
| Performance | Low | Can’t be used for high performance / large application, |
| Scalability | Low | Difficult to scale |
| Ease of Development | High | Easy to develop, structure is not complex at all |

1. Event-Driven Architecture

Usage:

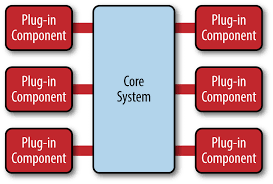
* For Application in which individual data blocks interacts with only a few modules
* Helps with User interface

Cons:

* Tight coupling
* Small modifications can lead to a redeployment of the entire application

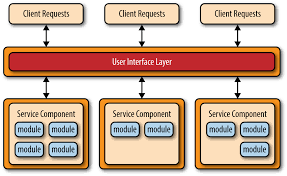
|  |  |  |
| --- | --- | --- |
| Analysis: |  |  |
| Agility | High | Easy to make changes due to isolated components |
| Ease of Deployment | High | Easy, decoupled components |
| Testability | Low | Each unit requires specialized testing tool to generate events |
| Performance | High | Can be used for high-end application, handles asynchronous events and queue |
| Scalability | High | Easy due to decoupled components |
| Ease of Development | Low | Asynchronous events is hard to develop |

1. Microkernel Architecture



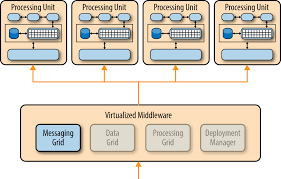
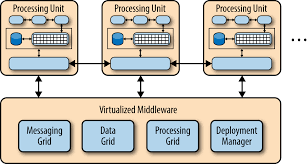
|  |  |  |
| --- | --- | --- |
| Analysis: |  |  |
| Agility | High | Easy to make changes due to loosely coupled plugin modules |
| Ease of Deployment | High | Easy, dynamic addition of plugin |
| Testability | High | Easy, each plugin can be tested separately |
| Performance | High | Easy customization, perform well |
| Scalability | Low | Cannot be scaled |
| Ease of Development | Low | Hard, needs thoughtful design |

1. Microservices Architecture



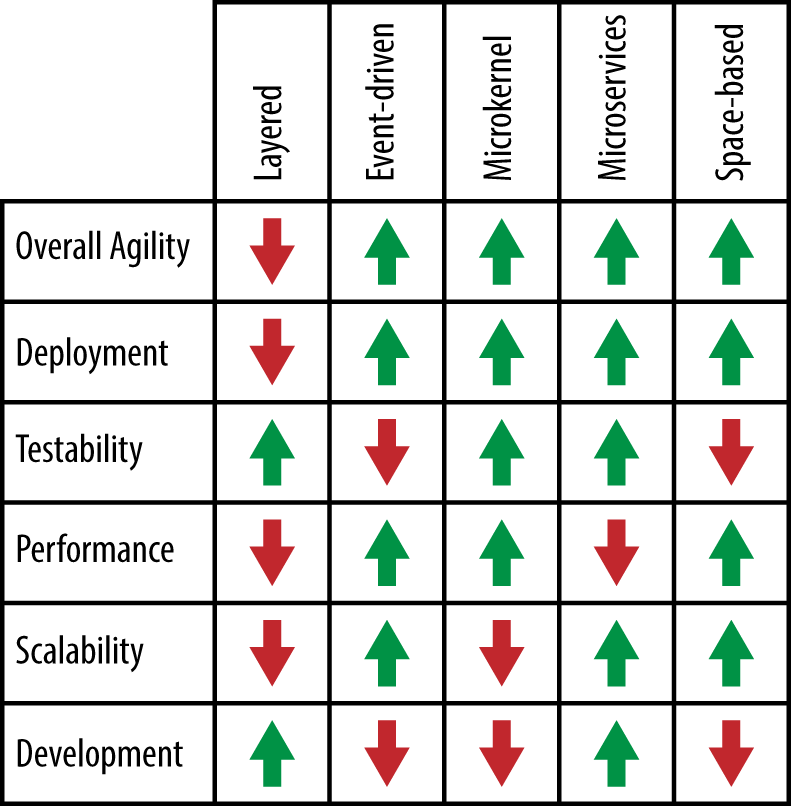
|  |  |  |
| --- | --- | --- |
| Analysis: |  |  |
| Agility | High | Easy to make changes due to loosely coupled |
| Ease of Deployment | High | High rate of development, each service doesn’t impact on other services |
| Testability | High | Easy, services are independent |
| Performance | Low | The architecture is not made for performance |
| Scalability | High | Easy to scale and add staff |
| Ease of Development | High | Easy to develop, each service is developed individually |

1. Space-Based Architecture

|  |  |  |
| --- | --- | --- |
| Analysis: |  |  |
| Agility | High | Easy for small application, |
| Ease of Deployment | High | Easy, dynamic cloud-based |
| Testability | Low | High user load test is time consuming |
| Performance | High | High performance due to in-memory data access and caching |
| Scalability | High | Little to no dependency |
| Ease of Development | Low | Complex to develop the caching and memory data grid |

Analysis Software Architecture:



<https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/>