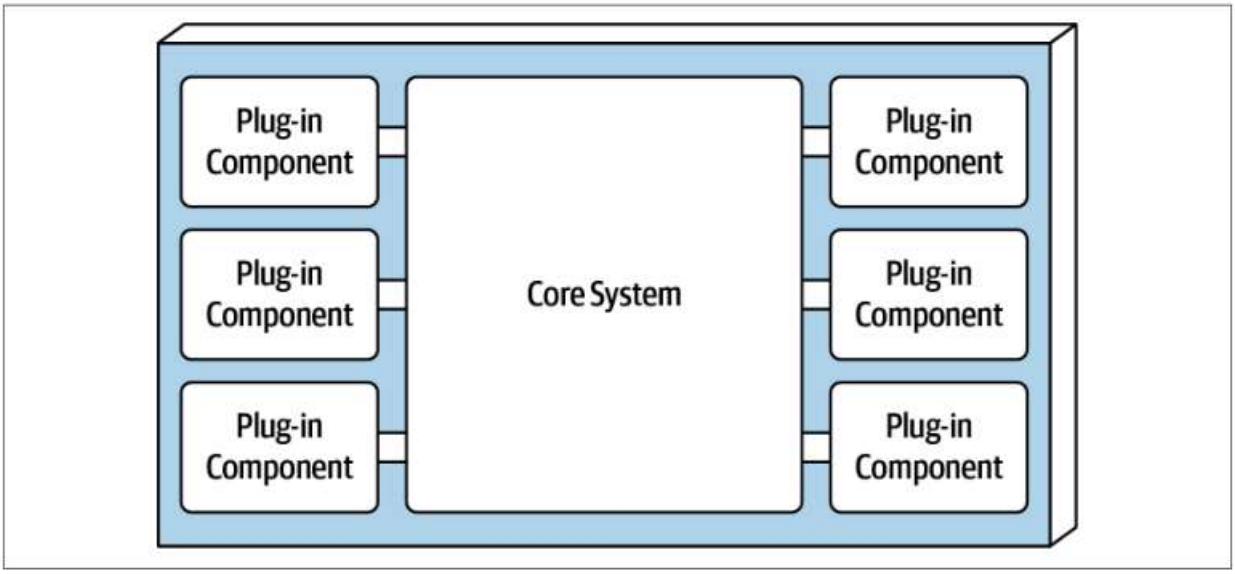
12 - Arquitetura de Microkernel

2025-12-20 20:10 pm

- Também chamado de arquitetura de *plug-in*
- Combina com software como produto.

Topologia

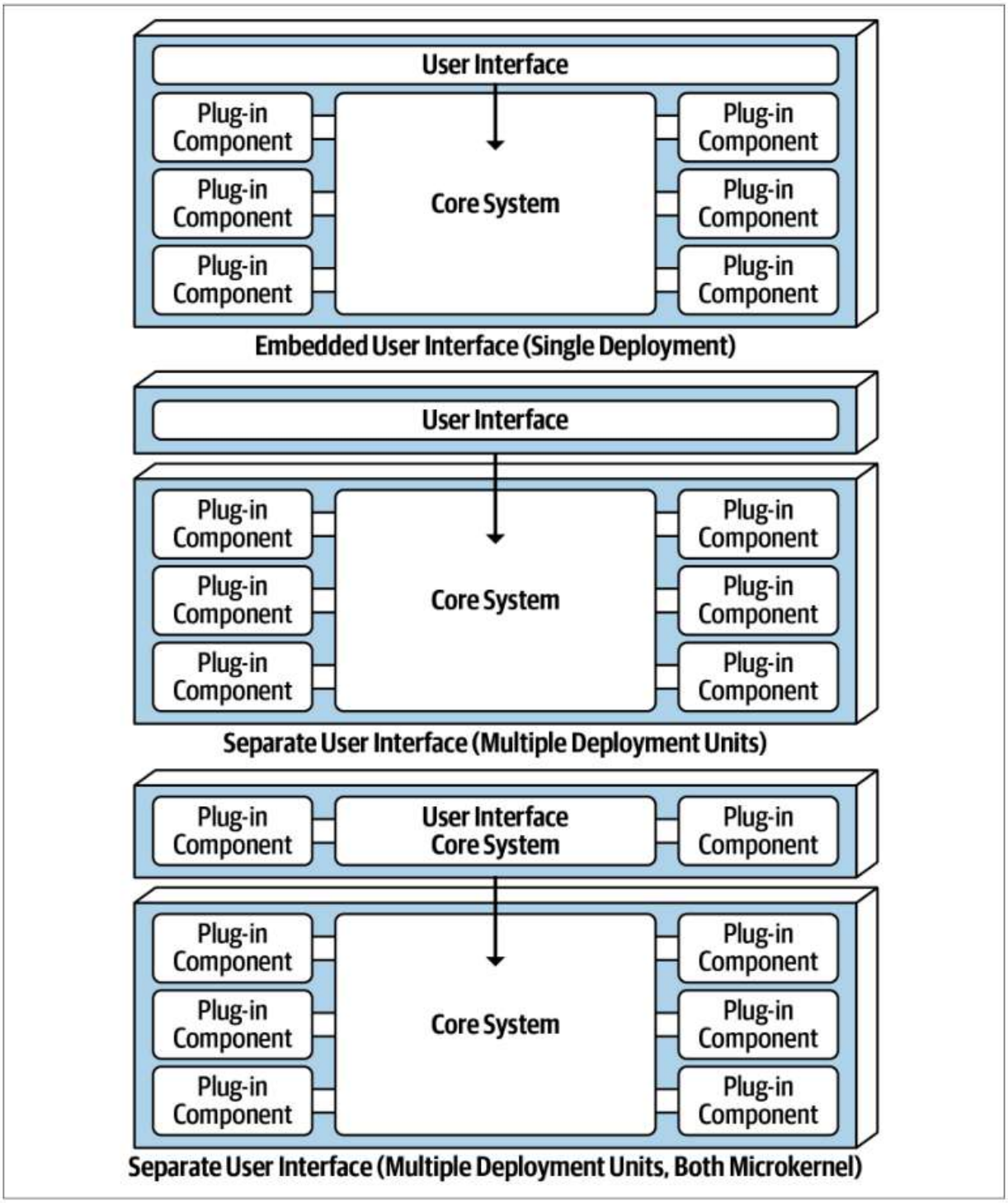
- Monolítico
- Relativamente simples
- Dois componentes:
    - **Core**
    - **Plug-in**s



- *Figure 12-1. Basic components of the microkernel architecture style*

Sistema Core

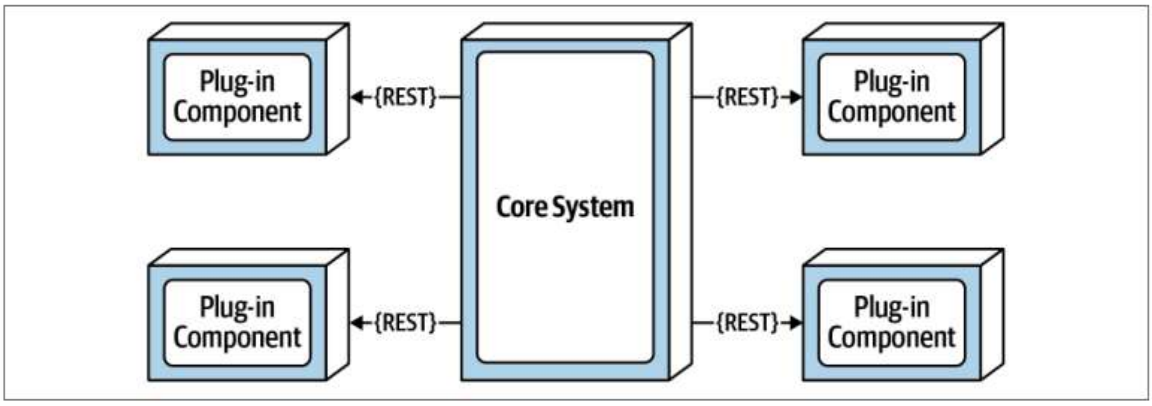
- Funcionalidade mínima para rodar o sistema. Ou então o menor *caminho feliz*.
    - e.g. numa IDE, o core pode ser um simples editor de texto, os plug-ins o complementam.
- Delega funcionalidades específicas aos plug-ins.
    - Plug-ins são independentes.
- Sistema core pode ter arquitetura em camadas ou um monolito modular. Ou mesmo distribuído.
- Core pode ser particionado tecnicamente ou por domínio.
- Pode interagir ou outros componentes eles próprios estruturados em microkernel.



- *Figure 12-3. User interface variants*

Componentes Plug-in

- Componentes independentes que contêm processamento especializado que estende o core system.
- Isola pedaços do código.
  - `app.plugin.<domain>.<context>`
- Plug ins runtime based vs compile based - exige ou não redeploy da aplicação
- Plug in pode ser um serviço avulso que comunica via REST com o Sistema Core.



- *Figure 12-6. Remote plug-in access using REST*
- Mais desacoplado e dessincronizado.
- Mais complexo

- Plug-in não costumam se comunicar com banco de dados central, sistema core gere isso.
  - Mas cada plug-in pode ter seu pequeno datastore particular que só ele acessa.

Registry

- Sistema core precisa saber quais plugins existem e como pegá-los.
  - Registry é uma forma de fazer isso
- Registry tem informações de nome, contrato de dados e endereço do plug-in
- Pode ser simples ou complexo.

```
Map<String, String> registry = new HashMap<String, String>();
static {
  //point-to-point access example
  registry.put("iPhone6s", "Iphone6sPlugin");

  //messaging example
  registry.put("iPhone6s", "iphone6s.queue");

  //restful example
  registry.put("iPhone6s", "https://atlas:443/assess/iphone6s");
}
```

Contratos

- Forma de transferir dados entre plugin e sistema core.
- Costuma ser padrão para um domínio de componentes de plug-in.
- Pode requerer um adapter entre um contrato de plugin não padrão e o contrato de plugin padrão.

Exemplos e Casos de Uso

- Jura, Eclipse IDE, Chrome, Firefox usam essa arquitetura.
- Quando usar para software não-produto?
  - Grande variações de regras de caso para caso pode ser bom ter plugin
  - E.g. sistema de seguros que têm regras diferentes por estado.

        - Permite manipular regras particulares com facilidade.
    - E.g. sistema de imposto que linka formulários diferentes.
        - Facilita mudanças na lei fiscal se cada regra é responsabilidade de um plugin isolado.

Características

Architecture characteristic	Star rating
Partitioning type	Domain and technical
Number of quanta	1
Deployability	★★★
Elasticity	★
Evolutionary	★★★
Fault tolerance	★
Modularity	★★★
Overall cost	★★★★★
Performance	★★★
Reliability	★★★
Scalability	★
Simplicity	★★★★
Testability	★★★

- Simples e barato.
- Monolítico, então pouco escalável e mais difícil de estender e deployar.
- Pode ser tanto particionado por técnica quanto por domínio.
    - Embora geralmente particionado por técnica.
- Altamente customizável.