Kresil - Kotlin Multi-Platform Library for Fault-Tolerance

Francisco Engenheiro, n.º 49428, e-mail: a49428@alunos.isel.pt, tel.: 928051992

Orientadores: Pedro Félix, e-mail: pedro.felix@isel.pt

Março de 2024

1 Introdução

TODO

2 Enquadramento Téorico

TODO: falar sobre agente, ambiente.. e principios de inteligencia artificial
Inteligência =¿ (implica) Autonomia - vice-versa não é verdadeiro

3 Projeto - Parte 1

3.1 Arquitetura de software

A arquitetura de software aborda a complexidade inerente ao desenvolvimento de software por meio de uma série de vertentes que estão interligadas.

3.1.1 Métricas

As métricas são medidas de quantificação da arquitectura de um software indicadoras da qualidade dessa arquitectura;

O acomplamento é uma métrica inter-modular que mede o grau de interdependência entre os módulos de um sistema.

- **Direção**: Unidirecional vs Bidirecional (uni representa menos acoplamento);
- **Visibilidade**: Menos visibilidade representa menos acoplamento;
- **Ordem de menos acoplamento para mais**: Herança → Composição → Agregação → Associação → Dependência.

A coesão é uma métrica intra-modular que determina o nível de coerência funcional de um subsistema/módulo, seja pela sua organização ou pela sua funcionalidade (e.g., single responsibility principle).

3.1.2 Princípios

Os princípios no contexto da arquitectura de software são um conjunto de convenções que orientam a definição da arquitectura de um software, garantindo a qualidade da arquitectura produzida.

Alguns exemplos:

- **Abstração**;
- **Modularização**: Ao qual está associado a decomposição e o encapsulamento dos diversos módulos;
- **Factorização**: Onde a arquitectura é dividida em camadas, cada uma com um conjunto de responsabilidades bem definidas. Pode ser estrutural (e.g, Herança) e Funcional (e.g, Delegação);

3.2 Processo de Desenvolvimento de Software

O processo de desenvolvimento de software consiste na criação da organização de um sistema de forma progressiva, através de diferentes níveis de abstracção:

- **Modelo (Conceptual)**: Representação abstrata do sistema, que define o que o sistema deve fazer, sem especificar como;
- **Arquitetura (Modelo Concreto)**: Representação concreta do sistema, que define como o sistema deve ser implementado;
- **Implementação**: Código fonte que implementa o sistema.

Consiste num processo iterativo, em que as diferentes actividades de desenvolvimento são alternadas ao longo do tempo em função do conhecimento e do nível de detalhe envolvido. Essa alternância poderá ser circular (i.e., implementação → arquitectura → modelo → implementação).

Tabela 1: Tipos de Implementação

Tipo	**Modelo Associado**	**Designação**
Estrutural	*UML*	Define a estrutura de um sistema, ou seja, a forma como os componentes se relacionam entre si.
Comportamental	*Sequence Diagram*	Define o comportamento de um sistema, ou seja, a forma como os componentes interagem e comunicam entre si.

Os diagramas de atividade ou sequência representam o fluxo de controlo de um sistema, ou seja, a sequência de atividades que um sistema executa. São compostos por:

- **Nós**: definem a forma como o fluxo de controlo é dividido (e.g., nó inicial, nó final, nó de decisão, nó de fusão, nó de bifurcação, nó de junção, nó de atividade);
- **Guardas**: definem as condições de transição
- **Partições**: definem a divisão do fluxo de controlo, que são as atividades realizadas por uma parte específica do modelo.

TODO: falar mais sobre UML