Arquitetura de Software

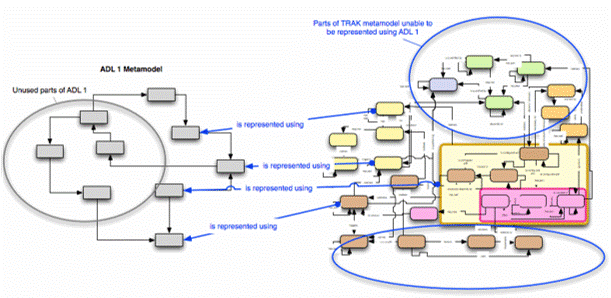
# Introdução

A arquitetura de software define o planejamento de toda a estrutura modular do software, atuando de forma análoga a uma planta baixa de uma construção, detalhando como os blocos operacionais serão construídos, as etapas dessa construção e como se interligam entre si.

As etapas de planejamento são definidas após a etapa de levantamento de requisitos e antes do design, e necessita de um alto nível de abstração, ou seja, a arquitetura deve prever o funcionamento dos componentes do sistema porem sem se preocupar (nesse momento) com detalhes da implementação, contemplar atributos necessários de eficiência, confiabilidade, reusabilidade e portabilidade.

# Linguagens ADL

São as linguagens utilizadas para descrever a arquitetura de um software, Architecture description language (ADL), que possuem um conjunto de regras e protocolos para a criação de diagramas de fluxo, tabelas organizacionais e outras formas de visualização que possam ser necessárias para descrever o modelo a ser construído.



Uma ADL deve possuir como pré-requisitos de usabilidade:

- Ser compatível para a comunicação entre todas as partes interessadas.

- Suportar as tarefas de criação da arquitetura, refinamento e validação.

- Fornece a base para a futura implementação, ou seja, deve permitir a inserção de informações e especificações das quais as especificações do sistema vão derivar.

- Suportar funcionalidades analíticas que permitam a geração de protótipos da implementação.

Exemplos de linguagem

- UML

- ABACUS

- UNICON

- ArchiMate

# Estilos de Arquitetura

Os estilos de arquitetura de software possuem peculiaridades de forma que sua escolha define como os componentes do sistema serão representados e visualizados num nível mais alto de abstração, permitindo que seja possível prever como as camadas ou módulos do software iram se comunicar, identificando suas responsabilidades.

Dentre os principais estilos podemos citar cinco grandes grupos:

- S**tructure**(layered, pipes and filters, component-based)

**- Shared memory**(role-based, blackboard, database-centric)

**- Messaging styles**(Implicit invocation, asynchronous messaging, publish-subscribe)

**- Adaptive systems**(microkernel style, reflection, domain-specific language)

**- Distributed systems**(service-oriented, peer to peer style, object request broker, cloud computing),

 - D**eployment** (client-server, n-tier)

Cada estilo contempla a resolução de classes de problemas especificas, ou seja, não existe uma única solução para todos os problemas e a competência do arquiteto de software passa pela identificação dos problemas, escolha das conceções e definição do melhor estilo a ser empregado.

# Padrões de Arquitetura

Os padrões de arquitetura têm por objetivo definir soluções reutilizáveis para problemas recorrentes de forma conceitual e flexível, de forma que dentro de um mesmo projeto vários padrões podem ser utilizados.

Os padrões arquiteturais mais conhecidos são:

- Model-View-Controller (MVC)

- Microkernel

- Three-tier

- Event-driven (EDA)

- Message Broker

- CQRS (Command and Query Responsibility Segregation)

# Referencias

<https://medium.com/sicreditech/arquitetura-de-software-estilos-e-padr%C3%B5es-de-design-50d62d684ef2>

<https://cio-wiki.org/wiki/Architecture_Description_Language_(ADL)>

<https://en.wikipedia.org/wiki/Client%E2%80%93server_model>