5 Lab: Vistas de arquitetura

Enquadramento

A arquitetura de software de um sistema informático é o conjunto de estruturas necessárias para raciocinar sobre o sistema, que engloba elementos de software, relações entre eles, e propriedades de ambos. A arquitetura é, grosso o modo, a **partição** sensata de um todo em **partes**, com **relações** específicas entre elas. Esta partição comunica a organização e um sistema e permite o trabalho cooperativo entre grupos de pessoas para resolver um problema maior do que qualquer uma delas poderia fazer individualmente. A UML pode ser utilizada para descrever diferentes tipos de informação próprios da documentação de arquitetura de software, tais como vistas de módulos (ou lógica), vistas de componentes e conectores (C&C), vistas de atribuição (de módulos a ambientes de execução). Essas vistas correspondem, de forma geral e respetiva, aos diagramas de pacotes, componentes e instalação.

Objetivos de aprendizagem

- aplicar o diagrama de pacotes para representar os "grandes módulos" de um sistema, aplicando a relação de dependência.
- representar "*artifacts*" executáveis como componentes e evidencia, num diagrama de componentes, a interligação (nomeando as interfaces providas/solicitadas).
- mostrar, num diagrama de instalação, a atribuição de "*artifacts*" de software a ambientes físicos de execução e as linhas de comunicação.

Preparação

- informação tutorial: diagramas de pacotes, componentes e instalação.

5.1

Considere o exemplo do Diagrama 1.

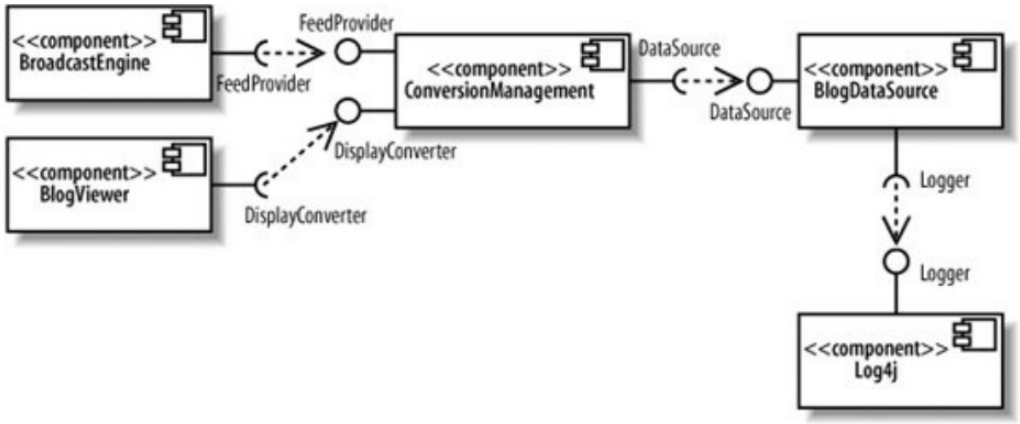

Diagrama 1: Componentes de uma aplicação (hipotética) de gestão de blogs[1].

A) Forneça uma leitura interpretativa do Diagrama 1.

B) O exemplo refere um componente designado "log4j". Pesquise mais informação sobre este componente ("Apache Log4j") e explique brevemente para que serve.

C) Este componente está disponível num "repositório" de componentes conhecido por Maven Central. Quais as coordenadas para o identificar, caso o quiséssemos incluir num projeto,

[1] Também é comum a conexão entre os componentes ser mostrada sem a seta a tracejado que liga a saída (interface provida) à entrada (interface requerida) dos componentes, como neste exemplo.

usando a “build tool” do Gradle?

5.2

Confira as seguintes fontes sobre a organização de um sistema segundo uma arquitetura por camadas:

— Larman: camadas frequentes numa arquitetura genérica e um exemplo.

— Clements: documentação de uma arquitetura por camadas com a UML ("A.2.4 Layered Style”)

Com base nos elementos essenciais de uma arquitetura por camadas identificados nas referências anteriores:

a) pesquise um **exemplo concreto** (na Internet) de uma arquitetura multicamada[2].

b) transcreva o diagrama para o Visual Paradigm, com adaptações, se necessário. Contextualize a resposta com uma explicação dessa arquitetura.

5.3

A Tabela 1 apresenta os principais componentes de um sistema hipotético, para facilitar a participação dos cidadãos na apresentação de ideias para consideração pelas instâncias de governo. O sistema é baseado num mecanismo de “chat bot” para atender os cidadãos e suporta diálogos em linguagem natural. O sistema engloba os módulos listado na Tabela 1.

Tabela 1

Componente	**Descrição**
Front End Client(*)	Este componente é o ponto de entrada para o Chatbot. É aqui que os utilizadores podem participar em conversas com o Chatbot.
Conversation Manager	Componente que trata da criação de novas sessões de conversação, bem como do funcionamento das conversas existentes.
NLU Module	Considerado como os ouvidos do Chatbot, este componente é responsável pela interpretação da linguagem natural.
Dialog Engine	Responsável por prever a próxima ação a tomar pelo Chatbot.
Action Server	Contém APIs para interagir com serviços externos. É aqui que reside a lógica para a realização de ações.
Idea DB (*)	Base de dados para guardar as ideias (dos cidadãos) recolhidas pelo Chatbot.
External Knowledge Base (*)	Várias fontes de dados que poderão ser utilizadas pelo Chatbot para fornecer respostas.

A) Tendo presente a informação anterior, considere também a seguinte descrição sobre a organização da solução e interações entre módulos. Modele num diagrama de componentes (evidenciado os serviços fornecidos/consumidos):

— os componentes marcados com (*) na Tabela 1 podem ser considerados externos aos serviços do módulo de Chatbot.

— o “Action Server” interage com a “Idea DB” para adicionar e para recuperar ideias. Também acede, quando necessário, o “External knowledge Base” para pesquisar informação.

[2] Não é necessário nem importante que o exemplo esteja já feito em UML. O importante é que reflita a ideia de camadas e módulos dentro das camadas.

— o “Dialog Engine” pode solicitar a realização de ações ao “Action Server”.

— o “Conversation Manager” é o orquestrador: solicita o “NLU Module” para analisar o texto recebido (“parsing”) e interage com o “Dialog Engine” passando atualizações (o DE é responsável por “seguir” a conversa). O CM fornece uma interface programática para os Clientes manterem uma conversação (“conversation handler”).

B) Sobre o mesmo sistema, considere a seguinte descrição e represente-a num diagrama adequado.

— cada módulo dá origem a um componente autónomo e é instalado num servidor dedicado para o serviço de Chatbot (excetuado os externos, marcados com *).

— os utilizadores finais acedem a partir de um *browser*, usando a aplicação associada ao “frontend” (que interage com o Conversation Manager).

— a base de dados é instalada num servidor dedicado e a base de conhecimento externa estás disponível numa infraestrutura externa. Ambas os recursos são acedidos por HTTP (pelo Action Server)

— O módulo de processamento de linguagem natural pode aceder a serviços de apoio, tais como reconhecimento de entidades em texto, categorização de conceitos, etc, disponíveis em *frameworks* de Inteligência Artificial, na Cloud. A comunicação é feita com REST APi, via HTTP.

Notas de estudo

Embora a principal utilização da UML não seja para documentar a arquitetura do software, é possível usar os seus diagramas para o fazer. Clements et al discutem formas de usar a UML na documentação de arquiteturas.[3]

Principais vistas:

1) Module View

Vistas de módulos mostram estruturas de arquitetura onde os elementos são unidades de implementação, ou módulos. Os módulos devem ser representados em UML como pacotes (ou classes ou interfaces, dependendo do nível de detalhe). Nas aulas TP, chamámos vistas lógicas.

2) Component-and-Connector View

Os componentes C&C devem ser representados utilizando componentes UML num diagrama de componentes. Os serviços providos/consumidos são evidenciados e nomeados.

3) Allocation View

As vistas de afetação apresentam mapeamentos entre elementos de software (de módulos ou vistas C&C) e elementos ambientais. Os elementos ambientais são elementos não-software (tais como nós de hardware) que estão de alguma forma associados aos elementos de software do sistema que está a ser concebido. Inclui a **vista de instalação**: os elementos ambientais de uma visão de instalação são elementos de hardware, tais como processadores, memória, e elementos de rede. Estes elementos podem ser representados em diagramas de instalação UML, utilizando nós.

[3] Cf. Anexo A , In: “Documenting Software Architectures: Views and Beyond, Second Edition”.