ANÁLISE E PROJETO DE SISTEMAS

Cleverson Ledur

Revisão técnica:

Jeferson Faleiro Leon
Desenvolvedor de Sistemas
Especialista em Formação Pedagógica de Professores
Professor de curso Técnico em informática

L475a Ledur, Cleverson Lopes
Análise e projeto de sistemas [recurso eletrônico] / Cleverson Lopes Ledur ; [revisão técnica: Jeferson Faleiro Leon]. – Porto Alegre : SAGAH, 2017.

ISBN 978-85-9502-179-2

1. Computação. 2. Projeto de sistemas. 3. Análise de projeto. I. Título.

CDU 004.41

Catalogação na publicação: Ana Paula M. Magnus – CRB 10/2052

Projetar a solução

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar o problema e adequar a solução conforme arquitetura a ser definida.
- Elaborar o projeto do problema dentro da arquitetura definida.
- Descrever o projeto da solução.

Introdução

O projeto de software é uma das etapas essenciais durante a criação de um novo sistema. É a partir dele que se faz a tradução dos requisitos de software em uma solução concreta que pode resolver os problemas dos clientes. Assim, no projeto de software, uma das mais importantes decisões será qual a arquitetura a ser implantada, por isso, esta será a primeira decisão a ser tomada. Após o entendimento dela, é necessário avançar para a aplicação do problema de software na arquitetura, com a apresentação de conceitos sobre o projeto de software, suas características e artefatos.

No entanto, tomar decisões em um projeto de software não gera o produto, então, é preciso documentar essas decisões, o que será estudado na última parte deste texto, com o documento de especificação de projeto, onde se descreve o que foi decidido e selecionado dentro dessa fase.

Arquitetura de software

A arquitetura de software refere-se às estruturas de alto nível de um sistema de software, à disciplina de criação de tais estruturas e à documentação dessas estruturas. As estruturas são necessárias para a criação do sistema de software, pois cada uma compreende elementos de software, relações entre eles e propriedades de ambos os elementos e suas relações. Assim, a arquitetura de

um sistema de software é uma metáfora análoga à arquitetura de um edifício (BASS; CLEMENTS; KAZMAN, 2012).

A arquitetura de software trata das escolhas estruturais fundamentais que são caras de mudar, uma vez implementadas. Elas incluem opções estruturais específicas de possibilidades no projeto de software (BASS; CLEMENTS; KAZMAN, 2012). Documentar arquitetura de software facilita a comunicação entre as partes interessadas, capta decisões iniciais sobre o projeto de alto nível e permite a reutilização de componentes entre projetos.

Características da arquitetura de software:

- **Atender *stakeholders*:** os sistemas de software devem atender a uma variedade de partes interessadas, como gerentes de negócios, proprietários, usuários e operadores. Essas partes interessadas têm suas próprias preocupações em relação ao sistema. O equilíbrio entre essas preocupações e a demonstração de como elas são abordadas faz parte da concepção do sistema. Isso implica que a arquitetura lida com uma ampla variedade de preocupações e partes interessadas, tendo uma natureza multidisciplinar.
- **Separação de necessidades:** o caminho estabelecido para que os arquitetos reduzam a complexidade é separar as necessidades que impulsionam o projeto. A documentação da arquitetura mostra que todas as necessidades das partes interessadas são abordadas modelando e descrevendo a arquitetura de diferentes pontos de vista associados às várias necessidades dos interessados. Essas descrições separadas são chamadas de vistas arquitetônicas.
- **Orientado pela qualidade:** abordagens de projeto de software clássicas foram conduzidas pela funcionalidade necessária e pelo fluxo de dados por intermédio do sistema (GARLAN; SHAW, 1994), mas a visão atual é de que a arquitetura de um sistema de software é relacionada aos seus atributos de qualidade, como tolerância a falhas, compatibilidade com versões anteriores, extensibilidade, confiabilidade, manutenção, disponibilidade, segurança, usabilidade e outras características. As preocupações das partes interessadas geralmente se traduzem em requisitos desses atributos de qualidade, que são chamados de requisitos não funcionais, requisitos extrafuncionais, requisitos comportamentais ou requisitos de atribuição de qualidade.

- **Estilos recorrentes:** como a arquitetura de construção, a disciplina de arquitetura de software desenvolveu maneiras-padrão de abordar as necessidades recorrentes. Esses modos-padrão são chamados por vários nomes, em vários níveis de abstração. Termos comuns para soluções recorrentes são o estilo arquitetônico, tática, arquitetura de referência e padrão arquitetônico.
- **Integridade:** existe a ideia de que a arquitetura de um sistema de software representa uma visão geral do que deve fazer e como deve fazê-lo. Essa visão deve ser separada de sua implementação. O arquiteto assume o papel de detentor da visão, certificando-se de que as adições ao sistema estão alinhadas com a arquitetura, preservando, assim, a integridade conceitual.

Atividades de definição de arquitetura

As atividades de suporte de arquitetura de software são realizadas durante as principais atividades dela. Essas atividades de apoio auxiliam um arquiteto de software a realizar análise, síntese, avaliação e evolução. Por exemplo, um arquiteto tem que reunir conhecimento, tomar decisões e documentar durante a fase de análise.

Você vai analisar algumas destas atividades a seguir:

- **Gestão do conhecimento e comunicação** é a atividade de explorar e gerenciar o conhecimento essencial para projetar uma arquitetura de software. Para isso, um arquiteto de software não trabalha de forma isolada. Ele recebem insumos, requisitos funcionais e não-funcionais, contextos de projeto de várias partes interessadas e fornecem resultados para as partes interessadas. O conhecimento da arquitetura de software é, muitas vezes, tácito e está retido nos líderes das partes interessadas. A atividade de gerenciamento de conhecimento de arquitetura de software trata de encontrar, comunicar e manter conhecimento. À medida que os problemas do projeto de arquitetura de software são intrincados e interdependentes, uma lacuna de conhecimento no raciocínio de projeto pode levar a um projeto de arquitetura de software incorreto.
- **Raciocínio de projetos e a tomada de decisão** são atividades de avaliação de decisões de projeto. Essas atividades são fundamentais para as três principais atividades de arquitetura de software. Isso implica reunir e associar contextos de decisão, formular problemas de decisão

de projeto, encontrar opções de solução e avaliar trocas antes de tomar decisões. Esse processo ocorre em diferentes níveis de granulometria de decisão, ao mesmo tempo, que avalia requisitos arquitetônicos significativos e decisões de arquitetura de software, análise de arquitetura de software, síntese e avaliação.

- **Documentação** é a atividade de gravação do projeto gerado durante o processo de arquitetura de software. Um projeto do sistema é descrito utilizando-se várias visualizações, que, frequentemente, incluem uma visão estática, que mostra a estrutura do código do sistema, uma exibição dinâmica, que apresenta as ações do sistema durante a execução e uma visão de implantação, que mostra como um sistema é colocado no hardware para execução.

Padrões da arquitetura de software

Um padrão de arquitetura é uma solução geral e reutilizável para um problema comum na arquitetura de software em um determinado contexto. Padrões são frequentemente documentados como padrões de projeto de software. Seguindo a arquitetura de construção tradicional, um padrão de arquitetura de software é um método de construção específico, caracterizado pelos fatores que o tornam notável. Um padrão arquitetônico define uma família de sistemas em termos de padrão de organização estrutural ou um vocabulário de componentes e conectores com restrições sobre como eles podem ser combinados. Os padrões arquitetônicos são "pacotes" reutilizáveis de decisões de projeto e restrições aplicadas a uma arquitetura para induzir qualidades desejadas (ANGELOV; GREFEN; GREEFHORST, 2009).

Veja a lista abaixo com os padrões de arquitetura de software mais conhecidos e utilizados (ANGELOV; GREFEN; GREEFHORST, 2009):

- *Blackboard*
- Cliente-servidor
- Baseado em componentes
- Centrado em dados
- Orientado aeventos (ou invocação implícita)
- Arquitetura em camadas (ou multicamadas)
- Aplicação monolítica
- *Peer-to-peer* (P2P)
- Tubos e filtros

- *Plug-ins*
- REST
- Baseado em regras
- Serviço orientado
- Arquitetura de microservices
- Arquitetura nada compartilhada
- Arquitetura baseada em espaço

Esses padrões foram definidos para facilitar a criação de sistemas, já que eles eram utilizados de forma recorrente. Ou seja, eles evitam que os projetistas necessitem criar ou solucionar problemas comuns que foram solucionados anteriormente, com necessidade de apenas escolher o padrão mais adequado ao problema apresentado.

Projeto de software: do problema à arquitetura

O projeto de software é o processo pelo qual é criada uma especificação de um artefato de software destinado a atingir objetivos no uso de um conjunto de componentes primitivos e sujeito a restrições. O projeto de software pode referir-se a "toda a atividade envolvida na conceituação, enquadramento, implementação, comissionamento e, em última análise, na modificação de sistemas complexos" ou também "a atividade que ocorre entre a etapa de especificação de requisitos e a programação" (SURYANARAYANA; SAMARTHYAM; SHARMA, 2015).

Ele geralmente envolve a solução de problemas e o planejamento de uma solução de software. Isso pode incluir um componente de baixo nível, um projeto de algoritmo e um projeto de arquitetura de alto nível.

A fase de projeto de software pode ser classificada em até três tipos, conforme citados a seguir:

Projeto estrutural: o projeto estrutural é uma versão abstrata em alto nível do sistema. Ele representa o software como um sistema formado por múltiplos componentes que interagem entre si. Nesse nível, os projetistas obtêm a ideia do domínio da solução proposta, observando a solução de forma estrutural.

Projeto de alto nível: o projeto de alto nível quebra o conceito de projeto estrutural de múltiplos componentes em uma visão menos abstrata de subsistemas e módulos. O projeto de alto nível concentra-se em como o sistema, juntamente com todos os seus componentes, pode ser implementado em forma

de módulos. Assim, reconhece a estrutura modular de cada subsistema, sua relação e interação entre si.

Projeto detalhado: o projeto detalhado trata da parte de implementação do que é visto como um sistema e de seus subsistemas nos dois projetos anteriores. É mais detalhado para módulos e suas implementações. Define a estrutura lógica de cada módulo e suas interfaces para se comunicar com outros módulos.

Modularização

A modulação é uma técnica para dividir um sistema de software em múltiplos módulos independentes e discretos, em que se espera que sejam capazes de executar tarefas independentemente. Esses módulos podem funcionar como construções básicas para todo o software. Os projetistas tendem a criar módulos de forma que possam ser executados e/ou compilados separadamente e de forma independente (SURYANARAYANA; SAMARTHYAM; SHARMA, 2015).

O projeto modular segue, involuntariamente, as regras da estratégia de solução de problemas "dividir e conquistar", porque existem muitos outros benefícios associados ao projeto modular de um software.

Agora, você acompanhará as vantagens da modularização:

- Pequenos componentes são mais fáceis de manter.
- O programa pode ser dividido com base em aspectos funcionais.
- O nível desejado de abstração pode ser trazido ao programa.
- Componentes com alta coesão podem ser reutilizados.
- Possibilidade de execução simultânea (paralela).
- Maior segurança.

Concorrência

Existem, basicamente, dois tipos de execução de software. A sequencial é onde todo o software deve ser executado sequencialmente. Por execução sequencial se quer dizer que a instrução codificada será executada uma após outra, o que implica que apenas uma parte do programa está sendo ativada em cada instante. Além da sequencial, há a execução paralela ou concorrente, onde um software, que tem vários módulos ou dados particionados, pode executar de forma paralela. Em relação à arquitetura do software, se tem a

divisão de módulos como importante fator na divisão do software para aplicar o processamento concorrente.

No projeto de software, a concorrência é implementada dividindo o software em múltiplas unidades de execução independentes, como módulos e executando-as em paralelo. Em outras palavras, a concorrência oferece capacidade para o software executar mais de uma parte do código em paralelo entre si. Por isso, é necessário que os programadores e projetistas reconheçam que esses módulos podem ser feitos de execução paralela.

Acoplamento e coesão

Quando um programa de software é modularizado, suas tarefas são divididas em vários módulos com base em algumas características. Como se sabe, os módulos são conjuntos de instruções que se juntam para realização de algumas tarefas. No entanto, são considerados como uma única entidade que pode se referir para trabalhar em conjunto. Existem medidas pelas quais a qualidade de um projeto de módulos e sua interação entre eles podem ser medidas. Logo, essas medidas são chamadas de acoplamento e coesão.

Coesão

A coesão é uma medida que define o grau de acoplamento dentro dos elementos de um módulo. Quanto maior a coesão, melhor é o projeto do sistema. Existem sete tipos de coesão que você irá ver a seguir:

Coesão coincidental: é uma coesão não planejada e aleatória que pode ser o resultado de quebrar o programa em módulos menores por motivos de modularização. Como não é planejado, pode ser uma confusão para os programadores e geralmente não é aceita.

Coesão lógica: quando elementos logicamente categorizados são agrupados em um módulo.

Coesão temporal: quando os elementos do módulo são organizados de forma que são processados em um ponto similar.

Coesão processual: quando os elementos do módulo são agrupados e executados sequencialmente para realizar uma tarefa.

Coesão comunicativa: quando os elementos do módulo são agrupados, executados sequencialmente e funcionam nos mesmos dados (informação).

Coesão sequencial: quando os elementos do módulo são agrupados porque a saída de um elemento serve como entrada para outro e assim por diante.

Coesão funcional: é considerado o maior grau de coesão e é altamente esperado. Os elementos do módulo em coesão funcional são agrupados, porque todos contribuem para uma única função bem definida,também pode ser reutilizado.

Acoplamento

O acoplamento é uma medida que define o nível de interconfiabilidade entre os módulos de um programa. Ele conta em que nível os módulos interferem e interagem uns com os outros. Quanto menor for o acoplamento, melhor será o programa.Existem cinco níveis de acoplamento, os veja a seguir:

Acoplamento de conteúdo é quando um módulo pode acessar ou modificar diretamente ou se referir ao conteúdo de outro módulo.

Acoplamento comum é quando vários módulos têm acesso à leitura e à gravação de alguns dados globais, podendo ser chamado também de acoplamento global.

Controle de acoplamento é quando dois módulos são chamados de controle acoplado, se um deles decidir a função do outro módulo ou mudar o fluxo de execução.

Acoplamento de selo é quando vários módulos compartilham a estrutura de dados comum e trabalham em diferentes partes.

Acoplamento de dados é quando dois módulos interagem por meio de dados passantes (como parâmetro). Se um módulo passa a estrutura de dados como parâmetro, o módulo de recepção deve usar todos os seus componentes.

Fique atento

De maneira ideal, nenhum acoplamento é considerado uma boa alternativa, e sim, a modularização do sistema.

Verificação do projeto

A saída do processo de projeto de software é a documentação do projeto, dos pseudocódigos, dos diagramas de lógica detalhados, dos diagramas de processo e da descrição detalhada de todos os requisitos funcionais ou não-funcionais. A próxima fase, que é a implementação de software, depende de todas as saídas mencionadas acima.

É, então, necessário verificar a saída antes de prosseguir para a próxima fase. O início de qualquer erro é detectado ou melhor, pode não ser detectado até o teste do produto. Se as saídas da fase de projeto estiverem na forma de notação formal, suas ferramentas associadas para verificação devem ser usadas, caso contrário, uma avaliação de projeto completa pode ser usada para verificação e validação.

Desse modo, na abordagem de verificação estruturada, os revisores podem detectar defeitos que podem ser causados por negligência de algumas condições. Por isso, uma eficiente revisão de projeto é importante para um bom projeto de software, com precisão e qualidade.

Documentação do projeto

Assim, o projeto do software é um processo pelo qual os requisitos de software são traduzidos em uma representação de componentes de software, interfaces e dados necessários para a fase de implementação. Nesse processo, cria-se o chamado **documento de especificação de projeto**. Ele mostra como o sistema de software será estruturado para satisfazer os requisitos.

É a principal referência para desenvolvimento de código e, portanto, deve conter todas as informações exigidas por um programador para escrever o código. O documento de especificação de projeto é realizado em duas etapas. A primeira é um projeto preliminar em que a arquitetura geral do sistema e a arquitetura de dados são definidas. Na segunda etapa, o projeto é detalhado em fases e dados mais detalhados. Em sequência, as estruturas são definidas e os algoritmos são desenvolvidos para a arquitetura determinada.

Agora, você verá a estrutura de um documento de especificação de projeto de software:

- **Índice**
- **Introdução**
  - Proposta
  - Escopo
  - Resumo / Visão geral
  - Referencial teórico
  - Definições / Acrônimos
- **Visão geral do sistema**
- **Arquitetura do Sistema**
  - Projeto arquitetural
  - Descrição da decomposição
  - Projeto racional
- **Projeto de Dados**
  - Descrição de dados
  - Dicionário de dados
- **Projeto de Componente**
- **Projeto de Interface**
  - Visão geral da interface
  - Telas
  - Ações e objetos
- **Matriz de Requisitos**
- **Apêndice**

Portanto, seguindo esse roteiro, você conseguirá fazer uma documentação bastante completa e consistente do um projeto de software. Obviamente, alguns itens podem não estar de acordo com o projeto, mas é interessante tentar seguir o padrão estipulado pela IEEE.

Referências

ANGELOV, S.; GREFEN, P.; GREEFHORST, D. A Classification of Software Reference Architectures: Analyzing Their Success and Effectiveness. In: JOINT WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE & EUROPEAN CONFERENCE ON SOFTWARE ARCHITECTURE. *Proceedings*... New York; IEEE, 2009. p. 141–150.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture In Practice*. 3. ed. Boston: Addison-Wesley, 2012.

GARLAN, D.; SHAW, M. *An Introduction to Software Architecture*. Pittsburgh: Carnegie Mellon University, 1994. Disponível em: <https://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf>. Acesso em: 26 set. 2017.

SURYANARAYANA, G.; SAMARTHYAM, G.; SHARMA, T. *Refactoring for Software Design Smells:* managing technical debt. Burlington: Morgan Kaufmann, 2015.258 p.

Leituras recomendadas

BRAUDE, E. *Projeto de software da programação à arquitetura:* uma abordagem baseada em Java. Porto Alegre: Bookman , 2009.

ENGHOLM JR., H. *Engenharia de Software na prática*. São Paulo: Novatec, 2010.

FREEMAN, P.; HART, D. A Science of design for software-intensive systems. *Communications of the ACM,* New York, *v.* 47, n. 8, p. 19–21, 2004.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.