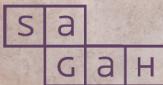


ARQUITETURA DE SISTEMAS

Paulo Antonio Pasqual Júnior



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

Introdução à arquitetura de sistemas

Objetivos de aprendizagem

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

- Reconhecer a arquitetura de *software* e sua importância.
- Identificar os tipos de arquiteturas.
- Relacionar a escolha do tipo de arquitetura para o desenvolvimento de sistemas.

Introdução

A arquitetura de sistemas é um elemento importante da engenharia de *software*, em que são definidos quais componentes farão parte de um sistema e como eles se comunicarão. Definir a arquitetura é importante principalmente para que a equipe de desenvolvimento tenha uma visão geral do que será o produto do *software*.

Existem diversos tipos de arquitetura de sistemas. Escolher uma arquitetura é fundamental para garantir um *template*, que será o norteador do desenvolvimento do sistema. Neste capítulo, você vai estudar a arquitetura de um *software*, verificando sua importância e identificando os tipos de arquitetura existentes. Por fim, você vai analisar a escolha do tipo de arquitetura no desenvolvimento de sistemas.

Conceitos fundamentais

Quando pensamos em arquitetura de sistemas, é comum fazermos analogias com a construção civil, assim como é comum comparar a engenharia de *software* com as outras engenharias. A **arquitetura de sistemas** é uma subárea da engenharia de *software*. Enquanto a primeira se preocupa com as características

do sistema, dos elementos necessários e das suas interrelações, a segunda se preocupa com todos os processos que envolvem o desenvolvimento de um produto de *software*, abrangendo a concepção, o gerenciamento do projeto e o desenvolvimento do produto.

Apesar de as analogias com a construção civil serem úteis para que entendamos esses conceitos, é importante ressaltar que arquitetura de *software* não é uma área independente, como é o caso da arquitetura no âmbito da construção civil. Pressman e Maxim (2016, p. 253) definem arquitetura de *software* da seguinte forma: “A arquitetura de *software* de um programa ou sistema computacional é a estrutura ou as estruturas do sistema que abrange os componentes de *software*, as propriedades externamente visíveis desses componentes e as relações entre eles”.

Pressman e Maxim (2016) ainda acrescentam que a arquitetura do sistema consiste em um modelo relativamente pequeno e compreensível acerca de como o sistema será estruturado e como os seus componentes vão se comunicar. Gallotti (2016) complementa essa definição ao dizer que, quando falamos da arquitetura de um *software*, estamos nos referindo à estrutura interna do seu sistema. Basicamente, ela explica como um *software* se organiza, funciona, além da forma como é implementado.



Fique atento

As analogias entre a arquitetura e a engenharia no âmbito da construção civil são comuns para que entendamos os processos no campo do desenvolvimento de sistemas. Contudo, é importante ressaltar que a arquitetura de sistemas não é uma área distinta da engenharia de *software*; na verdade, a arquitetura de sistemas é uma subárea da engenharia de *software*.

Ao fazermos uma analogia, mais uma vez, com a construção civil, o projeto arquitetônico se preocupa basicamente em satisfazer os requisitos do cliente e apresentar para ele uma ideia concreta do projeto. Por exemplo, imagine o projeto de uma casa; o arquiteto, após levantar os requisitos com o cliente, vai elaborar uma planta e uma visualização em 3D do que ele apreendeu a partir do que foi estabelecido pelo cliente. O arquiteto vai escolher os elementos que

ele julga ideais, incluindo formas e acabamentos, para que o projeto esteja mais próximo do que foi solicitado.

Nesse sentido, Sommerville (2007) explica que o projeto de arquitetura é o primeiro passo no processo do desenvolvimento de *software*, sendo o elo entre o projeto e a engenharia de requisitos, já que identifica os principais componentes estruturais do *software*. O autor ainda ressalta que a importância do **projeto de arquitetura de sistemas** está relacionada ao desempenho e à robustez, além da capacidade de manutenção e distribuição do sistema.

Normalmente, em uma empresa de desenvolvimento de *software*, os processos envolvendo o projeto de arquitetura e o desenvolvimento do sistema ocorrem paralelamente, e as funções são compartilhadas entre arquitetos de sistema e desenvolvedores de *software*. Dificilmente uma equipe será responsável apenas pela programação, enquanto outra fica responsável pela modelagem. Isso porque essa distinção de responsabilidades não faz sentido, já que, geralmente, o processo de desenvolvimento de *software* ocorre em ciclos iterativos, que podem modificar o projeto original.



Fique atento

Ao se considerar a arquitetura de um sistema, são analisados principalmente os requisitos apontados pela equipe responsável pelo levantamento de requisitos. Analisar detalhadamente os requisitos contribui para que a escolha da arquitetura esteja de acordo com os objetivos do desenvolvimento do sistema.

Pressman (2011) explica que a arquitetura de sistemas é importante por três motivos. Em primeiro lugar, as diversas representações da arquitetura de *software* permitem a comunicação entre todas as partes envolvidas no desenvolvimento de um sistema computacional. Em segundo lugar, porque a arquitetura implica em decisões durante a fase inicial que impactam todo o desenvolvimento do projeto e o produto final de *software*. Por fim, a arquitetura se constitui como um modelo mental de como o sistema será estruturado e como os seus componentes trabalharão em conjunto.

Nesse contexto, Sommerville (2007), baseado em outros autores, amplia a reflexão sobre a importância da arquitetura do sistema, apontando as três vantagens do projeto de *software* listadas a seguir.

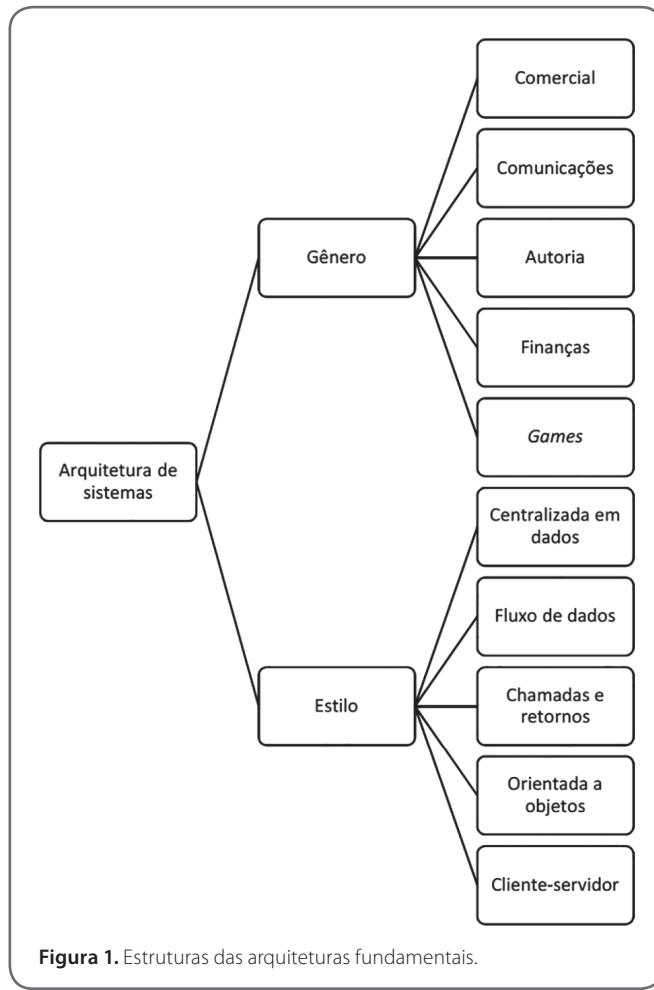
- **Comunicação de stakeholders:** refere-se à capacidade de a arquitetura representar, em alto nível, o sistema, garantindo assim uma comunicação entre as partes interessadas.
- **Análise do sistema:** a análise do sistema permite que a arquitetura torne explícitos alguns elementos do sistema ainda na fase inicial, o que permite à equipe analisar detalhes do sistema que não poderiam ser considerados sem um projeto arquitetural.
- **Reúso de componentes:** como a arquitetura do sistema mapeia os componentes e suas relações, o projeto de arquitetura viabiliza a reusabilidade de *software* para sistemas com requisitos semelhantes.

Sommerville (2007) ainda aponta que o uso de um projeto de arquitetura contribui para viabilizar discussões acerca do sistema, já que possibilita uma visão de alto nível do sistema. Além disso, o autor argumenta que o projeto de arquitetura possibilita uma forma de documentação do sistema, em que são explicitados os componentes, suas propriedades e suas interpelações.

A **escolha da arquitetura** pode contribuir para que o projeto nasça alinhado aos requisitos para os quais ele será desenvolvido. Definir um tipo de arquitetura de sistemas corrobora para que sejam detalhados e desenvolvidos componentes que poderão ser reutilizados mais tarde, além de garantir a manutenibilidade de cada um desses componentes de forma mais simples. Dessa forma, o arquiteto de *software* terá, dentre outras funções, a responsabilidade de mapear quais são esses componentes e como eles vão interagir com os outros elementos do sistema. Além disso, o arquiteto precisa ser capaz de identificar os componentes e saber quais deles podem ser reutilizados, melhorando o tempo de desenvolvimento.

Tipos de arquitetura de sistemas

As decisões que devem ser tomadas em relação às arquiteturas de sistemas podem ser classificadas utilizando diversos fatores. Uma visão dessa classificação pode ser observada na Figura 1.



No âmbito da arquitetura de sistemas, podemos elencar três elementos fundamentais para qualquer projeto.

- 1. Componentes:** unidades principais do sistema; podem ser caracterizados por módulos de *software* com responsabilidades específicas dentro do sistema.
- 2. Propriedades:** elementos que se referem às particularidades de cada componente.

3. Conectores: caracterizados por todas as formas de conexão entre os componentes de um sistema. Eles podem ser caracterizados por chamadas de métodos, envio e requisição de mensagens e outras formas que garantem a interação entre os componentes do sistema.

Entre os componentes, as propriedades e os conectores que podem ser usados no âmbito da arquitetura de sistemas, Pressman (2011) aponta as principais como “estruturas de arquitetura canônicas”, as quais se encontram listadas a seguir.

- **Estrutura funcional:** os componentes se referem a entidades que refletem funcionalidade. Os conectores se referem à capacidade das interfaces em transmitir dados para um componente e, desse modo, fornecer a capacidade de uso. Já as propriedades se referem à descrição dos componentes e à organização das interfaces.
- **Estrutura de implementação:** nessa estrutura, os componentes se referem a elementos para “empacotar” funcionalidades em vários níveis de abstração, como classes, funções, métodos, objetos, etc. Os conectores se referem a como esses elementos vão se comunicar dentro do sistema, transmitindo dados de controle e instâncias de objetos e compartilhando dados. As propriedades se referem a elementos de qualidades como manutenibilidade e reusabilidade.
- **Estrutura de concorrência:** nessa estrutura, os componentes são responsáveis por representar estruturas para tarefas paralelas ou *threads*. Os conectores “[...] incluem as relações *sincronizações-com*, *é-de-maior prioridade-que*, *envia-dados-a*, *não-é-possível-executar-sem* e *não-é-possível-executar-com*” (BASS, 2003 *apud* PRESSMAN, 2010, p. 235).
- **Estrutura física:** os conectores consistem nas interfaces entre os componentes de *hardware*, já as propriedades se referem a desempenho, largura da banda, entre outros aspectos.
- **Estrutura de desenvolvimento:** refere-se aos componentes, artefatos e outras fontes que são necessárias durante o desenvolvimento dos processos da engenharia de *software*. Os conectores têm a função de representar o relacionamento entre os artefatos e as propriedades, identificando as características de cada item. Já as propriedades identificam as características de cada item.

Outro elemento fundamental no processo de escolha da arquitetura é a definição do **gênero arquitetural** do sistema. Pressman (2011) entende gênero como uma categoria particular de sistemas; nesse sentido, inclui toda a variedade de produtos de *software*, como aplicativos, sistemas comerciais, de escritório e tantos outros. Cada um desses gêneros de arquitetura demandam a existência de estruturas específicas a serem modeladas e desenvolvidas. A seguir, estão listados alguns exemplos de gêneros de arquitetura de sistemas, com base em Pressman (2011).

- Inteligência artificial
- Comercial
- Comunicações
- Autoria de conteúdo
- Dispositivos
- Esportes e entretenimento
- Financeiros
- Jogos
- Governo
- Industriais
- Legais
- Médicos
- Sistemas operacionais
- Plataformas
- Científicos
- Ferramentas
- Transportes
- Serviços públicos

A partir da definição do gênero arquitetural, também devem ser definidos os **estilos de arquitetura** que poderão nortear o projeto. Existem diversos estilos de arquitetura de sistemas; Pressman (2011), ao exemplificar esses estilos, faz a analogia com os estilos arquitetônicos presentes nas casas americanas. Segundo o autor, quando se fala em estilo colonial americano com *hall* central, muitas pessoas familiarizadas com esse estilo arquitetônico já imaginarão do que se trata (PRESSMAN, 2011). Do mesmo modo, os estilos arquitetônicos no âmbito de sistemas consistem, segundo Pressman (2011, p. 234), em um “[...] um *template* para a construção”.

Ou seja, é por meio da definição de estilos que se podem considerar as características personalizadas a serem acrescentadas ao produto que será desenvolvido. Nesse sentido, o estilo arquitetônico de um sistema define o conjunto de componentes, as conexões entre eles, as restrições e os modelos semânticos que permitem que o projetista compreenda as propriedades gerais de um sistema. De acordo com Pressman e Maxim (2016), embora muitos sistemas tenham sido desenvolvidos nos últimos anos, a maior parte deles pode ser classificada em pequenos estilos de arquitetura.

- **Arquitetura centralizada em dados:** consiste em sistemas que compartilham o banco de dados ou um sistema de arquivos como elemento central do sistema. Todos os componentes do sistema se comunicam individualmente, tendo como elo apenas o banco de dados. A troca de informação é feita entre os componentes do sistema e o banco de dados.
- **Arquitetura de fluxo de dados:** é um tipo de arquitetura que é aplicada quando dados de entrada devem ser transformados passando por várias etapas, em um fluxo dentro do próprio sistema, até as etapas de saída. Geralmente, um componente se comunica com outro de forma restrita e interdependente.
- **Arquitetura de chamadas e retornos:** consiste em uma arquitetura em que o programa é desenvolvido de forma hierárquica, por meio de um programa principal, que pode acessar outros subprogramas e, assim, sucessivamente. Esse tipo de arquitetura é útil, pois essa estrutura é relativamente fácil de aumentar e modificar, segundo Pressman e Maxim (2016).
- **Arquiteturas orientadas a objetos:** são sistemas baseados em classes e objetos com certo grau de encapsulamento, em que a comunicação e a coordenação ocorrem entre os objetos por meio de mensagens.
- **Arquitetura em camadas:** praticamente todo sistema, independentemente da sua função, é baseado em camadas. Essa arquitetura consiste em níveis que vão da interface (camada mais externa) até a camada central (mais próxima da linguagem de máquina).

De maneira geral, a partir da escolha da arquitetura do sistema, ficarão evidenciadas as **camadas** que o sistema terá. Se a arquitetura for de um sistema operacional, obviamente as camadas serão diferentes de um *software* de gestão empresarial. Entre as camadas comuns em um *software*, temos:

- interface;
- camada de negócios;
- camada central.

Uma vez definida a arquitetura do sistema e as camadas que serão utilizadas, cada uma dessas camadas será desenvolvida com as suas especificidades.

Definir a arquitetura e as camadas de um sistema pode contribuir também para que os próximos projetos de *software* sejam desenvolvidos de forma mais rápida e com menor número de erros. Isso ocorre porque essas camadas podem ser reaproveitadas para outros sistemas similares.



Exemplo

Um exemplo bastante simples é o uso da **interface**. Imagine um sistema de cadastro e gerenciamento de clientes que foi desenvolvido para uma empresa X. Quando a equipe de desenvolvimento de *software* precisar desenvolver um outro *software* do mesmo gênero, a interface será muito próxima da que já está desenvolvida, assim como o banco de dados. Assim, bastará integrar o novo sistema às camadas já existentes. Como essas camadas já foram utilizadas outras vezes e testadas, a probabilidade da incidência de erros será muito menor do que se o sistema fosse desenvolvido desde o início.

No âmbito dos estilos de arquitetura, Sommerville (2007) ainda acrescenta a **arquitetura cliente-servidor**, que consiste em uma arquitetura baseada em serviços, ou seja, cada cliente acessa o servidor para ter acesso aos dados e funcionalidades específicas. Muitos sistemas possuem esse tipo de arquitetura; em especial, esse tipo de arquitetura privilegia sistemas distribuídos.

Arquitetura de sistemas e desenvolvimento de *software*

Quando você pensa em tipos de arquitetura, o que vem à sua mente? É possível que você imagine diferentes tipos de construções, sejam elas casas ou prédios. Enfim, podemos pensar em diversos tipos de arquitetura — clássica, colonial, moderna, entre outras. Mais uma vez pensando na analogia com a construção civil, quando um arquiteto começa a desenvolver um projeto, certamente ele já terá em mente a arquitetura em que ele se baseará. O mesmo acontece quando falamos em arquitetura de sistemas. Antes que o processo

de desenvolvimento seja iniciado, é provável que o arquiteto de sistemas, a partir dos requisitos apontados pela equipe, seja capaz de indicar um tipo de arquitetura a ser seguida.



Exemplo

Suponha que um cliente tenha solicitado um *game* para computador. Assim, o arquiteto de *software* deverá determinar quais componentes desse sistema serão necessários para que a aplicação funcione, bem como quais serão as formas de conectar cada um desses componentes. Da mesma forma que um *software* comercial, um *game* possui uma interface e um banco de dados, além das especificidades de um *game*, que são diferentes daquelas de outras arquiteturas. Por exemplo, o *game* poderá ser controlado por teclado, controle ou *mouse*? Poderá se comunicar com algum sensor de movimento? Quais outras questões são distintas de um *software* de gestão? Todas essas questões deverão ser consideradas para que a escolha, a modelagem e o desenvolvimento do sistema esteja de acordo com os requisitos necessários para viabilizar o projeto. Ao definir a arquitetura, deve-se mapear não só esses elementos, mas também as interpelações entre eles.

Para construir uma arquitetura, é importante considerar a utilização de diagramas de contexto arquitetural e de arquétipos; enfim, deve ocorrer o refinamento da arquitetura. O **diagrama de contexto arquitetural** (Figura 2) é utilizado para mapear e modelar a maneira como o *software* vai interagir com entidades externas, conforme aponta Pressman (2011). Nesse contexto, o sistema que está sendo desenvolvido é representado como o centro do processo, e são expressas no diagrama as relações com os sistemas apresentados a seguir.

- **Sistemas superiores:** consistem em sistemas que usam o sistema-alvo.
- **Sistemas subordinados:** sistemas que são utilizados pelo sistema-alvo.
- **Sistemas de mesmo nível:** consistem em sistemas que compartilham informação com o sistema-alvo.
- **Atores:** consistem em entidades, pessoas ou dispositivos que utilização o sistema, como se pode observar na Figura 2.

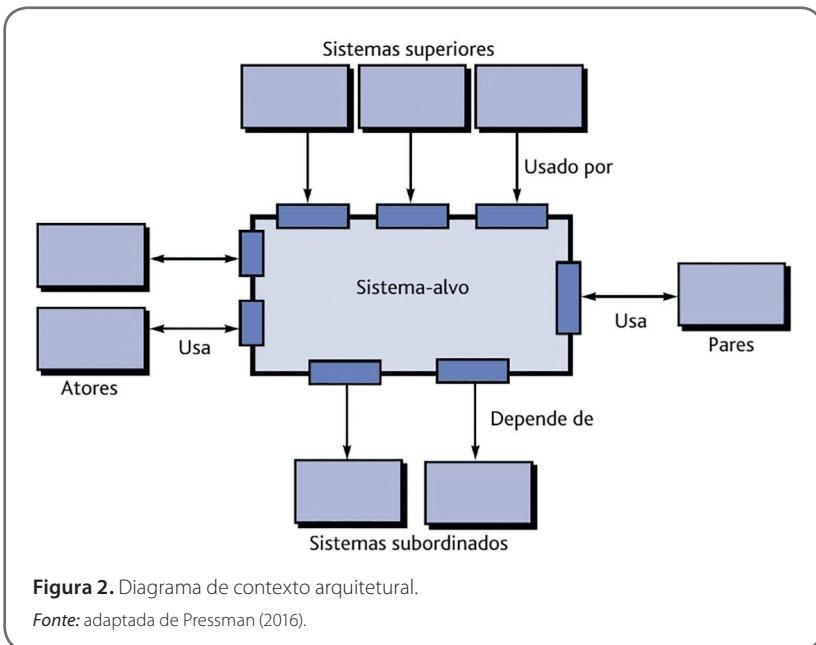


Figura 2. Diagrama de contexto arquitetural.

Fonte: adaptada de Pressman (2016).

Os **arquétipos** são utilizados para representar conceitos e conhecimentos de um domínio específico. São descritos como modelos formais e reutilizáveis do conceito do domínio, que são aplicados nos vários cenários em que esses conceitos são utilizados. Por exemplo, a pressão sanguínea pode ser representada pelos valores da pressão sistólica e diastólica, pela data e horário em que a pressão foi aferida e pelo local onde a pressão foi aferida. Sempre que a pressão sanguínea precisar ser utilizada, o modelo conceitual da pressão será utilizado. Os arquétipos formam a base da arquitetura, mas devem ser refinados à medida que a arquitetura é detalhada.

Na sequência, os componentes da arquitetura devem ser definidos com base no domínio da aplicação, na infraestrutura e no diagrama de contexto arquitetural. Os componentes da arquitetura devem, portanto, ser modelados. Por fim, uma das etapas finais do projeto de arquitetura consiste em desenvolver uma instância real do problema, mostrando que a estrutura e os componentes projetados são adequadas à necessidade do *software*.

Ao fim do processo de projeto de arquitetura, é comum que outro processo se inicie: o processo de **avaliação do projeto**. Nessa fase, são analisados os projetos de arquitetura e, a partir de uma série de princípios, são analisados os pontos fortes e fracos da arquitetura escolhida para o desenvolvimento do sistema. Esse processo pode ocorrer imediatamente após o projeto de arquitetura, ou poderá ocorrer durante a implementação do sistema.

Pressman (2011) aponta três modelos para a avaliação de um projeto de arquitetura. O primeiro corresponde a analisar o projeto desde os casos de uso, passando por fases de elicitação de requisitos e descrição de estilos, pela análise de atributos de qualidade, entre outras fases, até chegar ao ponto em que são analisados os pontos fortes das arquiteturas candidatas. Uma segunda forma, ainda segundo o autor, consiste em avaliar a complexidade da arquitetura a partir da análise dos componentes e de suas interdependências. Como última forma de avaliação, Pressman (2011) aponta a utilização de uma linguagem formal para o mapeamento da arquitetura, que pode contribuir para um processo de avaliação mais formal. Tendo esses processos finalizados, o sistema então poderá ser implementado, levando em consideração todos os aspectos elencados pela equipe de projeto e a avaliação da arquitetura de sistemas.

No âmbito do desenvolvimento do sistema, da escolha e do projeto de arquitetura, Sommerville (2007) ressalta a necessidade de se focar nos **requisitos não funcionais**, uma vez que eles estão diretamente vinculados à arquitetura do sistema. Nesse sentido, o autor lista alguns exemplos de requisitos não funcionais que podem ser influenciados pela arquitetura do sistema, são eles: desempenho, proteção, segurança, disponibilidade e manutenção.



Exemplo

Por exemplo, se o **desempenho** for um requisito não funcional, o sistema não poderá estar organizado em uma série de pequenos componentes, que levam tempo para serem acessados e geram um *feedback* tardio para o usuário. Do mesmo modo, se um requisito não funcional for a **segurança**, um módulo de segurança deve estar restrito ao menor número de componentes possível, o que possibilita maior segurança em relação aos dados desse sistema.

Sommerville (2007) ainda aponta que, muitas vezes, dependendo dos requisitos não funcionais, pode haver conflitos; por exemplo: o uso de grandes componentes melhora o desempenho, mas o uso de pequenos componentes melhora a manutenibilidade. Se desempenho e manutenibilidade forem requisitos não funcionais, é preciso avaliar uma arquitetura que possa contemplar, ao menos em partes, ambos os requisitos não funcionais.

O autor aponta que, muitas vezes, o projeto de arquitetura pode ser desnecessário, quando o sistema é relativamente pequeno e simples de desenvolver, ao passo que, no caso de sistemas críticos, o projeto de arquitetura contribui para a redução de possíveis falhas do sistema. Em síntese, o projeto de arquitetura se relaciona com o desenvolvimento do sistema, na medida em que fornece um modelo conceitual, que, ainda no início da fase de desenvolvimento, possibilita uma visão geral do sistema, com base não apenas nos requisitos funcionais, mas também nos requisitos não funcionais.



Referências

- GALLOTTI, G. M. A. (org.). *Arquitetura de Software*. São Paulo: Pearson Education do Brasil, 2016.
- PRESSMAN, R. S. *Engenharia de software: uma abordagem profissional*. 7. ed. Porto Alegre: McGraw-Hill, 2011.
- PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.
- SOMMERVILLE, I. *Engenharia de software*. 8. ed. São Paulo: Pearson Addison Wesley, 2007.

Leitura recomendada

- BARBOSA, G. M. G. *Um livro-texto para o ensino de projeto de arquitetura de software*. 2009. Dissertação (Mestrado em Ciência da Computação) — Centro de Engenharia Elétrica e Informática, Universidade Federal de Campina Grande, Campina Grande, SP, 2009.

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

Conteúdo:



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS