VANTAGENS E DESVANTAGENS DO DIAGRAMA DE PACOTES

DINIZ, Felipe.
GILI, Johnny.
GOULART, Luciano.
ROCHA, Gabriel.
ZANINI, Elaine de Oliveira.

Resumo

Neste trabalho é evidenciado as vantagens e desvantagens do diagrama de pacotes no desenvolvimento de software, onde para a elaboração do artigo, com base em pesquisas de fontes renomadas foram reunidas informações com o intuito de formar uma pesquisa bibliográfica qualitativa. O diagrama de pacotes é uma ferramenta útil que oferece inúmeras vantagens na modelagem de sistemas de software, permitindo uma melhor compreensão, organização, manutenção, criação e qualidade no desenvolvimento de projetos. Ele desempenha um papel fundamental na engenharia de software, proporcionando uma base sólida para a construção de softwares de alta qualidade e produtividade. Como resultados, observamos que o diagrama de pacotes se diferencia por ser um recurso especial na organização de sua forma de criação de modelos, simplificando o desenvolvimento de sistemas grandes ou complexos, contudo possui como desvantagens não apresentar os atores e funções as suas saídas e entradas. Conclui-se que dentre os diversos tipos de diagrama UML, o diagrama de pacotes é uma ferramenta que apresenta dentre suas vantagens, maior visibilidade e manutenção das grandes ou pequenas funções e subfunções de projetos.

PALAVRAS-CHAVE: Diagrama de pacotes; Desenvolvimento de software; UML; Projetos;

1 INTRODUÇÃO

A engenharia de software é a área profissional responsável pela criação, manutenção e desenvolvimento de sistemas, os profissionais da área podem trabalhar em diferentes funções, que vão desde a criação do código, até a documentação dos processos. Conforme os projetos foram ficando complexos, se tornou cada vez mais necessário que a qualidade do software fosse excepcional, para que não houvesse problemas na utilização dos usuários (XPEDUCAÇÃO, 2023).

É de extrema importância que na criação de um software a aplicação tenha qualidade e atenda aos requisitos que o cliente necessita, é justamente aí que entra a qualidade de software, que é dada por um conjunto de fatores a serem atendidos, nesse sentido é obrigatório que a aplicação esteja de acordo com os requisitos do cliente (MONITORIATEC, 2021).

Na década de 80 , as definições e nomenclaturas dos projetos eram definidas de forma não convencional, fazendo com que houvesse alguns conflitos nos projetos, principalmente nos mais complexos, devido a isso em 1994 começou-se a criação da UML, que serviria como uma forma de modelagem unificada e padronizada, criando alguns tipos de diagramas para atender diversas necessidades e resolvendo os problemas que existiam na época (DEVMEDIA, 2023).

Os diagramas UML podem atender diferentes necessidades quando se trata de documentação de processos e interações de um sistema. Dentre os tipos de diagramas existem aqueles que tratam das interações entre as classes de um sistema, basicamente lidam com os aspectos dinâmicos do sistema, existem também os diagramas estruturais que lidam os os aspectos estáticos do sistema, como classes e componentes. É dentro dos estruturais que existe o diagrama de pacotes, tem como objetivo descrever as relações e os comportamentos estruturais dos pacotes de um sistema (CREATELY, 2022).

O diagrama de pacotes é o tema principal deste artigo, que tem como objetivo evidenciar as vantagens e desvantagens do diagrama de pacotes no desenvolvimento de software, onde para a elaboração do artigo, com base em pesquisas de fontes renomadas foram reunidas informações com o intuito de formar uma pesquisa bibliográfica qualitativa.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ENGENHARIA DE SOFTWARE

Durante toda a década de 1980 a engenharia de software tinha em seu maior problema o custo de seus hardwares, não necessariamente o seu custo monetário, mas, pelo fato deles não terem o armazenamento de dados e processamento desejados, o que se prolonga de certa forma até a contemporaneidade, mas, não da mesma forma, logo, nos tempos atuais o maior problema é a qualidade, com soluções de criadas em softwares que tentam solucionar este problema de lógica (PRESSMAN, 2021).

Outro transformação na engenharia de software foi como foi aberto um novo escopo, na sua parte de planejamento, onde antes simples, mas, com dificuldade em seus processos de

desenvolvimento de software, na atualidade inverteu-se para uma complexidade de seus processos de planejamento, comunicação, como também seguinte destes, modelagem, construção e por fim entrega, mas, facilidade em seus desenvolvimentos de software pelas grande quantidade de ferramentas criadas e utilizadas atualmente (BEZERRA, 2002).

A engenharia de software é uma forma de conciliar diferentes engenharias de sistemas e da parte física o hardware, segundo Pressman (2021 p.31), “a engenharia de software é um rebento da engenharia de sistemas e de hardware, onde em 3 elementos básicos, ferramentas, procedimentos”, e por fim métodos, são o que possibilita uma base para a construção de software de alta qualidade produtividade.

Portanto, a engenharia de software desempenha um papel grande para fundamental no desenvolvimento de sistemas de software complexos e de alta qualidade, tendo em vista, que um dos papéis essenciais da engenharia de software é a organização eficiente e estruturada do desenvolvimento (BEZERRA, 2002).

2.2 QUALIDADE DE SOFTWARE

É um termo subjetivo de cada pessoa, mas, a qualidade é uma forma de expressão muito antiga, portanto, é um tema falado desde os tempos antigos do egípcios, como por exemplo, uma padrão de medida “cíbito”, que pegava o tamanho do braço do faraó vivo, que servia como padrão de medida para tudo, seguindo de certa forma um “padrão de qualidade” imposto, e isto, se seguiu para outras sociedades como a Roma e Grécia antiga até a atualidade com revoluções industriais e no Brasil as normas ABNT, portanto, a qualidade é um termo de um ou mais padrões que devem ser respeitados, mostrando que pelo tempo ela vem se tornando cada vez mais frequente e importante (DEVMEDIA, 2023).

A qualidade de software nos tempos atuais é mais buscada pela maior parte das empresas no ramo de engenharia de software, onde é buscado tanto a qualidade do software através da preocupação nos processos de desenvolvimento do software em si, quanto nos desenvolvedores de software, buscando como segundo Devmedia (2023), “atendimento às necessidades do cliente, conhecimento do processo para melhorá-lo ou também seguir padrão”, mas, a busca da qualidade não é somente dos desenvolvedores, mas, como também dos clientes

que ao longo do tempo estão ficando mais exigentes exigentes, logo, segundo Devmedia (2023), a qualidade de software é “usar boas metodologias para se desenvolver um produto.”.

2.3 COMO SURGIU A UML

No final da década de 80, as definições e nomenclaturas dos projetos de software muitas vezes eram decididos de acordo com o contentamento dos desenvolvedores, oque gerava muitos conflitos nos projetos, uma vez que não existia um padrão a ser seguido. Conforme a tecnologia e as linguagens de programação iam evoluindo e as aplicações ficando mais complexas começou a surgir uma necessidade de um modelo que suprisse todas as necessidades, pois os modelos existentes como o Booch, Oose e OMT não atendiam todas as necessidades de um projeto de software. Em meio a esse problema surgiram três nomes: Ivar Jacobson , Grady Booch e James Rumbaugh, considerados os idealizadores da UML. Esses três foram responsáveis por pegar as melhores ideias e uni-las criando uma linguagem unificada de modelagem, usando técnicas orientadas a objetos eles criaram uma técnica que ia desde o conceito até o sistema executável, que era efetiva tanto em pequenos projetos quanto em projetos mais complexos (DEVMEDIA, 2023).

A criação da UML se iniciou oficialmente em outubro de 1994 quando Rumbaugh se juntou a Booch na Rational Com o objetivo de unificar os métodos Booch e OMT, decorrido um ano de trabalho, foi lançado, em outubro de 1995, o esboço da versão 0.8 do *Unified Process* – Processo Unificado (como era conhecido). Nesta mesma época, Jacobson se associou à Rational e o escopo do projeto da UML foi expandido para incorporar o método OOSE. Nasceu então, em junho de 1996, a versão 0.9 da UML. Finalmente em 1997 a UML foi aprovada como padrão pela OMG(Object Management Group – um consórcio internacional de empresas que define e ratifica padrões na área de Orientação a Objetos (FELIZARDO, 2013).

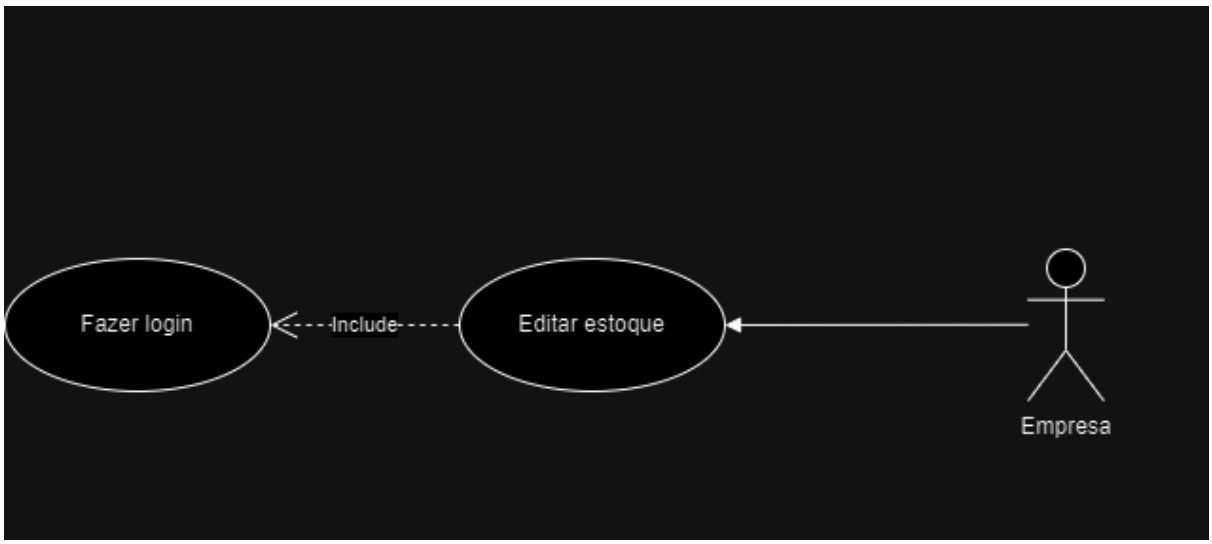
2.3.1 Principais diagramas UML

Diagramas podem ser entendidos como grafos, contendo nós, caminhos entre esses nós e textos. Esses nós podem ser símbolos gráficos (figuras bidimensionais) podendo ser

complexos na medida do possível. Os caminhos podem ser linhas com diversas características, podendo ter até diferentes decorações que podem servir como uma espécie de legenda, que demonstra o tipo e o objetivo daquela linha. E os textos podem aparecer de diversas formas tendo significados diferentes.

A Figura 1 mostra um exemplo de diagrama de causa de uso.

Figura 1: Exemplo de diagrama de casos de uso.



Fonte: ELABORADO PELOS AUTORES (2023).

O exemplo acima é uma ilustração de um diagrama de casos de uso, onde são usados dois tipos de linhas e setas, que possuem características diferentes onde cada uma tem um significado, fazendo com que não seja preciso escrever o que elas representam, uma vez que seus objetivos já estão representados pelas suas características. Dessa maneira fica muito mais prático e dinâmico demonstrar e representar quaisquer tipo de funcionalidade. Vale ressaltar que os diagramas não são utilizados somente na área de desenvolvimento de software, já que são muito úteis em diversas áreas como administração de empresas, por exemplo (GUDWIN, 2010).

2.3.2 Diagramas UML: Principais tipos

A *Unified Modeling Language*, ou Linguagem Unificada de Modelagem é, como o nome indica, é uma linguagem de notação utilizada para modelar e documentar as diversas fases do desenvolvimento de sistemas orientados a objetos. Para isso são usados uma série de elementos gráficos como retângulos, setas , balões e linhas que são utilizados para representar os diversos componentes de uma aplicação, suas interações e mudanças. Trata-se de uma linguagem de modelagem única, cujo papel é auxiliar a equipe de desenvolvimento a visualizar os diversos aspectos da aplicação, facilitando a compreensão do seu funcionamento (BETRYPE, 2022).

Existem duas categorias principais de diagramas : Diagramas de estrutura e diagramas de classes. Diagramas de estrutura mostram as coisas no sistema modelado. Em um termo mais técnico elas mostram diferentes objetos em um sistema. Os diagramas estruturais são : Diagrama de classes, Diagrama de componentes, Diagrama de implantação, Diagrama de objetos, Diagrama de pacotes, Diagrama de perfil e Diagrama de estrutura composta. Já os diagramas comportamentais são : Diagrama de caso de uso, Diagrama de atividades , Diagrama da máquina do estado, Diagrama de sequência, Diagrama de comunicação, Diagrama de visão geral da interação e Diagrama de tempo (CREATELY, 2022).

Dentre todos os tipos iremos abordar os principais utilizados no processo de modelagem de software.

2.3.3 Diagramas de casos de uso

Os diagramas de caso de uso são uma ferramenta de modelagem de software que tem como objetivo a visualização, especificação e documentação do comportamento de um sistema em uma perspectiva externa, de usuário. Eles são um dos diagramas mais utilizados na engenharia de software, pois permitem a comunicação eficaz entre os stakeholders envolvidos no desenvolvimento de um sistema, como usuários, desenvolvedores e gerentes (DEVMEDIA, 2023).

Segundo Pressman (2021), “Uma descrição da interação entre um ator e um sistema para alcançar um objetivo”. Os atores são entidades externas que interagem com o sistema, como usuários, outros sistemas ou dispositivos. Os objetivos são os resultados que o ator deseja alcançar ao interagir com o sistema.

Jackbon (2019), afirma que os diagramas de casos de uso são uma ferramenta valiosa para a comunicação e a compreensão dos requisitos de um sistema. Eles permitem que os stakeholders visualizem como o sistema será usado e quais são os seus objetivos.

Sommerville (2011), destaca que os diagramas de casos de uso são uma forma de descrever os requisitos funcionais de um sistema. Eles são úteis para identificar as funcionalidades que o sistema deve fornecer e para garantir que essas funcionalidades atendam às necessidades dos usuários.

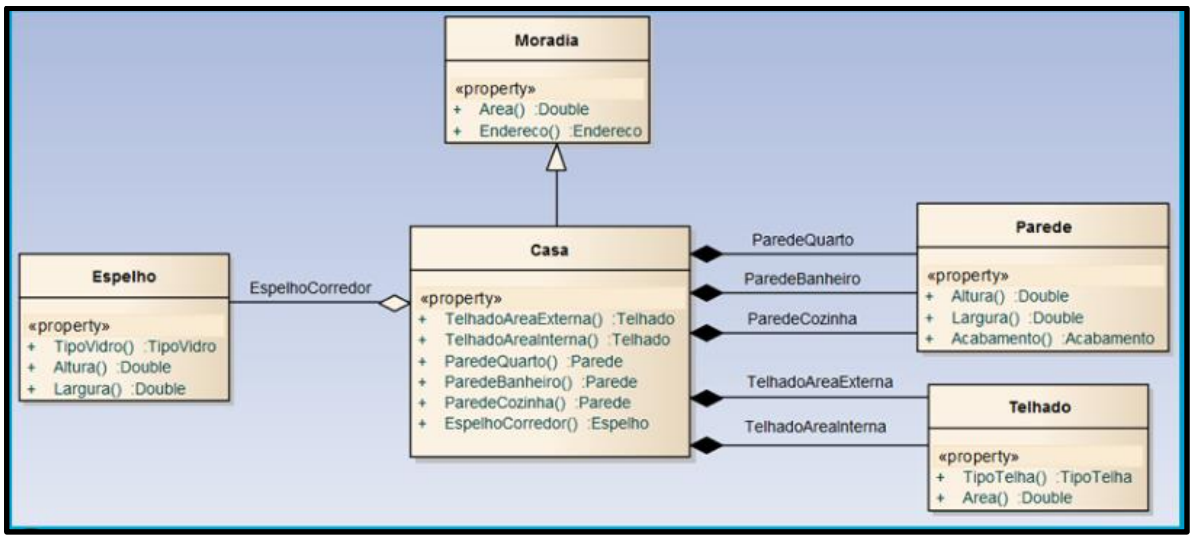
2.3.4 Diagrama de classe

Para entendermos melhor o que é o Diagrama de classes, é importante compreender o significado de classe. Na programação orientada a objetos, a classe funciona como um molde que molda os objetos e define características desses objetos, fazendo uma analogia é possível entender melhor como uma classe funciona, imaginando que você está escolhendo um carro para comprar,, você pode escolher diversos modelos com características diferentes, como por exemplo: cor, tipo de câmbio,quatro portas ou duas portas. Essas características são chamadas de atributos e os comportamentos do carro são chamados de métodos. Todos os tipos de carros nesse caso são considerados objetos onde cada carro tem suas características, porém, todos são pertencentes a classe carro ou na linguagem usada na engenharia de software são instâncias da classe carro (ALURA, 2023).

Entendendo melhor o conceito de classe em programação orientada a objetos, podemos compreender de maneira mais clara o que é e como funciona o diagrama de classes. Diagrama de classes é um tipo de diagrama uml que descreve um sistema visualizando os diferentes tipos de objetos dentro de um sistema e as relações estáticas que ocorrem entre eles. São normalmente usados para explorar os conceitos de domínio, compreender os requisitos de software e descrever processos detalhadamente(CREATELY, 2022)

A Figura 2 mostra um exemplo de diagrama de classes.

Figura 2: Exemplo de um diagrama de classes



Fonte: VENTURA (2018).

No exemplo acima temos uma classe moradia e o objeto casa, com os atributos parede, telhado e espelho, todos esses atributos estão ligados ao objeto casa pois eles interagem com casa, enquanto a casa é interligada em moradia, pois é uma instância da classe moradia.

2.3.6 Como e quando utilizar um diagrama de classes

É importante utilizar o diagrama de classes para modelar os objetos que compõem o sistema , mostrar as relações entre entre esses objetos e para descrever o que esses objetos fazem e quais serviços eles oferecem (IBM, 2021) .

Há algumas boas práticas que são importantes para se utilizar o diagrama de classes:

> 1- Os diagramas de classe tendem a ficar incoerentes à medida que crescem. É melhor evitar a criação de diagramas maiores e dividi-los em diagramas menores .
> 2- Usando a simples notação de classe, você pode criar rapidamente uma síntese de alto nível do seu sistema. Um diagrama detalhado pode ser criado separadamente, conforme necessário, e até mesmo ligado ao primeiro para fácil referência.

3- Quanto mais linhas se sobrepõem nos seus diagramas de classes, mais desordenada se torna. O leitor só vai ficar confuso ao tentar encontrar o caminho. Certifique-se que não há duas linhas que se cruzam.
4- Use cores para agrupar módulos comuns. Cores diferentes em classes diferentes ajudam o leitor a diferenciar entre os vários grupos (CREATELY, 2022)

2.3.7 Diagrama de atividades

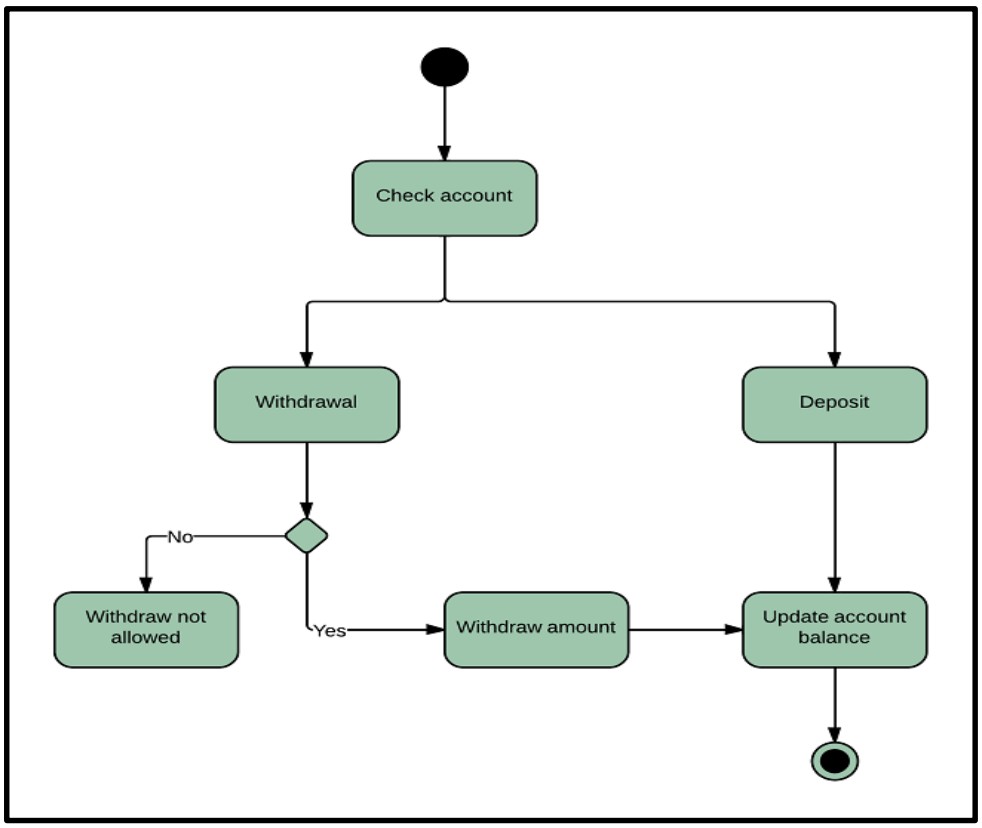
Os diagramas de atividades são uma ferramenta de modelagem de software que tem como objetivo a visualização, especificação e documentação do comportamento dinâmico de um sistema. Eles são uma forma de representar o fluxo de controles e de dados de um sistema.

Eles oferecem uma série de benefícios para os usuários, dentre eles demonstrar a lógica de um algoritmo, ilustrar um processo de negócio ou fluxo de trabalho entre usuários e o sistema, simplificar e melhorar qualquer processo ao esclarecer casos de uso complicados, entre outros.

Jackbon (2019), afirma que os diagramas de atividade são uma ferramenta valiosa para a comunicação e a compreensão do comportamento de um sistema. Eles permitem que os stakeholders visualizem como o sistema funciona e como os dados são processados.

A Figura 3 mostra um exemplo de um diagrama de atividades para um sistema bancário.

Figura 3: Exemplo de um diagrama de atividades para um sistema bancário



Fonte: LUCIDCHART (2023).

2.3.8 Diagrama de componentes

Um diagrama de componentes é um diagrama UML que representa os componentes de um sistema, subsistema ou mesmo componentes ou classes internas de um componente individual. Seu principal objetivo é mostrar uma visão geral do sistema, como ele está estruturado e como os componentes interagem entre si.

Um componente pode ser qualquer unidade de software que possa ser empacotada e reutilizada, como uma classe, um módulo, uma biblioteca ou um artefato de implantação. Os componentes são representados em diagramas de componentes por retângulos, com o nome do componente no interior.

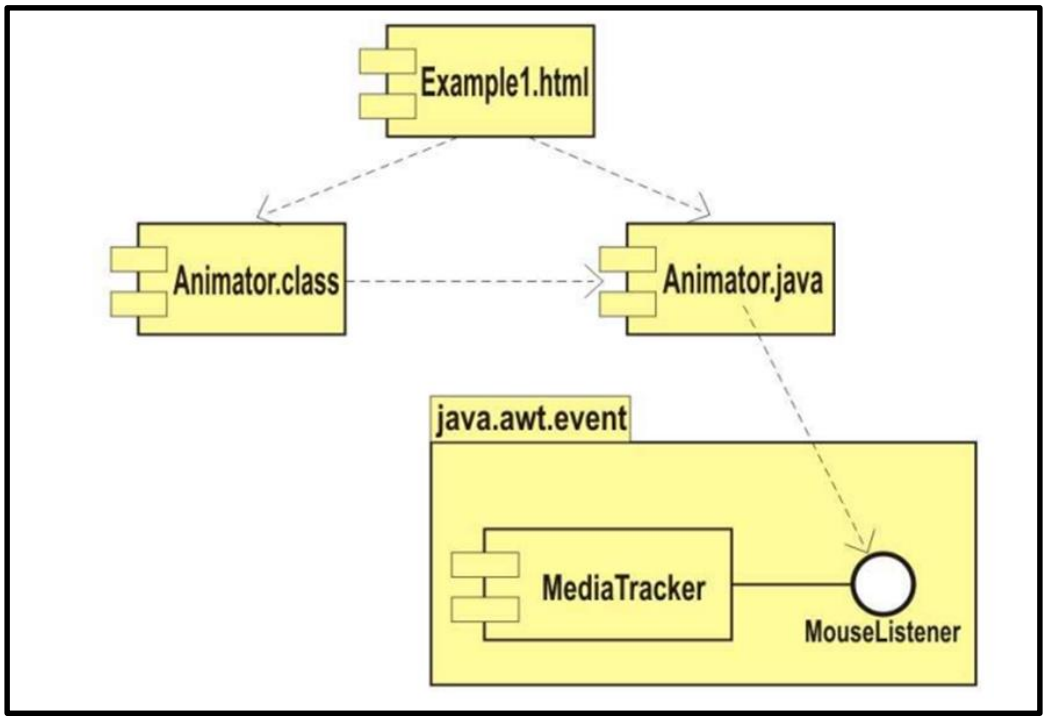
As relações entre componentes podem ser representadas por ligações, que indicam como os componentes interagem entre si. As ligações podem ser de vários tipos, como dependência, associação, agregação ou composição.

Os diagramas de componentes são úteis para uma variedade de propósitos, incluindo:

- Visualização da arquitetura de um sistema: Os diagramas de componentes fornecem uma visão geral do sistema, como ele está estruturado e como os componentes interagem entre si.
- Planejamento da implementação: Os diagramas de componentes podem ser usados para planejar a implementação do sistema, identificando os componentes que precisam ser criados ou reutilizados.
- Comunicação com clientes e stakeholders: Os diagramas de componentes podem ser usados para comunicar a arquitetura do sistema a clientes e stakeholders.

A Figura 4 mostra um exemplo de diagrama de componentes.

Figura 4: Exemplo de diagrama de componentes



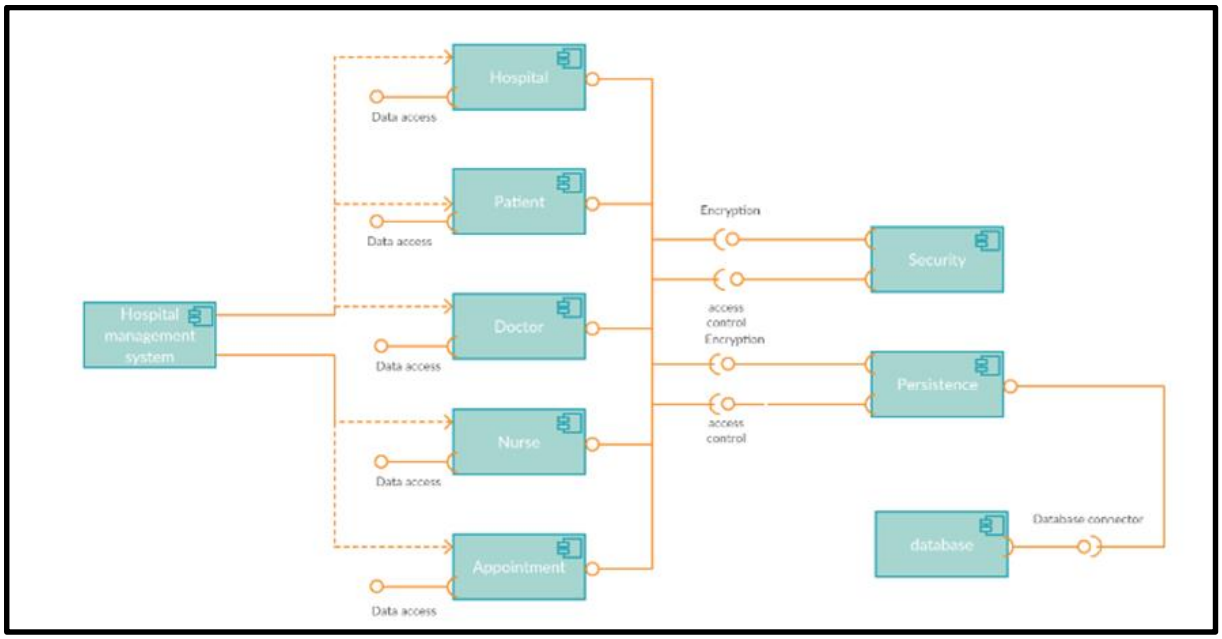
Fonte: RAMOS (2013).

Aqui estão alguns exemplos de diagramas de componentes:

- Diagrama de componentes de um sistema de gerenciamento de banco de dados: O diagrama pode mostrar os componentes do sistema, como o componente de banco de dados, o componente de interface do usuário e o componente de processamento de negócios.
- Diagrama de componentes de um sistema web: O diagrama pode mostrar os componentes do sistema, como o componente do servidor web, o componente do banco de dados e o componente do aplicativo web.
- Diagrama de componentes de um sistema de software embarcado: O diagrama pode mostrar os componentes do sistema, como o componente do hardware, o componente do firmware e o componente do software do sistema operacional.

A Figura 5 mostra um exemplo do Diagrama de Componentes para Sistema de Gestão Hospitalar.

Figura 5: Exemplo Sistema de Gestão Hospitalar



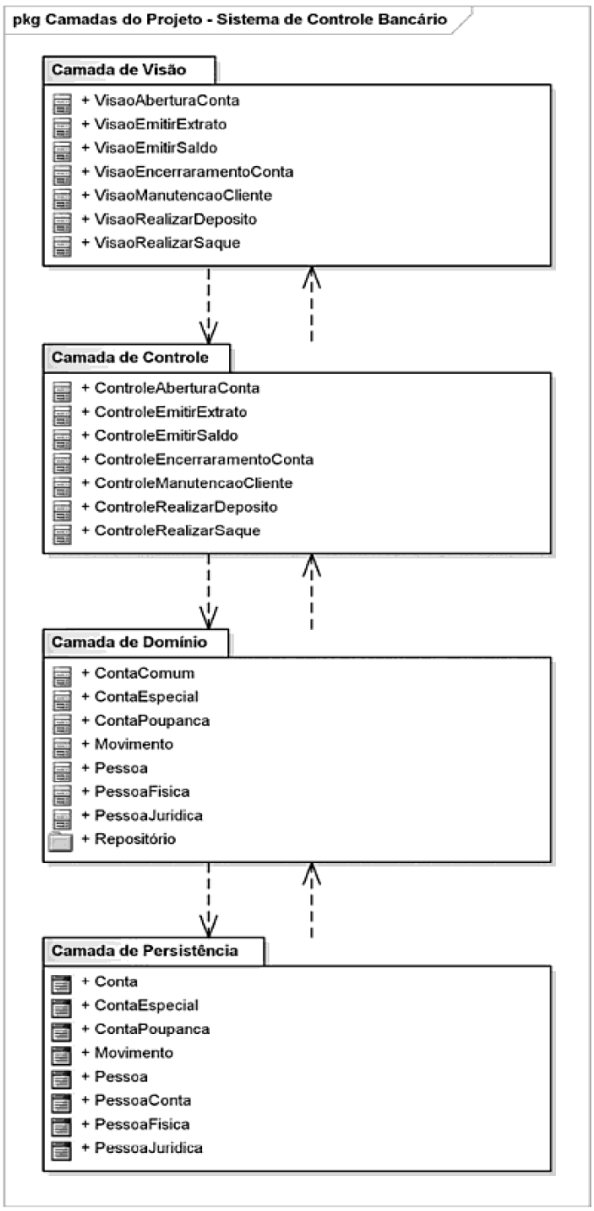
Fonte: CREATELY (2022).

2.4 Conceito do diagrama de pacotes e suas vantagens e desvantagens

O diagrama de pacotes é uma forma de estrutura que por meio de elementos representam segmentos que os próprios podem conter subsegmentos que podem continuar em diante em camadas ou não ter mais nenhum em diante, em um sistema sendo uma forma de “pacotes”, assim, os elementos podendo ter um ou mais segmentos, permitindo uma melhor forma de descrever os segmentos em múltiplos níveis diferentes, tendo a ideia que qualquer elemento pode ser ligados em um pacote, podendo os pacotes serem ligados a outros pacotes criando um diagrama que podem funcionar de forma conectados com outros diagramas ou de forma autônoma (GUEDES, 2018).

A Figura 6 demonstra um exemplo de diagrama de pacotes.

A Figura 6 mostra um exemplo de diagrama de pacotes.



Fonte: GUEDES (2018).

O diagrama de pacotes na arquitetura de projeto é a base de para a facilidade na manutenção de um sistema, onde os pacotes ajudam na separação das lógicas técnicas e de

aplicação, logo, a separação torna a mudança de segmentos mais fácil onde a sua alteração não irá atingir de forma negativa os demais segmentos (REZENDE, 2006).

Portanto, o diagrama de pacotes contribui na visualização, construção e como também ajuda na prática de documentação do projeto, tal como igualmente segundo Engiel (2011 p.6), “A base de representação usa um conjunto de construtos de modelagem e uma notação comum”, ademais, colabora na compreensão de sistemas e seus fluxos complexos, como também na tratar das suas funcionalidades de uma forma mais visível, entretanto, segundo Engiel (2011 p.6), como desvantagem, este diagrama “Pode não representar os atores, não representa os artefatos de entrada e saída das atividades”, assim, mesmo demonstrando de uma excelente forma, ele apresenta a incapacidade de demonstrar os atores e o início e fim das funcionalidades de forma mais expressiva.

3 METODOLOGIA

Para desenvolver este artigo foram utilizados periódicos, localizados pelo Google, regras que regem os diagramas UML e livros, sendo baseados em autores reconhecidos na área, assim possibilitando o levantamento bibliográfico dos dados,caracterizando-se uma pesquisa bibliográfica e exploratória.

As palavras chaves usadas foram: Diagramas UML, Diagrama de pacotes,engenharia de software,Diagrama de classes,História da UML, Diagrama de atividades, entre outros. O período da pesquisa ocorreu entre o mês de setembro e o mês de outubro de 2023.

A pesquisa se caracteriza por uma pesquisa bibliográfica que busca ser fundamentada em documentação e levantamento bibliográfico, colocando o pesquisador com o que já produziu a respeito do seu tema de pesquisa. A pesquisa também é exploratória, quando a pesquisa se encontra na fase preliminar, tem como finalidade proporcionar mais informações sobre o assunto que vamos pesquisar, possibilitando sua definição e seu delineamento (PRODANOV e FREITAS, 2013).

Em relação aos resultados , o trabalho caracteriza-se como pesquisa qualitativa. Menciona que a pesquisa qualitativa é subjetiva ao objeto de estudo, eleva-se sobre a dinâmica

e abordagem do problema pesquisado e visa descrever e entender de forma interpretativa os componentes de um sistema complexo (GIL, 1999).

4 ANÁLISES E DISCUSSÕES

Perante o que foi analisado, é um fato que os diagramas UML são uma forma base na utilização de organização de produção na engenharia de software, onde a produção em seus softwares a utilização dos diagramas traz uma grande vantagem na visualização dos requisitos, e planejamento e desenvolvimento do projeto.

Portanto, o diagrama de pacotes dentre os outros diagramas se diferencia por ser um recurso especial na organização pela sua forma de criação de modelos, simplificando o desenvolvimento de sistemas grandes ou complexos, de um meio onde a leitura de suas funções podem entendidas de forma mais fácil e lógica, como também o seu desenvolvimento por mostrar de forma mais simplória as funções, diminuindo as chances de erros e falhas no projeto, tanto pela falta de lógica como por erros acidentais, ou seja, criando um projeto mais coerente e com mais qualidade de software. Já em relação às suas desvantagens, a mais significativa se refere ao fato de não apresentar os atores e os artefatos de entrada e saída das atividades do sistema.

Diante do que foi posto em discussão, nota-se que o diagrama de pacotes fornece uma visão geral e diferenciado do projeto, onde o sistema em criação ou manutenção pode ser gerido de forma mais fácil pela eficiência de entendimento das funções e subfunções do sistema em questão, assim, acrescentando tanto qualidade de software quanto estabilidade no desenvolvimento do projeto, logo, atingindo prazos de forma mais assertiva e atendendo o cliente de forma mais satisfatória.

5 CONSIDERAÇÕES FINAIS

Diante do que foi proposto, dentre os diversos tipos de diagramas UML propostos, o diagrama de pacotes é uma ferramenta que apresenta nas suas grandes vantagens, onde se pode realizar projetamento, visibilidade e manutenção das grandes ou pequenas funções e

subfunções de projetos de softwares grandes ou pequenos sem causar danos ao projetos e seu desenvolvimento, mas, visando principalmente os projetos de softwares mais complexos, entretanto, das suas desvantagens são de não apresentar os atores e funções as suas saídas e entradas.

Portanto esse tipo de diagrama possibilita reduzir a incidência de erros pela sua sua forma de visualização e gerenciamento, assim, aumentando a qualidade durante o desenvolvimento e do produto final, por apresentar as funções e suas subfunções de forma mais lógica e compreensível, logo, trazendo a entrega de prazos mais assertivos.

REFERÊNCIAS

BEZERRA. EDUARDO. **Princípios De Análise E Projeto de Sistemas UML** 3 ed. Rio de Janeiro: Ltda, 1972 Disponível em: <https://www.academia.edu/35326870/Principios_De_Analise_E_Projeto_Eduardo_Bezerra_pdf> Acesso em: 05 out. 2023.

CAMPOS. JOSÉ; RIBEIRO. ANTONIO**.Desenvolvimento de Sistemas de Software.** Disponível em: <http://ideafix.di.uminho.pt/disciplinas/dss/0708/AulaT_20080108.pdf>. Acesso em: 05 out. 2023

CREATELY. **O Guia Fácil de Diagramas de Componentes.** Disponível em: <https://creately.com/blog/pt/diagrama/tutorial-de-diagrama-de-componentes-2/>. Acesso em: 03 out. 2023>.

DEVMEDIA. **Qualidade de software e Selenium IDE em testes Funcionais.** Disponível em: <https://www.devmedia.com.br/qualidade-de-software-e-selenium-ide-em-testes-funcionais/33225>. Acesso em: 03 Out. 2023.

DEVMEDIA. **Processo, Qualidade e Métricas de Desenvolvimento de Software.** Disponível em: <https://www.devmedia.com.br/processo-qualidade-e-metricas-de-desenvolvimento-de-software/26917>. Acesso em: 05 out. 2023.

DEVMEDIA. **Introdução a UML.** 2023. Disponível em: <https://www.devmedia.com.br/introducao-a-uml/6928>Acesso em: 05 out. 2023.

DEVMEDIA. **Diagrama de casos de uso.** 2023. Disponível em: <https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408> 2.3.2. Acesso em: 06 out. 2023.

ENGIEL, PRISCILA. **Apresentação de modelos de processos públicos – uma discussão sobre abordagens de representação de processos.** 2011. Disponível em: <https://seer.unirio.br/monografiasppgi/article/view/1778>. Acesso em: 06 out. 2023.

GUDWIN. RICARDO. **Introdução à Linguagem UML**. Disponível em: <https://www.dca.fee.unicamp.br/~gudwin/ftp/ea976/Estruturais2010.pdf> Acesso em: 05 out. 2023.

GILLEANES.GUEDES. **UML 2 - Uma Abordagem Prática** 3 ed. São Paulo, 2013. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=mJxMDwAAQBAJ&oi=fnd&pg=PA10&dq=diagrama+uml&ots=x9AVOhWNm3&sig=3GuXJ2US2RGx7KXxiw0gFKWQ9-w#v=onepage&q&f=false>. Acesso em: 05 out. 2023.

PRESSMAN. ROGER. **Engenharia de software.** 3 ed. São Paulo: LTDA, 1995.. Disponível em: <https://engenhariasoftwareisutic.files.wordpress.com/2016/04/engenharia-software-pressman.pdf> Acesso em: 06 out. 2023.

PROJETODIARIO. **História da uml.** 2023. Disponível em: <https://www.projetodiario.net.br/historia-da-uml/> Acesso em: 08 out. 2023.

RAMOS, RICARDO. **UML – Aula III Diagramas de Estado, Atividades, Componentes e Instalação.** Disponível em: <http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2013_1/UML_Aula3.pdf>. Acesso em: 04 out. 2023

REZENDE, DENIS. **Engenharia de Software e Sistemas de Informação** 3 ed. Rio de Janeiro: Brasport, 2005. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=rtBvl_L-

1mcC&oi=fnd&pg=PT23&dq=sommerville+engenharia+de+software&ots=9Bjo_H_oZo&sig=oL4yU03ur5w0C1auAVlD9s50r0A#v=onepage&q=diagrama%20de%20pacotes&f=false > Acesso em: 04 out. 2023