INTERFACE TECNOLÓGICA

ARQUITETURA ORIENTADA A MODELO: uma abordagem para o desenvolvimento de software

MODEL-DRIVEN ARCHITECTURE: an approach to software development

Larissa Bernardino – bernardinolarissaa@gmail.com
Faculdade de Tecnologia de Taquaritinga (Fatec) – Taquaritinga – São Paulo – Brasil

Daniela Gibertoni – daniela.gibertoni@fatectq.edu.br
Faculdade de Tecnologia de Taquaritinga (Fatec) – Taquaritinga – São Paulo – Brasil

DOI: 10.31510/infa.v21i1.1968
Data de submissão: 15/04/2024
Data do aceite: 10/03/2024
Data da publicação: 20/06/2024

RESUMO

Com a constante evolução tecnológica, a demanda por *softwares* de alta qualidade e complexidade vem crescendo a cada dia. As metodologias desempenham um papel importante na Engenharia de *Software*, é através delas que todo o sistema será construído. Este artigo apresenta a *Model Driven Architecture* (MDA), uma metodologia criada pela *Object Management Group* em 2001, que tem a proposta de atender todas as necessidades dos desenvolvedores. Ao longo do artigo, são destacados os benefícios da MDA, mostrando os exemplos de casos de uso de empresas que obtiveram sucesso ao aplicar essa metodologia. Além disso, são discutidos os desafios enfrentados na implementação dessa metodologia, juntamente com uma comparação entre a MDA e o Scrum, outra metodologia popular de desenvolvimento de *software*. O artigo oferece uma visão do que é a *Model Driven Architecture* e como ela pode ser promissora no mercado da Engenharia de *Software*.

Palavras-chave: *Model Driven Architecture*. Desenvolvedor. *Software*. Metodologia

ABSTRACT

With constant technological evolution, the demand for complex, high-quality software grows every day. Methodologies play an important role in Software Engineering, it is through them that the entire system will be built. This article presents Model Driven Architecture (MDA), a methodology created by the Object Management Group in 2001, which aims to meet all the needs of developers. Throughout the article, the benefits of MDA are highlighted, showing examples of use cases from companies that have been successful in applying this methodology. Furthermore, the challenges faced in implementing this methodology are discussed, along with a comparison between MDA and Scrum, another popular software development methodology. The article offers an insight into what Model Driven Architecture is and how it can be promising in the Software Engineering market

Keywords: Model Driven Architecture. Developer. Software. Methodology

1 INTRODUÇÃO

Nos tempos atuais com a grande evolução tecnológica estamos vendo uma crescente e alta demanda de desenvolvimento de *software* isso porque eles estão sendo cada vez mais presentes no mundo tecnológico que vivemos. Eles estão tomando um grande espaço dentro de grandes e pequenas empresas, seja para automatizar tarefas ou otimizar processos. Nisso entra a Engenharia de *Software*, ao longo dos anos ela tem se tornado parte essencial para o desenvolvimento de um *software*, isso porque ela se utiliza de metodologias, práticas e ferramentas que ajudam no momento do seu desenvolvimento.

Para a Engenharia de *Software* as metodologias são uma parte indispensável, porque essas metodologias definem as etapas, ferramentas e técnicas que serão utilizadas para garantir a eficiência e qualidade do *software*. Existem diversas metodologias que podem ser usadas no desenvolvimento de um *software*, como cascata, prototipação, espiral, incremental, além das metodologias tradicionais foram desenvolvidas metodologias ágeis como Kanban e *Scrum*, segundo a PMI (*Project Manager Institute)* (2017), afirma que 71% das empresas de engenharia de *software* utilizam metodologias ágeis, isso porque fornece maior flexibilidade e agilidade

Ao longo dos anos foram desenvolvidas várias outras metodologias para a engenharia de *software* entre elas está a arquitetura dirigida por modelos (*model-driven architecture*) que foi proposta pela *Object Management Group* em 2001.

Segundo Kleppe, o MDA é uma metodologia para desenvolvimento de *software*, é uma abordagem que se utiliza modelos como centro do processo de desenvolvimento, no MDA são criados modelos em níveis diferentes de abstração, que vão ser conectados para criar um sistema.

Porém a abordagem MDA não é muito utilizada, por conta da necessidade de ferramentas específicas para o seu uso e a curva de aprendizado.

O objetivo desse artigo é investigar o uso da abordagem MDA, mostrando como essa abordagem pode ser útil dentro do desenvolvimento de *software.*

Esse artigo fez o uso da pesquisa bibliográfica através de livros e artigos sobre o MDA e a Engenharia de *Software*.

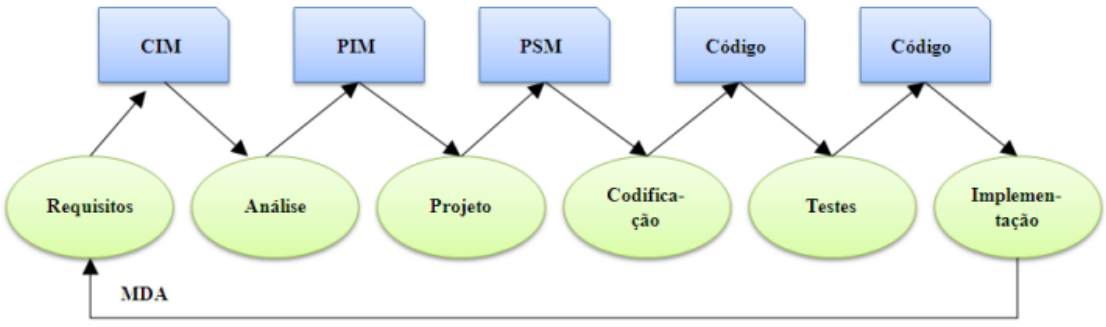
2 FUNDAMENTAÇÃO TEÓRICA

O MDA (*Model Driven Architecture*) é uma metodologia de desenvolvimento de *software* projetada para lidar com a complexidade de um sistema através do uso central de modelos, o MDA foi criado e implantado pela OMG (*Object Management Group*) em 2001.

A criação do MDA surgiu com a necessidade a eficiência do desenvolvimento de um *software*, no MDA os modelos são o centro para o desenvolvimento do *software*, ou seja, o MDA se concentra na utilização de modelos no ciclo de vida de desenvolvimento de um *software* permitindo a agilidade no planejamento do mesmo, também ajudando nas futuras manutenções e alterações que esse *software* poderá sofrer no futuro, o princípio do MDA é poder trabalhar em um nível de abstração alto, independente de tecnologias ou plataformas específicas, que vai transformar essa abstração em implementações concretas em diferentes plataformas que serão utilizadas. Na Figura 1, é possível observar o ciclo de vida do MDA, conforme representado no diagrama.

O MDA acontece em três passos principais, são eles o CIM (Modelo Independente de Computação), PIM (Modelo Independente de Plataforma) e PSM (Plataformas Específicas de Modelo),o MDA propõe que podemos fazer transformações automáticas de modelos, por exemplo pegar um modelo em PIM e fazer transformações em PSM e isso permite automatizar etapas de trabalho que eram feitas manualmente.

Figura 1 – Ciclo de vida do MDA



Fonte: (BORGES; SOUZA; SCHULZE; MURY, 2011, p 21.)

2.1 Empresas que utilizaram o MDA

O MDA conta com muitas histórias de sucessos, o site da OMG disponibiliza artigos sobre empresas que implementaram essa abordagem, entre elas podemos citar a CONQUEST,

INC., eles fornecem soluções de arquitetura empresarial para a Agência de Inteligência do Governo dos EUA. A Conquest e a Popkin utilizaram o MDA para separar o capital intelectual da implementação física, mantendo três níveis de modelos em um repositório comum. Isso permitiu o desenvolvimento e atualização independentes dos modelos, garantindo uma representação precisa da arquitetura empresarial. Além disso, automatizaram transformações entre os modelos, possibilitando uma transição eficiente ao longo do ciclo de vida do projeto, resultando em uma arquitetura robusta e escalável para a agência de inteligência. (OMG).

O Instituto Nacional do Câncer, a abordagem MDA foi empregada pelo Instituto Nacional do Câncer de várias maneiras para alcançar a interoperabilidade necessária e apoiar os pesquisadores em sua missão de encontrar a cura para o câncer até 2015. Eles utilizaram a abordagem MDA para padronizar, gerar e integrar automaticamente parte do código necessário para alcançar interoperabilidade entre os diversos projetos de pesquisa em câncer. Isso proporcionou uma base sólida para colaboração e avanço científico na busca pela cura do câncer. (OMG).

E a Codagen, a empresa optou pela abordagem MDA para separar a lógica de negócios dos sistemas, automatizar a geração de código e aumentar a produtividade. Isso envolveu a modelagem em um nível de abstração mais alto, permitindo a geração automática de código. Essa estratégia possibilitou enfrentar com sucesso os desafios de recriar os sistemas do cliente de forma eficiente. (OMG)

2.2Modelos MDA

A OMG considera 3 importantes modelos no MDA, o CIM, PIM e PSM, esses modelos permitem a criação de sistemas de *software* independentes de plataforma, alcançando maior flexibilidade e qualidade no desenvolvimento.

O CIM (*Computation Independent* Model) também é frequentemente chamado de modelo de negócio ou de domínio porque usa um vocabulário que é familiar aos especialistas no assunto. Ele apresenta exatamente o que se espera que o sistema faça, mas esconde toda a tecnologia da informação especificações relacionadas permaneçam independentes de como esse sistema será implementado.

O CIM desempenha um papel importante na colmatação da lacuna que normalmente existe entre esses especialistas do domínio e os tecnólogos da informação responsáveis para

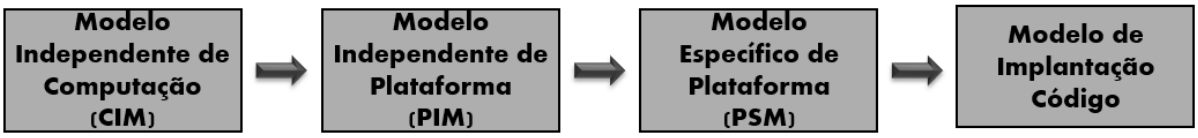
implementação do sistema (OMG, 2003), ou seja, o CIM está mais próximo e preocupado com os problemas do usuário.

O PIM (*Platform Independent Model*) apresenta um grau de independência suficiente para permitir seu mapeamento para uma ou mais plataformas. Isso geralmente é conseguido definindo um conjunto de serviços de uma forma que abstraia detalhes técnicos. Outros modelos então especificam uma realização desses serviços de uma maneira específica da plataforma (OMG, 2003), ou seja, o PIM está pensando no sistema que vai implementar, os modelos são que são independentes de plataformas vai se mapeando para plataformas específicas.

O PSM (*Platform Specific Model*) combina as especificações do PIM com os detalhes necessários para estipular como um sistema usa um determinado tipo de plataforma. Se o PSM não incluir todos os detalhes necessários para produzir uma implementação dessa plataforma é considerada abstrata, o que significa que depende de outras informações explícitas ou modelos implícitos que contêm os detalhes necessários (OMG, 2003), ou seja, o PSM está preocupado com plataformas específicas, por exemplo por qual plataforma onde ele está sendo desenvolvido e qual a linguagem que está sendo utilizada.

Na Figura 2, é possível ver os níveis de abstração do MDA mostrando como o Modelo Independente de Computação se situa no início do processo de desenvolvimento, passando para o Modelo Independente de Plataforma, gerando automaticamente o Modelo Específico de Plataforma que resulta na Implantação do Código.

Figura 2 – Níveis de abstração do MDA



Fonte: (Kardos; Drozdová, 2010.)

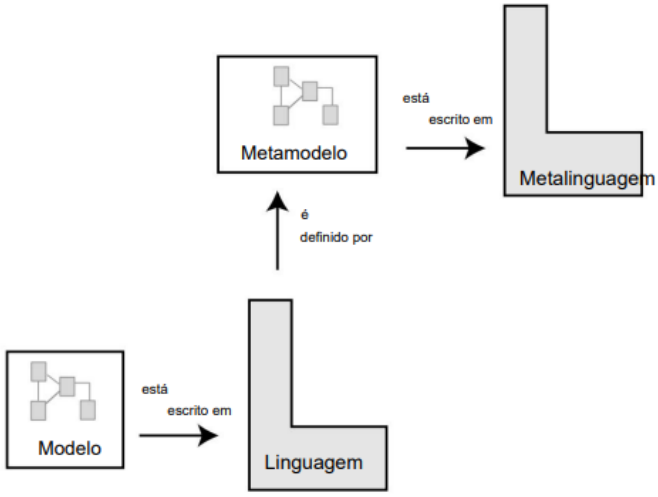
2.2.1 Metamodelos

A metamodelagem é muito importante no MDA, porque ela define as linguagens no contexto do MDA, os idiomas na metamodelagem são definidos através de metamodelos, o metamodelo define os conceitos, regras e restrições da linguagem.

Segundo Kleppe (2003), o metamodelo também deve ser escrita em uma linguagem bem definida, essa linguagem chama metalinguagem, uma metalinguagem desempenha um papel diferente de uma linguagem de modelagem na estrutura MDA, porque é uma linguagem

especializada para descrever linguagens de modelagem. Portanto, usamos um símbolo diferente para uma metalinguagem na estrutura MDA. Em segundo lugar, o metamodelo define completamente a linguagem. Portanto, não é necessário nem útil fazer a distinção entre a linguagem e o metamodelo que define a linguagem; para todos os efeitos práticos, eles são equivalentes. Como uma metalinguagem é ela própria uma linguagem, ela pode ser definida por um metamodelo escrito em outra metalinguagem. Em teoria, existe um número infinito de camadas de relações modelo-linguagem-metalinguagem.

Figura 3- Relação entre modelo e metamodelo



Fonte: (KLEPPE, 2003, p 84.)

Na Figura 3, conforme apresentado por Kleppe (2003), é possível ter uma compreensão visual da relação entre modelo e metamodelo,essa relação ilustra como os modelos são definidos por metamodelos.

2.3 Benefícios da implementação MDA

O MDA pode apresentar inúmeros benefícios em sua implantação, isso porque, como visto anteriormente ele otimiza processos e oferece uma abordagem sistemática e baseada em modelos para o desenvolvimento de *software*, que pode melhorar a qualidade, a produtividade e a adaptabilidade dos sistemas de *software*.

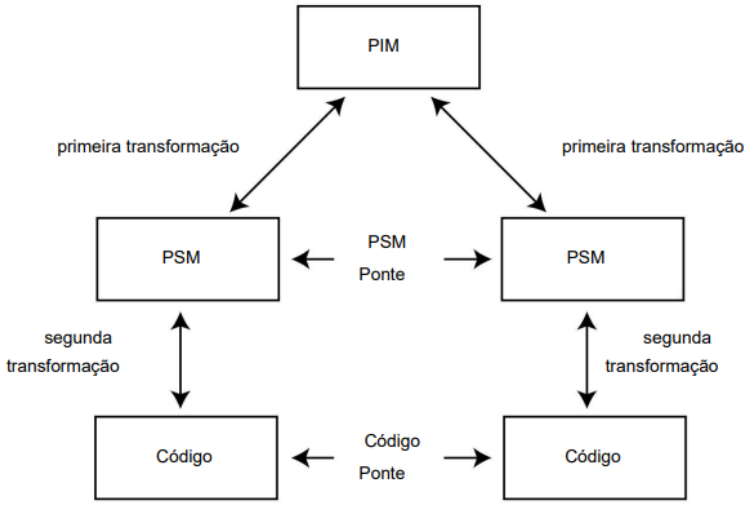
Segundo Kleppe (2003) os benefícios de sua implementação são:

a) **Produtividade:** Os desenvolvedores podem focar mais no PIM, ou seja, conseguem focar mais no sistema e nas melhorias e implantações que precisam ser realizadas, sem depender ou se preocupar com plataformas ou detalhes técnicos, já que esses detalhes serão especificados automaticamente na transformação do PIM para o PSM, isso dá menos trabalho aos desenvolvedores e ajuda na produtividade e fornece um sistema que se adapta as necessidades dos usuários e tem melhores funcionalidades em menor tempo.

b) **Portabilidade:** A portabilidade só pode ser alcançada se for focado no PIM, o que for descrito dentro de um nível PIM é portátil, então podem ser transformados em diferentes PSMs para diferentes plataformas, o que vai determinar a portabilidade é a disponibilidade e eficácia das ferramentas de transformação automatizadas. Para plataformas populares, muitas ferramentas estarão disponíveis. Para plataformas menos populares, talvez seja necessário usar uma ferramenta que suporte definições de transformação de plug-in e escrever você mesmo a definição de transformação. Para novas tecnologias e plataformas que chegarão no futuro, a indústria de software precisa de entregar as transformações correspondentes a tempo. Isso nos permite implantar rapidamente novos sistemas com a nova tecnologia, com base em nossos PIMs antigos e existentes. (KLEPPE, 2003)

c) **Interoperabilidade:** No MDA os modelos PSM transformados a partir do PIM podem se relacionar, esses relacionamentos são conhecidos como “pontes”, quando um PSM é destinado a diferentes plataformas isso impede a comunicação entre eles, isso exige a interoperabilidade, no MDA além dos PSMs gerados, é gerado as pontes necessárias para essa comunicação, permitindo a comunicação e integração entre esses PSMs como pode ser observado na Figura 4, onde é possível visualizar as pontes geradas para permitir a comunicação e integração entre os diferentes PSMs alcançando a interoperabilidade.

Figura 4- Interoperabilidade no MDA



Fonte: (KLEPPE, 2003, p 10.)

d) Manutenção e Documentação: No ciclo de vida do MDA os desenvolvedores focam no PIM que o nível mais alto de abstração, através do PIM que é gerado o PSM e por fim o código, portanto o PIM desempenha o papel de documentação do sistema, ele é mantido e atualizado mesmo após a criação do sistema, qualquer alteração feita no sistema altera o PIM e regenera o PSM e o código, apesar de que na prática, as alterações são feitas direto no PSM, existem ferramentas que conseguem manter o PIM e o PSM ligados, as alterações no PSM vão ser refletidas no PIM e a documentação vai continuar igual ao código real, o MDA facilita a disponibilidade de documentação de alto nível, mas ainda pode existir a necessidade de anotações adicionais para informações específicas.

2.5 Como a UML se encaixa no MDA

Apesar do MDA poder realizar o uso de outras linguagens de modelagem, a UML (*Unified Modeling Language)* é o padrão de linguagem de modelagem que o MDA segue, no MDA é importante que o modelo seja bem escrito.

A UML é uma ferramenta para criar modelos de sistemas, com a UML pode-se criar diagramas de classe, artefatos, interfaces, componentes, entre outros, e isso ajuda na estruturação de projetos de *software*, a UML ajuda na representação visual do sistema com seus principais componentes, assim ajudando a documentar o sistema.

A UML é usada no MDA na criação de um modelo PIM, com isso esse modelo será gerado automaticamente para o PSM, ou seja, irá conter detalhes específicos sobre sua implementação, como por exemplo, qual plataforma ele será direcionado como JAVA, .NET, CCM, SOAP.

A UML é utilizada no MDA porque permite aos desenvolvedores se concentrarem nos modelos de alto nível sem ter que se preocupar como será implementado, esses modelos também podem ser reutilizados em diferentes sistemas e qualquer mudança que seja realizada no sistema pode ser refletida automaticamente nos modelos e transformadas em código.

3 PROCEDIMENTOS METODOLÓGICOS

Esta pesquisa é pautada em levantamento bibliográfico, através de livros e artigos. A bibliografia base a ser utilizada é o livro "*MDA Explained: The Model Driven Architecture: Practice and Promise*." KLEPPE (2003), o site da *Standards Development Organization- OMG* que foram os criadores do método MDA e o livro "Engenharia de Software" SOMMERVILLE (2011). Foram selecionadas fontes com base em sua relevância. O livro Engenharia de Software de SOMMERVILLE é conhecido e usado para o estudo em Engenharia de Software e o portal da OMG, os pioneiros dessa abordagem, além dos artigos que continham os indicadores conceitos básicos, visão do processo de desenvolvimento, tecnologias envolvidas. Nessa pesquisa serão reunidos dados sobre o MDA, abordando a sua história, apontando seus benefícios e características, em vista de informar o leitor sobre o tema.

4 RESULTADOS E DISCUSSÃO

A pesquisa realizada sobre o MDA e sua aplicação na Engenharia de Software revela como essa abordagem pode ser útil e significativa na criação e desenvolvimentos de *software*. Os resultados obtidos na pesquisa mostram os casos de uso de instituições que implantaram o MDA e seu sucesso na aplicação por diferentes organizaçõescomo agências governamentais, instituições de pesquisa e empresas de consultoria, isso porque como dito anteriormente o MDA proporciona lidar com a complexidade dos sistemas de *software* e promover uma maior eficiência no desenvolvimento.

Comparando o MDA com outras metodologias, como por exemplo o Scrum, tanto o MDA quanto o Scrum oferecem abordagens diferentes para o desenvolvimento de *software*. O MDA é uma metodologia que utiliza um alto nível de abstração, o que é bom para criação de sistemas com uma complexidade muito alta,o MDA cria modelos que são mais fáceis de compreenderos requisitos e reutilizá-los se for necessário. Mas tem dificuldades na sua abordagem como a complexidade no começo da criação e manutenção dos modelos e a curva de aprendizado para a equipe que desejar utilizar essa metodologia. Já o Scrum oferece flexibilidade e entrega rápida de valor através de iterações curtas e priorização da lista de tarefas (*backlog*). Mas assim como o MDA, o Scrum também tem algumas dificuldades como o tempo para a entrega das tarefas e a pressão para que a equipe tenha tempo e boa comunicação.

A escolha da metodologia vai depender totalmente das necessidades da equipe, do ambiente e do projeto que trabalham.

Mas apesar dos benefícios, alguns desafios nessa abordagem são identificados, como a necessidade de ferramentas adequadas de transformação de modelos e a manutenção da sincronia entre os modelos de nível PIM e PSM, essas ferramentas precisam ser compatíveis com os padrões do MDA, a curva de aprendizado para a transição de qualquer outra metodologia para o MDA vai exigir que os desenvolvedores aprendam os conceitos e técnicas básicas em relação a modelagem e a transformação de modelos, também a criação para os modelos iniciais requer um investimento significativo de tempo e recursos, a manutenção e a atualização desse modelos também deve ser levada em conta e é importante que para a implementação do MDA ser bem sucedida os padrões definidos pela OMG (*Obejct Management Group*) devem ser seguidos.

5 CONCLUSÃO

Diante da pesquisa é possível concluir que, no mundo atual com a complexidade e alta demanda de desenvolvimentos de *softwar*ea importância de se ter uma metodologia que traga vantagens e melhorias para esse desenvolvimento é cada vez mais significativa. A abordagem MDA apesar de trazer grandes desafios, mas também traz também grandes benefícios, como produtividade e facilidade na realização desses *softwares.*

Os exemplos de casos de uso apresentados mostraram grande potencial para abordagem MDA de lidar com sistemas complexos e mostrando uma maior eficiência no desenvolvimento permitindo assim uma adaptação facilitada a mudanças que podem ocorrer no futuro, isso

porque quando concentrado o foco em modelos de alto nível que independem de plataforma especificas isso resulta em software com mais versatilidade, com qualidade e que podem ser desenvolvidos em menor tempo.

O MDA oferece uma abordagem promissora para o desenvolvimento de software, proporcionando uma estrutura baseada em modelos que pode melhorar a qualidade, a produtividade e a adaptabilidade dos sistemas de software. Se seus padrões e técnicas forem bem compreendidos a sua implementação no mercado de Engenharia de Software pode se tornar muito promissor.

REFERÊNCIAS

BORGES H. P.; SOUZA J. N.; SCHULZE B; MURY A. R. **UMA ARQUITETURA BASEADA EM MODELOS** - MDA. UMA ARQUITETURA BASEADA EM MODELOS - MDA, 2011. Disponível em: <https://livroaberto.ibict.br/bitstream/1/866/2/Uma%20arquitetura%20baseada%20em%20modelos%20-%20MDA.pdf. >Acesso em: 18 jan. 2024.

JUNIOR P. J. **MDA**:Uma visão geral. MDA:Uma visão geral, 2 fev. 2017. Disponível em: <https://tecnopode.blogspot.com/2017/02/mda-visao-geral-parte-i.html?m=0> Acesso em: 27 mar. 2024.

KLEPPE G. A.; WARMER B. J.; BAST W**. MDA Explained:** The Model Driven Architecture: Practice and Promise. 1º ed. EUA. Addison Wesley. 2003

SANTANDER**. Metodologias de desenvolvimento de software:** o que são?.Metodologias de desenvolvimento de software: o que são?, 12 abr. 2023. Disponível em: <https://www.santanderopenacademy.com/pt_br/blog/metodologias-de-desenvolvimento-de-software.html>. Acesso em: 05 mar. 2024.

SANTOS G. M. **Arquitetura orientada a modelos (MDA) e sua aplicação no processo de desenvolvimento de *software*.** Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 06, Ed. 07, Vol. 12, pp. 121-131. Julho de 2021. Disponível em:https://www.nucleodoconhecimento.com.br/ciencia-da-computacao/aplicacao-no-processo> Acesso em: 20 mar. 2024.

STANDARS DEVELOPMENT ORGANIZATION. **MDA** – A arquitetura preferida para um mundo em mudança. OMG Standards Development Organization. EUA. Disponível em:<https://www.omg.org/mda/products_success.htm**>.** Acesso em: 05 abr. 2024.

STANDARS DEVELOPMENT ORGANIZATION. **MDA** – A arquitetura preferida para um mundo em mudança. OMG Standards Development Organization. EUA. Disponível em: <https://www.omg.org/mda/faq_mda.htm>. Acesso em 13 nov. 2023.

SOMMERVILLE I. **Engenharia de Software**. 9ª ed. São Paulo. Person Education. 2011