UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTAÇÃO E SISTEMAS

MARINA DE OLIVEIRA MARGARIDA

AVALIAÇÃO DA APLICABILIDADE DA UML COMO UMA ADL DE SOFTWARE

João Monlevade
2018

MARINA DE OLIVEIRA MARGARIDA

AVALIAÇÃO DA APLICABILIDADE DA UML COMO UMA ADL DE *SOFTWARE*

Monografia apresentada ao curso Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Orientador: Diego Zuquim Guimarães Garcia

João Monlevade

2018

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

ATA DE DEFESA

Aos 11 dias do mês de julho de 2018, às 12 horas e 00 minutos, na sala C203 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pela aluna **Marina de Oliveira Margarida,** sendo a Comissão Examinadora constituída pelos professores: Prof. Dr. Diego Zuquim Guimarães Garcia, Profa. Ma. Daniela Rodrigues Dias e Prof. Dr. Euler Horta Marinho.

A candidata apresentou a monografia intitulada: *"AVALIAÇÃO DA APLICABILIDADE DA UML COMO UMA ADL DE SOFTWARE".* A comissão examinadora deliberou, por unanimidade, pela aprovação da candidata, com nota 9,0 (NOVE), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pela graduanda.

João Monlevade, 11 de julho de 2018.

Prof. Dr. Diego Zuquim Guimarães Garcia
Professor Orientador/Presidente

Profa. Ma. Daniela Rodrigues Dias
Professora Convidada

Prof. Dr. Euler Horta Marinho
Professor Convidado

Marina de Oliveira Margarida
Graduanda

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

Curso de Sistemas de Informação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

AVALIAÇÃO DA APLICABILIDADE DA UML COMO UMA ADL DE SOFTWARE

Marina de Oliveira Margarida

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

Prof. Dr. Diego Zuquim Guimarães Garcia
Departamento de Computação e Sistemas - UFOP

Profa. Ma. Daniela Rodrigues Dias
UFOP

Prof. Dr. Euler Horta Marinho
Departamento de Computação e Sistemas - UFOP

João Monlevade, 11 de julho de 2018

DEDICATÓRIA

À minha mãe, Maria de Lourdes, e ao meu irmão, Arthur. Obrigada.

AGRADECIMENTOS

Agradeço a Deus por acompanhar meus passos, amparar-me sempre que precisei de uma ajudinha lá de cima e nos dias em que as nuvens estão mais densas e cinza, me manda o Sol e toda a luz para iluminar minha mente, meus caminhos e minha vida inteira.

Agradeço aos meus pais, pelo suporte e incentivo aos meus estudos, em especial à minha mãe Maria de Lourdes pelo carinho e por passar-me tanta força para vencer os desafios. Agradeço ao meu irmão, Arthur, principalmente pelas risadas e brincadeiras.

Agradeço ao meu falecido avô e padrinho, José Benedito Pastor, que sempre incentivou-me a estudar e até os seus últimos dias de vida contagiou-me com seu otimismo e fé inabaláveis. Agradeço também à minha avó e madrinha Nélia, pelo amor e atenção que dá a mim e a todos os membros da família Pastor.

Agradeço aos amigos que fiz dentro e fora da Universidade nesses anos de estudo, pelas risadas, conselhos, auxílio e chocolates.

Agradeço ao meu Orientador, Diego Zuquim pelo apoio e conhecimento compartilhado.

Agradeço às pessoas que fazem parte da minha vida, que torcem por mim, e tornam o meu dia-a-dia mais doce.

Por fim, agradeço a tudo e todos que de alguma forma contribuíram e contribuem para o alcance dos meus objetivos!

“Se não puder voar, corra. Se não puder correr, ande. Se não puder andar, rasteje, mas continue em frente de qualquer jeito.”

(MARTIN LUTHER KING)

RESUMO

Diversos projetos têm utilizado o desenvolvimento baseado em arquitetura, pois este estilo apresenta muitos benefícios como controle e rastreabilidade dos requisitos funcionais e não funcionais, e das expectativas dos *stakeholders*. Nesse contexto pode-se observar que um importante elemento nesse desenvolvimento são as *Architecture Description Languages* (ADLs), ou Linguagens de Descrição Arquitetural, usadas para dar suporte ao armazenamento de descrições arquiteturais, estilos e requisitos do sistema em uma arquitetura. Apesar de todos os benefícios que proporcionam, seu uso não se popularizou. A *Unified Modeling Language* (UML), uma linguagem gráfica usada para visualizar, especificar, construir, documentar e comunicar artefatos de *software* ganhou ascensão tanto no meio acadêmico quanto industrial como uma ADL. O tema deste trabalho sugere a análise e o estudo da UML como uma ADL de *software* por meio da revisão bibliográfica, para a posterior avaliação dos benefícios e limitações da mesma no desempenho deste papel. Pôde-se concluir que a UML se tornou o padrão industrial para se documentar arquiteturas, por apresentar uma série de benefícios em relação às ADLs. Os benefícios e limitações do seu uso foram descobertos e pontuados nesta obra.

Palavras-chave: UML. ADL. Arquitetura de *software*. Descrição arquitetural.

ABSTRACT

Several projects have used architecture-based development because this style has many benefits such as control and traceability of functional and non-functional requirements and stakeholder expectations. In this context it can be observed that an important element in this development is Architecture Description Languages (ADLs), used to support the storage of architectural descriptions, styles and system requirements in an architecture. Although all the benefits that provide its use was not popularized. The Unified Modeling Language (UML), a graphical language used to visualize, specify, construct, document, and communicate software artifacts, has gained momentum both in academia and industry as an ADL. The subject of this work suggests the analysis and the study of the UML as a software ADL through a bibliographic review, for the later evaluation of the benefits and limitations of the same in the performance of this paper. It was concluded that the UML has become the industry standard for documenting architectures because it has a number of benefits in relation to ADLs. The benefits and limitations of its use have been discovered punctuated in this work.

Keywords: UML. ADL. Software architecture. Architectural description.

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AADL	–	*Architecture Analysis & Design Language*
ADL	–	*Architecture Description Language*
IEEE	–	*Institute of Electrical and Electronics Engineers*
OCL	–	*Object Constraint Language*
SysML	–	*Systems Modeling Language*
TI	–	Tecnologia da Informação
UML	–	*Unified Modeling Language*

Sumário

1 Introdução

Atualmente os *softwares* têm papel fundamental para os negócios, a ciência e a engenharia. É impossível pensar em uma sociedade modernizada sem a sua presença. Diversos setores, serviços e organizações fazem uso intensivo dele, com um amplo número de finalidades.

De acordo com Sommerville (2011) os sistemas de *software* são abstratos e intangíveis. Eles não são restringidos pelas propriedades dos materiais, nem governados pelas leis da física ou pelos processos de manufatura. Tal falta de restrições pode fazê-los se tornarem extremamente complexos, difíceis de administrar e modificar. Neste contexto tem-se a Engenharia de *Software*, uma disciplina de engenharia, que visa minimizar os problemas em relação ao ambiente do seu desenvolvimento, levando em consideração os fatores humanos, processos, planos de processos e ferramentas para que a construção de *software* que atendam os requisitos solicitados dentro dos prazos definidos e custos previsíveis, tornando-se essencial para estudar e dirigir o processo completo do seu desenvolvimento.

Uma das fases do desenvolvimento de *software* é o Projeto de Arquitetura, que se preocupa em como um sistema de *software* deve ser organizado e em como será a sua estrutura geral, visando identificar os principais componentes estruturais de um sistema, bem como a maneira que eles são interligados e a relação entre os mesmos, desempenhando um papel fundamental na gerência da complexidade do sistema a ser desenvolvido. A valorização dessa fase é de extrema importância, pois nesta etapa ocorre a descrição arquitetural, onde são criados conjuntos de artefatos que refletem uma visão do sistema. O elemento responsável por gerar essa representação são as chamadas *Architecture Description Languages* (ADLs), linguagens formais usadas para a realização da descrição arquitetural. Alguns exemplos dessas linguagens são a C2, Rapide e Wright, como mencionado em Nakagawa (2006).

A Unified Modeling Language (UML) tem sido amplamente aceita tanto nas universidades quanto nas indústrias como uma ADL de *software*, como dito em Avgerigou, Guelfi e Medvidovic (2005), e muitos trabalhos têm abordado o mapeamento dos conceitos de ADLs tradicionais em notações orientadas a objetos através de técnicas da UML por apresentar benefícios, como por exemplo, a

familiaridade dos desenvolvedores com a mesma, abordada em Garlan, Cheng e Kompanek (2002), no entanto, segundo os mesmos autores, algumas limitações acerca de sua aplicabilidade para a realização efetiva deste papel podem ser identificadas, demonstrando a necessidade de estudá-la e avaliar as vantagens e desvantagens do seu uso como uma ADL para facilitar na sua escolha, ou na escolha de outra ADL para modelar a arquitetura.

1.1 Problema

Diversos projetos têm utilizado o desenvolvimento baseado em arquitetura, pois este estilo apresenta muitos benefícios como controle e rastreabilidade dos requisitos funcionais e não funcionais, e das expectativas dos *stakeholders*. Nesse contexto pode-se observar que um importante elemento do desenvolvimento são as ADLs, usadas para dar suporte ao armazenamento de descrições arquiteturais, estilos e requisitos do sistema em uma arquitetura. A análise precoce do sistema, por meio da sua descrição arquitetural é fortemente recomendada, no entanto, de acordo com Babar, Zhur e Jeffery (2004), existe pouco consenso sobre uma definição universalmente aceita do que seria uma ADL.

Entende-se que uma ADL é uma linguagem utilizada para representar a arquitetura de um *software*, elemento, que preferencialmente deve ser representado sob diferentes visões, como a estrutural, comportamental, a de módulos, a de implantação, dados e implementação (NAKAGAWA, 2006), etc., que são escolhidas de acordo com o julgamento das pessoas envolvidas no projeto e com as necessidades do mesmo. A UML, uma linguagem gráfica usada para visualizar, especificar, construir, documentar e comunicar artefatos de *software* (BOCH; JACOBSON; RUMBAUGH, 2006), apesar de ter sido inicialmente criada para fornecer um *design* detalhado de projetos de *software*, se mostra potencialmente aplicável em outras etapas da produção do mesmo, como a realização da descrição arquitetural, no entanto, muitos estudos e artigos apontam limitações que a mesma possui. O objetivo do presente trabalho então é analisar a UML como uma ADL de *software*, levantando os benefícios e limitações da mesma no desempenho deste papel.

1.2 Objetivos

1.2.1 Objetivo geral

O presente trabalho tem como objetivo avaliar a aplicabilidade da UML como uma ADL de *software*.

1.2.2 Objetivos específicos

- Realizar a Revisão Bibliográfica sobre Arquitetura de *Software*, Modelagem e o Estado da Arte.
- Descobrir, compreender e avaliar as limitações e benefícios do uso da UML como uma ADL de *software.*
- Elaborar dois quadros comparativos que sintetizam as limitações e benefícios do uso da UML para realizar a descrição arquitetural.

1.3 Justificativa

A arquitetura de *software* é alvo de pesquisas desde o início dos anos 90 (OZKAYA; KLOUKINAS, 2013). Desde então, algumas ADLs surgiram e desapareceram, mas nenhuma se tornou popular, exceto poucas de domínio específico (PANDEY, 2010). Existem algumas queixas em relação às ADLs de *software*, por parte dos arquitetos, como notações complexas e pouco chamativas (PANDEY, 2010), que não permitem boa comunicação entre os *stakeholders* (MALAVOLTA et al., 2013). Em contrapartida, a UML tem sido amplamente utilizada na indústria para realizar a modelagem arquitetural de sistemas de *software* (MALAVOLTA et al., 2013; OZKAYA; KLOUKINAS, 2013). Isso se dá por ela ser uma linguagem extensível e flexível, suportada por várias ferramentas (LEVIN, 2009). Mesmo sendo a linguagem mais utilizada pela indústria para descrever arquiteturas, existem algumas limitações acerca da sua aplicabilidade como uma ADL, portanto, a avaliação das vantagens, desvantagens, pontos fortes e fraquezas do uso da mesma para este papel se mostra interessante e válida.

A grande importância e impacto que a valorização e realização da descrição arquitetural sobre as demais fases do desenvolvimento de *software* aliadas às barreiras encontradas pelos profissionais sobre o uso das ADLs, e ao

grande interesse existente em relação à UML foram os grandes motivadores para a escolha do tema, e para a confecção do presente trabalho. Com este trabalho, pretende-se então, descobrir-se as limitações e benefícios do uso da UML para a modelagem de Arquiteturas de *Software*.

1.4 Estrutura do trabalho

Neste primeiro capítulo foi realizada a introdução do tema, os objetivos do trabalho, a justificativa e a estruturação do presente trabalho.

No segundo capítulo é exposta a metodologia deste trabalho.

No terceiro capítulo será realizada a Revisão da Literatura sobre os temas: Arquiteturas de *Software*, ADLs, Modelagem, UML e o Estado da Arte do uso da UML como uma ADL de *software*.

No quarto capítulo serão apresentados os resultados da pesquisa.

No quinto capítulo serão expostas as conclusões do trabalho.

2 Metodologia

Para que os objetivos deste trabalho fossem alcançados, foi realizada a revisão da literatura, sua importância é ressaltada em Porfírio (2012), por permitir uma visão clara e o entendimento relacionado ao tema. A revisão da literatura contribui para conhecimento das publicações existentes sobre o tema, aspectos já abordados, verificação de opiniões similares e diferentes, a respeito ou relacionados ao tema da pesquisa (MORESI, 2003).

Uma revisão sobre arquitetura de *software*, ADLs, modelagem e UML foi realizada e adicionada ao trabalho para introduzir o tema principal, que é a análise da aplicabilidade UML como ADL de *software*, ou seja, para realizar a descrição arquitetural. Para que tal propósito fosse efetivado foram lidas 12 (doze) obras onde a UML foi usada, ou avaliada como tal, para que o levantamento das vantagens e desvantagens do uso da mesma para este fim fosse explicitado no presente trabalho. A literatura abrange desde artigos científicos, teses de mestrado e livros.

Para a descoberta, compreensão e avaliação das limitações e benefícios do uso da UML como uma ADL de *software*, realizou-se uma pesquisa qualitativa onde verificou-se argumentos importantes sobre o Estado da Arte. As obras foram consultadas nas bibliotecas digitais *ACM Digital Library*[1], *IEEE Xplore Digital Library*[2] e na base de dados bibliográfica *Google Acadêmico*[3]. A faixa de tempo estipulada para a seleção das obras foi 1999 – 2017, escolhida objetivando-se encontrar uma possível evolução do uso da UML como uma ADL de *software* ao longo dos anos. As palavras-chave utilizadas na pesquisa foram “UML”, “ADL”, “*Software architecture*” e “*Architectural description*”.

2.1 Etapas

Esta obra foi realizada nas seguintes etapas:

- Revisão bibliográfica do tema “Arquitetura de *software*”;
- Revisão bibliográfica sobre as ADLs de *software*;

[1] dl.acm.org/

[2] https://ieeexplore.ieee.org/

[3] https://scholar.google.com.br

- Revisão bibliográfica do tema “Modelagem”;
- Revisão bibliográfica sobre a UML;
- Seleção e filtragem de obras aonde a UML foi utilizada, ou avaliada como uma ADL de *software*, onde as vantagens, desvantagens, benefícios, ou limitações do uso da mesma foram levantados.

3 Revisão da Literatura

3.1 Arquitetura de *software*

Desde o final da década de 80, pesquisas voltadas para a arquitetura de *software* começaram a surgir motivadas pela necessidade de se estudar a produção de *software*s em larga escala (SHAW; CLEMENTS, 2006) evoluindo de meras descrições qualitativas e observações empíricas de como sistemas eram organizados, até o amadurecimento de técnicas de análise, exploração de notações e ferramentas, contribuindo para que a mesma ofereça um meio efetivo para desenvolvimento de *software*s complexos.

Como defendido em Oliveira (2007), a arquitetura representa um elemento essencial no projeto e construção de *software*s (ou sistemas) complexos, pode ser compreendida como um conjunto de componentes organizados e representados em alto nível, e a maneira com que eles se relacionam. Segundo Navasa (2005, apud OLIVEIRA, 2007), a arquitetura quando bem representada garante a satisfação de requisitos nas áreas de desempenho, confiabilidade, portabilidade, escalabilidade e interoperabilidade.

A principal característica das arquiteturas de *software* é o fato de descreverem a estrutura dele como um todo (OLIVEIRA, 2007), permitindo assim a tomada de decisões de alto nível, proporcionar melhor visualização da composição do sistema e a interação entre cada parte do mesmo. Em outras palavras, descreve a estrutura e organização do sistema e provê a elucidação a respeito como ele se comporta, fornecendo uma base sólida sobre qual o *software* pode ser construído.

A arquitetura de *software* pode ser representada por meio de diferentes visões cada qual com seu nível de relevância e detalhamento para um ou mais *stakeholders* em prol de uma melhor comunicação e compreensão sobre se o sistema irá abordar seus anseios. Um único modelo pode ser muito abrangente, e ter seu entendimento dificultado, onde pode-se observar a necessidade de diferentes visões serem descritas.

Cada visão descreve uma determinada parte da arquitetura e permite reduzir a quantidade de informações que o arquiteto trata em um dado momento (OLIVEIRA, 2007). O autor ainda defende que diversas visões podem ser criadas de

acordo com as necessidades do projeto, mas que Clements (2003, apud OLIVEIRA, 2007) propõe as visões de módulo, execução, implementação, implantação e dados.

Kaur e Singh (2011) discorrem em seu trabalho sobre a importância da arquitetura de *software* para o desenvolvimento de sistemas, pois a estrutura e comportamento de *software* são definidos com a arquitetura, ajuda no enfoque de elementos importantes a serem observados e desenvolvidos, balanceia as necessidades dos *stakeholders* e influencia positivamente a estrutura da equipe na escolha das pessoas que irão compor o time, de modo que o mesmo esteja alinhado com a arquitetura.

3.1.1 Linguagens de Descrição Arquitetural

De acordo com Shaw e Clements (2006), Linguagens de Descrição Arquitetural (*Architetural Description Languages*), ou ADLs de *software* são linguagens formais utilizadas para representar a arquitetura de *software*. As ADLs surgiram através da necessidade de formalizar e padronizar a descrição de arquiteturas e visões. Por falta de padronização, cada qual surgiu com seu conjunto de elementos e domínios específicos (OLIVEIRA, 2007).

Como defendido em Soares Neto (2003), ADLs são como notações que permitem a descrição precisa e a análise das propriedades de uma arquitetura de sistema de *software*, oferecendo diferentes visões em diferentes níveis de abstração.

Em seu trabalho, Babar, Jeffery e Zhur (2004) afirmaram que existe pouco consenso sobre uma definição universalmente aceita do que seria uma ADL. Tal fato traz consigo a dificuldade de classificar ou não uma linguagem como uma ADL de *software*, juntamente com o fato de que não existe um conceito absoluto de o que é arquitetura de *software*. A escolha da linguagem mais apropriada para realização da descrição arquitetural deve ser guiada por algum método de avaliação onde devem ser observados quais serão os elementos fundamentais a serem considerados e descritos, realizado pelo próprio usuário, visto que tais métodos de avaliação são muito pessoais e subjetivos. Mesmo na presença dessa dificuldade, a descrição arquitetural não pode ser ignorada, já que seu papel para o sucesso da implementação de um sistema.

Apesar da falta do consenso universalmente aceito sobre o que é uma ADL de *software*, o trabalho de Shaw et al. (1995) aponta propriedades que se espera encontrar nas mesmas como: capacidade de representar componentes e conectores, abstração e encapsulamento, tipos e checagem de tipos e aptidão para acomodar ferramentas de análise. Além disso, Luckham e Vera (1995) apontam uma série de requisitos que consideram essenciais para que uma ADL seja "suficientemente poderosa". São elas: abstração de componentes, abstração de comunicações, habilidade de modelar arquiteturas dinâmicas, capacidade de prover suporte hierárquico de refinamento, etc.

3.1.1.1 Principais elementos de uma descrição arquitetural

A descrição arquitetural é composta por alguns elementos. Soares Neto (2003) descreveu os seguintes: componentes, que são unidades computacionais, ou de armazenamento que possuem geralmente, um estado associado e podem ser representados em diferentes níveis de abstração; interfaces de componentes, que são pontos de interação entre componentes e o restante do sistema, especificam serviços providos por um componente, e provê meios para especificar a necessidade dos componentes. Tais interfaces são chamadas "portas" (GARLAN; CHENG; KOMPANEK, 2002); conectores, elementos responsáveis por representar interações entre componentes, e as regras dessas interações; interface de conectores, que são pontos de interação entre si e outros componentes, e até mesmo conectores; configurações arquiteturais, ou também chamadas de sistemas arquiteturais, são grafos conexos representados por componentes e conectores responsáveis por descrever a estrutura arquitetural, onde é possível verificar a conexão entre componentes, compatibilidade entre interfaces, se a comunicação é permitida pelos conectores, e se a semântica da mesma gera um comportamento esperado. Além destes, Garlan, Cheng e Kompanek (2002) descreveram mais dois elementos: propriedades, que representam elementos adicionais além da estrutura sobre as partes de uma descrição arquitetônica, que variam de acordo com a ADL utilizada, e são utilizadas para representar antecipadamente aspectos não funcionais de um projeto arquitetural, e os estilos arquiteturais que representam uma família de sistemas relacionados e normalmente define um *design* de vocabulário de elementos, como componentes, conectores, portas, e propriedades.

3.1.1.2 Breve descrição das ADLs

Pandey (2010) divide as ADLs em dois grupos, o primeiro compreende as “ADLs iniciais”, que surgiram durante os primeiros anos da pesquisa sobre arquitetura de *software*, onde pesquisadores experimentavam estruturas para se realizar a descrição arquitetural, presentes nas mesmas. Dentre elas estão nomes como Wright, Rapide, C2, dentre outras. O segundo compreende as “ADLs recentes”, que geralmente têm seu foco em geração de código, e foram desenvolvidas a partir da experiência adquirida com as “ADLs iniciais”. Dentre as “ADLs recentes” estão SOFA, AADL, PRISMA, dentre outras. As ADLs acima mencionadas serão descritas brevemente nos tópicos abaixo.

A. Wright

   De acordo com Ribeiro (2017), Wright é uma ADL de propósito geral, voltada para a análise de protocolos de comunicação. Concentra-se em conectores explícitos, verificação automática de propriedades e formalização de estilos arquiteturais. Como mencionado em Ribeiro (2017), Wright não oferece suporte para a implementação de modelos, sendo usada somente como linguagem de modelagem. Além dessas características, é uma ADL que possui semântica formal, mas não fornece suporte para requisitos não funcionais, além de que só define uma visão arquitetural.

B. Rapide

   Conforme Ribeiro (2017), Rapide é uma ADL com ênfase em simulação de sistemas distribuídos, sendo capaz de arquitetar, analisar, simular e gerar código. Apesar se seus benefícios, a linguagem não provê suporte a conectores, limitando sua capacidade de descrição. Fornece apenas uma visão arquitetural, e não fornece suporte para a implementação de modelos, sendo usada somente como uma linguagem de modelagem.

C. C2

   Para Ribeiro (2017), C2 é uma ADL que preocupa-se com aplicações dirigidas a eventos, possui sintaxe textual e gráfica, permite a visualização do comportamento de uma arquitetura em execução, mas

não fornece suporte a requisitos não funcionais, nem define uma semântica formal.

D. SOFA

Segundo Ribeiro (2017), SOFA é uma linguagem voltada para o suporte a estilos arquiteturais. Em sua versão 2.0, possui semântica baseada em meta-modelos, apresentando características como suporte a geração automática de modelos, geração de código para componentes e configuração de uma aplicação. Na linguagem SOFA conectores fornecem a lógica de comunicação, e possibilitam diferentes estilos de comunicação e de requisitos não funcionais. Fornece suporte a apenas a visão de implementação.

E. AADL

Como mencionado em Ozkaya e Kloukinas (2013), AADL é uma ADL que suporta *hardware* e *software*, especializada em sistemas embarcados, que tornou-se a ADL mais amplamente utilizada (quando desconsidera-se a UML como uma ADL). De acordo com Ribeiro (2017), AADL apresenta descrições textuais e gráficas, que podem ser combinadas, e suporta visões arquiteturais para descrever modelos.

F. PRISMA

Como mencionado em Ribeiro (2017), PRISMA é uma linguagem orientada a aspectos que combina Engenharia de *Software* baseada em componentes e Engenharia de *Software* baseada em aspectos. Dentre as características dessa ADL estão: a geração automática de código fonte e especificação requisitos não funcionais, além de possuir sintaxe textual e gráfica.

3.2 Modelagem

Modelo é uma abstração de algo, com a finalidade de entendê-lo antes de construí-lo. Permite omitir detalhes não essenciais, e é mais fácil manipulá-lo do que à entidade original (BLAHA; RUMBAUGH, 2006). Ainda, segundo Blaha e Rumbaugh (2006), a abstração provida pelos modelos permite lidar com complexidade. Os autores defendem que modelos são usados há milhares de anos

por engenheiros, artistas e artesãos para experimentar projetos antes de executá-los. Para a montagem de sistemas de *software*s não é diferente, os desenvolvedores precisam abstrair visões do sistema, montar e verificar se os modelos satisfazem aos requisitos estabelecidos, e acrescentar detalhes gradativamente, para que no fim, os mesmos sejam transformados em implementações.

Modelos são simplificações da realidade, construídos para facilitar a compreensão do sistema de *software* em desenvolvimento (BOOCH; RUMBAUGH; JACOBSON, 2006). Permitem destacar elementos de grande importância, e omitir componentes não relevantes de acordo com os níveis de abstração trabalhados, podem descrever sistemas sob diferentes aspectos. Modelos podem ser estruturais, enfatizando a organização do sistema, ou comportamentais, enfatizando a dinâmica.

A montagem de modelos, ou modelagem, permite com que objetivos de grande importância sejam alcançados. Ajuda na visualização do sistema, permite a especificação da estrutura e comportamento do mesmo, direciona a sua construção, documenta as decisões tomadas (BOOCH; RUMBAUGH; JACOBSON, 2006), e ainda permite o teste de entidades antes de construí-las, facilita a comunicação com clientes, reduz a complexidade, separando um pequeno número de aspectos importantes a serem tratados de uma só vez (BLAHA; RUMBAUGH, 2006).

Booch, Rumbaugh e Jacobson (2006) sugerem quatro princípios da modelagem:

- Modelos devem ser bem escolhidos: modelos escolhidos corretamente solucionaram eficazmente os problemas de desenvolvimento, ao passo que os inadequados podem causar confusões e priorizar questões irrelevantes.
- Cada modelo poderá ser expresso em diferentes níveis de precisão: o nível de detalhamento dos modelos pode mudar de acordo com a necessidade dos *stakeholders* ao qual o modelo será exibido.
- Os melhores modelos estão relacionados à realidade: os modelos devem ser feitos de modo que exista clara conexão com a realidade, e, além disso, caso a conexão não seja fiel, deve-se saber exatamente como esses modelos diferem-se do “mundo real”.
- Nenhum modelo único é suficiente: para compreender a estrutura de sistemas de *software* complexos, é preciso recorrer a várias visões complementares e

inter-relacionadas, que poderão conter aspectos estruturais e comportamentais, onde seu conjunto representará a base do projeto.

Como o enfoque deste trabalho se dá na modelagem arquitetural realizada pela UML, é importante frisar que para Taylor, Medvidovic e Dashofy (2009, apud LEVIN, 2009), modelos de arquitetura são artefatos que capturam algumas ou todas as decisões de *design* que compreendem a arquitetura de um sistema. A modelagem arquitetural é a reificação e documentação das decisões de um projeto, uma notação de modelagem arquitetural é uma linguagem ou meio de capturar decisões de *design*. Para Levin (2009), sem um modelo a arquitetura é incompreensível. Notações de modelagem podem ser ricas e ambíguas, altamente formais, semanticamente fracas, ou semanticamente formais, a autora discorre também sobre a possibilidade de combinação das notações para se descrever arquiteturas.

3.2.1 Unified Modeling Language (UML)

A UML é uma linguagem de modelagem introduzida em 1994, sendo a unificação de três métodos orientados a objetos: o método de Booch, a Técnica de Modelagem de Objetos de Rumbaugh (OMT – *Object Modeling Technique*) e a Engenharia de *Software* Orientada a Objetos de Jacobson (OOSE – *Objected Oriented Software Engineering*) (BOOCH; RUMBAUGH; JACOBSON, 2006). Onde seus idealizadores buscaram equilibrar simplicidade e expressividade para alcançar seus principais objetivos, sendo eles visualizar, especificar, construir e documentar sistemas orientados a objetos.

3.2.1.1 Visão geral da UML

De acordo com os idealizadores da UML, Booch, Rumbaugh e Jacobson (2006), ela é uma linguagem expressiva podendo abranger as visões necessárias para o desenvolvimento e implantação de diversos tipos de sistemas, como sistemas de informação corporativos, aplicações WEB, até sistemas embutidos de tempo real, e, ao mesmo tempo simples de se compreender e usar.

Para Booch, Rumbaugh e Jacobson (2006), a UML é uma linguagem destinada a:

- Visualização: permite fazer a modelagem textual e gráfica, possibilitando ao usuário visualizar o desenho e a comunicação entre objetos.
- Especificação: permite a construção de modelos específicos, sem ambiguidades, e completos.
- Documentação: abrange a documentação da arquitetura, permite expressão de requisitos, e a realização de testes.

3.2.1.2 Elementos da UML para descrever arquiteturas

Garlan, Cheng e Kompanek (2002) sumarizam em sua obra alguns elementos da UML que podem ser usados para descrever arquiteturas, item propositalmente escolhido, visto que o objetivo do trabalho é avaliar a aplicabilidade da UML como uma ADL de *software*. São estes:

- Classes e objetos: classes são construções que descrevem a visão lógica de um sistema. Possuem propriedades na forma de atributos, fornecem serviços abstratos na forma de operações, e podem estar logicamente ligada a outras classes. São usadas em diagramas de classes para descrever uma visualização estática de um sistema. As instâncias das classes são chamadas de objetos, que podem ser incluídos em diagramas de classes/objetos para representar casos reais, ou prototípicos da visão estática do projeto, ou usado nas chamadas colaborações para representar comportamento em cenários particulares;
- Interfaces: coleções de operações que especificam os serviços de uma classe, componente, ou subsistema, e definem algum aspecto do comportamento de um elemento, como um conjunto de especificações de operação;
- Colaborações: especificam como elementos como classes e interfaces cooperam para oferecer algum comportamento agregado, podem apresentar tanto aspectos comportamentais, quanto estruturais.
- Componentes: usados para descrever peças físicas, implantáveis do sistema. Podem expor suas funcionalidades por meio de

interfaces e são associados uns aos outros por meio de dependências.

- Pacotes: são mecanismos de agrupamento usados para particionar grandes modelos da UML em partes gerenciáveis;
- Relacionamentos: indicam dependência entre elementos. Alguns exemplos são: associações, que descrevem o relacionamento entre classes e os papéis que elas desempenham nos relacionamentos; generalizações, que representam relações entre objetos pais e objetos filhos; realizações, que são relações semânticas onde elementos de modelagem especificam uma espécie de contrato onde outro elemento deve garantir a execução do mesmo.
- Estereótipos: a UML fornece um mecanismo para descrever formas especializadas de outros tipos de modelo com o objetivo de permitir sua extensão a conceitos de domínio específico. Normalmente, estereótipos são definidos para restringir o uso ou semântica de outro tipo de modelo.

3.3 Estado da Arte

O presente trabalho consiste na leitura de 12 obras onde a UML foi utilizada para realizar o papel de uma ADL de *software*, ou ao menos avaliada como tal para que as vantagens e pontos fortes, desvantagens e fraquezas, que podem estar explícitas ou implícitas sejam levantadas, permitindo assim, a avaliação da mesma como uma ADL de *software*. Os artigos analisados são abordados separadamente.

3.3.1 *Using the UML for Architectural Description*

O trabalho de Hilliard (1999) sugere que há um grande interesse no uso da UML para a realização de descrição arquitetural e analisa a sua aplicabilidade para tal objetivo de acordo com as práticas recomendadas pelo IEEE Architecture Working Group et al. (1999), intitulada *“Recommended Practice for Architectural Description”*, também conhecida por IEEE P1471, cuja finalidade é fornecer

orientações relacionadas à descrição de arquiteturas de *software* tanto no meio industrial e acadêmico. O autor examina várias abordagens para usar a UML no atendimento dos requisitos da IEEE P1471. Em sua pesquisa, o autor explicita que a UML adequa-se bem à IEEE P1471, pois fornece um amplo conjunto pronto para modelar diversos aspectos das arquiteturas de *software*, porém, possui falhas também, que são causadas pela generalidade de sua ontologia.

3.3.2 *Describing Software Architecture with UML*

O trabalho de Hofmeister, Nordi e Soni (1999), relatou a experiência do uso da UML para descrever a arquitetura da parte de processamento de imagens de um sistema de aquisição de imagem em tempo real, de acordo com as visões arquiteturais que julgavam mais importantes baseados em estudos e experiências anteriores. Nesta obra, os autores buscavam uma notação clara e que fosse igualmente legível para arquitetos, desenvolvedores e gerentes, e decidiram eleger quatro diferentes visões arquiteturais a serem descritas pela UML: conceitual, de módulo, de execução e de código.

Na visão conceitual, a arquitetura é descrita em termo de elementos de domínio, nela seriam projetados os requisitos funcionais do sistema. Na visão de módulos é descrita a decomposição do *software* e sua divisão em camadas. A visão de execução mapeia os módulos para imagens em tempo de execução, definindo a comunicação entre os mesmos e atribuindo-lhes recursos físicos. A visão de código captura a forma que módulos e interfaces são mapeados para arquivos de origem, e imagens de tempo de execução são mapeados para arquivos executáveis. Os autores frisam e sua obra, que cada uma dessas quatro visões possui elementos específicos que precisam ser nomeados e sua interface, atributos e relacionamentos devem ser descritos. Nas seções posteriores, é mostrado como a UML foi usada para descrever as quatro visões elegidas. Ao final foram apontados em quais quesitos a UML foi eficiente para efetivar o papel de uma ADL de *software*. Como vantagens do uso da UML, foi mencionado que ela descreveu bem a estrutura estática do sistema, e sequências particulares de atividades. Como desvantagens, identificou-se que a UML foi deficiente para descrever mapeamentos diretos, como

correspondência entre elementos de diferentes pontos de vista, protocolos, portas em componentes e sequências gerais de atividades.

3.3.3 *Reconciling the Needs of Architectural Description with Object-Modeling Notations*

Garlan, Cheng e Kompanek (2002) afirmaram que usar a UML como uma ADL tem benefícios, mas que existem questões em aberto sobre se as soluções orientadas a objetos são suficientemente expressivas para isso. Em seu trabalho eles descreveram quatro estratégias para representar a estrutura arquitetural por meio da UML, para ao menos tentar que tais representações sejam tão expressivas quanto às feitas ADLs “tradicionais”.

As estratégias criadas focam-se nos componentes, que segundo os autores, são elementos centrais de um projeto de descrição arquitetural, e para cada estratégia escolhida, foram consideradas sub-alternativas para representar os outros elementos arquiteturais como portas, conectores e sistemas. Cada estratégia foi descrita em uma seção, e são elas:

- “Classes e objetos”: onde tipos de componentes são representados por classes UML, e suas instâncias representadas por objetos;
- “Classes e classes”: onde tanto os tipos de componentes, quanto suas instâncias são representadas por classes UML;
- “Componentes UML”: onde tipos de componentes são representados como componentes UML e instâncias de componentes como instâncias de componentes UML;
- “Subsistemas”: onde os tipos de componentes são representados como subsistemas UML e instâncias de componentes, como instâncias dos subsistemas UML.

Além disso, foram definidos também os critérios de avaliação, onde idealmente, os autores gostariam que fossem exibidas três características:

- Correspondência semântica: o mapeamento deve respeitar a semântica da UML, o modelo deve ser inteligível tanto para arquitetos, quanto para ferramentas baseadas em UML;

- Clareza visual: as descrições arquiteturais resultantes na UML devem trazer clareza ao *design* do sistema, evitar confusões visuais e realçar os principais detalhes do *design*;
- Completude: os conceitos arquiteturais abordados pelos autores devem ser representáveis em UML.

Por meio desta obra, Garlan, Cheng e Kompanek (2002) chegaram à conclusão de que a UML não fornece o melhor caminho para a codificação e elementos arquiteturais. As estratégias mostradas por eles possuem pontos fortes e fracos, e geralmente, uma exigência deve ser priorizada em detrimento à outra quando se diz respeito à clareza e a completude. Tais fatores implicam que a escolha de qual decisão tomar depende de quais aspectos da arquitetura precisam ser descritos. Além disso, é falado que todas as estratégias possuem alguma forma de incompletude, ou incompatibilidade semântica. Outro problema apontado foi a dificuldade de representar conceitos arquiteturais, como portas, conectores, e subestruturas.

3.3.4 *Documenting Architectural Connectors with UML 2*

O trabalho de Ivers et al. (2004) aborda sobre o fato de a popularidade da UML (1.x) ter feito com que ela fosse usada para representar arquiteturas e que usuários adotaram algumas convenções para representar elementos arquiteturais, ou criaram *profiles* para especializar a UML, obtendo-se assim, resultados mistos e em alguns casos decepcionantes. Ainda, o autor menciona que as mudanças trazidas pela UML 2.0 melhoraram de forma significativa a adequação para a realização da descrição arquitetural. No trabalho é avaliado o quão bem o conceito de conectores em UML 2.0 satisfaz a quatro critérios expostos a seguir: correspondência semântica, onde elementos de modelagem UML devem mapear intuitivamente conceitos arquiteturais; clareza visual, para evitar confusões, e realçar os principais detalhes do projeto; completude, que possibilita que características arquiteturais relevantes sejam expressáveis pela UML; suporte às ferramentas. Os autores analisam elementos padrões da UML que podem ser usados para representar conectores (Associações, Classes de Associações, e Classes), e quão bem eles satisfazem os quatro critérios acima mencionados. Ao final da obra, é

mencionado que a revisão da UML que resultou na UML 2.0 melhorou consideravelmente com a introdução de novos elementos, como portas, mas que modelar conectores ainda continua sendo problemático.

3.3.5 *Using UML with OCL as ADL*

Navarčik (2005) discorre sobre o fato de se existir um número considerável de ADLs, mas que mesmo assim, seu uso é muito baixo. Em paralelo a isso, a UML é analisada como um padrão industrial, mas não totalmente qualificada para ser considerada uma ADL. O objetivo do artigo é fornecer o mapeamento entre a UML, e a *Object Constraint Language* (OCL[4]), para o empenho do papel de uma ADL de *software*.

O autor frisa que apesar de inicialmente ter sido criada para apoiar a análise e o *design* de *software*s orientados a objetos, sua revisão, que consiste na UML 2.0, aprimorou o suporte para modelagens de arquiteturas de sistemas de *software*. No artigo, ainda é dito que apesar do amplo uso industrial da UML, sozinha nunca foi identificada expressiva o suficiente para ser considerada uma ADL, mesmo que estratégias de descrição arquitetural com UML tenham sido propostas na literatura.

De acordo com o autor, OCL é parte integrante da UML (2.0), e é comumente usada para definir restrições do sistema modelado. Sua incorporação fornece limitações extra sintáticas e semânticas, o que torna o modelo mais claro, porém, mais complexo. Ainda é dito que OCL permite maior raciocínio e é capaz de captar aspectos dinâmicos do sistema.

Na obra há uma seção onde o criador do artigo discute sobre se a UML utilizada em conjunto com a OCL configura uma ADL, e curiosamente, ele diz que apesar de essa junção fornecer todos os recursos esperados por ADLs, classificar a UML como tal, seria limitar os recursos da linguagem, que é capaz de realizar outros aspectos e níveis de detalhe da modelagem que nada tem a ver com a arquitetura.

[4] Linguagem notacional para análise e *design* de software, usada em conjunto com a UML para especificar a semântica de blocos de construção com precisão. (KAUR; SINGH, 2011)

3.3.6 *In practice: UML software architecture and design description*

A fim de levantar a maneira com que profissionais utilizam a UML em projetos, e as armadilhas de tal feito, Lange, Chaudron e Muskens (2006) propõe em sua obra, levantar tais respostas através de um questionário anônimo, e ainda, analisaram modelos UML em quatorze estudos de caso na Indústria, focando-se na qualidade dos modelos UML em projetos reais.

É interessante ressaltar que os autores deram especial atenção aos critérios que os entrevistados usavam para determinar quando as atividades de modelagem devem terminar. Os mais relatados foram, em ordem: a completude, passar por alguma revisão, ou inspeção, e os prazos a serem atendidos, no entanto não existia nenhum critério objetivo para verificar se um modelo estava completo, ou se satisfazia alguma noção tangível de qualidade. Como os resultados mostraram que a completude era o critério considerado o mais importante para a maioria dos entrevistados, foram investigados os problemas causados por modelos incompletos, sendo eles: má qualidade do modelo, produtos que não atendem os requisitos e esforços excessivos aplicados aos testes.

Na obra, também são mencionados o problemas com descrições UML identificados nos questionários e entrevistas adicionais, como informações dispersas, incompletude, uso informal, falta de convenções de modelagem, entre outros. Também foram identificados os defeitos em modelos industriais em UML, verificados por uma ferramenta que analisou uma série de estudos de caso, implementada pelos autores, onde puderam constatar que havia várias violações nas regras de modelagem em UML.

Por fim, foram apresentados alguns pontos em que as práticas UML devem melhorar, como prover meios para detectar falhas, omissões e inconsistências, maior uniformidade na modelagem, e mais consistência entre modelos UML e requisitos do sistema.

3.3.7 *Modeling in Software Architecture*

Levin (2009) em seu trabalho aborda a importância da modelagem arquitetural para a compreensão do *software* a ser desenvolvido, e reúne em seu trabalho notações e linguagens usadas para descrever a arquitetura. As notações

abordadas abrangeram desde o uso da linguagem natural, estilos gráficos informais, como diagramas em Power Point, UML, ADLs como Rapide, Koala, e ADLs extensíveis, como Acme.

Logo no início da subseção que se refere à UML, são apresentadas algumas desvantagens e vantagens do seu uso para realizar a modelagem arquitetural. É dito que em suas versões anteriores, o foco se dava na modelagem do design, o que configura uma limitação, pois a arquitetura é igualmente importante por afetar todos os outros estágios do ciclo de vida do *software*. Tal limitação requer o uso de outras notações em conjunto com a UML para permitir a total modelagem da arquitetura de sistemas (TAYLOR; MEDVIDOVIC; DASHOFY, 2009, apud LEVIN, 2009). Outra característica apontada que pode caracterizar um problema é a ambiguidade das construções de modelagem, que apesar de trazer generalidade e flexibilidade na notação, se faz necessária adição de informações complementares para entender-se a semântica de determinado diagrama, além de ser limitada para modelar requisitos não funcionais. Como benefícios, a autora apontou a grande quantidade de elementos para descrever arquiteturas, como classes, associações, estados, etc., capacidade de modelar múltiplos pontos de vista, suporte para a modelagem de elementos estáticos e dinâmicos, grande variedade de ferramentas disponíveis, e o fato de a UML ser uma linguagem amplamente utilizada.

Na seção que debate sobre a escolha de uma ADL de *software*, a autora se refere à UML como uma notação flexível, e extensível. Menciona que a linguagem oferece variedade de tipos de diagramas, entidades, relacionamentos e a capacidade de estender os diagramas com anotações torna a linguagem hábil para modelar aplicações em múltiplos domínios. Fala-se em sua obra, que a UML não está vinculada a nenhum estilo arquitetural, o que possibilita o seu uso em múltiplos contextos. Outra vantagem se dá no fato de ela ser uma linguagem amplamente utilizada, fazendo com que existam múltiplas aplicações de código aberto e comerciais para modelagem usando a notação gráfica UML. Além do mais, existem ferramentas capazes de expressá-la em notação textual.

A autora expôs características de diversas linguagens em sua obra, como Darwin, Rapide, Wright, Koala, Weaves, AADL, Acme, ADML, xADL, e DFN, e ainda assim, informou que na época da pesquisa, a UML parecia ser a linguagem mais extensível e flexível, a mais amplamente utilizada, suportada por várias ferramentas,

que possui uma comunidade de usuários disponíveis, além de tutoriais de uso para facilitar seu aprendizado.

3.3.8 *Architecture Description Languages (ADLs) vs UML: A Review*

Pandey (2010) relata em seu trabalho que desde o surgimento da arquitetura de *software*, algumas ADLs surgiram e desapareceram, mas que nenhuma se tornou popular exceto poucas de domínio específico. Em contrapartida, a UML, que em alguns casos sequer é aceita como uma ADL, ou aceita com hesitação, tornou-se o padrão industrial para modelar arquiteturas.

Em seu trabalho, o autor questiona alguns profissionais sobre se a UML é ou não uma ADL de *software*, e pôde concluir que a maior parte dos profissionais a aceita como uma, mas que reconhecem as fraquezas da mesma.

Na obra são apresentados diversos pontos fortes e fracos das ADLs. Como pontos fortes, são mencionadas: a formalidade e a forma inequívoca ao representar a arquitetura, a natureza textual apresentada pela maioria delas as torna legíveis para máquinas e adequadas para a automação, suas características formais de representação permitem a análise e correção da arquitetura, e o suporte a geração automática de sistemas de *software*. Como pontos fracos o autor pontua que: a maioria das ADLs é textual e pouco atraente para arquitetos de *software*s de domínios diferentes pelos quais foi criada, a maioria foi criada para documentar a arquitetura de *software*s de domínio específico, falta de apoio de ferramentas, não suportam várias visões arquiteturais, e a maioria é bastante restritiva, impondo um modelo arquitetural que muitas vezes não é adequado para os usuários.

São também apresentados os pontos fortes e fracos da UML. Os pontos fortes são: a UML é uma linguagem gráfica, e consequentemente mais chamativa, suporta várias visões que são muito úteis para *stakeholders*, existem várias ferramentas disponíveis para a UML, e é uma linguagem de propósito geral e tem sido utilizada de forma eficaz em vários âmbitos da engenharia. Como fraquezas, são apontadas: a UML não é adequada para a análise automatizada de verificação e validação da arquitetura e sua semântica informal que pode tornar-se uma fonte de ambiguidade e inconsistência em alguns casos.

O autor conclui em sua obra que a popularização da UML se dá por todas as vantagens que ela oferece, mas que ainda existe a necessidade de que alguns recursos das ADLs sejam incorporados a ela para torná-la ainda mais expressiva. É dito ainda, que a UML possui quase todas as características das ADLs, e do ponto de vista da compreensão humana a UML pode servir como a melhor ferramenta para documentar a arquitetura de sistemas de *software*s.

3.3.9 *UML (Unified Modeling Language): Standard Language for Software*

Kaur e Singh (2011) primeiramente discorrem sobre a influência da construção de uma sólida visão arquitetural para o sucesso, ou fracasso de um projeto, também é dito que a arquitetura configura uma ponte entre requisitos e a implementação. Os autores discorrem sobre as contribuições da arquitetura de *software* em diversos aspectos do desenvolvimento. O foco do artigo se dá na padronização da UML para o desenvolvimento de *software*s, e a posterior discussão dos benefícios e fraquezas desta linguagem. Os benefícios apontados pelos autores são a análise primária do modelo, desenvolvimento baseado em componentes, fácil de usar, fazendo com que o arquiteto tenha controle sobre a arquitetura em construção, impactando beneficamente na construção do *software*. Fraquezas também foram discutidas, os autores mencionaram que a semântica da UML é “bastante solta”, que a maneira de descrever nomes de relacionamentos deveria ser melhorada, que a linguagem lida apenas parcialmente com identificadores únicos, que é necessário aprimorar a infraestrutura de comparação e mesclagem de arquivos de diagrama, além de que, de acordo com os autores, UML é só sintaxe e não diz nada a respeito de como criar modelos.

3.3.10 *Are We There Yet? Analyzing Architecture Description Languages for Formal Analysis, Usability, and Realizability*

Ozkaya e Kloukinas (2013) relatam em sua obra, que apesar de a pesquisa sobre arquitetura de *software* estar ativa desde o início dos anos 90, e algumas dezenas de ADLs de *software* terem surgido ao longo deste tempo, ao contrário do que se esperava, elas não foram amplamente adotadas. De acordo com os autores, isso se dá pelo fato de que as linguagens desenvolvidas possuem uma série de

deficiências que impedem sua aplicação prática. Praticantes relataram que as ADLs disponíveis são difíceis de usar, dificultando a construção de arquiteturas grandes e complexas, e não suportam análise automática. Diante de tal situação, é proposta então a comparação entre algumas ADLs “iniciais”, e as mais recentes, de acordo com as propriedades que os idealizadores da pesquisa julgaram como cruciais a serem apresentadas pelas mesmas. Dentre os resultados obtidos, foram relatados que não houve grandes evoluções entre as ADLs atuais e iniciais, se comparadas pelos critérios adotados pelos autores. Em contrapartida, a UML é mais usada que todas as 13 ADLs analisadas no artigo, provavelmente por ser mais bem conhecida, e extensivamente usadas em projetos de baixo nível, mesmo que se saiba do fraco suporte arquitetural fornecido, além da semântica informal.

3.3.11 *What Industry Needs from Architectural Languages: A Survey*

O trabalho Malavolta et al. (2013) enfatiza que apesar de existir uma quantidade razoável de ADLs, existem dúvidas sobre se elas atendem as necessidades percebidas pelo usuário. Os autores construíram um questionário, respondido por 48 arquitetos cuidadosamente selecionados, de 40 empresas de TI, e 15 países diferentes, para analisar a percepção dos participantes em relação aos pontos fortes, limitações e necessidades em relação à descrição arquitetural, a fim de fornecer direcionamento o desenvolvimento de ADLs futuras. Dentre os dados coletados, foi descrito que a UML é amplamente utilizada pela indústria, tendo seu uso relatado por 86% dos entrevistados, mas apesar da grande proporção, a linguagem não convergiu para um padrão de notação para modelar arquiteturas. Pôde-se notar também, que de acordo com a entrevista feita, existe a divergência de opiniões dos usuários da UML para descrever a arquitetura sobre considerá-la ou não uma ADL. Dado o foco da entrevista, os autores identificaram a necessidade de notações formais, que concedem análises e automatização de tarefas, permitirem também maior comunicação entre os *stakeholders*, visto que quando os arquitetos têm que escolher, eles priorizam linguagens simples e intuitivas, e bem apoiadas por ferramentas em vez de uma linguagem com grande formalidade, o que explica o forte uso da UML na indústria.

3.3.12 Uma técnica baseada e SysML para modelar a arquitetura de sistemas embarcados de tempo real

Em sua tese de Mestrado, Ribeiro (2017), propõe o uso de um *profile* da UML intitulado *Systems Modeling Language*, ou SysML para modelar a arquitetura de sistemas de tempo real automotivos (sistema de controle de *airbags* e de faróis). Em sua obra, a autora ressalta a importância que a arquitetura de *software* tem para a comunicação dos *stakeholders*, e que apesar disto, as ADLs não são amplamente usadas pela indústria por fatores como a dificuldade de expressar arquiteturas de sistemas complexos. Ainda é dito que tal fato fez com que a indústria utilizasse a UML amplamente, apontando também, uma das grandes vantagens da linguagem, que é a sua extensibilidade, permitindo a criação e o uso de *profiles* para superar as limitações da mesma, já que se tratando de sistemas embarcados de tempo real, a UML sozinha não consegue representar características cruciais, como as restrições de tempo real. Em sua obra, a autora também comparou e apontou as vantagens da SysML em relação a outra linguagem de descrição arquitetural, a AADL, que objetiva analisar e especificar sistemas embarcados complexos de tempo real, considerada a ADL mais utilizada pelos arquitetos (quando desconsidera-se que a UML é uma ADL). A autora conclui que na técnica proposta em sua tese, de usar a UML em conjunto com a SysML fornece elementos para descrever requisitos de *software*, relacionamentos com o sistema, permite gerenciamento das mudanças, maior facilidade ao rastrear requisitos, e a efetiva comunicação dos *stakeholders*, aspecto importante, visto que equipes de desenvolvimento de sistemas embarcados de tempo real são compostas por ampla diversidade de profissionais envolvidos.

4 Resultados e Discussões

Após a leitura das 12 obras, puderam ser observadas as vantagens e desvantagens do uso da UML como uma ADL de *software*, que serão descritas nas próximas subseções.

4.1 Limitações

A desvantagem do uso da UML para realizar a modelagem arquitetural se dá nas fraquezas abaixo pontuadas:

- **Deficiente para descrever elementos da descrição arquitetural**

  As versões antigas da UML se mostraram deficientes para descrever elementos importantes da descrição arquitetural, como portas conforme dito em Hofmeister, Nordi e Soni (1999), portas e conectores, como em Garlan, Cheng e Kompanek (2002). A revisão que resultou na UML 2.0 permitiu a modelagem arquitetural de maneira mais natural, introduziu novos elementos à linguagem, como portas, mas que modelar conectores continuou sendo problemático (IVERS et al., 2004).

- **Convenções para representar elementos nem sempre são completas e satisfatórias**

  Ivers et al. (2004) menciona que a popularidade das antigas versões da UML aliada ao seu suporte arquitetural deficiente fez com que usuários adotassem convenções de modelagem, que por vezes apresentavam resultados decepcionantes. Na UML 2.0 o problema parece perdurar, visto que Lange, Chaudron e Muskens (2006) relatam que existe a falta de convenções de modelagem, fazendo com que profissionais usem a UML de acordo com seus hábitos individuais.

- **Múltiplas visões**

  Apesar das vantagens da representação de múltiplas visões, Hofmeister, Nordi e Soni (1999) relataram a dificuldade de se realizar o mapeamento de

elementos entre as diferentes visões arquiteturais. Lange, Chaudron e Muskens (2006), apontaram a dispersão de informações pelos diferentes visões como um problema.

- **Incompletude**

  A incompletude se dá nas técnicas de modelar a arquitetura propostas por Garlan, Cheng e Kompanek (2002). Em alguns casos a UML só é utilizada para modelar parte da arquitetura (LANGE; CHAUDRON; MUSKENS, 2006).

- **Foco no *design***

  A UML não foi desenvolvida para descrever arquiteturas, e não convergiu para um padrão de modelagem arquitetural (MALAVOLTA et al., 2013). De acordo com Levin (2009), o foco no *design* dado pela UML se configura em uma desvantagem do seu uso, visto que o projeto arquitetural é igualmente importante por impactar todas as etapas da construção do *software*. Com o foco no *design*, o aprimoramento do suporte arquitetural fica em segundo plano, podendo nunca se tornar plenamente satisfatório.

- **Semântica informal**

  O fato da UML não possuir semântica formal é uma desvantagem apontada em Levin (2009), Pandey (2010), Kaur e Singh (2011), Ozkaya e Kloukinas (2013), e pode contribuir com ambiguidade e inconsistências na arquitetura.

- **Não é adequada para a análise automatizada**

  Por ser uma linguagem gráfica, a UML não é adequada para o processamento automatizado como dito em Levin (2009) e Pandey (2010). Tal fato se torna uma desvantagem em relação a muitas ADLs, que suportam análise automatizada para a verificação e validação da arquitetura (PANDEY, 2010).

- **Limitada para a modelagem de requisitos não funcionais (LEVIN, 2009)**

  Ao modelar a arquitetura de *software*, deseja-se obter controle e rastreabilidade dos requisitos funcionais e não funcionais.

4.1.1 Limitações – Síntese

O Quadro 1 apresenta uma síntese das limitações descobertas neste trabalho sobre o uso da UML para a descrição arquitetural.

Quadro 1 – Limitações do uso da UML para descrever arquiteturas

Limitações	Referências
Deficiente para descrever elementos das descrições arquiteturais.	(HOFMEISTER; NORDI; SONI, 1999), (GARLAN; CHENG; KOMPANEK, 2002), (IVERS et al., 2004)
Convenções de representação de elementos incompletas e insatisfatórias	(IVERS et al., 2004), (LANGE; CHAUDRON; MUSKENS, 2006)
Representação da arquitetura em múltiplas visões e as dificuldades que podem acompanhá-las	(HOFMEISTER; NORDI; SONI, 1999), (LANGE; CHAUDRON; MUSKENS, 2006)
Incompletude	(GARLAN; CHENG; KOMPANEK, 2002), (LANGE; CHAUDRON; MUSKENS, 2006)
Foco no design	(LEVIN, 2009), (MALAVOLTA et al., 2013)
Semântica informal	(LEVIN, 2009), (PANDEY, 2010), (KAUR; SINGH, 2011), (OZKAYA; KLOUKINAS, 2013)
Inadequada para a análise automatizada	(LEVIN, 2009), (PANDEY, 2010)
Limitada para a modelagem de requisitos não funcionais	(LEVIN, 2009)

Fonte: Elaborado pela autora.

4.2 Benefícios

A UML é considerada a notação padrão *de facto* para documentar a arquitetura, isso se dá pelas vantagens que apresenta que incluem, mas não se limitam as que seguem:

- **UML 2.0**

  As versões antigas da UML eram caracterizadas pelo seu baixo suporte arquitetural, fazendo com que muitas vezes os desenvolvedores adotassem convenções que não auxiliavam no alcance de resultados satisfatórios. O surgimento da UML 2.0 trouxe maior adequação para a realização da descrição arquitetural (IVERS et al., 2004; NAVARČIK, 2005; LEVIN, 2009). Tal versão fornece 13 tipos de diagramas que permitem a modelagem da arquitetura em diferentes níveis de abstração (LANGE; CHAUDRON; MUSKENS, 2006). A UML também conta com grande variedade de elementos para descrever arquiteturas, e é capaz de representar aspectos estáticos e dinâmicos em uma arquitetura (LEVIN, 2009).

- **Generalidade**

  De acordo com Pandey (2010), a UML é uma linguagem de propósito geral utilizada eficazmente em quase todos os domínios da Engenharia de *Software*.

- **Fortemente apoiada por ferramentas**

  Existe um extenso conjunto de ferramentas disponíveis para apoiar a modelagem, inclusive a arquitetural, com a UML (HILLIARD, 1999; LEVIN, 2009; MALAVOLTA et al., 2013). Segundo Levin (2009), a ampla adoção da UML ao longo dos anos, contribuiu para o aumento desse número, provendo grande disponibilidade de ferramentas de código-aberto, ou comerciais para os profissionais.

- **Extensibilidade**

  Dentre as vantagens da UML está na capacidade de extensão através da criação e uso de *profiles* (RIBEIRO, 2017). Conforme Malavolta et al. (2013), diversos *profiles* foram propostos para aprimorar a modelagem da arquitetura, e consequentemente eliminar algumas limitações da UML na descrição arquitetural. Navarčik (2005) aborda o mapeamento entre UML e OCL para atuarem como uma ADL de *software*, alcançando benefícios como criação de modelos mais claros, permitindo maior raciocínio e capacidade de captar aspectos dinâmicos do sistema. Ribeiro (2017) propôs o uso da SysML, um profile da UML para modelar a arquitetura de sistemas embarcados de tempo real, atingindo muitos benefícios como a modelagem de múltiplas visões arquiteturais, identificação de requisitos, e a efetiva comunicação dos *stakeholders*.

- **Amplamente adotada**

  A UML é considerada a língua padrão para descrever arquiteturas, e na pesquisa de Levin (2009), Ozkaya e Kloukinas (2013) e de Malavolta et al. (2013) é apontada como a linguagem mais utilizada para descrever arquiteturas em relação à todas as outras ADLs estudadas por eles.

- **Linguagem gráfica**

  Segundo Pandey (2010), o fato de a UML ser uma linguagem gráfica a torna alvo de interesse dos desenvolvedores, pois melhora claramente a compreensão da arquitetura desenvolvida.

- **Permite a modelagem de diversas visões**

  Uma das grandes vantagens da UML se dá por ela permitir a modelagem arquitetural em diferentes pontos de vista (LEVIN, 2009; PANDEY 2010). Tal fato se torna uma vantagem em relação a algumas ADLs que não permitem tal feito (PANDEY, 2010). A modelagem em mais de um ponto de vista é

primordial para conciliar expectativas, e estabelecer comunicação entre *stakeholders*.

- **Flexibilidade**

  As ADLs em sua maioria possuem natureza restritiva impondo um modelo arquitetural muitas vezes inapropriado para os profissionais. Tal fato contribuiu para que elas não se popularizassem (PANDEY, 2010). Em contrapartida, a UML possui natureza flexível, não está vinculada a nenhum estilo arquitetural, podendo ser usada em vários contextos (LEVIN, 2009).

- **Simples de ser entendida e utilizada**

  Malavolta et al. (2013) mostra em seu trabalho que um dos fatores que fizeram profissionais abandonarem as ADLs é o excesso de formalidade presente nelas, o que prejudica a comunicação entre *stakeholders*. Ozkaya e Kloukinas (2013) apontam que dentre os motivos para a não popularização das mesmas se dá pela dificuldade do seu uso. Conforme Pandey (2010), a UML é simples de ser utilizada apesar de ter quase todas as características das ADLs, e do ponto de vista da compreensão humana pode servir como a melhor notação para documentar arquiteturas.

4.2.1 Benefícios – Síntese

O Quadro 2 apresenta uma síntese dos benefícios descobertos neste trabalho sobre o uso da UML para a descrição arquitetural.

Quadro 2 – Benefícios do uso da UML para descrever arquiteturas

(continua)

Benefícios	Referências
Revisão da linguagem que resultou na UML 2.0	(IVERS et al., 2004), (NAVARČIK, 2005), (LANGE; CHAUDRON; MUSKENS, 2006), (LEVIN, 2009),

Quadro 2 – Benefícios do uso da UML para descrever arquiteturas

(conclusão)

Benefícios	Referências
Generalidade	(PANDEY, 2010)
Fortemente apoiada por ferramentas	(HILLIARD, 1999), (LEVIN, 2009), (MALAVOLTA et al., 2013)
Extensibilidade	(NAVARČIK, 2005), (MALAVOLTA et al., 2013), (RIBEIRO, 2017)
Amplamente adotada	(LEVIN, 2009), (MALAVOLTA et al., 2013), (OZKAYA; KLOUKINAS, 2013)
Linguagem gráfica	(PANDEY, 2010)
Permite a modelagem de diferentes visões	(LEVIN, 2009), (PANDEY 2010)
Flexibilidade	(LEVIN, 2009), (PANDEY 2010)
Simples de ser entendida e utilizada	(PANDEY, 2010), (MALAVOLTA et al., 2013), (OZKAYA; KLOUKINAS, 2013)

Fonte: Elaborado pela autora.

5 Conclusões

Com este trabalho foi possível realizar a revisão bibliográfica sobre diversos conceitos da Engenharia de *Software*, como arquitetura de *software*, modelagem, ADLs e UML.

A arquitetura de *software* tem sido estudada desde o início dos anos 90 e se tornou elemento primordial de grande impacto para as outras fases do desenvolvimento de *software*. Quando bem descrita, ela facilita o entendimento do sistema a ser construído, auxiliando no levantamento dos requisitos funcionais e não funcionais e no balanceamento das expectativas dos *stakeholders*.

As ADLs fornecem notações formais para que descrições arquiteturais sejam feitas, e uma quantidade considerável das mesmas surgiu desde que os estudos se voltaram para a arquitetura de *software*, mas seu uso não foi popularizado por diversos motivos. Em contrapartida, a UML, inicialmente criada para fornecer o *design* detalhado de projetos de *software* se tornou a língua padrão para a modelagem da arquitetura, mesmo apresentando uma série de limitações. Diante deste fato intrigante foram selecionadas algumas obras da literatura para que fossem levantadas as vantagens e desvantagens do uso da UML como uma ADL de *software*.

Como desvantagens, foram identificadas as limitações de se descrever elementos arquiteturais, incompletude, semântica informal da notação, dentre outras. Como vantagens, foram identificadas a extensibilidade, a possibilidade de modelar a arquitetura sob diferentes visões, o fato de ser uma notação gráfica, dentre outras.

Curiosamente, alguns fatores foram identificados tanto como vantagem, ou ponto forte, quanto como desvantagem, ou fraqueza, como por exemplo, o fato de a UML ser uma linguagem gráfica a torna chamativa aos olhos dos profissionais e facilita o entendimento dos modelos criados, porém, não é adequada para a análise automatizada.

Através dos resultados obtidos pôde-se concluir que a UML não foi inicialmente projetada para descrever arquiteturas, mas pode ser considerada uma ADL de *software*, pois com ela profissionais são capazes de modelar e documentar arquiteturas em diferentes visões e modelar aspectos estáticos e dinâmicos das mesmas, além disso, é uma linguagem visual, amplamente utilizada e apoiada por

ferramentas, extensível e flexível. A linguagem possui pontos fortes e fracos, como todas as outras ADLs, e sua adoção ou não está sujeita ao julgamento da equipe quanto às necessidades de modelagem arquitetural do projeto.

Referências

AVGERIOU, Paris; GUELFI, Nicolas; MEDVIDOVIC, Nenad. Software architecture description and UML. In: **International Conference on the Unified Modeling Language**. Springer, Berlin, Heidelberg, 2004. p. 23-32.

BABAR, Muhammad Ali; ZHU, Liming; JEFFERY, Ross. A framework for classifying and comparing software architecture evaluation methods. In: **Software Engineering Conference, 2004. Proceedings. 2004 Australian**. IEEE, 2004. p. 309-318.

BLAHA, Michael; RUMBAUGH, James. **Modelagem e projetos baseados em objetos com UML 2**. Elsevier, 2006

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Elsevier Brasil, 2006.

GARLAN, David; CHENG, Shang-Wen; KOMPANEK, Andrew J. Reconciling the needs of architectural description with object-modeling notations. **Science of Computer Programming**, v. 44, n. 1, p. 23-49, 2002.

HILLIARD, Rich. Using the UML for architectural description. In: **International Conference on the Unified Modeling Language**. Springer, Berlin, Heidelberg, 1999. p. 32-48.

HOFMEISTER, Christine; NORD, Robert L.; SONI, Dilip. Describing software architecture with UML. In: **Software Architecture**. Springer, Boston, MA, 1999. p. 145-159.

IEEE ARCHITECTURE WORKING GROUP et al. IEEE P1471/D5. 0 Information Technology—Draft Recommended Practice for Architectural Description, August 1999. **Available by request from http://www. pithecanthropus. com/awg**.

IVERS, James et al. Documenting architectural connectors with UML 2. In: **Software Architecture Description & UML Workshop**. 2004. p. 1.

KAUR, Harpreet; SINGH, Pardeep. UML (Unified Modeling Language): standard language for software architecture development. In: **International Symposium on Computing, Communication, and Control**. 2011.

LANGE, Christian FJ; CHAUDRON, Michel RV; MUSKENS, Johan. In practice: UML software architecture and design description. **IEEE software**, v. 23, n. 2, p. 40-46, 2006.

LEVIN, Jenya. Modeling in software architecture. **Ottawa-Carleton Institute for Computer Science, Ottawa**, 2009.

LUCKHAM, David C.. ; VERA, James. An event-based architecture definition language. **IEEE transactions on Software Engineering**, v. 21, n. 9, p. 717-734, 1995.

MALAVOLTA, Ivano et al. What industry needs from architectural languages: A survey. **IEEE Transactions on Software Engineering**, v. 39, n. 6, p. 869-891, 2013.

MORESI, Eduardo et al. Metodologia da pesquisa. **Brasília: Universidade Católica de Brasília**, v. 108, p. 24, 2003.

NAKAGAWA, Elisa Yumi. **Uma contribuição ao projeto arquitetural de ambientes de engenharia de software**. 2006. Tese de Doutorado. Universidade de São Paulo.

NAVARČIK, Matúš. Using UML with OCL as ADL. In: **IIT. SRC 2005: Student Research Conference**. p. 175.

OLIVEIRA, Cap Marcelo Luz Sande E. **Modelagem De Arquitetura De Software Orientada A Aspectos Com Uml 2.0**. 2007. Tese de Doutorado. Instituto Militar De Engenharia.

OZKAYA, Mert; KLOUKINAS, Christos. Are we there yet? Analyzing architecture description languages for formal analysis, usability, and realizability. In: **Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on**. IEEE, 2013. p. 177-184.

PANDEY, R. K. Architectural description languages (adls) vs uml: a review. **ACM SIGSOFT Software Engineering Notes**, v. 35, n. 3, p. 1-5, 2010.

PORFÍRIO, Gustavo. **Revisão da literatura**. 2012.

RIBEIRO, Quelita Araújo Diniz da Silva. **Uma técnica baseada em SysML para modelar a arquitetura de sistemas embarcados de tempo real.** 2017. 85 f. Dissertação (Pós-Graduação em Ciência da Computação) - Universidade Federal de Sergipe, São Cristóvão, SE, 2017.

SHAW, Mary; CLEMENTS, Paul. The golden age of software architecture. **IEEE software**, v. 23, n. 2, p. 31-39, 2006.

SHAW, Mary et al. Abstractions for software architecture and tools to support them. **IEEE transactions on software engineering**, v. 21, n. 4, p. 314-335, 1995.

SOARES NETO, Carlos de Salles. **Descrição Arquitetural da Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação.** 2003. 111 f. Dissertação (Mestrado) - Curso de Pós-graduação em Informática, Pontifícia Universidade Católica, Rio de Janeiro, 2003.

SOMMERVILLE, Ian. **Engenharia de software**. Tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama–. 2011.

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

TERMO DE RESPONSABILIDADE

Eu, **Marina de Oliveira Margarida**, declaro que o texto do trabalho de conclusão de curso intitulado *"AVALIAÇÃO DA APLICABILICABILIDADE DA UML COMO UMA ADL DE SOFTWARE"* é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 11 de julho de 2018

Marina de Oliveira Margarida
Assinatura do aluno

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

DECLARAÇÃO DE CONFORMIDADE

Certifico que o(a) aluno(a) Marina de Oliveira Margarida, autor do trabalho de conclusão de curso intitulado "AVALIAÇÃO DA APLICABILIDADE DA UML COMO UMA ADL DE SOFTWARE" efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

João Monlevade, 20 de Julho de 2018.

Professor (a) Orientador (a)