LEVI MORAES DE SOUZA

UM ESTUDO COMPARATIVO ENTRE MÉTODOS DE SELEÇÃO DE ARQUITETURA DE SOFTWARE BASEADOS EM ATRIBUTOS DE QUALIDADE

São Paulo
2015

LEVI MORAES DE SOUZA

UM ESTUDO COMPARATIVO ENTRE MÉTODOS DE SELEÇÃO DE ARQUITETURA DE SOFTWARE BASEADOS EM ATRIBUTOS DE QUALIDADE

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

São Paulo
2015

LEVI MORAES DE SOUZA

UM ESTUDO COMPARATIVO ENTRE MÉTODOS DE SELEÇÃO DE ARQUITETURA DE SOFTWARE BASEADOS EM ATRIBUTOS DE QUALIDADE

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Prof. Dr[a]. Denise Lazzeri Gastaldo Bombonatti

São Paulo
2015

DEDICATÓRIA

Dedico este trabalho á Deus, fonte de
toda a sabedoria e conhecimento.

AGRADECIMENTOS

À Universidade de São Paulo – USP que oferece educação e ensino de excelente qualidade a mais de um século, por formar bons profissionais e ser uma das maiores alavancas que movem o progresso científico, político e tecnológico no Brasil.

À Escola Politécnica da Universidade de São Paulo – EPUSP que é um berço de conhecimento de ponta com uma infraestrutura notável, formando engenheiros que ajudaram a construír grande parte dos empreendimentos que meus olhos enxergam hoje no Brasil e no mundo.

Ao PECE – Programa de Educação Continuada em Engenharia que abriu as portas para estudantes que anseiam obter uma educação de ponta através dos seus cursos de especialização, em especial o MBA em Tecnologia de Software, por seu excelente corpo docente, que concerteza me ajudou a me tornar uma pessoa melhor e um profissional mais qualificado e preparado para os desafios constantes que são impostos no mundo corporativo.

À Deus fonte de todo ar que eu respiro e de todas as coisas que vivem e que foram construídas, criador dos céus e da terra, sem a tua presença eu não sou nada, com a tua presença o impossível se torna possível na minha vida.

Aos meus pais que são pessoas maravilhosas, sem eles concerteza eu não teria chegado até aqui e faço questão de expressar para eles toda a minha gratidão e amor.

À Prof. Dr. Denise Lazzeri Gastaldo Bombonatti por ter me auxiliado com muito empenho na execução desse trabalho, sua excelente capacidade analítica, sua agradável habilidade em lidar com pessoas e sua paciência em esclarecer detalhes, fazem de você uma excelente professora com distinção de mérito, me sinto honrado por ter sido orientado por você, este trabalho é fruto de todas as suas recomendações e esforços em me auxiliar nesta jornada.

O único lugar onde o sucesso vem
antes do trabalho é no dicionário.
(Albert Einstein)

RESUMO

Diversos arquitetos de software já tiveram que escolher entre um ou outro método para definir, refinar e selecionar uma arquitetura para um sistema de software. O processo de escolha do método mais adequado pode ser direcionado através de um conjunto de questões que possam ressaltar as características destes métodos, fazendo com que os arquitetos de software possam utilizar o método mais adequado para a resolução do seu problema. Este trabalho apresenta um *framework* que permite realizar a comparação entre métodos de seleção de arquiteturas de software definidas com base em atributos de qualidade. O *framework* apresentado é derivado do NIMSAD (*Normative Information Model-based System Analysis and Design*), que oferece informações para uma análise minuciosa de vantagens e desvantagens dos métodos que estão sendo comparados. Este trabalho selecionou três métodos da literatura que serão comparados. Concluiu-se com a aplicação do NIMSAD que as diferenças entre os métodos comparados ficam visíveis em todas as questões formuladas, tal situação, apoia a tomada de decisão, direcionando o avaliador a escolher o método mais adequado após o *framework* ser aplicado.

Palavras chave: método, *framework*, arquitetura, atributos de qualidade.

ABSTRACT

Several software architects have had to choose between one or another method to define, refine and select an architecture for a software system. The process of choosing the most appropriate method can be directed through a set of questions that can highlight the characteristics of these methods, so that software architects can use the most appropriate method for the resolution of your problem. This paper presents a framework that allows for the comparison of software architectures selection methods defined on the basis of quality attributes. The presented framework is derived from NIMSAD (Normative Information Model-based System Analysis and Design), which provides information for trend analysis of advantages and disadvantages of the methods being compared. This work selected three methods of literature that will be compared. Concluded with the application of NIMSAD that the differences between the methods compared are visible on all the questions asked, this situation supports decision making, directing the evaluator to choose the most appropriate method after the framework is applied.

Keywords: method, framework, architecture, quality attributes.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE DIAGRAMAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery,* em português, Associação para computação de máquina.
ADD	*Attribute Driven Design*, em português, projeto orientado ao atributo.
ADL	*Architecture Description Language*, em português, linguagem de descrição de arquietutra.
ALMA	*Architecture Level Modifiability Analysis,* em português, Análise do nível de mutabilidade arquitetural.
AMDAQ	Análise de múltiplas decisões de atributos de qualidade utilizando equivalência hipotética.
ATAM	*Architecture Tradeoff Analysis Method,* em português, Método de análise de vantagens e desvantagens da arquitetura.
CA	Comparação de arquiteturas.
COCOMO II	*Construtive Cost Model II,* em português, Modelo de Custo Construtivo II.
FPA	*Function Point Analysis,* em português, Análise de Ponto de Função.
IEEE	*Institute of Electrical and Electronics Engineers,* em português, Instituto de Engenheiros, Eletricistas e Eletrônicos.
ISO	*International Organization for Standardization,* em português, Organização Internacional de Normalização.
MVC	*Model View Controler*, em português, Modelo, Visão, Controlador.
NIMSAD	*Normative Information Model-based System Analysis and Design,* em português, modelo baseado em informação normativa de análise e projeto de sistemas.
PASA	*Performance Assesment of Software Architecture,* em português, Avaliação do desempenho da arquitetura de software.
SAAM	*Scenario-Based Architecture Analysis,* em português, Análise da arquitetura baseada em cenário.
SEI	*Software Engineering Institute,* em português, Instituto de Engenharia de Software.

SUMÁRIO

1. INTRODUÇÃO

Este capítulo apresenta as motivações, objetivo, as justificativas e a estrutura do trabalho.

1.1 Motivações

Uma das atividades mais importantes contidas no processo de desenvolvimento de software é a definição da arquitetura do sofware mais adequada para um determinado domínio, que atenda interesses e expectativas das partes interessadas. Porém, para possibilitar a identificação da arquitetura mais adequada diversos métodos baseados em atributos de qualidade podem ser utilizados. Neste contexto, algumas perguntas precisam ser respondidas:

- Existe algum *framework* que suporte a seleção de um método para arquitetura de software baseado em atributos de qualidade que seja mais adequado?
- Este *framework* permite a identificação de vantagens e desvantagens entre os métodos analisados?
- Este *framework* pode ser customizado?

Este trabalho se propõe a responder a estes questionamentos apresentando um *framework* de avaliação de métodos derivado do NIMSAD (*Normative Information Model-based System Analysis and Design*).

Ao analisar o *framework* derivado do NIMSAD foi possível entender que ele poderia ajudar as partes interessadas na realização da comparação dos métodos, sendo que, o foco deste trabalho é a comparação de métodos de seleção de arquitetura.

Vale ressaltar que este trabalho é o primeiro passo de um trabalho maior, sendo que o primeiro passo é a escolha do método mais adequado para seleção de uma arquitetura de software com base em atributos de qualidade, depois aplicá-lo em um contexto real de um sistema de indicação de vagas. Este sistema de indicação de vagas é um sistema para dispositivos móveis que realiza indicação de vagas de estacionamento para os motoristas da cidade, e finalmente coletar os benefícios.

1.2 Objetivo

O objetivo deste trabalho é comparar métodos de definição e seleção de arquitetura de sofware com base em atributos de qualidade através de um *framework* que permite realizar a comparação entre métodos, o NIMSAD. Tendo como base um conjunto de componentes e elementos pré-definidos propostos pelo *framework*, serão identificadas as vantagens e desvantagens entre os métodos estudados.

1.3 Justificativas

Derivações do NIMSAD já foram aplicadas em diferentes domínios de avaliação, as evidências podem ser apresentadas com o exemplo apresentado por Babar e Gorton (2004) que aplicaram uma derivação do *framework* NIMSAD para comparar métodos de avaliação de arquitetura de software.

Também Matinlassi (2004) apresenta outra evidência da aplicação de um *framework* derivado do NIMSAD para comparar métodos que verificam a arquitetura de linhas de produtos.

Kitchenham e Babar (2007) apresentam evidência da aplicação de um *framework* derivado do NIMSAD, o *framework* criado por Kitchenham e Babar (2007) apresentam algumas características iguais às do NIMSAD, porém, a customização adotada pelos autores permitiu a criação de um segundo *framework* mais relevante para um domínio específico que herda características do NIMSAD.

Existem outros estudos com *frameworks* semelhantes, que embora não sejam derivados do NIMSAD, utilizam conceitos semelhantes na estruturação do *framework*, por exemplo os estudos apresentados por Dobrica e Niemela (2002) apresentam um *framework* que possibilita a comparação de métodos com base em um conjunto de questões de avaliação semelhantes a estrutura do *framework* NIMSAD.

Por outro lado, este estudo, realiza uma customização do *framework* NIMSAD. Para comparar métodos de seleção de arquitetura de software com base nos atributos de qualidade, o *framework* utilizado na comparação neste trabalho, é apresentado por Babar e Gorton (2004). A aplicação do NIMSAD na comparação de métodos para seleção de arquitetura ainda não foi explorada, inclusive Babar e

Gorton (2004), Matinlassi (2004), Kitchenham e Babar (2007) apresentam o *framework* sendo aplicado em contextos diferentes, portanto, neste trabalho, o *framework* é aplicado para auxiliar a identificar o método de seleção de arquitetura de software mais adequado com base nos elementos definidos no *framework* de avaliação.

Este trabalho apresenta um *framework* que pode auxiliar o avaliador a realizar a comparação entre métodos que apresentam a mesma finalidade, embora os outros autores mencionados anteriormente já utilizaram este *framework* para realizar a comparação de métodos em outros trabalhos, o *framework* apresentado neste trabalho vai realizar a comparação de métodos de seleção de arquitetura de software baseados em atributos de qualidade, ou seja, durante as pesquisas não foi identificado em nenhum dos artigos pesquisados a utilização do NIMSAD na comparação de métodos de seleção de arquitetura, portanto, utilizaremos neste trabalho o uma customização do *framework* com esta finalidade.

1.4 Metodologia

A pesquisa foi realizada em computador particular, a partir da utilização da rede *wireless* do PECE, diversos artigos acadêmicos foram baixados a partir da consulta a sites de pesquisa acadêmica, os principais sites de referência que contém a maior parte dos artigos baixados são:

- IEEE Xplore http://ieeexplore.ieee.org/Xplore/home.jsp
- ACM Digital Library http://dl.acm.org/
- Software Engineering Institute http://www.sei.cmu.edu/
- ScienceDirect http://www.sciencedirect.com/

Alguns *ebooks* foram utilizados na pesquisa, os *ebooks* utilizados foram obtidos no site da Amazon:

- Amazon http://www.amazon.com/

O passo a passo das atividades realizadas para viabilizar o estudo que está sendo apresentado, está descrito a seguir:

1) **Definição do tema de pesquisa e do assunto a ser estudado**. A definição do tema e do assunto do trabalho foi firmada em reuniões de orientação. Identificou-se a necessidade de realizar uma pesquisa nesta área de concentração.

2) **Pesquisa de artigos *online* nas redes do PECE**. A pesquisa foi realizada com computador particular do aluno dentro da biblioteca e laboratório de informática do PECE, a pesquisa de artigos foi realizada através de palavras-chaves que remetem ao tema de estudo em questão.
3) ***Download* dos artigos que apresentam relevância ao tema proposto**. O *download* dos artigos foi realizado com base na aderência apresentada pelo mesmo ao tema de pesquisa proposto, em reuniões semanais de orientação, ficou definido que diversos artigos relacionados ao tema de pesquisa deveriam ser baixados para estudo e a identificação de vantagens e desvantagens entre um e outro foi realizada.
4) **Escolha dos artigos mais relevantes para o tema proposto**. Após o download dos artigos e da leitura dos mesmos, foi realizada uma seleção dos artigos que mais apresentam detalhes relacionados ao tema de pesquisa proposto, ou seja, uma segunda seleção foi realizada para identificar os artigos mais relevantes dentre outros que apresentam aderência em relação ao tema de pesquisa proposto.
5) **Investigação em busca de um *framework* que permita realizar a comparação entre métodos**. Após a identificação dos artigos mais relevantes, que fariam parte do estudo proposto, foi necessário realizar uma busca por um *framework* ou método que permitisse realizar a comparação de outros métodos, vários artigos com exemplos de *frameworks* foram lidos para possibilitar a seleção.
6) **Seleção do *framework* que será objeto de estudo no trabalho**. Após a leitura de alguns artigos que apresentavam informações sobre alguns *frameworks* foi escolhido um *framework* de comparação derivado do NIMSAD. Detectamos que este *framework* poderia ser utilizado no processo de comparação dos métodos que seriam avaliados durante o estudo. A escolha foi decidida em conjunto em reunião de orientação.
7) **Obtenção de informações sobre o *framework* com o próprio autor do *framework***. O autor do *framework* escolhido foi contatado, o objetivo deste contato foi entender realmente se o *framework* pode ser usado para realizar a comparação entre métodos com finalidades distintas ou semelhantes. O contato com o autor foi importante para entender como o método pode ser utilizado e em qual domínio ele pode ser aplicado.

8) **Escolha dos métodos de seleção de arquitetura a serem comparados**. Após realizarmos uma pesquisa sobre métodos de seleção de arquitetura com base nos atributos de qualidade, diversos métodos foram encontrados, porém apenas três foram escolhidos, esta escolha foi baseada na relevância e pertinência dos artigos ao tema do trabalho em questão.
9) **Reunião semanais com a orientadora, para verificar o progresso do trabalho**. Semanalmente eram realizadas reuniões com a orientadora, onde o *status* do progresso do trabalho entrava em discussão. Novas atividades foram propostas por orientadora e aluno durante as reuniões, com o objetivo de acrescentar mais conteúdo relevante à pesquisa em questão.

Esta foi à metodologia adotada durante o processo de construção deste estudo.

1.5 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas e a estrutura do trabalho.

O Capítulo 2 ATRIBUTOS DE QUALIDADE E ARQUITETURA DE SOFTWARE apresenta um conjunto de conceitos que definem o que é atributo de qualidade, arquitetura de software, requisito não-funcional, atributo de qualidade, padrões arquiteturais e visões arquiteturais, o capítulo 2 é a base de pesquisa para o desenvolvimento do trabalho.

O Capítulo 3 MÉTODOS PARA SELEÇÃO DE ARQUITETURA DE SOFTWARE BASEADOS EM ATRIBUTOS DE QUALIDADE apresenta detalhadamente os métodos que serão objetos de comparação, os procedimentos necessários para executar cada método são apresentados passo a passo.

O Capítulo 4 COMPARAÇÃO DE MÉTODOS PARA SELEÇÃO DE ARQUITETURA DE SOFTWARE, UTILIZANDO UM *FRAMEWORK* DE AVALIAÇÃO descreve um *framework* que permite realizar uma comparação entre os métodos que são apresentados no capítulo 3, após a aplicação prática do *framework* é possível que

as vantagens e as desvantagens identificadas durante a comparação sejam analisadas.

O Capítulo 5 CONSIDERAÇÕES FINAIS descreve os pontos positivos e as limitações detectadas após a aplicação do *fremework* de comparação, as contribuições do trabalho e os trabalhos futuros são apresentados.

REFERÊNCIAS relaciona as bibliográfias pesquisadas e que são a base da construção do trabalho.

2. ATRIBUTOS DE QUALIDADE E ARQUITETURA DE SOFTWARE

Este capítulo tem o objetivo de apresentar os conceitos relacionados aos atributos de qualidade, suas classificações bem como os conceitos de arquitetura de software, estilos e padrões arquiteturais e seus benefícios.

2.1 Atributos de Qualidade

Alguns autores, muitas vezes, utilizam termos diferentes ao se referir aos atributos de qualidade. Chung et al. (2000), Sommerville (2011), Pressman (2011) utilizam o termo requisito não-funcional. A norma ISO/IEC 25010 (2010) utiliza os termos característica e subcaracterística de qualidade. Bass; Clements e Kazman (2012) utilizam o termo atributo de qualidade. Nestes casos, os termos atributos de qualidade, requisitos não-funcionais, características e subcaracterísticas de qualidade são utilizados como sinônimos.

Neste trabalho optou-se pelo uso do termo atributo de qualidade conforme definição de Bass; Clements e Kazman (2012) por apresentar uma abordagem voltada a arquitetura de software, contexto deste trabalho. Os atributos de qualidade presentes em uma arquitetura de software são definidos por partes interessadas.

Conforme a ISO/IEC 25010 (2010) as partes interessadas que determinam se um software é de qualidade ou não com base no comportamento do software.

Um conjunto de individuos ou organização que tem interesse real no sistema, podem ser definidos como partes interessadas. (ISO/IEC 42010, 2011)

São as partes interessadas que tomam as decisões que vão impactar no projeto da arquitetura, inclusive as decisões das partes interessadas afetam e definem os requisitos não-funcionais.

Segundo Bass; Clements e Kazman (2012) os requisitos não-funcionais do sistema que podem ser testados ou medidos podem ser denominados de atributos de qualidade, um exemplo de atributo de qualidade pode ser **desempenho**, **disponibilidade** e etc. A partir dos objetivos das partes interessadas, os atributos de qualidade são definidos para o sistema.

Os atributos de qualidade devem fazer parte de um sistema de software, porém, cada sistema de software é criado para atender alguma necessidade, ou

seja, nem todos os atributos de qualidade serão necessários em um sistema de software, porém qualquer sistema de software contém algum atributo de qualidade.

Chung et al. (2000) ressalta que a insatisfação das partes interessadas e o alto volume de custos oriundos de um mau planejamento do projeto, podem ser sinais de alerta avisando que os requisitos não-funcionais foram mal definidos, portanto um software de má qualidade pode ser entregue se os requisitos não-funcionais não estiverem bem definidos, por isso é necessária uma averiguação nos requisitos não-funcionais durante o desenvolvimento do sistema de software.

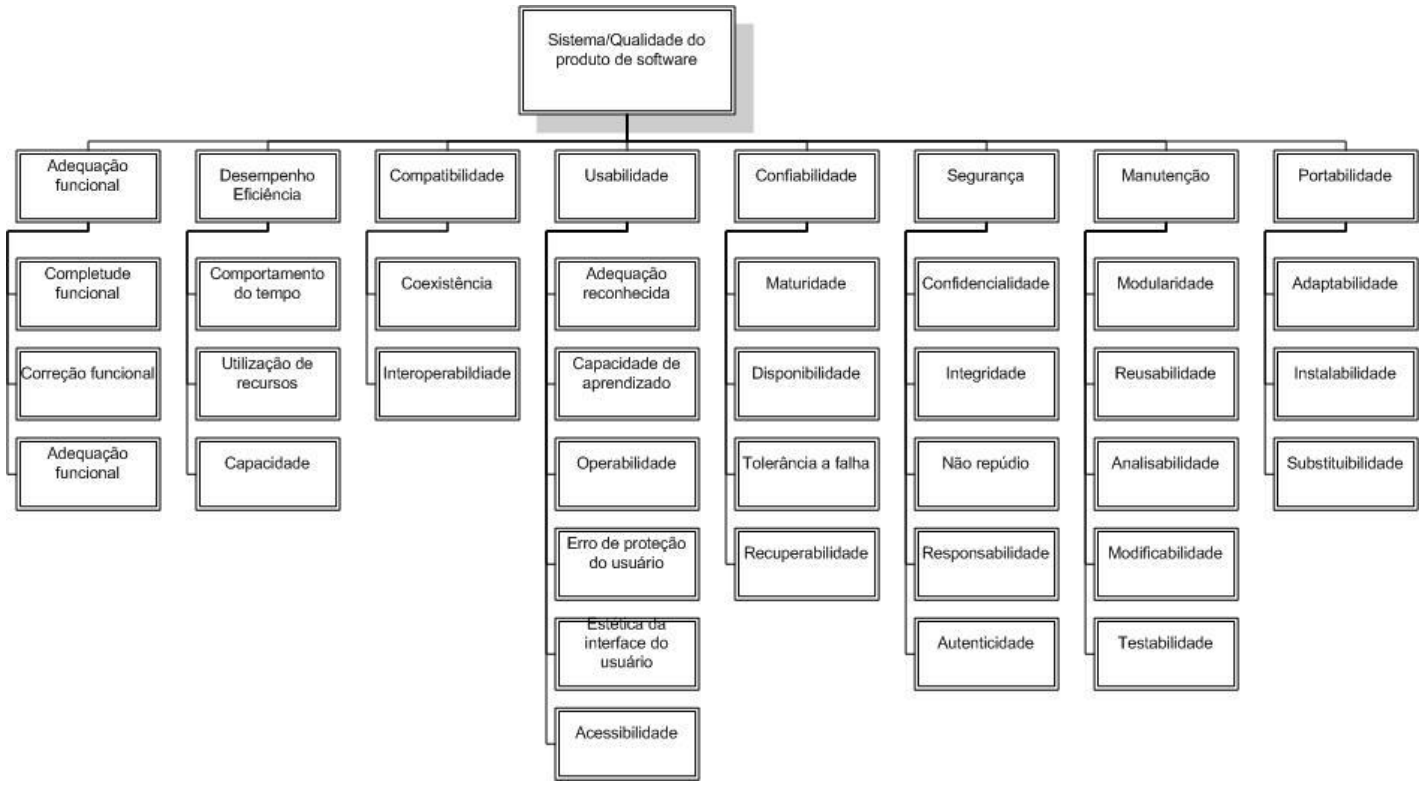
Sommerville (2011) afirma que um requisito não-funcional é oriundo de conjunto de limitações que são impostas ao sistema de software, estas limitações estão relacionadas a normas, políticas, decisões da compania, interesse das partes interessadas e necessidades esperadas, de tal forma os requisitos não-funcionais se aplicam em todos os elementos do sistema de software.

2.2 Classificações dos Atributos de Qualidade

A ISO/IEC 25010 (2010) define um padrão de classificação das características de qualidade de um sistema ou software. Este padrão apresenta os atributos de qualidade que possívelmente podem ser requiridos em um sistema de software.

A Figura 1 lista as características de qualidade relacionadas com o modelo de qualidade do produto apresentado na norma ISO/IEC 25010 (2010):

Figura 1 – Modelo de Qualidade do Produto



Fonte: ISO/IEC 25010 (2010, p.4, tradução nossa, adaptada por nós)

As características de qualidade mencionadas na Figura 1 são descritas da seguinte maneira:

> ***Adequação funcional****:* grau em que um produto ou sistema fornece funções que correspondam às necessidades explícitas e implícitas, quando utilizado sob condições especificadas;
> ***Eficiência de desempenho****:* desempenho em relação à quantidade dos recursos utilizados sob condições estabelecidas;
> ***Compatibilidade****:* grau em que um produto, sistema de software ou componente pode trocar informações com outros produtos, sistemas de software ou componentes, e/ou realizar suas funções necessárias, enquanto compartilha o mesmo ambiente de hardware ou software;
> ***Usabilidade****:* grau ao qual um produto ou sistema de software pode ser usado por usuários especificos para atingir metas especificadas com eficácia, eficiência e satisfação em um contexto de uso especificado;
> ***Confiabilidade****:* grau em que um sistema de software, o produto ou componente executa as funções especificadas sob condições especificadas para um período de tempo especificado;
> ***Segurança****:* grau ao qual um produto ou sistema de software protege informações e dados, de modo que as pessoas ou outros produtos ou sistemas de software tenham o grau de acesso aos dados apropriado para os seus tipos e níveis de autorização;
> ***Manutenibilidade****:* grau de eficácia e eficiência com que um produto ou sistema de software pode ser modificado pela equipe responsável pela manutenção;
> ***Portabilidade****:* grau de eficácia e eficiência com que um sistema de software, o produto ou componente pode ser transferido a partir de um

> hardware, software ou outro ambiente operacional ou para outro uso. (ISO/IEC 25010, p.10-15, tradução nossa).

Conforme Bass; Clements e Kazman (2012) é possível que o arquiteto de software tenha uma lista dos atributos de qualidade mais comumente utilizados à sua disposição. Esta lista existe e pode ser encontrada na ISO/IEC 25010 (2010) a lista mencionada pode ser vista na Figura 1, ela exibe a classificação dos atributos de qualidade. Bass; Clements e Kazman (2012) afirmam que essa lista pode servir de apoio para identificação de atributos de qualidade. Nesta direção os autores ressaltam que uma lista de atributos de qualidade pode servir para vários propósitos, um dos propósitos é a coleta de requisitos, onde a lista poderia servir como uma base para verificar se nenhum requisito foi esquecido ou negligenciado.

Neste contexto uma lista de atributos de qualidade serve também para estabelecer medidas, ou seja, o arquiteto de software pode consultar a lista de atributos de qualidade para verificar quais são os possíveis atributos de qualidade que o sistema de software a ser construído pode incorporar e consequentemente escolher os que se aplicam ao sistema de software, com a finalidade de verificar o quanto um sistema de software atende às necessidades das partes interessadas.

Porém, vale ressaltar que não existe uma lista completa de atributos de qualidade. (BASS; CLEMENTS e KAZMAN, 2012). Nenhuma lista de atributos de qualidade é completa, e, este é um incoveniente no mapeamento dos atributos de qualidade, pois estes atributos podem variar conforme o interesse das partes interessadas, ou seja, uma parte interessada pode solicitar um atributo de qualidade que não está descrito na norma ISO/IEC 25010 (2010). Ainda neste contexto Bass; Clements e Kazman (2012) afirmam que o atributo de qualidade “baixa capacidade”, em inglês, *lowability*, é um atributo de qualidade difícil de encontrar em qualquer lista ou norma de atributos de qualidade.

De acordo com Chung et al. (2000) os atributos de qualidade podem ser subjetivos, ou seja, eles podem ser interpretados de formas diferentes por pessoas diferentes, portanto a definição de um atributo de qualidade pode ser abrangente ou especifica. A importância de um atributo de qualidade depende do sistema de software em análise.

Chung et al. (2000) criaram uma lista ampla dos possíveis requisitos não-funcionais que podem ser utilizados como referência, essa lista pode ser visualizada na Tabela 1.

Tabela 1 - Lista de Requisitos não-funcionais

Atributos de Qualidade		
Acessibilidade	Aditividade	Acoplamento
Acessibilidade	Agilidade	Ajustabilidade
Adaptabilidade	Capacidade de evolução	Auditabilidade
Analisabilidade de impacto	Clareza	Capacidade
Capacidade	Compatibilidade	Capacidade de sobrevivência
Capacidade de aprendizado	Comunalidade	Completude
Capacidade de execução	Conceitualidade	Concisão
Coesividade	Confiabilidade	Consistência
Compreensibilidade	Configurabilidade	Coordenação de custo
Compreensibilidade	Consistência externa	Custo da análise do domínio
Confidencialidade	Consistência interna	Custo da comunicação
Custo de análise de risco	Controlabilidade	Custo de acompanhamento de projetos
Custo de inspeção	Convivialidade	Custo de execução
Custo de integração de componentes	Custo	Custo de hardware
Custo de manutenção	Custo da reforma	Custo de planejamento
Custo de prototipagem	Custo de desenvolvimento	Degradação do serviço
Custo de reengenharia	Custo de funcionamento	Desempenho de armazenamento secundário
Dependabilidade	Custo de software	Desempenho durante o período de pico
Desempenho do espaço de dados	Decompatibilidade	Desempenho no espaço de código
Desempenho médio	Desempenho	Desempenho no tempo
Disponibilidade	Desempenho de espaço de armazenamento	Elasticidade
Distributividade	Desempenho de memória principal	Especificidade
Espaço delimitado	Desempenho uniforme	Exatidão
Estabilidade	Diversidade	Formalidade
Extensibilidade	Eficiência	Informatividade
Generalidade	Envolvibilidade	Integridade

Atributos de Qualidade		
Interoperabilidade	Escalabilidade	Intuição
Manutenibilidade	Espaço de desempenho	Manutenibilidade
Melhoria da capacidade	Estabilidade do projeto	Maturidade
Movimento frequente	Fiança	Mobilidade
Operacionalidade	Fidelização de clientes	Modularidade
Planejamento de tempo	Flexibilidade	Naturalidade
Pontualidade	Independência	Período de desempenho
Precisão	Mensurabilidade	Personalização
Presteza	Modularidade	Portabilidade
Produtividade	Observabilidade	Rastreabilidade
Proteção	Orientação	Reconfigurabilidade
Responsividade	Padronização	Recuperabilidade
Segurança	Plasticidade	Reutilização
Substituibilidade	Prestação de contas	Robustez
Suportabilidade	Previsibilidade	Similaridade
Susceptibilidade	Recuperabilidade	Subjetividade
Tempo de análise do domínio	Replicabilidade	Tempo de coordenação
Tempo de avaliação do cliente	Sensibilidade	Tempo de desenvolvimento
Tempo de comunicação	Sustentabilidade	Tempo de gestão de processos
Tempo de teste	Taxa de transferência	Tempo de produção de software
Treinabilidade	Tempo de análise de risco	Tempo de prototipagem
Usabilidade	Tempo de inspeção	Tempo de resposta
Validade	Tempo de integração de componentes	Testabilidade
Variabilidade	Tempo de manutenção	Tolerância a falhas
Verificabilidade	Tolerância	Transparência
Viabilidade	Transmissibilidade	Uniformidade
Visibilidade	Versatilidade	

Fonte: Chung et al. (2000, p.160, tradução nossa)

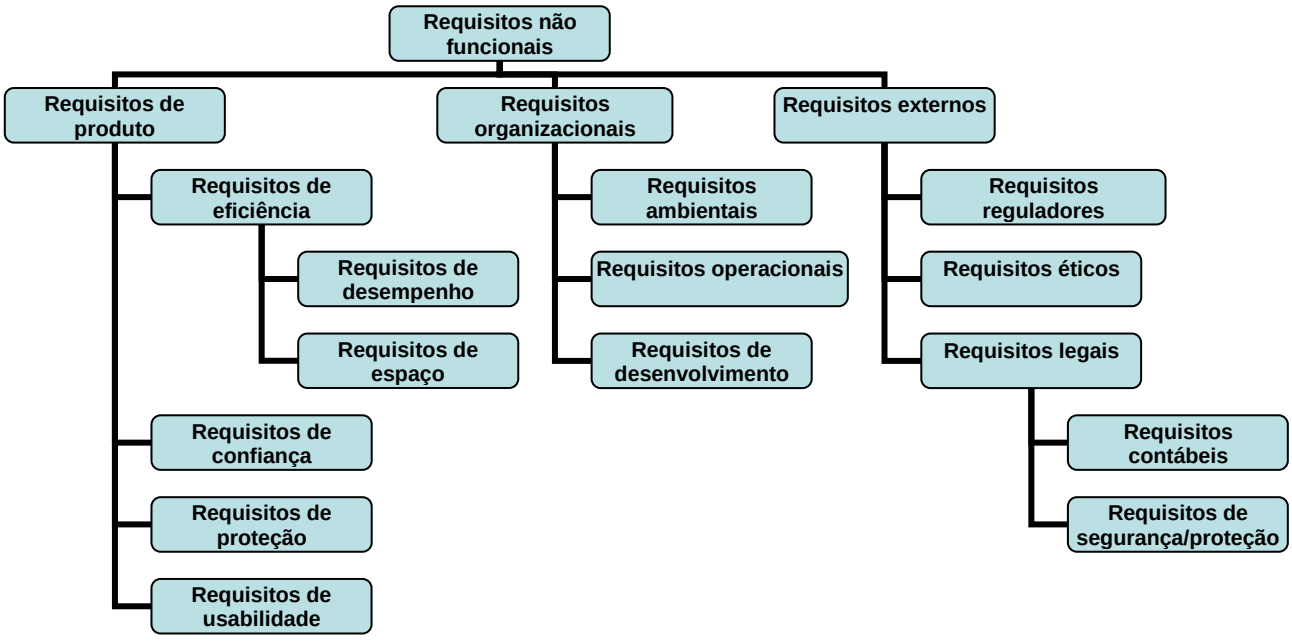
Ao analisar-se a Tabela 1 percebe-se a existência de uma lista ampla de requisitos não-funcionais sugeridos por Chung et al. (2000). Esta lista não possui classificação por categorias, pois o intuito é apenas apresentar possíveis tipos de requisitos não-funcionais que podem ser úteis na definição de um sistema de software. Sommerville (2011) ressalta que a origem dos requisitos não-funcionais, muitas vezes está relacionada a fatores externos como mudança de Lei, alteração de normas, mudança no planejamento estratégico da empresa, ou premência do usuário.

Ainda Sommerville (2011) relata que os requisitos não-funcionais tem origem nas características requisitadas para um software, e que existem diferentes tipos de requisitos:

- Requisitos de produto: São requisitos que afetam o desempenho do software frente a um estímulo.
- Requisitos organizacionais: A partir de decisões políticas e das normas da companhia, estes requisitos são definidos.
- Requisitos externos: Requisitos que dependem da Lei dou de outros fatores externos como normas ou regulamentos.

A Figura 2 ilustra os tipos de requisitos não-funcionais conforme a categorização de Sommerville (2011, p.61)*:*

Figura 2 - Tipos de requisitos não-funcionais



Fonte: Sommerville (2011, p.61, adaptada por nós)

De acordo com Sommerville (2011) a identificação de detalhes passíveis de testes, podem ser detectados após a análise dos requisitos não-funcionais.

Somerville (2011) apresenta na Tabela 2 os requisitos não-funcionais como **propriedade**, pois os requisitos não-funcionais apresentados na Tabela 2 podem ser testados, por este motivo foram denominados de propriedades por Sommerville (2011), embora para este trabalho o termo definido é atributo de qualidade.

A Figura 2 apresentada por Sommerville (2011) difere-se da Tabela 2, pois os requisitos não-funcionais apresentados na Figura 1 não apresentam qual requisito não-funcional pode ser testado, entretanto a Tabela 2 apresenta os requisitos não-funcionais que podem ser testados.

Tabela 2 - Métricas para especificar requisitos não-funcionais

Propriedade	Medida
Velocidade	Transações processadas/segundo. Tempo de resposta de usuário/evento. Tempo de atualização de tela
Tamanho	Megabytes. Número de chips de memória ROM.
Facilidade de uso	Tempo de treinamento. Número de *frames* de ajuda.
Confiabilidade	Tempo médio para falha. Probabilidade de indisponibilidade. Taxa de ocorrência de falhas. Disponibilidade.
Robustez	Tempo de reinício após falha. Percentual de eventos que causam falhas. Probabilidade de corrupção de dados em caso de falha.
Portabilidade	Percentual de declarações dependentes do sistema-alvo. Número de sistemas-alvo

Fonte: Sommerville (2011, p. 63)

Ao analisar-se a Tabela 2 percebe-se que os requisitos não-funcionais são descritos de forma a serem testados, as métricas apresentadas servem de base para o teste dos requisitos não-funcionais.

De acordo com Sommerville (2011) estas métricas podem servir para identificar o que as partes interessadas esperam do sistema de software, ou seja,

após a métrica é possível obter condições para o teste e consequentemente validar os resultados com as expectativas das partes interessadas. Porém vale ressaltar que as métricas apontadas por Sommerville (2011) estão relacionadas com alguns dos atributos de qualidade apresentados por Chung et al. (2000), na mesma direção a ISO/IEC 25010 (2010) apresenta no modelo de qualidade do produto exibido na Figura 1, atributos de qualidade relacionados aos apresentados por Sommerville (2011), ou seja, os atributos de qualidade apresentados são métricas que servem para avaliar o comportamento do sistema, a partir destas métricas é possível avaliar a qualidade do sistema de software.

2.3 Arquitetura de Software

De acordo com Rozanski e Woods (2012) um dos maiores problemas quando o assunto em questão é arquitetura de sistemas de software é a definição do uso do termo arquitetura.

O termo arquitetura pode ser utilizado em diferentes contextos, entretanto, Rozanski e Woods (2012) ressaltam que ao se analisar um sistema de software, as verificações cruciais a serem realizadas referem-se a segregação de responsabilidades entre as funções que compõe o software, a habilidade do sistema se comunicar com outros sistemas em ambiente externo e o nível de relacionamento entre as funções que compõe a arquitetura do software.

Uma definição amplamente aceita para o termo arquitetura é encontrada na ISO/IEC 42010 (2011, p. 2, tradução nossa), que define arquitetura de um sistema de software como um “conjunto de conceitos ou propriedades de um sistema embutidos nos elementos do sistema de software, no ambiente do sistema de software, nos relacionamentos, e nos princípios de sua concepção e evolução”.

A necessidade das partes interessadas são as diretrizes de uma arquitetura, pois a partir das necessidades das partes interessadas é possível avaliar se o ambiente que o sistema de software está operando é viável e consequentemente se os elementos que compõe a arquitetura mantém um relacionamento entre eles, estas representações idealizam uma arquitetura de software. (PRESSMAN, 2011).

De acordo com Bass; Clements e Kazman (2012) as relações entre os elementos de um sistema e os comportamentos destes elementos, compõe a

arquitetura do software. É fato que existe relacionamento entre elementos de um sistema de software, por mais sucinto que o sistema possa parecer ele representa uma arquitetura.

De acordo com Sommerville (2011), os requisitos não-funcionais impactam de forma considerável a arquitetura de um sistema de software, a relação entre os elementos permite identificar a dependência existente entre os componentes da arquitetura. Nesta mesma direção Sommerville (2011) enfatiza que o projeto da arquitetura se concentra em proporcionar um entendimento aceitável da organização e da estrutura do sistema de software. Neste contexto Rozanski e Woods (2012) enfatizam que durante o processo de desenvolvimento do software é possível identificar onde a arquitetura do software se situa, ela é construída após a fase de especificação e antes da fase de implementação.

2.3.1 Visões Arquiteturais

Segundo a ISO/IEC 42010 (2011) uma visão arquitetural apresenta a arquitetura de software a partir da concepção das exigências particulares embutidas na solução sistêmica.

De acordo com a ISO/IEC 42010 (2011) um ponto de vista arquitetural defini a forma de interpretar as visões arquiteturais a partir das exigências específicas de um sistema de software, o ponto de vista arquitetural procura apresentar uma determinada necessidade do sistema de software sob uma perspectiva.

Conforme a ISO/IEC 42010 (2011) os termos ponto de vista arquitetural e visão arquitetural podem ser erroneamente utilizados como sinônimos, pois, existem diferenças fundamentais entre esses dois conceitos.

“Um ponto de vista é a maneira de olhar para o sistema de software, enquanto que uma visão arquitetural é o resultado da aplicação do ponto de vista sob uma necessidade particular do sistema de software” (ISO/IEC 42010, 2011, p.20, tradução nossa).

Contudo, Rozanski e Woods (2012) ressaltam que as propriedades mais notáveis que compõe a arquitetura, tem a possibilidade de serem expostas através de uma visão arquitetural, de tal forma que seja possível atender as expectativas das partes interessadas.

O documento que contém a descrição da arquitetura pode conter um conjunto conexo de pontos de vistas arquiteturais documentados, ou seja, exemplos ou padrões podem ser documentados, o intuito é disponibilizar um documento detalhado com os pontos de vistas arquiteturais, este tipo de documentação auxilia na criação de uma visão arquitetural. (ROZANSKI; WOODS, 2012).

De acordo com com Bass, Clements e Kazman (2012) o principal conceito associado a documentação da arquitetura de um sistema de software, se baseia na visão arquitetural, ou seja, a documentação da arquitetura de um sistema de software deve apresentar a descrição detalhada das visões apresentadas e definidas pelos arquitetos de software, de tal maneira que essas visões possam ser compreendidas pelas partes interessadas.

Para este trabalho o conceito adotado para conceituar visão arquitetural é o mencionado pela ISO/IEC 42010 (2011), por ser uma norma conhecida e também por apresentar diversas definições amplamente aceitas para os termos relacionados a engenharia de software.

Vale ressaltar que as visões arquiteturais são totalmente dependentes das metas das partes interessadas, ou seja, não é possível estabelecer uma lista de visões que podem ser utilizadas em qualquer arquitetura de software, pois as visões arquiteturais estão relacionadas aos objetivos individuais das partes interessadas, de tal forma, diferentes visões respaldam diferentes objetivos e diferentes usuários. (BASS; CLEMENTS; KAZMAN, 2012).

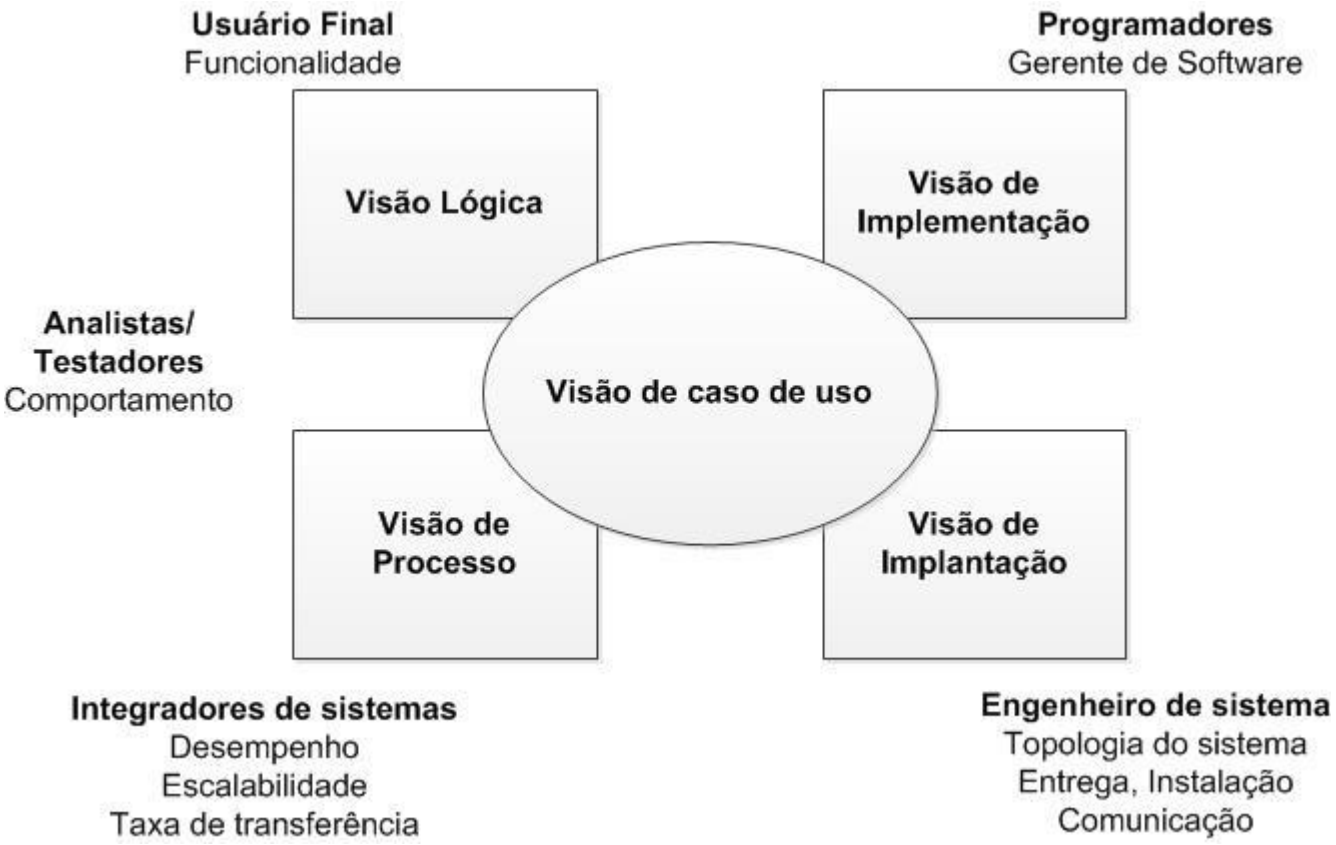
De acordo com Bass, Clements e Kazman (2012) as visões arquiteturais apresentam diferentes níveis de atributos de qualidade, de tal forma que, os atributos de qualidade são o principal guia para as partes interessadas decidirem quais visões devem ser documentadas.

Entretanto, Kruchten (2003) enfatiza que o alvo principal de uma visão arquitetural refere-se ao entendimento da arquitetura de um sistema de software sob uma determinada perspectiva.

Kruchten (2003) ressalta que uma arquitetura deve ser exposta de modo a todas as partes interessadas compreenderem o que a arquitetura representa. Pois, diferentes partes interessadas, podem ter diferentes conceitos a respeito da arquitetura apresentada, por isso é importante que a arquitetura possa ser criada de modo a possibilitar o entendimento comum.

Kruchten (2003) sugere uma representação arquitetural com 5 visões, essa abordagem é amplamente conhecida. A Figura 3 apresenta o modelo de visão de arquitetura 4+1 (quatro mais um):

Figura 3 – O Modelo de visão arquitetural 4+1



Fonte: Kruchten (2003, cap.5, seção – "Representação da arquitetura", tradução nossa)

Kruchten (2003, cap.5, seção-"Representação da arquitetura") descreve cada uma das 5 visões arquiteturais apresentadas no modelo de visão 4+1 da seguinte forma:

> **Visão lógica:** aborda os requisitos funcionais do sistema de software, em outras palavras, o que o sistema de software deve entregar para seus usuários finais. É uma abstração do modelo de projeto e identifica os principais pacotes de projeto, subsistemas e classes.
> **Visão de implementação:** descreve a organização dos módulos estáticos de software (código fonte, arquivos de dados, componentes, executáveis e outros artefatos de acompanhamento). Ela aborda as questões de facilidade de desenvolvimento, gerenciamento de ativos de software, reutilização, subcontratação, e componentes de prateleira.
> **Visão de processo:** aborda os aspectos simultâneos do sistema em tempo de execução de tarefas, *threads*, ou processos, bem como suas interações. Esta visão aborda questões como a concorrência, paralelismo, sistema de software ligando e desligando, tolerância a falhas e distribuição de objetos.

> **Visão de entrega:** descreve como os vários arquivos executáveis e outros componentes de tempo de execução são mapeados para as plataformas subjacentes ou nós de computação. Aborda questões como a implantação, instalação e desempenho.
> **Visão de casos de uso:** esta visão desempenha um papel especial no que diz respeito à arquitetura, pois contempla cenários-chave ou casos de uso. Inicialmente, estes são usados para impulsionar a descoberta e projeto da arquitetura nas fases de iniciação e de elaboração, e serão utilizados em fases posteriores para validar as diferentes visões. Os cenários agem para ilustrar no documento de arquitetura de software como as demais visões irão trabalhar. (KRUCHTEN, 2003, cap.5, seção-“Representação da arquitetura”, tradução nossa).

Os **outros artefatos de acompanhamento** descritos por Kruchten (2003) na definição da **visão de implementação** pode ser um documento que contém a descrição da arquitetura do software, a partir deste artefato é possível compreender o objetivo do sistema de software que está sendo construído conforme o ponto de vista da parte interessada no projeto da arquitetura.

2.4 Benefícios da Arquitetura de Software

De acordo com Kruchten (2003) a arquitetura de soffware tem um papel importante na tomada de decisões.

Kruchten (2003) apresenta algumas características da arquitetura de software que podem influenciar na tomada de decisões, Krutchten (2003) relata que a arquitetura tem um impacto no sistema de software, influenciando na estrutura do sistema de software. A arquitetura possibilita uma visão dos relacionamentos entre os elementos que compõe o sistema de software, a arquitetura de software permite compreender o ambiente de execução do sistema de software, também, os atributos de qualidade fazem parte da arquitetura de software, ou seja, os atributos de funcionalidade, desempenho, segurança entre outros, podem fazer parte da arquitetura, de tal maneira, que os atributos de qualidade se tornam essenciais na definição da arquitetura.

Contudo Bass; Clements e Kazman (2012) enfatizam a importância da arquitetura sob uma perspectiva técnica e as principais razões para utilizar a arquitetura, seguem as mais importantes razões para o uso da arquitetura:

1. Os atributos de qualidade guiam a construção da arquitetura.

2. As decisões arquiteturais devem ter um impacto profundo na arquitetura de modo a possibilitar mudanças futuras na solução arquitetural.
3. Se a arquitetura de software estiver bem construída existe uma grande possibilidade de se obter exito na implementação.
4. Todas as decisões arquiteturais devem estar em um documento com a descrição da arquitetura, tal documento, permite um entendimento uniforme do que está sendo construído, ou seja, as partes interessadas envolvidas no projeto arquitetural devem compreender de forma unanime o projeto arquitetural que está sendo construído.
5. Um projeto de arquitetura aprovado pelas partes interessadas, pode servir como exemplo em outros projetos, ou seja, novos projetos não precisariam começar do zero a construção da arquitetura, a partir de um modelo pronto testado e validado é possível inciar o projeto com mais agilidade.
6. Quando a representação da arquitetura está bem definida, o entendimento do que deve ser implementado se torna mais simples para os desenvolvedores.

De acordo com Pressman (2011) uma arquitetura de software é importante, ela serve como uma planta de uma casa, ou seja, a arquitetura permite visualizar o que será executado em um contexto geral, ela apresenta uma visão geral do que será feito e garante que o entendimento do projeto da arquitetura seja compreendido de forma correta.

2.5 Considerações do Capítulo

Esta seção apresenta uma síntese dos principais termos apresentados no capítulo 2.

Ao analisar as menções dos autores Chung et al. (2000) e Bass; Clements e Kazman (2012) percebe-se que ambos expressam pontos de vista semelhantes no que diz respeito à abrangência e dificuldade na identificação dos atributos de qualidade. Embora existam normas e padrões conhecidos, com definições a respeito dos atributos de qualidade mais comumente utilizados, nota-se que existe uma

complexidade considerável no mapeamento dos possíveis atributos de qualidade que podem existir em um sistema de software. Percebe-se também que não é tarefa fácil classificar os atributos de qualidade, pois conforme Chung et al. (2000), a identificação de um atributo de qualidade depende do sistema de software em questão, e, portanto a necessidade de qualidade requerida em um sistema de software deve ser avaliada caso a caso, com base nas expectativas das partes interessadas. Sommerville (2011) enfatiza que a origem de um atributo de qualidade surge a partir das necessidades das partes interessadas, e também dos interesses de uma organização.

Neste trabalho optou-se pelo uso do termo atributo de qualidade referindo-se se a um requisito não-funcional que pode ser definido, testado e/ou calculado. Para identificar uma arquitetura de sofware com base nos atributos de qualidade, é necessário identificar os atributos de qualidade aplicáveis com base nas expectativas de cada parte interessada. Nesta direção Bass; Clements e Kazman (2011) ressaltam que um atributo de qualidade deve apresentar caracterísiticas próprias e ser testável, portanto ficou definido que o termo “atributo de qualidade” será utilizado para referir-se aos requisitos não-funcionais.

1. O que é atributo de qualidade?

Atributo de qualidade é um termo cujo conceito refere-se a uma propriedade que pode ser testada ou medida, para fins de exemplo, o atributo de qualidade de desempenho pode ser utilizado, por exemplo, quando uma parte interessada desejar que o desempenho de uma determinada função do sistema seja alto, onde o tempo ideal para o sistema responder a um determinado evento seja de 2 segundos. Desta forma, o atributo de qualidade pode ser medido com base na expectativa da parte interessada.

2. O que é arquitetura de software?

Arquitetura de software refere-se a representação da estrutura do sofware, ou seja, apresenta os papéis de cada elemento que compõe o sofware, dentro de um determinado ambiente e os relacionamentos entre estes elementos. Vale ressaltar que a arquitetura de software surge a partir de várias necessidades oriundas das

partes interessadas, ou seja, a partir do momento que uma necessidade é identificada, essa necessidade deverá ser suprida no sistema de software, e, a arquitetura do software, deve apresentar uma solução que atendas às necessidades das partes interessadas.

3. O que são visões arquiteturais?

Visão arquitetural refere-se aos aspectos da solução que o sistema de software apresenta a partir dos interesses definidos pelas partes interessadas, ou seja, a visão arquitetural existe a partir do momento que é possível viabilizar uma solução arquitetônica para o sistema de software. A partir das necessidades especificas das partes interessadas, a visão arquitetural deve representar uma solução a nível sistêmico para uma necessidade específica, com base nos interesses das partes interessadas.

4. O que é ponto de vista arquitetural?

Ponto de vista arquitetural refere-se à necessidade de alguma parte interessada que deve ser representada na arquitetura do software. Em um projeto de arquitetura de software, várias partes interessadas podem ter diferentes interesses, isso significa que existirão diferentes pontos de vistas em relação à arquitetura do software. Neste contexto, a arquitetura do software deve representar os diferentes pontos de vistas. Entretanto, os pontos de vistas devem ser documentados e descritos, pois, a partir da descrição do ponto de vista arquitetural é possível proporcionar um entendimento comum para as partes interessadas. Após os pontos de vistas arquiteturais serem definidos com base nas necessidades das partes interessadas, é possível estabelecer as visões arquiteturais.

3. MÉTODOS PARA SELEÇÃO DE ARQUITETURA DE SOFTWARE BASEADOS EM ATRIBUTOS DE QUALIDADE

Neste capítulo serão apresentados três métodos que permitem definir e/ou selecionar uma arquitetura de software a partir dos atributos de qualidade exigidos pelas partes interessadas. Os métodos apresentados auxiliam na identificação dos atributos de qualidade que fazem parte das exigências das partes interessadas e consequentemente disponibilizam arquiteturas de software candidatas, que devem ser verificadas pelo arquiteto de software.

3.1 Método ADD – *ATTRIBUTE DRIVEN DESIGN* (Projeto Orientado ao Atributo)

Conforme Bass; Clements e Kazman (2012, p.316, tradução nossa) o ADD é um método interativo que em cada interação auxilia o arquiteto a realizar as seguintes tarefas:

- Escolher uma parte do sistema para projetar.
- Decompor todos os requisitos arquiteturais significativos.
- Criar e testar um projeto para a parte escolhida.

“A saída do ADD não é uma arquitetura completa em todos os detalhes, mais uma arquitetura em que as principais abordagens do projeto foram selecionadas e controladas. Produzindo uma arquitetura funcional de forma antecipada e rápida”. (BASS; CLEMENTS; KAZMAN, 2012, p.316, tradução nossa).

3.1.1 Entradas do ADD

Bass; Clements e Kazman (2012) enfatizam que as entradas do ADD são os requisitos funcionais, atributos de qualidade e as restrições do projeto, estas três entradas devem ser conhecidas antes do inicio do processo de construção da arquitetura. Bass; Clements e Kazman (2012) enfatizam que o aumento do volume de requisitos que são identificados em um projeto, está relacionado ao aumento do conhecimento das partes interessadas, pois à medida que o projeto vai sendo construído e as regras de negócio amadurecendo, surgem novas ideias e mais conhecimentos, dessa forma novas solicitações, tornam-se requisitos e mudanças podem acontecer a todo o instante.

Bass; Clements e Kazman (2012) relatam que existe uma maneira de elencar os requisitos que servirão como base para projetar a arquitetura, porém a seleção dos requisitos se baseia no ASR (*architecturally significant requirements)* ou “requisitos arquiteturais significativos”, ou seja, a partir de uma lista de requisitos definidos para o sistema, deve ser extraída outra lista com os requisitos mais significativos, denominados de ASRs. A partir dos ASRs é possível iniciar o ADD, porém algumas restrições existem ao aplicar este conceito.

“Se o conjunto de ASR mudar durante a execução do ADD, então o projeto da arquitetura precisa ser revisto como um todo.” (BASS; CLEMENTS; KAZMAN, 2012, p.317, tradução nossa).

Bass; Clements e Kazman (2012) enfatizam que deve ser dada prioridade para os atributos de qualidade no levantamento dos ASRs, os autores relatam que para aplicar o ADD é necessário compreender o contexto do que está sendo realizado em torno do projeto da arquitetura.

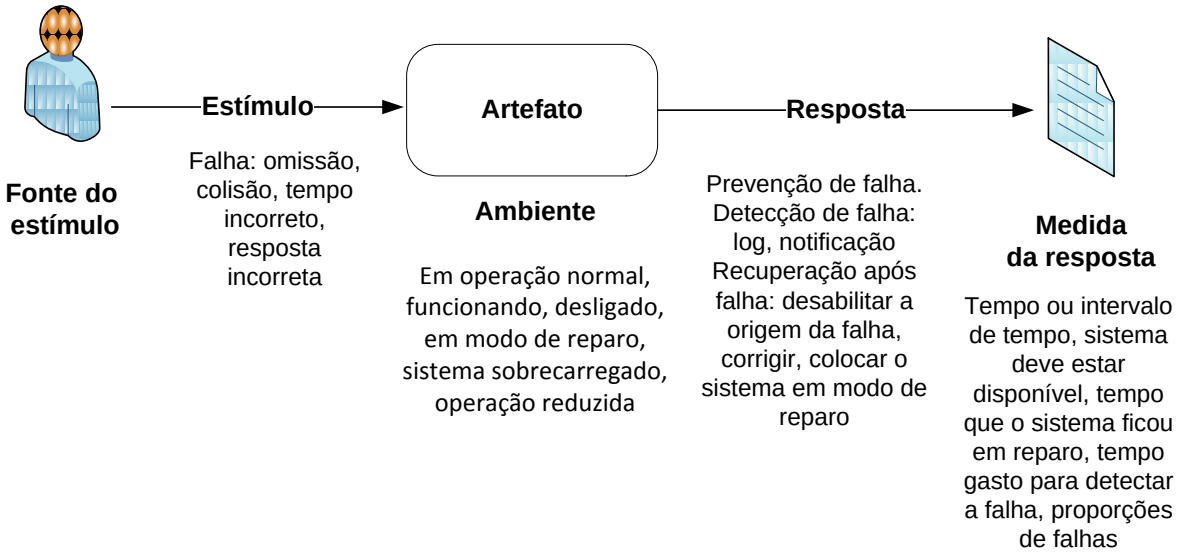
3.1.1.1 Criação de Cenários

Bass; Clements e Kazman (2012) apresentam a técnica de criação de cenários, ou seja, a classificação dos atributos de qualidade é realizada através de cenários, o cenário é composto por seis passos:

1. **Origem do estímulo**. É alguma entidade (um humano, sistema de computador, ou outro ator que gera um estímulo.
2. **Estímulo**. O estímulo é uma condição que requer uma resposta assim que ela chega no sistema.
3. **Ambiente**. O estimulo ocorre dentro de certas condições que dependem do ambiente. O sistema pode estar sobrecarregado ou em operação normal, ou algum outro estado relevante.
4. **Artefato**. Algum artefato é estimulado. Este artefato pode ser, um conjunto de sistemas, o sistema inteiro, ou parte do sistema.
5. **Resposta**. A resposta é a atividade empreendida como resultado da chegada do estímulo.
6. **Medir a resposta**. Quando a resposta ocorre, ela deve ser medida de alguma forma, de maneira que o requisito possa ser testado. (BASS; CLEMENTS; KAZMAN, 2012, p.69, tradução nossa):

A partir da execução dos cenários é possível identificar os atributos de qualidade que serão necessários para o sistema, o refinamento nos atributos de qualidade surge após a aplicação das táticas, o Diagrama 1 apresenta um exemplo da execução de um cenário:

Diagrama 1 - Um cenário genérico para o atributo de qualidade disponibilidade



Fonte: Bass; Clements e Kazman (2012, p.70, tradução nossa, adaptado por nós).

A partir do Diagrama 1 é possível verificar as partes em que um cenário é dividido, ao seguir estas etapas, é possível aferir a consistência do atributo de qualidade de disponibilidade com a necessidade apresentada pelas partes interessadas.

3.1.2 Saídas do ADD

De acordo com Bass; Clements e Kazman (2012) a saída do ADD, produz um rascunho de uma visão arquitetural, os autores relatam que as visões que serão produzidas, dependem do projeto que está sendo construído, tais projetos passam por uma seleção e enfim após a construção do projeto um esboço da visão é apresentado. As visões arquiteturais apresentam os elementos e as relações entre esses elementos, contudo as informações trocadas entre os elementos da arquitetura precisam ser documentadas, pois essas informações são a base para o entendimento das visões arquiteturais. As visões arquiteturais quando documentadas, permitem que o desfecho do método apresente uma arquitetura consistente. (BASS; CLEMENTS; KAZMAN, 2012).

3.1.3 Os Passos do ADD

Conforme Bass; Clements e Kazman (2012, p.318, tradução nossa) o ADD é um método composto por 5 passos:

1. Escolha um elemento do sistema para projetar.
2. Identifique os ASRs (requisitos arquiteturalmente significativos) para o elemento escolhido.
3. Gere uma solução de projeto para o elemento escolhido.
4. Armazene os requisitos restantes e selecione a entrada para a próxima iteração.
5. Repita os passos 1- 4 até que todos os ASRs tenham sido satisfeitos.

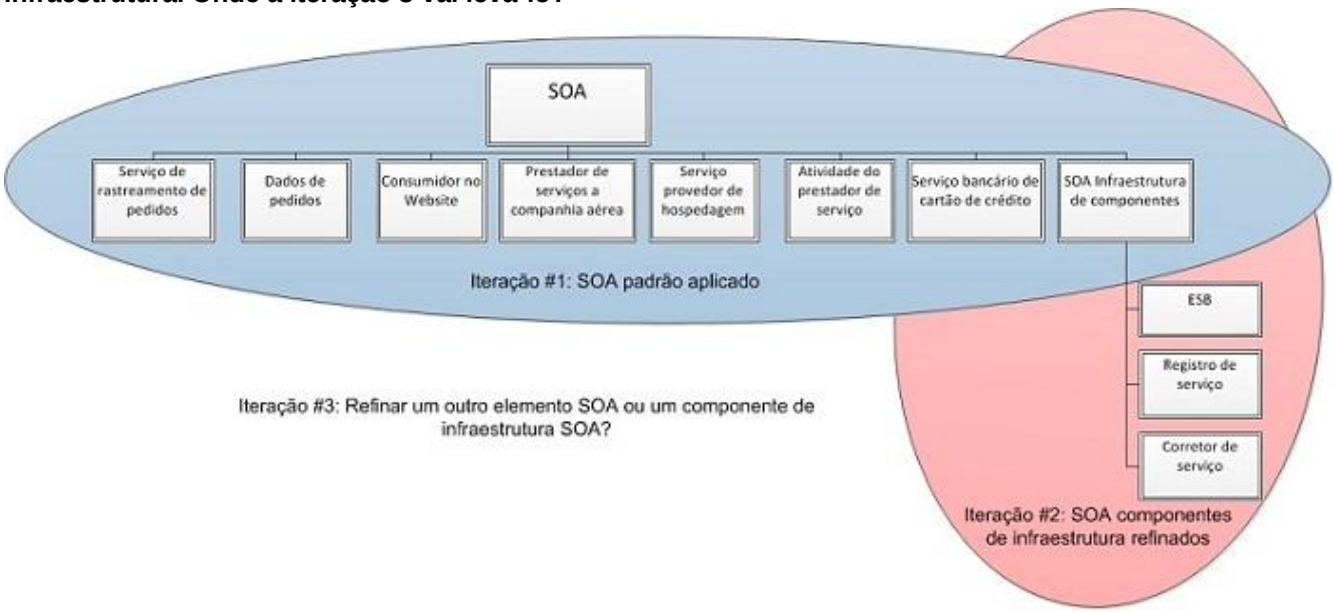
3.1.4 Passo 1: Escolha um Elemento do Sistema Para Projetar

De acordo com Bass; Clements e Kazman (2012) o ADD tem inicio a partir da identificação do elemento que será modelado no projeto arquitetural. Em príncipio a abordagem apresentada pelo ADD para gerar um esboço arquitetural enfatiza que o elemento que deve passar pelos passos do ADD, é o sistema como um todo, ou seja, cada elemento do sistema deve ser modelado para que seja possível definir um esboço de uma visão arquitetural do sistema.

Através de várias iterações através dos passos do ADD as decisões de projeto são influenciadas pelos elementos identificados e escolhidos durante esta iteração, pois os elementos escolhidos contém os ASRs e um dos argumentos das decisões de projeto para a escolha de um elemento é perceber se o elemento escolhido satisfaz os ASRs contidos em si próprio, ou seja, os elementos que satisfazem os ASRs serão candidatos no passo 1 do ADD, estes elementos estarão disponíveis para escolha. (BASS; CLEMENTS; KAZMAN, 2012).

Bass; Clements e Kazman (2012) enfatizam que a composição do sistema como um todo, já é representada de forma inicial através do agrupamento dos elementos que integram a primeira iteração do ADD, este conjunto de elementos estão relacionados a um padrão arquitetural e o refinamento dos elementos que compõe o padrão arquitetural, ou seja, a Figura 4 apresenta as iterações do ADD em um sistema hipotético que utiliza o padrão SOA (Arquitetura orientada a serviço):

Figura 4 - Iteração 1 aplicado o padrão SOA. Iteração 2 aperfeiçoou os componentes de infraestrutura. Onde a iteração 3 vai levá-lo?



Fonte: Bass; Clements e Kazman (2012, p.319, adaptada por nós)

Conforme Bass; Clements e Kazman (2012) a primeira iteração refere-se à aplicação do padrão SOA (elipse azul) no sistema como um todo, é possível perceber que os elementos filhos a partir da hierarquia apresentada na Figura 4 referem-se a elementos que satisfazem as características da aplicação do padrão SOA, por exemplo: provedores de serviços e os componentes da infraestrutura SOA. Na próxima iteração (elipse rosa) um dos elementos filhos do SOA é decomposto, no caso a infraestrutura de componentes é decomposta. Na próxima iteração é possível escolher entre refinar outro elemento SOA (elipse azul) ou um dos componentes filhos da infraestrutura SOA (elipse rosa). Percebe-se que os elementos apresentados na Figura 4 são exibidos em uma árvore de decomposição.

Bass; Clements e Kazman (2012) enfatizam que nem sempre a iteração do ADD vai tratar o sistema como um todo, pois, em alguns momentos pode ser necessário manter e melhorar um sistema que já está em funcionamento, ou seja, o projeto pode surgir a partir da evolução de um sistema já existente, desta forma, não será necessário criar o sistema a partir do zero, em muitos casos a empresa oferece um sistema que já apresenta funções semelhantes ao que será construído, neste contexto a aplicação do primeiro passo do ADD, não se aplica no sistema como um todo, porém se aplica ao elemento que foi reaproveitado do sistema que já existia e que será evoluído.

Nesta direção Bass; Clements e Kazman (2012) enfatizam que geralmente a primeira iteração do ADD refere-se a todo o sistema, porém é importante observar que o passo 1 do ADD afirma que pelo menos um dos elementos deve ser escolhido e adicionado ao projeto.

Bass; Clements e Kazman (2012) ressaltam duas importantes estratégias de refinamento que ajudam a prosseguir de forma correta com o ADD: “primeira amplitude” e “primeira profundidade”.

“Primeira amplitude significa que todos os elementos de segundo nível são projetados antes de qualquer elemento de terceiro nível, e assim por diante”. (BASS; CLEMENTS; KAZMAN, 2012, p.319, tradução nossa).

“Primeira profundidade significa que uma cadeia descendente está completa antes de iniciar uma segunda cadeia descendente.” (BASS; CLEMENTS; KAZMAN, 2012, p.319, tradução nossa).

A abordagem “primeira amplitude” pode ser compreendida na Figura 4 a partir da elipse azul, a elipse azul contém os elementos de segundo nível alinhados lado a lado, portanto ao aplicar essa abordagem todos os elementos de segundo nível contidos na elipse azul da Figura 4 deveriam ser projetados antes dos elementos filhos da infraestrutura de componentes que estão na elipse rosa da Figura 4.

Entretanto a abordagem “primeira profundidade” pode ser mais bem compreendida a partir da elipse rosa, onde um determinado elemento de segundo nível é escolhido e pode ser decomposto em diversos outros elementos filhos, até chegar ao último elemento da hierarquia, quando a decomposição do elemento chegar ao fim, então é possível retornar aos elementos de segundo nível contidos na elipse azul, escolher um deles e decompor até o último da hierarquia novamente. (BASS; CLEMENTS; KAZMAN, 2012).

Bass; Clements e Kazman (2012) recomendam a estratégia de “primeira amplitude”, pois diversos grupos de trabalhos, podem receber as atividades a serem executadas de forma mais rápida, outra vantagem da estratégia de “primeira amplitude”, refere-se aos elementos que estão correferidos na mesma categoria, ou seja, o relacionamento entre estes elementos são considerados na abordagem de primeira amplitude.

3.1.5 Passo 2: Identifique os ASRs Para o Elemento Escolhido

A arquitetura é impactada de forma profunda pelos ASRs, de tal maneira, que a ausência do ASR, gera mudanças significativas na arquitetura. Antes de identificar os ASRs é necessário identificar os atributos de qualidade que fazem parte do escopo do projeto da arquitetura, ou seja, é preciso definir quais serão os atributos de qualidade que farão parte do escopo do projeto da arquitetura que será construída. Para identificar os atributos de qualidade para o sistema existe uma técnica denominada identificação de táticas arquiteturais. Estas táticas referem-se a uma série de atividades definidas, que permitem identificar os atributos de qualidade que farão parte da arquitetura do software. (BASS; CLEMENTS; KAZMAN, 2012).

> Uma tática é uma decisão de projeto que influencia no resultado esperado de um atributo de qualidade, as táticas afetam diretamente a resposta do sistema para algum estímulo. Táticas transferem portabilidade para um projeto, alto desempenho para outro, e integrabilidade para um terceiro. (BASS; CLEMENTS; KAZMAN, 2012, p.70, tradução nossa).

Existe um conjunto de táticas que podem ser utilizadas para definir um atributo de qualidade, porém cabe ao arquiteto de software selecionar a tática mais adequada para refinar um atributo de qualidade, entretanto a seleção da tática deve ser baseada em uma análise dos pontos positivos e negativos de um determinado atributo de qualidade. Para um dado atributo de qualidade, o arquiteto de software deverá verificar qual a tática mais adequada para refinar este atributo de qualidade, neste contexto, o custo para o desenvolvimento deve ser levado em consideração na escolha da tática. (BASS; CLEMENTS; KAZMAN, 2012).

Após os atributos de qualidade serem definidos através das táticas, a seleção dos ASRs do sistema de software, pode ser realizada.

3.1.6 Passo 3: Gerar uma Solução de Projeto Para o Elemento Escolhido

O passo 3 é o passo principal do ADD, pois nele o elemento escolhido é projetado para ser refinado e consequentemente testado. Nesta fase da iteração já existe o elemento escolhido e os ASRs para o elemento escolhido, ou seja, existe uma lista de requisitos definidos para o elemento escolhido, contudo, com base nos requisitos definidos para o elemento escolhido, um projeto deverá ser criado para atender os requisitos exigidos para o elemento selecionado. Entretanto existem

técnicas que permitem identificar um projeto candidato para um determinado elemento. (BASS;CLEMENTS;KAZMAN, 2012).

A seleção da solução de projeto arquitetural para o elemento escolhido é geralmente baseada em um padrão arquitetural, porém a escolha do padrão arquitetural deve ser orientada pelas táticas. As táticas permitem a exploração dos pontos fortes e fracos dos atributos de qualidade, com base no contexto do projeto a ser construído. Ao fim da execução da tática uma lista de verificação é criada, sendo que esta lista de verificação contém detalhes do atributo de qualidade que auxiliam na escolha da solução de projeto arquitetural para o elemento escolhido. A lista de verificação auxilia nas decisões de projeto para um determinado atributo de qualidade. (BASS;CLEMENTS;KAZMAN, 2012).

Bass; Clements e Kazman (2012, p.73, tradução nossa) apresentam a lista de verificação de forma categorizada:

1. Alocação de responsabilidades
2. Modelo de coordenação
3. Modelo de dados
4. Gerenciamento de recursos
5. Mapeamento entre os elementos arquitetônicos
6. Ligação entre os tempos de decisão
7. Escolha da tecnologia

Estas sete categorias de decisões de projeto aplicadas, encaminham para a melhor escolha de uma visão arquitetural baseada em atributos de qualidade. Vale ressaltar que a aplicação da lista de verificação depende de uma definição prévia das táticas apresentadas no passo 2, na seção “3.1.5 - Passo 2: Identifique os ASRs Para o Elemento Escolhido”. Esta lista de verificação ajuda o arquiteto na verificação da incorporação dos requisitos prioritários, incluindo os atributos de qualidade. Ao escolher uma solução de projeto arquitetural candidata, outras exigências e necessidades podem ser atendidas, através do projeto escolhido, ou seja, é possível que o projeto escolhido seja semelhante a outros projetos já utilizados em uma empresa ou laboratório, então, não é recomendável que seja construído um projeto a partir do início para resolver um problema já conhecido, onde existem formas de resolver o problema definidas anteriormente. (BASS;CLEMENTS;KAZMAN, 2012).

Bass; Clements e Kazman (2012) enfatizam que a solução de projeto arquitetural candidata pode incorporar um padrão arquitetural, ou seja, uma solução comum para um determinado problema, nesta direção é possível perceber que um projeto candidato pode ajudar a resolver uma série de ASRs de uma só vez, pois nele podem estar inseridas soluções aplicáveis em outros projetos e que podem ser reusadas. Os autores enfatizam que se uma solução de projeto arquitetural candidata conseguir atender vários ASRs de uma vez, ele por si só é mais vantajosa.

3.1.7 Passo 4: Verificar e Refinar Requisitos e Gerar Subsídios Para a Próxima Iteração

Bass; Clements e Kazman (2012) enfatizam que nem todos os requisitos previstos podem ser atendidos de forma adequada, por este motivo um teste deve ser realizado nos requisitos com seus atributos de qualidade.

O elemento escolhido no passo 1 da iteração do ADD, deve ser analisado e validado no passo 4 do ADD, é nesta etapa onde é possível realizar a verificação e validação dos elementos definidos no passo 1. Após o teste ser realizado, um dos possíveis resultados é um caminho de retorno, ou seja, é possível que a solução apresentada no passo 3 não atenda todos os ASRs propostos, desta forma o passo 4 do ADD identifica quais os requisitos e atributos de qualidade que não foram satisfeitos e estabelece um trajeto que permite identificar a origem da inconsistência. Desta forma, surge a necessidade do projeto ser revisado, com o intuito de verificar requisitos faltantes e seus impactos para a solução arquitetural. (BASS; CLEMENTS; KAZMAN, 2012).

A Tabela 3 sumariza os tipos de problemas e ações recomendadas para cada caso:

Tabela 3 - Ações recomendadas para problemas com as hipóteses atuais.

Tipo de ASR não encontrado	Ação recomendada
1. Atributo de qualidade requerido	Deve-se considerar a aplicação de (mais) táticas para melhorar o projeto no que se refere a atributo de qualidade. Para cada tática candidata, questione: • Esta tática vai melhorar o comportamento do atributo de qualidade do atual projeto de forma

Tipo de ASR não encontrado	Ação recomendada
	adequada? • Esta tática deve ser utilizada em conjunto com outra tática? • Quais são os pontos fortes e fracos da aplicação desta tática?
2. Responsabilidade funcional	Adicionar responsabilidades tanto para módulos existentes ou para módulos recém-criados: • Atribuir a responsabilidade a um módulo que contém responsabilidades similares; • Quebrar um módulo em partes quando é demasiado complexo; • Atribuir a responsabilidade a um módulo que contém responsabilidades similares com características de atributos de qualidade similares – por exemplo, comportamento temporal similar, requisitos de segurança similares, ou requisitos de disponibilidade similares.
3. Restrição	Modificar o projeto ou mitigar a restrição: • Modificar o projeto para aceitar a restrição; • Mitigar a restrição.

Fonte: Bass; Clements e Kazman (2012, p.322, tradução nossa, adaptada por nós)

Bass; Clements e Kazman (2012) enfatizam que no passo 4 é possível identificar os requisitos que não foram satisfeitos no projeto, ou seja, os atributos de qualidade, requisitos funcionais e as restrições, que não foram identificadas no projeto, precisam ser identificadas e inclusas no elemento que foi modelado, pois desta forma é possível chegar a um refinamento mais consistente do elemento escolhido, onde todos os ASRs definidos possam ser satisfeitos.

3.1.8 Passo 5: Repita os Passos 1 á 4 até Concluir

Bass; Clements e Kazman (2012) relatam que deve estar claro para as partes interessadas que os requisitos foram atendidos, o arquiteto deve verificar se todos os ASRs foram inseridos nas iterações do ADD, desta forma, o ADD é finalizado. O ADD só termina quando um esboço da arquitetura está disponível, ou seja, após as iterações nos passos do ADD deve ser possível identificar uma visão arquitetural para o sistema em análise.

Nesta direção Bass; Clements e Kazman (2012) expressam que a decisão de iniciar a implementação com base no esboço criado, depende do nível de confiança que o arquiteto possui em relação à equipe de desenvolvimento, pois o esboço pode estar claro o suficiente para o arquiteto, entretanto, pode não estar claro para a equipe de desenvolvimento. Caso o esboço da arquitetura esteja claro para a equipe de desenvolvimento, a implementação pode ser iniciada, porém o arquiteto precisa avaliar se realmente o entendimento está claro e compreendido pelas partes interessadas. Contudo, caso o esboço arquitetural não esteja claro para a equipe de desenvolvimento, o arquiteto precisará refinar a arquitetura, e, este refinamento pode ficar claro após mais um nível de detalhamento, ou após vários níveis de detalhamento. A clareza da solução vai depender do arquiteto e da equipe de desenvolvimento.

3.2 Método de Análise de Múltiplas Decisões de Atributos de Qualidade Utilizando Equivalência Hipotética

O método apresentado por Zayaraz e Thambidurai (2005) é derivado do método de análise de múltiplas decisões de atributos de qualidade utilizando equivalência hipotética, que é apresentado por See (2002, 2004).

Zayaraz e Thambidurai (2005) customizam o método original apresentado por See (2002, 2004).

De acordo com Zayaraz e Thambidurai (2005, p.167, tradução nossa) ”a arquitetura de um sistema de software suporta requisitos funcionais e seus atributos de qualidade”.

Zayaraz e Thambidurai (2005) enfatizam que a origem dos requisitos em um sistema de software é baseada nas necessidades das partes interessadas no sistema, pois existem diferentes necessidades que precisam ser atendidas e devem ser contempladas na arquitetura de um sistema de software.

Nesta direção Zayaraz e Thambidurai (2005) enfatizam que as expectativas das partes interessadas são distintas, contudo, a arquitetura de software deve contemplar e satisfazer as necessidades de todas as partes interessadas. Esta não é uma tarefa simples, pois em um projeto, diferentes partes interessadas possuem diferentes papéis, por exemplo, o engenheiro de sistema, pode esperar uma solução arquitetural diferente do analista de negócio, da mesma forma que o desenvolvedor

do sistema de software pode ter uma expectativa diferente do gerente da área comercial e assim por diante.

Zayaraz e Thambidurai (2005) propõem um método de análise de múltiplas decisões de atributos qualidade utilizando equivalência hipotética. Este método detalha os passos necessários para selecionar uma arquitetura de um sistema de software, com base em seus atributos de qualidade definidos pelas partes interessadas. Um dos pontos relevantes do método é a atribuição de pesos para os atributos de qualidade. Estes pesos, quando calculados de forma correta, conforme cálculos previamente definidos, permitem identificar uma arquitetura de software candidata com maior capacidade de atender os atributos de qualidade definidos. A atribuição dos pesos é baseada nas decisões das partes interessadas.

3.2.1 Proposta do *Framework* e Resultados Experimentais

Zayaraz e Thambidurai (2005, p.168, tradução nossa) apresentam a seguir os passos necessários para executar o método:

1. Identificação dos atributos de qualidade das partes interessadas.
2. Definir (fixar) um limite aceitável para cada atributo de qualidade.
3. Normalização.
4. Identificação da força de preferência.
5. Determinação do peso de preferência.
6. Conversão dos valores dos atributos de qualidade para arquiteturas candidatas.
7. Cálculo das pontuações cumulativas.
8. Seleção da arquitetura.

3.2.1.1 Classificação das Partes Interessadas

Zayaraz e Thambidurai (2005) relatam que as decisões em um projeto de software, são baseadas em um conjunto de exigências das partes interessadas, e estas exigências precisam ser acatadas no momento da escolha da arquitetura, porém é possível que um determinado projeto de software contenha muitas partes interessadas. Com base neste cenário o método de análise de múltiplas decisões de

atributos de qualidade utilizando equivalência hipotética tem a finalidade de identificar os principais requisitos a serem atendidos pela arquitetura a partir de um conjunto amostral pertencente às partes interessadas.

De acordo com Zayaraz e Thambidurai (2005) a amostragem é realizada através da identificação das partes interessadas que apresentam peculiaridades correspondentes, estas peculiaridades são identificadas com base na escolha dos atributos de qualidade mais relevantes para as partes interessadas, contudo é necessário que os papéis das partes interessadas sejam claros. A Tabela 4 apresentada na seção 3.2.1.2 exibe a divisão que é realizada por papéis.

3.2.1.2 Identificação dos Atributos de Qualidade por Parte Interessada

Zayaraz e Thambidurai (2005) apresentam um exemplo com três grupos de partes interessadas a serem consideradas: gerente, técnico e usuário. A Tabela 4 apresenta um exemplo de um grupo de partes interessadas com diferentes interesses relacionados aos atributos de qualidade.

Tabela 4 - Atributos de qualidade por grupo.

Grupo	Atributos de qualidade		
Gerente	Custo	Tamanho da equipe	Tempo de desenvolvimento
Técnico	Manutenibilidade	Confiabilidade	Tempo de resposta
Usuário	Funcionalidade	Usabilidade	Compreensibilidade

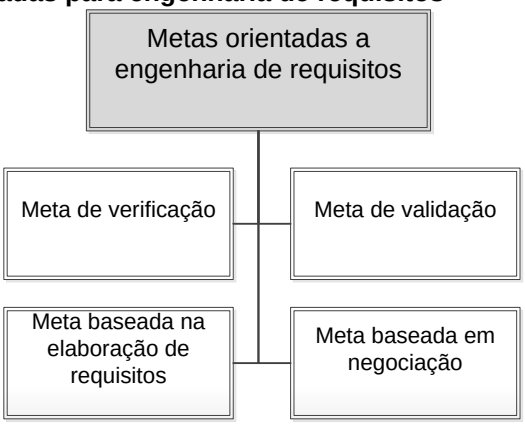
Fonte: Zayaraz e Thambidurai (2005, p.168, tradução nossa, adaptada por nós).

Zayaraz e Thambidurai (2005) apresentam um exemplo de partes interessadas e atributos de qualidade que são representados na Tabela 4, a partir do exemplo na Tabela 4 é possível perceber que diferentes partes interessadas podem apresentar diferentes requisitos.

Zayaraz e Thambidurai (2005) mencionam uma técnica que pode ser utilizada para identificar os atributos de qualidade exigidos pelas partes interessadas, a técnica sugerida é a meta orientada para engenharia de requisitos.

A Figura 5 apresenta metas que podem ser guiadas pela engenharia de requisitos, a Figura 5 foi criada com base nas metas apresentadas por Lamsweerde (2001), as metas apresentadas na Figura 5 também contém sub metas, porém somente as metas gerais estão sendo apresentadas.

Figura 5 - Metas orientadas para engenharia de requisitos



Fonte: Lamsweerde (2001, criada por nós, tradução nossa).

De acordo com Lamsweerde (2001) a Figura 5 pode ser descrita da seguinte forma:

- Meta de verificação: uma verificação é realizada nos requisitos com o intuito de constatar se todos os requisitos que foram definidos podem satisfazer as metas definidas pelas partes interessadas.
- Meta de validação: Cenários são criados para validar as metas que foram definidas pelas partes interessadas, um conjunto de hipóteses é estabelecido para validar as metas.
- Meta baseada na elaboração de requisito: Esta meta verifica se existe alguma anormalidade ou conflito entre as metas que foram definidas pelas partes interessadas (requisitos que foram especificados). O documento que contém os requisitos é analisado e as situações que apresentarem ambiguidade ou inconsistência são revisadas.

- Meta baseada na negociação: refere-se as regras de negócios (requisitos) que geram conflitos e que precisam ser resolvidos através de um consenso das partes interessadas.

Conforme Lamsweerde (2001) a partir de um conjunto de metas é possível identificar o que as partes interessadas esperam do sistema, quais são os atributos de qualidade esperados, quais são as funções que o sistema deve executar e estas necessidades são identificadas como metas. Vale ressaltar que cada meta apresentada na Figura 5 contém uma técnica específica que permite viabilizar a eficácia da meta, ou seja, para cada meta existe um procedimento a ser realizado, tal procedimento permite que a meta seja alcançada.

3.2.1.3 Definição de um Limite Aceitável Para Cada Atributo de Qualidade

Zayaraz e Thambidurai (2005) enfatizam que os atributos de qualidade precisam ter limites estabelecidos para serem utilizados no método apresentado, ou seja, é necessário medir o atributo de qualidade, o intuito é permitir que o atributo de qualidade seja testado conforme os limites (medidas) definidos e expostos as partes interessadas. O objetivo é que o limite (medida) definido para o atributo seja aceito pelas partes interessadas.

Zayaraz e Thambidurai (2005, p.168, tradução nossa) afirmam que “um exemplo de limite aceitável para o atributo de qualidade compreensibilidade para o grupo de usuários apresentado na Tabela 9, seria um limite de 2-8 horas de aprendizado do sistema”. Estas horas representam o limite mínimo e máximo respectivamente definido para o usuário apreender a utilizar o sistema por si próprio.

3.2.1.4 Normalização

Zayaraz e Thambidurai (2005) enfatizam que quando vários atributos de qualidade são apresentados no projeto, distintas formas de medir um atributo de qualidade podem ser aplicadas a estes atributos de qualidade, ou seja, cada atributo de qualidade pode expressar um limite (medida), porém quando o limite é definido, recomenda-se que não seja mais alterado.

Nesta direção Zayaraz e Thambidurai (2005) recomendam que os limites sejam definidos com valores máximo e mínimo, durante a normalização. Um

exemplo de normalização que poderia ser aplicado refere-se ao atributo de qualidade compreensibilidade. Foi mencionado na seção “3.2.1.3 - Definição de um Limite Aceitável Para Cada Atributo de Qualidade”, que o limite para o atributo de qualidade poderia ser de 2 a 8 horas de aprendizado. Ao se aplicar a normalização no limite informado, o resultado seria “0=2” e “8=100”, ou seja, 2 horas corresponde ao valor mínimo do limite na normalização, zero, e, 8 horas representa o valor máximo do limite na normalização, cem. Desta forma, zero aplica-se ao valor mais baixo e cem se aplica ao valor mais alto do limite.

3.2.1.5 Identificação da Preferência Mais Forte

Zayaraz e Thambidurai (2005) enfatizam que uma escala linear de preferência não pode ser utilizada como base para estabelecer as diretrizes definidas pelas partes interessadas sobre as medidas dos atributos de qualidade.

“Escala linear é aquela que possui passo e degrau constantes. Na escala linear estabelece-se uma correspondência entre a unidade de comprimento na escala e o valor da grandeza representada”. (INSTITUTO TECNOLÓGICO DA AERONÁUTICA, entre 1996 e 2015).

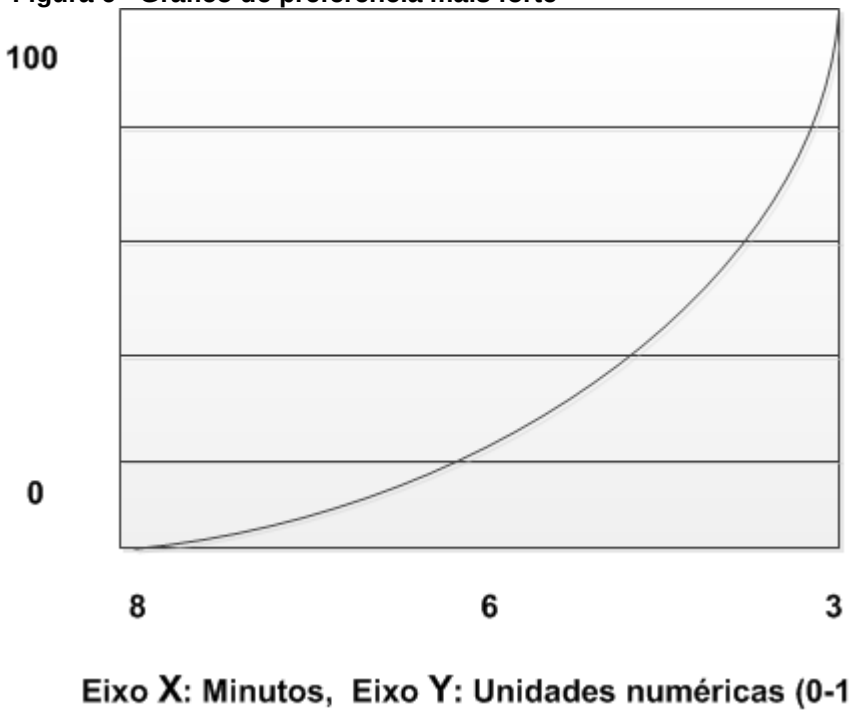
Contudo, a escala não linear pode ser compreendida da seguinte maneira, por exemplo, se 85%, do grupo técnico, apresentado na Tabela 9, preferem diminuir o tempo de resposta de 10 segundos para 8 segundos e 15% preferem diminuir o tempo de resposta de 5 segundos para 3 segundos, o resultado final não pode ser esperado de uma escala linear de preferência, pois a escala linear apresenta os dados ou informações numéricas em sequência sem atribuir prioridade ou peso aos dados que estão sendo exibidos, ou seja, um exemplo poderia ser uma régua de 10 centímetros, onde 1 centímetro representa o ínicio e 10 centímetros representa o final, ao analisar esta régua não é possível afirmar qual centímetro é mais importante, ou seja, não é possível afirmar que 1 centímetro é mais importante que 10 centímetros, pois na régua não são atribuídos pesos que indicam importância as medidas que são apresentadas, de tal forma, não é possível identificar qual medida é mais importante. (ZAYARAZ; THAMBIDURAI, 2005).

Neste contexto, Zayaraz e Thambidurai (2005) enfatizam que uma função de força não-linear pode direcionar na identificação das preferências das partes interessadas.

“Para registrar a preferência mais forte, um questionário é preparado e distribuído para todas as partes interessadas e consequentemente os resultados são obtidos.” (ZAYARAZ; THAMBIDURAI, 2005, p.168, tradução nossa).

Conforme Zayaraz e Thambidurai (2005) após a tabulação dos dados dos questionários ser realizada é possível imputar as informações em um gráfico com as preferências que mais se destacaram, o gráfico que é gerado é denominado de gráfico de preferência mais forte, a Figura 6 apresenta um exemplo do gráfico de preferência mais forte:

Figura 6 - Gráfico de preferência mais forte



Fonte: Zayaraz e Thambidurai (2005, p.168, tradução nossa, adaptado por nós)

A Figura 6 auxilia no processo de refinamento dos atributos de qualidade, por exemplo, caso o atributo de qualidade “desempenho/tempo de resposta”, fosse o objeto de avaliação da Figura 6, é possível identificar que 100 partes interessadas envolvidas no processo de refinamento dos atributos de qualidade, preferem um tempo de resposta de 3 minutos, onde o eixo x representa os minutos (máximo e mínimo tempo de respostas) e o eixo y representa a quantidade de pessoas que fazem parte do processo de escolha.

Conforme Zayaraz e Thambidurai (2005, p.169, tradução nossa) “gráficos de preferência mais forte são gerados para todos os atributos de qualidade pertencentes aos grupos apresentados na Tabela 4”.

3.2.1.6 Determinação do Peso de Preferência

Conforme Zayaraz e Thambidurai (2005) a abordagem de análise hipotética é baseada em suposições de alternativas baseadas nas preferências das partes interessadas. Os atributos de qualidade recebem um peso com a finalidade de determinar a prioridade do atributo de qualidade. Os pesos são definidos com base nas preferências das partes interessadas, de forma hipotética, ou seja, o peso é definido com base nas preferências do grupo de usuários. Nesta abordagem a atribuição do peso é baseada na intuição das partes interessadas.

Zayaraz e Thambidurai (2005, p.169, tradução nossa) apresentam um exemplo onde os pesos são aplicados aos atributos de qualidade:

> Considere o grupo técnico e seus três atributos de qualidade, tempo de resposta, reusabilidade, manutenibilidade. No entanto, o grupo técnico pode considerar outras métricas pertinentes para a qualidade do projeto, por exemplo, coesão e acoplamento. No momento, os atributos de qualidade escolhidos pelo grupo podem estar em conflito e este conflito seria refletido na arquitetura (ZAYARAZ; THAMBIDURAI, 2005, p.169, tradução nossa).

Zayaraz e Thambidurai (2005, p.169, tradução nossa) apresentam quatro arquiteturas hipotéticas (A, B, C, D) e os valores atribuídos a cada uma delas são exibidos na Tabela 5:

Tabela 5 - Arquiteturas hipotéticas

Pesos e Atributos de qualidade				
Arquitetura	**Tempo de resposta (W_1)**	**Confiabilidade ($W_{2)}$**	**Manutenibilidade ($W_{3)}$**	**Pontuação total**
A	100	0	0	$100W_1$
B	0	0	40	$40W_3$
C	10	10	40	$10W_1 + 10W_2 + 40W_3$
D	0	0	50	$50W_3$

Fonte: Zayaraz e Thambidurai (2005, p.169, tradução nossa, adaptada por nós)

Zayaraz e Thambidurai (2005, p.169, tradução nossa) “ressaltam que existe uma similaridade equivalente entre as arquiteturas A e B, e entre as arquiteturas C e D que podem ser expressas nas seguintes equações:”

$$100W_1 = 40W_3 \quad (1)$$

$$10W_1 + 10W_2 + 40W_3 = 50W_3 \quad (2)$$

Zayaraz e Thambidurai (2005) enfatizam que um atributo de qualidade pode ter um nível de importância idêntico a outros atributos de qualidade, ou seja, para as partes interessadas escolher qual é mais prioritário, a normalização deve ser realizada. O intuito é quantificar a importância dos atributos de qualidade, através dos pesos atribuídos a cada um. Zayaraz e Thambidurai (2005, p.169) ressaltam que a equação após a normalização e atribuições de pesos, ficaria da seguinte forma:

$$W_1 = 0.2, W_2 = 0.3, W_3 = 0.5 \quad (3)$$

> Portanto o peso de preferência para tempo de resposta é 0.2, o peso de preferência para confiabilidade é 0.3, e o peso de preferência para manutenibilidade é 0.5. De forma similar os pesos de preferência para os atributos de qualidade que pertencem aos papéis gerente e usuário são computados. (ZAYARAZ; THAMBIDURAI, 2005, p.169, tradução nossa).

3.2.1.7 Conversão dos Valores dos Atributos de Qualidade Para as Arquiteturas Candidatas

Zayaraz e Thambidurai (2005) ressaltam que os valores normalizados dos atributos de qualidade e as preferências que são exibidas no gráfico de preferência mais forte podem ser utilizados para atingir os valores que serão atribuídos às arquiteturas candidatas.

Zayaraz e Thambidurai (2005) apresentam um exemplo de como os valores podem ser atribuídos aos atributos de qualidade, apresentados pelo grupo técnico.

A Tabela 6 apresenta este exemplo, contendo um conjunto de valores obtidos após os limites (medidas) dos atributos de qualidade serem normalizados, os valores apresentados são valores hipotéticos baseados na preferência do grupo técnico.

Conforme Zayaraz e Thambidurai (2005, p.169, tradução nossa) o cálculo é efetuado da seguinte maneira: “a pontuação total de cada arquitetura é calculada realizando a soma de todas as linhas, porém antes de efetuar a soma é necessário multiplicar o valor que está na linha pelo respectivo peso”.

Tabela 6 - Tabela de Valores

Pesos e Atributos de Qualidade				
Arquitetura	**Tempo de resposta (0.2)**	**Confiabilidade (0.3)**	**Manutenibilidade (0.5)**	**Pontuação total**
A	100	20	30	**41**
B	45	37	47	**43,6**
C	25	32	100	**64,6**

Fonte: Zayaraz e Thambidurai (2005, p.169, tradução nossa, adaptada por nós)

3.2.1.8 Cálculo das Pontuações Cumulativas

Zayaraz e Thambidurai (2005, p.169) enfatizam que a pontuação total representada na Tabela 6 pode ser alcançada através da seguinte fórmula:

$$\sum_{i=1}^{K} (Wi * Vi) \qquad (4)$$

As descrições das variáveis que compõe a equação são apresentadas da seguinte forma:

W = Peso do atributo i
V = Valor do atributo
K = Número de atributos na tabela

O próximo passo é calcular a média dos pontos acumulados para cada arquitetura. De acordo Zayaraz e Thambidurai (2005, p.169) os pontos acumulados para cada arquitetura são calculados a partir da seguinte fórmula:

$$\sum_{X=1}^{m} (TS\, j, x) \qquad (5)$$

As descrições das variáveis que compõe a equação são apresentadas da seguinte forma:

TS = Pontuação Total

M = Número de papéis

Zayaraz e Thambidurai (2005) enfatizam que a partir da fórmula de pontos acumulados, é possível identificar uma arquitetura que melhor satisfaça as necessidades das partes interessadas.

Zayaraz e Thambidurai (2005, p.169, tradução nossa) ressaltam que a satisfação total pode ser identificada através da seguinte equação:

Satisfação total = pontuação acumulada / número de papéis (6)

Neste contexto a pontuação acumulada deve ser calculada com base nas pontuações totais obtidas na Tabela 6 dividida pelos três papéis apresentados na Tabela 4. O resultado final do cálculo é apresentado na Tabela 7.

Tabela 7 - Satisfação Total

Arquitetura	Satisfação total
A	13.6
B	14,5
C	21,5

Fonte: Zayaraz e Thambidurai (2005, p.169, adaptada por nós)

Zayaraz e Thambidurai (2005) relatam que a arquitetura que apresenta o maior resultado na Tabela 7 é a que mais satisfaz as necessidades das partes interessadas. Desta forma, a arquitetura escolhida é a arquitetura C, pois ela possui o maior resultado de satisfação para as partes interessadas.

3.3 Método de Comparação de Arquiteturas, Baseado na ISO-9126-1 que Especifica os Atributos de Qualidade.

Losavio et al. (2003) apresenta um método para avaliação de arquiteturas de software com base nos atributos de qualidade definidos para a arquitetura proposta.

O método se baseia na ISO 9126-1 (1998) apud Losavio et al. (2003) que apresenta os principais atributos de qualidade mais utilizados em sistemas de softwares. A ISO 9126-1 (1998) apud Losavio et al. (2003) apresenta seis características de qualidade que estão definidas em um modelo de qualidade.

Losavio et al. (2003) ressaltam que a partir do modelo de qualidade apresentado pela ISO 9126-1 (1998) apud Losavio et al. (2003) é possível criar um processo que permite identificar qual a arquitetura é a mais adequada para um sistema de software com base nos atributos de qualidade.

Losavio et al. (2003) enfatizam que os requisitos funcionais e não funcionais do sistema devem ser conhecidos para que seja possível estabelecer as metas que a arquitetura do sistema de software deve atender. Nesta direção Losavio et al. (2003) ressaltam que a qualidade pode apresentar diferentes significados para diferentes partes interessadas, as visões de qualidade são individuais e cada parte interessada pode ter uma visão que o produto de software deverá satisfazer.

Conforme Losavio et al. (2003, p.136, tradução nossa)

> A visão do usuário está relacionada a qualidade do produto final, a visão do desenvolvedor está relacionada a qualidade dos produtos intermediários gerados durante o processo de desenvolvimento para diferentes partes interessadas, a visão do gerente pode estar relacionada a requisitos de *marketing.*(LOSAVIO et al., p.136, tradução nossa).

O modelo de qualidade da ISO 9126-1(1998) apud Losavio et al. (2003) pode ser visualizado na Tabela 8.

Tabela 8 - Características do modelo de qualidade da ISO 9126-1

Características	Descrição
Funcionalidade	A capacidade do produto de software de fornecer funções que satisfaçam necessidades específicas e implícitas, quando o software é usado em condições específicas.
Confiabilidade	A capacidade do produto de software para manter o seu nível de desempenho sob condições estabelecidas, para um determinado período de tempo.
Usabilidade	A capacidade do produto de software ser compreendido, aprendido, usado e atraente para o usuário, quando usado sob condições específicas.
Eficiência	A capacidade do produto de software apresentar o devido desempenho, em relação á quantidade dos recursos utilizados, sob determinadas condições.
Manutenibilidade	A capacidade do produto de software ser modificado. Modificações podem incluir correções, melhorias ou adaptações do software e alterações no ambiente nos requisitos e na especificação funcional.
Portabilidade	A capacidade do produto de software ser

Características	Descrição
	transferido de um ambiente para outro.

Fonte: ISO 9126-1 apud Losavio et al. (2003, p.137, tradução nossa)

Losavio et al. (2003, p.138) apresentam na Tabela 9 as subcaracterísticas dos atributos de qualidade apresentados na Tabela 8:

Tabela 9 - SubCaracterísticas do modelo de qualidade da ISO 9126-1

Características de qualidade	Sub Características
Funcionalidade	Conveniência, Precisão, Interoperabilidade, Segurança, Observância.
Confiabilidade	Maturidade, Tolerância a falhas, Capacidade de recuperação, Observância.
Usabilidade	Compreensibilidade, Apreensibilidade, Operacionalidade, Observância.
Eficiência	Comportamento do tempo, Comportamento do recurso, Observância.
Manutenibilidade	Analisabilidade, Mutabilidade, Estabilidade, Testabilidade, Observância.
Portabilidade	Adaptabilidade, Instalabilidade, Co-existência, Substituibilidade, Observância.

Fonte: ISO 9126-1 apud Losavio et al. (2003, p.138, tradução nossa)

Losavio et al. (2003) enfatizam que o método a ser aplicado para identificar a arquitetura de software baseado no modelo de qualidade da ISO 9126-1 (1998) apud Losavio et al. (2003), deve ser aplicado com base nas necessidades previamente definidas para o sistema, ou seja, os requisitos devem estar claros para que seja possível identificar se a arquitetura que está sendo refinada, atende aos requisitos previamente definidos. Nesta direção Losavio et al. (2003) enfatizam que os componentes que compõe a arquitetura, devem seguir o modelo de qualidade da ISO 9126-1 (1998) apud Losavio et al. (2003), pois a partir dos atributos de qualidade apresentados e identificados na arquitetura de software é possível coletar métricas, que exibem de forma quantitativa a aderência da arquitetura ao atributo de qualidade presente no modelo de qualidade da ISO 9126-1 (1998) apud Losavio et al. (2003).

Losavio et al. (2003) discutem que o resultado final da aplicação do método culmina na identificação dos atributos de qualidade na arquitetura de software identificada, ou seja, quando a arquitetura de software contemplar os atributos de qualidade apresentados no modelo de qualidade da ISO 9126-1 (1998) apud Losavio et al. (2003), esta arquitetura poderá ser considerada satisfatória.

A Tabela 10 apresenta um resumo das atividades que devem ser executadas pelo método:

Tabela 10 - Método para comparação de arquiteturas, baseado na ISO 9126-1 de 1998 que especifica os atributos de qualidade.

Atividades:
1. Analise os principais requisitos funcionais e atributos de qualidade do sistema, para que seja possível estabelecer as metas de qualidade; 2. Use o modelo de qualidade ISO 9126-1 de 1998 apresentado na Tabela 8 para definir a arquitetura. O modelo de qualidade ISO 9126-1 ele deve ser utilizado como um *framework*. Algumas métricas podem ser escritas com base em componentes específicos ou conectores; 3. Apresente as primeiras arquiteturas candidatas; 4. Construa uma tabela de comparação para as arquiteturas candidatas; 5. Priorize as características de qualidade, tendo em conta os requisitos de qualidade do sistema e as metas de qualidade. A customização da ISO 9126-1 para o domínio do problema pode ser usada para organizar hierarquicamente as características; 6. Analise os resultados apresentados na tabela, de acordo com as prioridades obtidas no passo 5; 7. Selecione a arquitetura candidata inicial, entre as outras arquiteturas candidatas que foram avaliadas, com base na análise anterior; 8. Se for necessária uma análise refinada, cenários podem ser utilizados, considerando apenas as características de qualidade relevantes para o domínio do problema obtido no passo 5.

Fonte: Losavio et al. (2003, p.147, tradução nossa, adaptada por nós)

3.4 Considerações do Capítulo

No capítulo 3 foram apresentados os métodos que permitem selecionar uma arquitetura de software com base nos atributos de qualidade. Três métodos foram apresentados e definidos, segue abaixo o resumo dos métodos apresentados:

1. Resumo do ADD (Projeto orientado ao atributo).

Este método é composto por 5 passos:

1. Escolha um elemento do sistema para projetar;

2. Identifique os ASRs (requisitos arquiteturalmente significativos) para o elemento escolhido;
3. Gere uma solução de projeto para o elemento escolhido;
4. Armazene os requisitos restantes e selecione a entrada para a próxima iteração;
5. Repita os passos 1- 4 até que todos os ASRs tenham sido satisfeitos.

A seleção da arquitetura no ADD é realizada no passo 3, nesta etapa as táticas e a lista de verificação já foram aplicadas e os ASRs já foram definidos, possibilitando a criação de um esboço de uma visão arquitetural. O ADD tem um escopo mais amplo do que os demais métodos, pois ele considera também a definição da arquitetura, ou seja, ele permite que a arquitetura seja construída do ínicio através da aplicação de táticas e seleções de ASRs que são exigidas durante a execução do método.

2. Resumo do Método de análise de múltiplas decisões de atributos de qualidade utilizando equivalência hipotética.

Este método é composto por 9 passos:

1. Classificação das partes interessadas;
2. Identificação dos atributos de qualidade das partes interessadas;
3. Definir (fixar) um limite aceitável para cada atributo de qualidade;
4. Normalização;
5. Identificação da força de preferência;
6. Determinação do peso de preferência;
7. Conversão dos valores dos atributos de qualidade para arquiteturas candidatas;
8. Cálculo das pontuações cumulativas;
9. Seleção da arquitetura.

A seleção da arquitetura no método de análise de múltiplas decisões de atributos de qualidade utilizando equivalência hipotética, é realizada no passo 9. No passo 9, as arquiteturas que foram objeto de avaliação são apresentadas e a

arquitetura de software que apresentar a maior pontuação baseada nos cálculos realizados nos passos anteriores é selecionada.

3. Resumo do método para comparação de arquiteturas, baseado na ISO-9126-1 que especifica os atributos de qualidade.

Este método é composto por 8 passos:

1. Analise os principais requisitos funcionais e não-funcionais do sistema, para que seja possível estabelecer os requisitos de qualidade e as metas de qualidade;
2. Use o modelo de qualidade ISO 9126-1 de 1998 apresentado na Tabela 13 para definir a arquitetura ele deve ser utilizado como um *framework*. Algumas métricas podem ser escritas com base em componentes específicos ou conectores;
3. Apresente as primeiras arquiteturas candidatas;
4. Construa uma tabela de comparação para as arquiteturas candidatas;
5. Priorize as características de qualidade, tendo em conta os requisitos de qualidade do sistema e as metas de qualidade. A personalização da ISO 9126-1 para o domínio do problema pode ser usada para organizar hierarquicamente as características;
6. Analisar os resultados apresentados na tabela, de acordo com as prioridades obtidas no passo 5;
7. Escolha a arquitetura candidata inicial, entre as outras arquiteturas candidatas que foram avaliadas, com base na análise anterior;
8. Se for necessária uma análise mais fina, cenários ou abordagens que se baseiam em perfil poderiam ser utilizadas, considerando apenas as características de qualidade relevantes para o domínio do problema obtido no passo 5.

A seleção da arquitetura no método para comparação de arquiteturas, baseado na ISO-9126-1 que especifica os atributos de qualidade, é realizada no passo 7. No passo 7 as arquiteturas candidatas são disponibilizadas para seleção, porém a arquitetura de software selecionada é aquela que apresenta maior aderência aos atributos de qualidade apresentados pela ISO-9126-1 (1998).

4. COMPARAÇÃO DE MÉTODOS PARA SELEÇÃO DE ARQUITETURA DE SOFTWARE, UTILIZANDO UM *FRAMEWORK* DE AVALIAÇÃO

Após serem apresentados os métodos para seleção da arquitetura no capítulo 3 será realizada a comparação entre os métodos apresentados. A comparação será realizada com o objetivo de apresentar as diferenças entre os métodos que foram apresentados no capítulo 3 a partir de um *framework* de avaliação denominado NIMSAD (*Normative Information Model-based System Analysis and Design)*. Esta comparação pretende auxiliar no processo de escolha de um método de seleção de arquitetura de software com base nos atributos de qualidade. Conforme Jayaratna apud Matinlassi (2004) o *framework* NIMSAD pode ser utilizado para avaliar diferentes metodologias na engenharia de software.

4.1 Um Framework de Avaliação: NIMSAD

Conforme Kheong e Jayaratna (entre 1995 e 2014) o NIMSAD é um *framework* que auxilia na resolução de problemas gerais.

O *framework* NIMSAD auxilia na avaliação de metodologias. O NIMSAD avalia a estrutura, passos, formas, natureza e outros itens da metodologia que será avaliada, conforme a necessidade do objeto de avaliação. (KHEONG; JAYARATNA, entre 1995 e 2014).

> Uma metodologia é um sistema de práticas, técnicas, procedimentos e regras utilizadas por aqueles que trabalham em uma disciplina, especificação do processo a seguir, juntamente com os produtos de trabalho a serem gerados e utilizados, além da consideração das pessoas e ferramentas envolvidas, durante um esforço de desenvolvimento. (ISO/IEC 24765, 2010, p.216, tradução nossa).

“Metodologia é definida como uma maneira explícita de estruturação (racionalização) pensamento e ação, que envolve um pensamento crítico e criativo ao mesmo tempo.” (JAYARATNA, 1994 apud KOSKINEN et al., 2004, p.7, tradução nossa). Para o contexto deste trabalho a definição escolhida para o termo metodologia, é a definição apresentada por Jayaratna (1994) apud Koskinen et al. (2004), pois está mais aderente ao contexto deste trabalho.

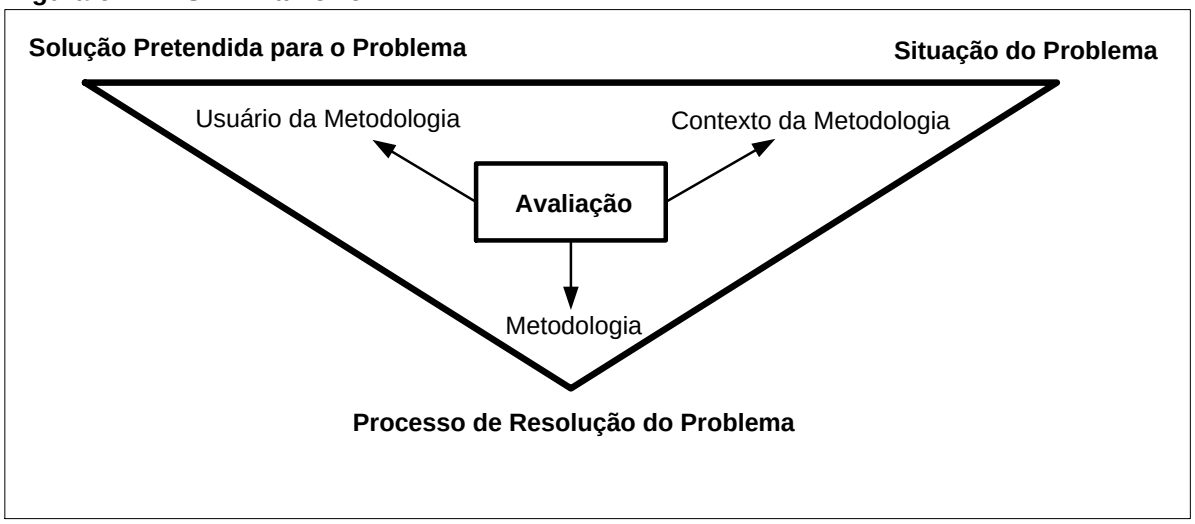
“Um método é uma implementação de uma operação, uma forma de verificar como uma possível combinação de características e valores produzem um resultado.” (ISO/IEC 24765, 2010, p.216, tradução nossa).

Vale ressaltar que o NIMSAD é utilizado também para avaliação de métodos, ou seja, o NIMSAD pode ser utilizado para comparar métodos e metodologia, e neste trabalho o NIMSAD será utilizado na comparação de métodos. É possível perceber que a aplicação do NIMSAD realizada por Babar e Gorton (2004) e Matinlassi (2004) são baseadas na comparação de métodos, nesta direção, para o contexto prático de aplicação neste trabalho, o termo a ser utilizado para referir-se aos objetos de comparação será método, pois será realizada a comparação de métodos.

Kheong e Jayaratna (entre 1995 e 2014) ressaltam que o NIMSAD é composto por quatro elementos principais, que são: situação do problema, solução pretendida para o problema, processo de resolução do problema e avaliação. O NIMSAD auxilia na resolução de problemas, contribuindo de forma substancial com as soluções geradas durante todo o processo de resolução do problema.

A Figura 6 apresenta os quatro elementos chave do NIMSAD, vale ressaltar que a palavra utilizada para referir-se ao método do NIMSAD na Figura 6 é “Metodologia”, ou seja, a palavra “Metodologia” foi mantida na Figura 6, para manter-se a originalidade da ilustração criada por Jayaratna apud Kheong e Jayaratna (entre 1995 e 2014).

Figura 6 - NIMSAD *Framework*



Fonte: Jayaratna apud Kheong e Jayaratna (entre 1995 e 2014, p.4, tradução nossa, adaptada por nós).

Situação do Problema

Conforme Kheong e Jayaratna (entre 1995 e 2014, p. 4, tradução nossa) “o contexto particular de um problema é representado pelo elemento ‘situação do problema’ do *framework*”.

Kheong e Jayaratna (entre 1995 e 2014) enfatizam que existem diferentes elementos que devem ser levados em consideração na identificação de um problema.

“Estes elementos podem ser considerados pessoas, processos, informação, tecnologia, fluxo de material e estruturas.” (KHEONG; JAYARATNA, entre 1995 e 2014, p. 4, tradução nossa).

De acordo com Kheong e Jayaratna (entre 1995 e 2014) os elementos apresentados relacionam-se, este relacionamento pode possibilitar muitas interpretações do contexto do problema. Contudo, as diferentes compreensões que são geradas a partir do relacionamento entre os elementos, viabilizam o entendimento do problema, pois a partir da análise das diferentes interpretações é possível perceber quais são as condições que geraram o problema e em paralelo definir quais serão as soluções para o problema.

Processo de Resolução do Problema

Confome Kheong e Jayaratna (entre 1995 e 2014) existe a necessidade de compreender as diversas formas de resolver um problema, ou seja, a maneira pela qual o problema será resolvido precisa estar clara. Nesta direção o elemento “processo de resolução de problema” do NIMSAD procura apresentar as etapas que são necessárias para solucionar um problema. A Tabela 11 apresenta as três fases do processo de resolução do problema:

Tabela 11 - Fases do processo de resolução do problema

Fases	Estágios
1. Formulação do problema	• Compreensão da situação do problema • Realizar o diagnóstico • Definindo o contorno prognóstico • Definindo os problemas • Derivando sistemas especulativos
2. Solução do projeto	• Execução do projeto lógico • Execução do projeto físico

Fases	Estágíos
3. Implementação do projeto	• Implementando o projeto

Fonte: Kheong e Jayaratna (entre 1995 e 2014, p. 5, tradução nossa)

A Tabela 11 apresenta três fase e oito estágios do processo de resolução do problema, estas fases e estes estágios auxiliam na identificação do problema, e na análise do problema, ao executar as atividades presentes na Tabela 11 é possível discernir o problema de forma especulativa, gerando meios para planejar um conjunto de ações pertinentes, para uma solução adequada para o problema em análise. O processo de resolução do problema conduz o executor do *framework* a encontrar a solução pretendida para o problema, através de um conjunto de atividades organizadas. (KHEONG; JAYARATNA, entre 1995 e 2014).

Solução Desejada Para o Problema

Kheong e Jayaratna (entre 1995 e 2014) ressaltam que neste elemento as partes interessadas são representadas através dos papéis que as mesmas desempenham, de forma a conduzi-lás a definir e solucionar o problema.

Conforme Kheong e Jayaratna (entre 1995 e 2014) este elemento é executado pelo solucionar do problema desejado, ou seja, é um indíviduo ou grupo que podem apresentar ideias e conceitos distintos em relação à solução pretendida para o problema. Vale ressaltar que a tarefa de solucionar um problema envolve a participação de várias partes interessadas, pois a partir dos conceitos e idéias das várias partes interessadas, é possível definir uma solução aceitável para o problema que consequentemente será avaliada.

“O solucionador do problema desejado pode estar envolvido nas fases de implementação, avaliação ou no processo de resolução do problema, juntamente com outras partes interessadas (cliente, dono do problema, beneficiário).” (KHEONG; JAYARATNA, entre 1995 e 2014, p.5, tradução nossa).

De acordo com Kheong e Jayaratna (entre 1995 e 2014) a solução do problema é influenciada por atributos oriundos da personalidade individual dos solucionadores do problema.

Kheong e Jayaratna (entre 1995 e 2014, p.5, tradução nossa) apresentam as características pessoais chaves dos solucionadores do problema, que influenciam o processo de definição e resolução do problema:

- Processo perceptivo;
- Valores e ética;
- Motivações e preocupações;
- Capacidade de raciocínio;
- Experiências;
- Habilidades e conjunto de conhecimentos;
- Estruturação do processo, incluindo metodologias;
- Papéis;
- Modelos e *frameworks*.

A solução pretendida para o problema é direcionada a partir das decisões dos solucionadores do problema, as partes interessadas. Contudo, vale ressaltar que ela é alcançada com base nas características individuais dos envolvidos na fase de solução do problema, pois a partir das experiências e atributos identificados é possível avaliar e compreender a solução mais adequada para o problema. (KHEONG; JAYARATNA, entre 1995 e 2014).

Avaliação

Kheong e Jayaratna (entre 1995 e 2014) ressaltam a importância da necessidade de avaliação dos elementos que compõe o NIMSAD, a avaliação é responsável por verificar e avaliar a eficácia dos três elementos do *framework*. A Figura 6 apresenta a relação do elemento de avaliação com os outros elementos do *framework*. Ao analisar a Figura 6 é possível perceber que o elemento de avaliação está relacionado com: solução pretendida para o problema, situação do problema, processo de resolução do problema.

A aplicação da avaliação é realizada em três momentos distintos, antes da interferência do solucionador do problema pretendido atuar, durante a interferência do solucionador do problema pretendido e após a interferência ser concretizada pelo solucionador do problema pretendido. (KHEONG; JAYARATNA, entre 1995 e 2014).

De acordo com Kheong e Jayaratna (entre 1995 e 2014, grifo nosso) a avaliação do processo de resolução do problema, a avaliação da situação do problema e a avaliação da solução pretendida para o problema são realizadas através de **questões**, estas questões que são formuladas com base no problema em

avaliação, servem de base para compreender o problema e também a solução do problema, permitindo identificar os impactos oriundos das intervenções das partes interessadas e as tranformações que acontecem durante a aplicação do *framework*.

4.1.1 Características Genéricas do *Framework*

Conforme apresentado na Figura 6, o NIMSAD apresenta quatro elementos chave que são necessários no processo de avaliação. Os quatro elementos chave são a base para a elaboração das questões de avaliação elaboradas para contextualizar, definir, avaliar e solucionar o problema.

A Figura 6, corresponde exatamente a análise efetuada por Koskinen et al. (2004) ao apresentar a relação entre os elementos do NIMSAD.

Cada elemento é apresentado conforme descrição apresentada por Koskinen et al. (2004):

- O contexto da metodologia permite compreender o motivo da utilização da metodologia, ou seja, os detalhes mais relevantes são destacados. O entendimento do âmbito do problema é realizado no contexto da metodologia. O contexto da metodologia está relacionado com o elemento situação do problema, apresentado na Figura 6.
- O usuário da metodologia é quem resolve o problema, ou seja, é este elemento que exerce o papel de solucionar do problema, de tal forma, este elemento está relacionado com o elemento solução pretendida para o problema apresentado na Figura 6. O caminho que o usuário da metodologia utiliza para tomar as suas decisões precisa ser compreendido, ou seja, quais são os motivos que levam o usuário da metodologia tomar uma decisão. Nesta mesma direção, é preciso entender quais são as competências e o conhecimento do usuário sobre a metodologia avaliada em questão.
- A metodologia em si está relacionada com o elemento, processo de resolução do problema, ou seja, apresenta o caminho necessário para resolver o problema.

A avaliação é responsável por avaliar os outros três elementos do NIMSAD conforme apresentado na Figura 6.

Segundo Jayaratna, 1994 apud Koskinen et al. (2004) estes quatro elementos são repletos de questões, ou seja, uma grande quantidade de questões são aplicadas para avaliar as particularidades do método em análise.

Conforme Koskinen et al. (2004) a lista de possíveis questões apresentadas por Jayaratna (1994) é ampla, nesta direção, um resumo das questões apresentadas por Jayaratna (1994) foi elaborado por Avison e Fitzgerald 1995 apud Koskinen et al. (2004) para possibilitar exemplificar o *framework* de forma resumida.

Vale ressaltar que as questões que são apresentadas na Tabela 12 são apresentadas por Forsell et al. 1999 apud Koskinen et al. (2004) com base na abordagem originalmente criada por Avison e Fitzgerald 1995 apud Koskinen et al. (2004).

Tabela 12 - O Framework NIMSAD e sua interpretação

Elementos do NIMSAD como representado por Forsell et al. (1999)	**Questões**	**Aplicabilidade e reinterpretação dos elementos necessários no NIMSAD, para a correta avaliação das abordagens gerais de resolução de problemas**
Contexto da Metodologia		
Situação de uso	Que tipo de situação ocasiona o uso da Metodologia?	OK, mais o escopo é geralmente mais restrito
Inicie usando a Metodologia	Quais eventos iniciam o uso da Metodologia?	OK
Clientes e proprietários do problema	Quem são os clientes e os proprietários do problema?	OK
Descrição do contexto	Como o contexto é descrito?	OK, mais o papel do contexto é menos saliente
Cultura e politícas do uso da Metodologia	Qual é a cultura e as politícas de uso da metodologia?	OK, mais a importância deste elemento, geralmente não é explicada
Riscos na descrição do contexto	Quais riscos são identificados pela metodologia durante a descrição do contexto?	OK, mas o papel do contexto é completamente limitado
Riscos da Metodologia	Quais são os riscos em utilizar a metodologia?	OK
Usuário da Metodologia		
Motivações e valores do usuário	Quais são as motivações e valores do usuário?	OK

Elementos do NIMSAD como representado por Forsell et al. (1999)	Questões	Aplicabilidade e reinterpretação dos elementos necessários no NIMSAD, para a correta avaliação das abordagens gerais de resolução de problemas
Necessidade de raciocínio abstrato	Qual nível de raciocínio abstrato é necessário para o usuário da metodologia?	OK
Habilidades necessárias	Quais habilidades são necessárias ao usuário da metodologia para realizar as tarefas requeridas no uso da metodologia?	OK
Metodologia		
Fronteiras e situação do problema	Como a metodologia auxilia na compreensão de uma situação específica e na definição da fronteira?	OK, mais geralmente não é considerada como muito importante
Análise da situação	Como o usuário da metodologia analisa qual o tipo de sistema é necessário?	OK, mas é uma espécie de análise de estado atual e inclui como elemento central as condições prévias para a aplicação do método
Prognóstico para o sistema	Como o usuário da metodologia faz um prognóstico do sistema a ser construído?	O prognóstico para a solução do problema, responde as seguintes perguntas: onde queremos ir? Quais são os objetivos?
Definição do problema	Como o usuário da metodologia define os problemas que precisam ser resolvidos?	Ok, mais na maioria dos casos triviais (as questões poderiam ser reutilizadas para resolver o problema baseando-se nas questões já realizadas no elemento anterior).
Definido sistemas imaginários	Como operar um sistema que precisa ser especificado?	OK, mas o processo para descrever o estado de destino geralmente não é necessário (e não é suportado)
Projeto (originalmente: projeto lógico e físico separado)	Esta fase está concluída? Como o usuário da metodologia implementa	Projeto, geralmente não é abstrato. O papel do projeto é completamente

Elementos do NIMSAD como representado por Forsell et al. (1999)	Questões	Aplicabilidade e reinterpretação dos elementos necessários no NIMSAD, para a correta avaliação das abordagens gerais de resolução de problemas
	esta fase?	restrito (consistindo de subfases produzindo entradas para o próximo elemento)
Implementação do projeto	Está fase é descrita? O que está incluído nesta fase?	Implementação: Processo de resolução do problema atual
Avaliação		
Avaliação (originalmente: casos ou cenários são aplicados antes / durante / após a intervenção de forma separada)	Como os outros elementos (apresentados pelo NIMSAD) são avaliados?	Avaliação (externa, interna) esta Metodologia não oferece suporte interno a “reflexão” ou para a “avaliação” própria

Fonte: Forsell et al. 1999 apud Koskinen et al. (2004, p.8, tradução nossa)

A Tabela 12 contém uma coluna denominada de “Aplicabilidade e reinterpretação dos elementos necessários no NIMSAD, para a correta avaliação das abordagens gerais de resolução de problemas”, esta coluna é utilizada por Forsell et al. 1999, para avaliar os elementos do próprio NIMSAD, ou seja, Forsell et al. 1999, verifica se o elemento é coerente ou não, baseando-se em abordagens que são geralmente utilizadas para resolução de problemas no contexto de software. Contudo, vale ressaltar, que as principais características a serem observadas para o contexto deste trabalho, estão baseadas nas colunas “Elementos do NIMSAD como representado por Forsell et al. (1999)” e “Questões”, pois, Forsell et al. 1999 apud Koskinen et al. (2004) apresentam nestas colunas os elementos originais do NIMSAD de forma resumida.

Forsell et al. 1999 apud Koskinen et al. (2004, p.9, tradução nossa) definem avaliação “interna” e “externa” da seguinte maneira:

> 1) Avaliação interna: refere-se a avaliação da reflexão, ou seja, com base nas sugestões informadas pelos usuários, bem como a longo prazo através de um modelo iterativo baseado em feedback, a avaliação é realizada.
> 2) Avaliação externa: reúne todos os outros tipos de avaliações, possívelmente feitas, por exemplo, (provas, testes, simulações, através da utilização de conhecimentos especializados, experiências de campo, estudos de caso, feedback do usuário e referências industriais. (FORSELL et al., 1999 apud KOSKINEN et al., 2004, p.9, tradução nossa).

De acordo com Koskinen et al. (2004) é importante ressaltar a importância do contexto da metodologia, pois é no contexto que é possível compreender a circunstância do problema que é necessário resolver. No objetivo da metodologia é possível identificar as razões que permitem identificar a situação do problema, para consequentemente solucioná-lo.

Koskinen et al. (2004) ressaltam que o contexto da metodologia está relacionado com o elemento situação do problema. Na mesma direção relatam que o elemento usuário da metodologia, está relacionado com o elemento solução pretendida para o problema, e que a metodologia está relacionada com o elemento do processo de resolução do problema, e por fim a avaliação está relacionada com todos os elementos.

4.1.2 Exemplos de Casos Reais de Aplicação do NIMSAD

Na seção - 4.1.1 Características Genéricas do *Framework*, foi necessário utilizar o termo metodologia, para conceituar o uso do NIMSAD conforme os termos originais definidos por Jayaratna (1994), contudo a partir desta seção 4.1.2 será utilizado o termo método, pois o NIMSAD também é utilizado na avaliação de métodos, conforme apresentado a seguir.

É importante ressaltar os contextos em que o NIMSAD foi utilizado através da análise do trabalho de outros autores.

O NIMSAD já foi utilizado na integra para auxiliar na seleção de métodos que auxiliam na resolução de um problema. Derivações do NIMSAD já foram criadas para auxiliar o processo de avaliação para o contexto de um problema específico.

Segundo Jayaratna apud Matinlassi (2004, p.2, tradução nossa) “o *framework* NIMSAD pode ser utilizado para avaliar métodos em qualquer categoria”.

Matinlassi (2004) realiza uma customização no NIMSAD original, apresentando o *framework* com características customizadas para avaliação de PLAs (Arquiteturas de linha dos produtos):

A customização realizada no NIMSAD por Matinlassi (2004) altera o nome do componente original do NIMSAD “Avaliação” para “Validação”. O intuito desta mudança, conforme Matinlassi (2004), está baseada no fato de existirem poucas aplicações do *framework* onde o elemento “Avaliação” é utilizado para avaliar o

contexto do método, usuário ou conteúdo do método, desta forma, Matinlassi (2004) encontrou maior conveniência em utilizar o termo validação para legitimar a saída do método, através de questionamentos que permitem validar os resultados.

Babar e Gorton (2004) ressaltam que o NIMSAD ajuda na escolha do método mais adequado para utilização em um determinado domínio. Babar e Gorton (2004) utilizam o *framework* NIMSAD para realizar a comparação de métodos de avaliação de arquitetura de software, por exemplo, um dos métodos avaliados é o ATAM - *Architecture Tradeoff Analysis Method* (Método de análise das vantagens e desvantagens da arquitetura) e o SAAM - Scenario-Based Architecture Analysis (Análise da arquitetura baseada em cenário).

De acordo com Babar e Gorton (2004) o uso do NIMSAD na comparação dos métodos de avaliação de arquitetura, foi utilizado com o intuito de guiar o usuário do *framework* a escolher o método mais adequado a partir da análise dos resultados obtidos.

Ao utilizar o NIMSAD para realizar as comparações entre os métodos de avaliação de arquitetura de software Babar e Gorton (2004) realizaram uma customização no *framework*. Com base no domínio do estudo realizado pelos mesmos, os nomes de dois componentes originais do NIMSAD foram alterados.

O primeiro componente corresponde ao “Usuário da Metodologia”, que foi renomeado para “Partes interessadas”, conforme Babar e Gorton (2004), esta mudança foi realizada, pois o método de avaliação de arquitetura de software, para que possa ser executado, depende da opinião das partes interessadas, incluindo pessoas com papéis e necessidades distintas, de tal forma que o método não se limita ao contexto do usuário, ele abrange as partes interessadas também, que exercem influência determinante nos métodos de avaliação da arquitetura de software.

O segundo componente renomeado por Babar e Gorton (2004) corresponde a “Avaliação” que foi alterado para “Confiança”. De acordo com Babar e Gorton (2004) esta mudança foi realizada devido à necessidade de apresentar um método confiável para o usuário, ou seja, Babar e Gorton (2004) informam que é mais importante possuir um método confiável do que um método avaliável, nesta direção, as perguntas realizadas neste componente são todas relacionadas à confiabilidade do método.

Koskinen et al. (2004) utilizam o NIMSAD na sua totalidade, sem customizações. Vale ressaltar que é realizado um resumo das perguntas que serão aplicadas conforme o domínio da avaliação realizada pelos autores.

A abordagem apresentada por Koskinen et al. (2004) realiza a comparação entre métodos que dão origem a estimativa de rentabilidade a partir da modernização no software e também compara os métodos que dão suporte a esta modernização. Dentre os diversos métodos comparados por Koskinen et al. (2004) pode-se destacar dois que são amplamente conhecidos:Custos de modernização:

- COCOMO II (*Construtive Cost Model II* – Modelo de Custo Construtivo II)
- FPA (*Function Point Analysis* – Análise de Ponto de Função)

Conforme Koskinen et al. (2004) ao final da aplicação do *framework* NIMSAD as diferenças relacionadas as estimativas de esforços de modernização são apresentadas, pois as diferenças existentes nos métodos são expostas. As abordagens apresentadas por Koskinen et al. (2004) referem-se ao contexto de software.

4.1.3 Aplicação do *Framework* na Comparação de Métodos de Seleção de Arquitetura de Software Baseados em Atributos de Qualidade.

Neste trabalho será utilizado a adaptação do NIMSAD proposta por Babar e Gorton (2004), pois os elementos definidos correspondem à necessidade de avaliação esperada para os métodos apresentados no capítulo 3.

Babar e Gorton (2004) consideram que o NIMSAD pode ser customizado conforme o domínio de um problema ou necessidade, nesta direção eles realizam uma customização no *framework*, conforme seção “4.1.2 Exemplos de casos reais de aplicação do NIMSAD.”

A Tabela 13 apresenta o *framework* de avaliação apresentado por Babar e Gorton (2004) que servirá de base para o *framework* que será utilizado neste trabalho.

Tabela 13 - Os componentes e atributos do framework e as questões de avaliação.

Componente	Elemento	Breve explicação
Contexto	**Definição de arquitetura de software**	O método apresenta de forma explícita alguma definição de arquitetura de software?
	Meta especifica	Quais são os objetivos particulares dos

Componente	Elemento	Breve explicação
		métodos?
	Atributos de qualidade	Quantos e quais atributos de qualidade são cobertos pelo método?
	Fase aplicável	Qual é a fase de desenvolvimento mais adequada para se aplicar o método?
	Entrada e saída	Quais são as entradas necessárias e as saídas produzidas?
	Domínio da aplicação	Qual é o domínio de aplicação onde geralmente o método é mais aplicado?
Partes interessadas	**Benefícios**	Quais são os benefícios que o método oferece para as partes interessadas?
	Partes interessadas	Quais grupos de interessados são obrigados a participar da avaliação?
	Suporte ao processo	O método oferece um volume de suporte adequado para executar várias atividades?
	Questões sócio técnicas	Como o método lida com questões não técnicas, por exemplo, (questões sociais, organizacionais)?
	Recursos necessários	Quantos homem-dia são necessários? Qual o tamanho da equipe de avaliação?
Conteúdo	**Atividades do método**	Quais são as atividades a serem realizadas e em qual ordem elas devem estar para alcançar os objetivos?
	Descrição da arquitetura de software	Qual forma de descrição arquitetural é recomendada? (formal, informal, particular, ADL, visões etc.)?
	Abordagens de avaliação	Que tipos de abordagens de avaliação são usadas pelo método?
	Ferramenta de apoio	Existem ferramentas ou repositório de experiência que auxiliam na execução do método e dos seus artefatos?
Confiabilidade	**Maturidade do método**	Qual o nível de maturidade do método (Criação, desenvolvimento, aperfeiçoamento ou inativo)?
	Validação do método	O método foi validado? Como tem sido validado?

Fonte: Babar e Gorton (2004, p.2, tradução nossa, adaptada por nós).

A Tabela 13 apresenta os componentes e elementos utilizados por Babar e Gorton (2004). A descrição de cada elemento apresentada na coluna breve explicação apresenta o contexto de cada elemento que será objeto de avaliação no método que será comparado, ou seja, o objetivo da avaliação.

Porém, para realizar a validação dos métodos apresentados no capítulo 3, além de utilizar o NIMSAD proposto por Babar e Gorton (2004), foram identificados novas questões de avaliação com o intuito de adequar o *framework* à comparação dos métodos apresentados no capítulo 3. Novas questões foram adicionadas no

framework com o intuito de apresentar uma avaliação com um nível de completude aderente às necessidades de avaliação consideradas neste trabalho.

Neste trabalho serão utilizadas as perguntas que estão contidas na Tabela 13 para realizar a comparação de métodos de seleção de arquitetura de software com base em atributos de qualidade. Embora Babar e Gorton (2004) utilizam as perguntas que estão na Tabela 13 para avaliar métodos de avaliação de arquitetura (métodos com escopo diferente do nosso), iremos utilizar as mesmas perguntas pois elas se aplicam na avaliação dos métodos comparados neste trabalho, elas apresentam coerência em suas definições e auxilia a distinguir de forma nítida as diferenças existentes entre um método e outro. Outro detalhe importante refere-se a etapa de execução dos métodos no ciclo de desenvolvimento do software, os métodos serão executados na fase de representação da arquitetura, de tal maneira que as perguntas que foram elaboradas por Babar e Gorton (2004) na Tabela 13, auxilia a distinguir de forma consistente e a identificar o método de seleção de arquitetura de software com base em atributos de qualidade, com maior precisão. Por isso neste trabalho vamos manter as perguntas elaboradas por Babar e Gorton (2004) para avaliar métodos de seleção de arquitetura de software.

A Tabela 14 apresenta o *framework* NIMSAD apresentado por Babar e Gorton (2004) com questões e elementos adicionais, ou seja, além das questões e elementos propostos por Babar e Gorton (2004), foram adicionadas questões e elementos chave que vão auxiliar no processo de avaliação dos métodos apresentados no capítulo 3. Estas questões estão ressaltadas no componente conteúdo.

Tabela 14 - Framework base para comparação dos métodos do capítulo 3, conforme customização nossa e de Babar e Gorton (2004).

Componente	Elemento	Breve explicação
Contexto	**Definição de arquitetura de software**	O método apresenta de forma explícita alguma definição de arquitetura de software?
	Meta especifica	Quais são os objetivos particulares dos métodos?
	Atributos de qualidade	Quantos e quais atributos de qualidade são cobertos pelo método?
	Fase aplicável	Qual é a fase de desenvolvimento mais adequada para se aplicar o método?
	Entrada e saída	Quais são as entradas necessárias e as saídas produzidas?
	Domínio da aplicação	Qual é o domínio de aplicação onde

Componente	Elemento	Breve explicação
		geralmente o método é mais aplicado?
Partes interessadas	**Benefícios**	Quais são os benefícios que o método oferece para as partes interessadas?
	Partes interessadas	Quais grupos de interessados são obrigados a participar da avaliação?
	Suporte ao processo	O método oferece suporte adequado para executar suas atividades?
	Questões sócio técnicas	Como o método lida com questões não técnicas, por exemplo, (questões sociais, organizacionais)?
	Recursos necessários	Quantos homem-dia são necessários? Qual deve ser o tamanho da equipe?
Conteúdo	**Atividades do método**	Quais são as atividades a serem realizadas e em qual ordem elas devem estar para alcançar os objetivos?
	Descrição da arquitetura de software	Qual forma de descrição arquitetural é recomendada? (formal, informal, particular, ADL, visões etc.)?
	Abordagens de avaliação	Que tipos de abordagens de avaliação são usadas pelo método?
	Ferramenta de apoio	Existem ferramentas ou repositório de experiência que auxiliam na execução do método e dos seus artefatos?
	Detalhamento	Qual o detalhamento existente na seleção da arquitetura candidata com base nos atributos de qualidade?
	Técnicas utilizadas	Quais são as técnicas utilizadas para selecionar a arquitetura?
Confiabilidade	**Maturidade do método**	Qual o nível de maturidade do método (Criação, desenvolvimento, aperfeiçoamento ou inativo)?
	Validação do método	O método foi validado? Como tem sido validado?

Fonte: Babar e Gorton (2004, p.2, tradução nossa, adaptada por nós).

Os dois elementos e as duas questões que foram inseridas por nós na Tabela 14 são apresentadas abaixo:

- **Detalhamento** - Qual o detalhamento existente na seleção da arquitetura candidata com base nos atributos de qualidade?
- **Técnicas utilizadas** - Quais são as técnicas utilizadas para selecionar a arquitetura?

Estes dois elementos foram inseridos, pois embora os métodos apresentam finalidades semelhantes, a maneira com que o resultado é alcançado por cada um é diferente. Portanto surgiu a necessidade de revelar o nível de granularidade de cada método, isto é, até que ponto as atividades assemelham-se e até que ponto elas se diferem. Da mesma forma, vale ressaltar que os métodos apresentam técnicas

distintas para alcançar resultados semelhantes, o resultado dos métodos apresentados deve ser a escolha da arquitetura de software mais adequada conforme os atributos de qualidade que foram definidos, os três métodos apresentados no capítulo 3 tem este objetivo, porém apresentam formas diferentes para alcançar o resultado.

4.2 Método 1: Avaliação do ADD Utilizando o NIMSAD Adaptado

A seguir será apresentado um resumo da avaliação do ADD a partir do *framework* sugerido na Tabela 14. A resposta para cada elemento do *framework* é definida abaixo conforme as caraterísticas do ADD:

Definição de arquitetura de software. De acordo com Software Engineering Institute (entre 2009 e 2012) o ADD é uma abordagem que permite definir uma arquitetura de software a partir dos atributos de qualidade, desde que um processo definido seja seguido para alcançar esta finalidade.

Metas específicas. Conforme Bass; Clements e Kazman (2012) a principal meta do ADD é gerar um esboço da arquitetura, uma visão arquitetural, de forma organizada, a partir dos atributos de qualidade mais significativos, que são escolhidos para refinar uma determinada parte do sistema. O ADD permite a definição e seleção de uma arquitetura de software com base em atributos de qualidade.

Atributos de qualidade. Conforme Bass; Clements e Kazman (2012) múltiplos atributos de qualidade podem ser cobertos pelo ADD. Porém as táticas apresentadas pelos autores se limitam a sete atributos de qualidade, que são: disponibilidade, interoperabilidade, modificabilidade, desempenho, segurança, testabilidade e usabilidade.

Fase aplicável. Segundo Bass; Clements e Kazman (2012) a aplicação do ADD pode ser iniciada, quando todos os ASRs forem conhecidos, ou seja, após a definição dos principais requisitos que farão parte da arquitetura do software o ADD pode ser aplicado. O ADD se situa entre duas fases, ou seja, ele deve estar após a fase de definição de requisitos e antes da implementação do código fonte. (Fase da representação da arquitetura de software).

Entrada e saída: Conforme Bass, Clements e Kazman (2012) as entradas e saídas do ADD são:

- Entradas
    - Requisitos funcionais
    - Atributos de qualidade
    - Restrições de projeto
- Saídas
    - Um conjunto de escorços de visões arquiteturais formam a saída do ADD.

Domínio da aplicação. De acordo com Software Engineering Institute (entre 2009 e 2012, tradução nossa) “O método ADD tem sido utilizado para domínios de aplicações que vão desde sistemas de informação até sistemas embarcados”.

Benefícios. “O ADD permite que os projetistas compreendam logo no inicío do processo os pontos fortes e fracos dos atributos de qualidade, orienta-os a projetar uma arquitetura que irá satisfazer as necessidades apresentadas nos atributos de qualidade e nos requisitos funcionais”. (SOFTWARE ENGINEERING INSTITUTE, entre 2009 e 2012, tradução nossa). Outro benefício é que a visão arquitetural gerada é consistente com os ASRs definidos pelas partes interessadas, ou seja, a aplicação do ADD permite que os principais requisitos da arquitetura do software estejam presentes na visão arquitetural gerada ao final da execução do método, de forma que as expectativas das partes interessadas sejam atendidas. O ADD utiliza um conjunto de táticas padronizadas para identificar os atributos de qualidade que compõe a arquitetura do software, para validar se os atributos de qualidade foram contemplados, uma lista de verificação é criada, com o intuito de apresentar quais atributos de qualidade e características vão compor a arquitetura. Este conjunto de estratégias confere consistência à arquitetura do software, de forma a atender as expectativas das partes interessadas.

Partes interessadas. Conforme Bass; Clements e Kazman (2012) as partes interessadas no processo de avaliação pode ser qualquer pessoa que obtém benefício com o sucesso do sistema, elas podem ser: o arquiteto de software, gerente de projeto, usuário final, cliente, e até aqueles que apresentam interesse comercial em relação ao sistema. Desta forma, é possível perceber que as partes interessadas, podem ser múltiplas, depende do contexto do projeto.

Suporte ao processo. O ADD oferece algumas técnicas que auxiliam no desempenho das atividades necessárias para a execução do método, as técnicas são (BASS; CLEMENTS; KAZMAN, 2012, p.69, tradução nossa):

- Cenários - Na seção “3.1.1.1 Criação de cenários”, é apresentada uma definição para esta técnica.
- Táticas – Na seção “3.1.5 Passo 2: Identifique os ASRs para o elemento escolhido”, é apresentada uma definição para esta técnica.
- Listas de verificação – Na seção “3.1.6 Passo 3: Gerar uma solução de projeto para o elemento escolhido” é apresentada uma definição para esta técnica.

Questões sócio técnicas. De acordo com Bass; Clements e Kazman (2012) as metas de negócio impactam profundamente os ASRs. Os autores ressaltam que os interesses da organização tem influencia na definição dos ASRs. Desta forma é possível compreender que o método ADD é influenciado pelos interesses de negócios da organização.

Recursos necessários. Bass; Clements e Kazman (2012) não informam a quantidade de recursos necessários.

Atividades do método. Segundo Bass; Clements e Kazman (2012) o ADD é compostos de cinco passos que devem ser seguidas na ordem para que o método tenha êxito:

> 1. Escolha um elemento do sistema para projetar.
> 2. Identifique os ASRs (requisitos arquiteturalmente significativos) para o elemento escolhido.
> 3. Gere uma solução de projeto para o elemento escolhido.
> 4. Armazene os requisitos restantes e selecione a entrada para a próxima iteração.
> 5. Repita os passos 1- 4 até que todos os ASRs tenham sido satisfeitos.
>
> (BASS; CLEMENTS; KAZMAN, 2012, p.318, tradução nossa).

Descrição da arquitetura de software. Conforme Bass; Clements e Kazman (2012) a documentação da arquitetura é construída a partir de um conjunto de visões arquiteturais.

Abordagens de avaliação. “É uma abordagem para a definição da arquitetura de software baseada em um processo definido para o projeto da arquitetura, com base nos atributos de qualidade.” (SOFTWARE ENGINEERING INSTITUTE, entre 2009 e 2012, tradução nossa). De acordo com Bass; Clements e Kazman (2012) diversos cenários podem ser criados para identificar os atributos de qualidade, cada cenário identifica: a origem do estímulo, o estímulo, o ambiente, o artefato, a resposta e a **medição da resposta**. Em seguida, táticas são utilizadas, estas táticas são decisões que impactam o projeto da arquitetura, as táticas

verificam de forma abrangente a consistência de um determinado atributo de qualidade. Padrões arquiteturais que satisfazem a um determinado atributo de qualidade podem ser identificados durante o projeto da arquitetura.

Ferramenta de apoio. Os autores do método não apresentam ferramentas que podem auxiliar na utilização do método.

Detalhamento. O detalhamento do elemento refere-se ao escopo do método em relação à seleção da arquitetura, ou seja, o detalhamento procura apresentar o nível de seleção que é realizado pelo método, por exemplo, o ADD é um método que define a arquitetura de software e também seleciona a arquitetura de software, ou seja, ele permite que uma arquitetura seja construída a partir do ínicio, e dentro desta construção a seleção da arquitetura de software é realizada, porém vale ressaltar que a seleção da arquitetura no ADD é feita através das táticas, ou seja, a aplicação correta das táticas permite alcançar uma arquitetura de software adequada para o sistema de software com base nos atributos de qualidade que foram definidos e que são aplicados através das táticas. A definição e a seleção da arquitetura são realizadas internamente no ADD, através de um conjunto de técnicas, que envolvem cenários, táticas e listas de verificação.

Técnicas utilizadas. As técnicas utilizadas para realizar a avaliação que estão no contexto do ADD são: criação de cenários, táticas e listas de verificação.

Maturidade do método. É um método que já foi aplicado e testado em diversos estudos, conforme o elemento de **validação** apresenta abaixo.

Validação do método. O ADD é um método que já foi validado por outros estudos de casos e artigos. Alguns estudos que apresentam uma descrição do ADD são os apresentados por Wojcik et al. (2006), eles apresentam um detalhamento do ADD, revisando os passos do ADD e apresentando informações relevantes em relação ao processo de definição do projeto. Nord et al. (2004) apresentam o ADD a partir da análise do RUP, *Rational Unified Process* (Processo unificado racional).

Também Wood (2007) apresenta uma aplicação prática do ADD, esta aplicação foi publicada pelo SEI (*Software Engineering Institute*).

Os estudos relacionados ao ADD escritos por Wojcik et al. (2006), Nord et al. (2004) e Wood (2007) foram publicados pelo SEI.

4.3 Avaliação do Método de Análise de Múltiplas Decisões de Atributos de Qualidade Utilizando Equivalência Hipotética.

A seguir será apresentado um resumo da avaliação do método **AMDAQ** (Método de Análise de Múltiplas Decisões de Atributos de Qualidade Utilizando Equivalência Hipotética). A partir do *framework* sugerido na Tabela 14, a resposta para cada elemento do *framework* é definida abaixo conforme as caraterísticas do AMDAQ:

Definição de arquitetura de software. O método não apresenta uma definição particular de arquitetura.

Metas específicas. Conforme Zayaraz e Thambidurai (2005) os diferentes atributos de qualidade definidos pelas partes interessadas, são as entradas principais para o processo de seleção de arquitetura de software. O método apresentado tem o intuito de apresentar como a seleção da arquitetura de software mais adequada é realizada com base nos atributos de qualidade definidos pelas partes interessadas.

Atributos de qualidade. De acordo com Zayaraz e Thambidurai (2005) Múltiplos atributos de qualidade podem fazer parte do processo de seleção apresentado pelo método, pois os atributos de qualidade dependem das exigências das partes interessadas e do contexto do sistema que está sendo desenvolvido.

Fase aplicável. O AMDAQ se situa entre duas fases, ou seja, ele deve estar após a fase de definição de requisitos e antes da implementação do código fonte. (Fase da representação da arquitetura de software).

Entrada e saída. De acordo com Zayaraz e Thambidurai (2005) as entradas necessárias para execução do método são as arquiteturas candidatas. A saída do método é uma arquitetura de software que atende as expectativas das partes interessadas.

Domínio da aplicação. Não informado pelos autores.

Benefícios. Conforme Zayaraz e Thambidurai (2005) as principais vantagens de utilizar o AMDAQ, são:

a) Oferece às partes interessadas um papel fundamental na seleção da arquitetura;

b) Permite alterar e incluir uma arquitetura candidada facilmente na execução do método com a adição de apenas uma linha nas tabelas;
c) De uma maneira científica é possível anteferir os atributos de qualidade.

Partes interessadas. Conforme Zayaraz e Thambidurai (2005) as partes interessadas que apoiam a execução do método, são escolhidas com base nos papéis que as mesmas desempenham e no nível de responsabilidade necessário dentro do contexto de aplicação do método. Neste sentido, é possível compreender que as partes interessadas podem ser diversas, depende do contexto do projeto de software que está sendo construído.

Suporte ao processo. O AMDAQ, não apresenta recursos que apoiam a execução do método.

Questões sócio técnicas. Não informado.

Recursos necessários. Não informado.

Atividades do método. Conforme Zayaraz e Thambidurai (2005) o AMDAQ, tem nove atividades:

> A. Classificação das partes interessadas.
> B. Identificação dos atributos de qualidade das partes interessadas.
> C. Definir (fixar) um limite aceitável para cada atributo de qualidade.
> D. Normalização.
> E. Identificação da força de preferência.
> F. Determinação do peso de preferência.
> G. Conversão dos valores dos atributos de qualidade para arquiteturas candidatas.
> H. Cálculo das pontuações cumulativas.
> I. Seleção da arquitetura.
> (ZAYARAZ; THAMBIDURAI, 2005, p.168, tradução nossa).

Descrição da arquitetura. Não informado.

Abordagens de avaliação. O método considera os atributos de qualidade esperados pelas partes interessadas, esses atributos de qualidade, ganham pesos, e a partir de fórmulas matemáticas é possível calcular a importância de um atributo de qualidade para arquitetura do software em questão, conforme seção 3.2.1.6 Determinação do peso de preferência.

Ferramenta de apoio. Não informado.

Detalhamento. O detalhamento deste método está na forma de identificar a arquitetura de software mais adequada, ou seja, este método não cria uma arquitetura a partir do ínicio, devem existir arquiteturas candidatas prontas para a escolha, porém a escolha é realizada com base nos atributos de qualidade que foram definidos pelas partes interessadas. A partir do momento que os atributos de

qualidade são definidos, os mesmos passam por uma avaliação e em seguida os atributos preferenciais são escolhidos, e a arquitetura de software, que atender os atributos de qualidade definidos pelas partes interessadas, será a escolhida.

Técnicas utilizadas. Cáculos matemáticos que vão dar origem aos principais atributos de qualidade que deverão compor a arquitetura escolhida, conforme apresentações na seção 3.2.1.6 Determinação do peso de preferência. E também na seção 3.2.1.7 Conversão dos Valores dos Atributos de qualidade para as Arquiteturas Candidatas. E por fim na seção 3.2.1.8 Cálculo das pontuações cumulativas.

Maturidade do método. Osterlind et al. (2013) realizam a avaliação de uma arquitetura empresarial utilizando os conceitos teóricos apresentados no método escrito por Zayaraz e Thambidurai (2005), desta forma, percebe-se que o método de Zayaraz e Tambidurai (2005) pode ser considerado maduro, pois já foi fruído no artigo de Osterlind et al. (2013) publicado pela IEEE.

Validação do método. O AMDAQ é um método validado por Zayaraz e Thambidurai (2005), pois através de um exemplo prático, os autores aplicam e validam os resultados do método.

4.4 Avaliação do Método de Comparação de Arquiteturas, Baseado na ISO/IEC-9126-1 (1998) que Especifica os Atributos de Qualidade.

A seguir será apresentado um resumo da avaliação do **CA** (Método de Comparação de Arquiteturas, Baseado na ISSO/IEC-9126-1(1998) que Especifica os Atributos de Qualidade). A partir do *framework* sugerido na Tabela 14. A resposta para cada elemento do *framework* é definida abaixo conforme as caraterísticas do CA:

Definição de arquitetura de software. O método não apresenta uma definição particular de arquitetura.

Metas específicas. Conforme Losavio et al. (2003) o método apresentado, auxilia na escolha da arquitetura de software mais adequada entre as candidatas, com base nos atributos de qualidades mais notáveis, apresentados pela ISO/IEC 9126-1 (1998).

Atributos de qualidade. De acordo com Losavio et al. (2003) o CA apresenta seis atributos de qualidade que devem ser levados em consideração na escolha da arquitetura, a funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Losavio et al. (2003) enfatizam que os sub-grupos dos atributos de qualidade podem ser levados em consideração na escolha da arquitetura candidata.

Fase aplicável. É aplicável após a definição dos requisitos funcionais e dos atributos de qualidade e antes da fase de implementação, ou seja, ele deve estar após a fase de definição de requisitos e antes da implementação do código fonte. (Fase da representação da arquitetura de software).

Entrada e saída. Conforme Losavio et al. (2003) a entrada do método depende da correta definição dos mais prioritários requisitos funcionais e não funcionais, a partir da priorização dos requisitos mais importantes é possível consolidar os atributos de qualidade. A saída do método é a eleição de uma arquitetura adequada, que satisfaça os atributos de qualidade que foram definidos.

Domínio da aplicação. Não informado.

Benefícios. Conforme Losavio et al. (2003) o método permite uma associação simples com o RUP, ou seja, ele poderia fazer parte do processo de desenvolvimento de software informado no RUP. Nesta mesma direção Losavio et al. (2003) enfatizam que o método apresenta um auxilio valioso na tipificação dos atributos de qualidade**.**

Partes interessadas. O contexto do método CA apresentado por Losavio et al. (2003) ressalta que a qualidade pode ser visualizada a partir de diferentes partes interessadas com diferentes perspectivas, de tal forma, o usuário, o desenvolvedor, e o gerente podem apresentar necessidades diferentes em relação ao sistema de software. Contudo as partes interessadas podem ser diferentes, isso depende do contexto do projeto de software.

Suporte ao processo. Conforme Losavio et al. (2003) cenários e outras abordagens que permitem um refinamento mais profundo podem ser aplicadas nos atributos de qualidade identificados, porém Losavio et al. (2003) não apresentam um exemplo com a descrição completa dos cenários que podem ser aplicados, contudo eles mencionam a aplicação de cenários oriundas de Kazman; Klein e Clements (2000) que utilizaram a técnica na aplicação do ATAM.

Questões sócio técnicas. Não informado.

Recursos necessários. Não informado.

Atividades do método. Conforme Losavio et al. (2003) a execução do CA, depende da execução das seguintes atividades:

> 1. Analise os principais requisitos funcionais e não funcionais do sistema, para que seja possível estabelecer os requisitos de qualidade e as metas de qualidade.
> 2. Use o modelo de qualidade ISO/IEC-9126-1 (1998) para definir a arquitetura, ele deve ser utilizado como um *framework*. Algumas métricas podem ser escritas com base em componentes específicos ou conectores.
> 3. Apresente as primeiras arquiteturas candidatas.
> 4. Construa uma tabela de comparação para as arquiteturas candidatas.
> 5. Priorize as características de qualidade, tendo em conta os requisitos de qualidade do sistema e as metas de qualidade. A personalização da ISO 9126-1 para o domínio do problema pode ser usada para organizar hierarquicamente as características.
> 6. Analisar os resultados apresentados na tabela, de acordo com as prioridades obtidas no passo 5.
> 7. Escolha a arquitetura candidata inicial, entre as outras arquiteturas candidatas que foram avaliadas, com base na análise anterior.
> 8. Se for necessária uma análise mais fina, cenários ou abordagens que se baseiam em perfil poderiam ser utilizadas, considerando apenas as características de qualidade relevantes para o domínio do problema obtido no passo 5. (LOSAVIO et al., 2003, p.147, tradução nossa).

Descrição da arquitetura. Conforme Losavio et al. (2003) visões são utilizadas para representar a arquitetura, as visões apresentadas por Losavio et al. (2003) são baseadas nas visões do usuário do sistema, e do arquiteto do sistema, com na base na visão do usuário e do arquiteto, a arquitetura é modelada.

Abordagens de avaliação. Os atributos de qualidade selecionados são baseados no modelo de qualidade apresentado pela ISO/IEC 9126-1 (1998), conforme apresentado na seção 3.3 Método de comparação de arquiteturas, baseado na ISO-9126-1 que especifica os atributos de qualidades. E podem ser refinados a partir da utilização de cenários, que são apresentados por Kazman; Klein e Clements (2000).

Ferramenta de apoio. Os autores do método não apresentam ferramentas que podem auxiliar na utilização do método.

Detalhamento. O detalhamento deste método está na forma de identificar a arquitetura de software mais adequada, ou seja, este método não cria uma arquitetura a partir do ínicio, devem existir arquiteturas candidatas prontas para a escolha, porém a escolha é realizada com base nos atributos de qualidade que foram definidos com base na ISO/IEC 9126-1 (1998). A partir do momento que os atributos de qualidade são definidos, os mesmos passam por uma avaliação e em seguida as arquiteturas candidatas são comparadas com os atributos de qualidade, ou seja, é verificada qual a arquitetura que satisfaz de forma mais adequada os

atributos de qualidade que foram definidos, ao final a arquitetura que atender as metas estabelecidas é selecionada.

Técnicas utilizadas. Modelo de qualidade apresentado pela ISO/IEC-9126-1 e opcionalmente a criação de cenários apresentados por Kazman; Klein e Clements (2000).

Maturidade do método. É um método maduro. Losavio (2002) utiliza o CA, apresentado por Losavio et al. (2003) no refinamento dos atributos de qualidade que irão compor a arquitetura de software selecionada, ambos os artigos foram publicados pelo *Journal Of Object Technology* (Jornal da Tecnologia de Objetos).

Validação do método. O CA é um método validado pelos próprios autores, ou seja, Losavio et al. (2003), apresentam um estudo de caso com a aplicação do método, neste estudo de caso apresentado por Losavio et al. (2003) eles aplicam o método em um sistema de monitoramento da bolsa de valores, e em seguida eles apresentam os resultados da aplicação da técnica neste estudo de caso.

4.5 Comparação Entre os Métodos

Nesta seção é apresentada a comparação dos métodos após a aplicação do *framework* de avaliação de metodologias, os métodos que serão comparados são os seguintes:

- Método 1: ADD
- Método 2: Análise de múltiplas decisões de atributos de qualidade utilizando equivalência hipotética (AMDAQ)
- Método 3: Comparação de arquiteturas (CA)

Para preencher a Tabela 15 será utilizada a seguinte nomenclatura:

- ✓ **Atende =** o elemento é informado no método e é coberto nas atividades do método.
- ✓ **Não atende =** o elemento não é informado no método.

A Tabela 15 apresenta um resumo da comparação individual das análises apresentadas nas seções 4.2, 4.3 e 4.4

Tabela 15 - Comparação elementar dos métodos ADD, AMDAQ, CA.

Componente	Elemento	Método 1 (ADD)	Método 2 (AMDAQ)	Método 3 (CA)
Contexto	**Definição de arquitetura de software**	Atende	Não atende	Não atende
	Meta especifica	Atende	Atende	Atende
	Atributos de qualidade	Atende	Atende	Atende
	Fase aplicável	Atende	Atende	Atende
	Entrada e saída	Atende	Atende	Atende
	Domínio da aplicação	Atende	Não atende	Não atende
Partes interessadas	**Benefícios**	Atende	Atende	Atende
	Partes interessadas	Atende	Atende	Atende
	Suporte ao processo	Atende	Atende	Atende
	Questões sócio técnicas	Atende	Não atende	Não atende
	Recursos necessários	Não atende	Não atende	Não atende
Conteúdo	**Atividades do método**	Atende	Atende	Atende
	Descrição da arquitetura de software	Atende	Não atende	Atende
	Abordagens de avaliação	Atende	Atende	Atende
	Ferramenta de apoio	Não atende	Não atende	Não atende
	Detalhamento	Atende	Atende	Atende
	Técnicas utilizadas	Atende	Atende	Atende
Confiabilidade	**Maturidade do método**	Atende	Atende	Atende
	Validação do método	Atende	Atende	Atende

Fonte: Babar e Gorton (2004, p.2, tradução nossa, criada por nós).

4.5.1 Contexto

Conforme Rozanski e Woods (2012) a partir do momento que a arquitetura do software está bem definida, a compreensão do que está sendo construído torna-se mais simples para as partes interessadas.

Em relação ao elemento **definição de arquitetura do software**, o ADD apresenta uma definição da arquitetura, nesta direção o AMDAQ, não apresenta uma definição particular da arquitetura. Na mesma direção o CA, baseado na ISO/IEC-9126-1 (1998) que especifica os atributos de qualidades, não apresenta uma definição particular da arquitetura.

Ao tratar sobre o elemento **metas específicas** do ADD é possível perceber que elas diferem das metas do AMDAQ e do CA. A meta principal do ADD é apresentar um esboço da arquitetura do software, ou seja, visões arquiteturais do elemento que foi escolhido para ser projetado com base nos ASRs, a partir da definição e seleção de uma arquitetura com base nos atributos de qualidade. Contudo, a **meta específica** do AMDAQ tem o objetivo de selecionar uma

arquitetura de software adequada com base nos atributos de qualidade esperados. A **meta específica** do CA é selecionar uma arquitetura com base nos atributos de qualidade, com base no modelo de qualidade apresentado pela ISO/IEC-9126-1 (1998). Porém, os três métodos permitem selecionar uma arquitetura com base nos atributos de qualidade, embora os três métodos apresentam abordagens diferentes para atingir o resultado final.

Em relação aos **atributos de qualidade** uma das tarefas embutidas no ADD é a utilização de táticas. As táticas são aplicadas em sete atributos de qualidade, que são: disponibilidade, interoperabilidade, modificabilidade, desempenho, segurança, testabilidade e usabilidade. Neste contexto é recomendável que as táticas sejam aplicadas para estes atributos de qualidade, pois foram estes que foram utilizados por Bass; Clements e Kazman (2012) na aplicação original das táticas. Nesta mesma direção o AMDAQ, pode conter múltiplos atributos de qualidade, pois está baseado na preferência das partes interessadas, porém, vale ressaltar que Zayaraz e Thambidurai (2005) executam o método a partir dos seguintes atributos de qualidade: manutenabilidade, confiabilidade, tempo de resposta, funcionalidade, usabilidade, compreensibilidade. O CA utiliza como referência a ISO/IEC-9126-1 (1998). Porém, vale ressaltar, que os três métodos podem lidar com múltiplos atributos de qualidade, sendo que o ADD é o mais completo.

A **fase de aplicação** dos métodos está aderente à ênfase apresentada por Rozanski e Woods (2012) e Babar e Gorton (2004) que enfatizam que a localização da arquitetura do software deve estar entre as fases de especificação dos requisitos e construção do sistema. O ADD, o AMDAQ, e o CA, devem ser aplicados, após os requisitos serem definidos até a fase de projeto. A situação em comum entre os três métodos é que todos somente podem ser aplicados quando os requisitos já estão definidos, ou que ao menos exista clareza sobre os requisitos a serem contemplados por eles.

O ADD e o CA, apresentam **entradas** semelhantes, ambos os métodos somente podem iniciar quando os principais requisitos forem definidos. Contudo, o AMDAQ, não apresenta de forma explícita as **entradas** do método, embora Rozanski e Woods (2012) afirmem que seja necessário definir os requisitos antes de começar a construir a arquitetura. Uma característica em comum entre os três métodos é que as **saídas** apresentadas pelos três referem-se a uma arquitetura que

atende aos atributos de qualidade definidos e as expectativas das partes interessadas.

Os autores dos métodos não fazem restrições ao **domínio de aplicação** dos métodos, sendo que o objetivo final é apresentar uma arquitetura adequada aos requisitos e atributos de qualidade, porém os autores dos CA e do AMDAQ, não informam explícitamente o domínio de aplicação dos métodos.

4.5.2 Partes Interessadas

Os **benefícios** apresentados pelos métodos ADD, e pelo AMDAQ e pelo CA, são distintos entre si, porém uma característica em comum deve ser destacada, os três métodos procuram apresentar uma arquitetura de software que atenda as expectativas das partes interessadas com base nas exigências que foram definidas.

Ao analisar-se o ADD, e o AMDAQ, e o CA é possível perceber que os três métodos não restringem os perfis das **partes interessadas** que farão parte do processo de definição da arquitetura. Diversas **partes interessadas** com diferentes papéis podem fazer parte do processo de seleção da arquitetura, entretanto, o ADD ressalta as partes interessadas mais comuns, o arquiteto de software, o gerente de projeto, o usuário final e o cliente.

O ADD oferece **suporte ao processo** a partir de um conjunto de técnicas auxiliares, que fazem parte do processo de avaliação, dentre as técnicas apresentadas, pode se destacar a técnica de criação de cenários. A técnica de criação de cenários também é mencionada pelo CA, porém a técnica de cenários mencionada no CA é diferente da técnica de cenários utilizada no ADD, ou seja, Losavio et al. (2003) referem-se aos cenários apresentados por Kazman; Klein, e Clements (2000) na execução do ATAM, porém o detalhamento destes cenários não são apresentados por Losavio et al. (2003). Ambos os métodos podem usar cenários para identificar as respostas que o sistema deve retornar para determinados estímulos. Entretanto o AMDAQ, não apresenta técnicas auxiliares que podem apoiar a execução do método. Vale ressaltar que o ADD, fornece um número maior de técnicas de apoio durante o processo de definição e seleção da arquitetura, em relação ao CA.

Babar e Gorton (2004) enfatizam que o elemento referente as **questões sócio técnicas**, impactam de forma expressiva a avaliação da arquitetura de

software, estes assuntos não técnicos, podem ser fatores políticos, preocupações gerenciais, investimento financeiro entre outros.

Somente o ADD apresenta um tratamento para assuntos não técnicos, ou seja, conforme Bass; Clements e Kazman (2012) os ASRs que são apresentados no ADD, são impactados pelas metas de negócio da organização. Embora a arquitetura seja influenciada por assuntos não técnicos, com exceção do ADD, os outros métodos não apresentam um tratamento para assuntos não técnicos.

Em relação ao elemento **recursos necessários**, nenhum dos métodos apresentados fornece informações sobre custo ou quantidade de recursos necessários para a execução.

4.5.3 Conteúdo

Todos os métodos possuem um conjunto de **atividades definidas**. Um **conjunto de atividades** distintas são executadas pelos métodos apresentados, embora algumas atividades possam apresentar alguma semelhança, nem todas tem a mesma finalidade, por exemplo, o ADD, apresenta a técnica de geração de cenários para auxiliar na identificação dos atributos de qualidade que serão gerados, a partir dos estímulos que chegam no sistema, contudo, outra técnica de criação de cenários pode ser utilizada no CA, ou seja, a técnica utilizada no ADD é a técnica apresentada na seção “3.1.1.1 Criação de cenários”, por outro lado a técnica utilizada no CA é a apresentada por Kazman; Klein, e Clements (2000) na execução do ATAM. Porém, a forma de utilizar esta técnica no ADD está claramente definida, ou seja, um conjunto de tarefas padronizadas estão incorporadas no processo de criação de cenários que o ADD utiliza. No CA é permitida a utilização de cenários para refinar a análise dos atributos de qualidade, porém nenhuma instrução detalhada é oferecida, de maneira que permita ilustrar como se constrói um cenário. Neste contexto, vale ressaltar que o AMDAQ, não informa a necessidade de criação de cenários para a seleção de uma arquitetura de software com base nos atributos de qualidade.

A identificação dos atributos de qualidade no ADD é realizada da seguinte forma: os requisitos mais significativos ASRs são escolhidos para fazer parte da arquitetura logo no início, assim, o ADD pode ser iniciado mesmo que todos os requisitos ainda não tenham sido definidos. Entretanto o AMDAQ informa que os

atributos de qualidade precisam ser definidos pelas partes interessadas, e diferentemente do ADD, pesos são atribuídos para identificar os atributos de qualidade mais significativos com base nas expectativas das partes interessadas. Neste contexto, o CA, difere dos outros métodos, pois utiliza um conjunto de atributos de qualidade definidos pela norma ISO/IEC 9126-1 (1998) como base de averiguação da arquitetura, ou seja, as arquiteturas de software são comparadas, a partir dos atributos de qualidade apresentados pela norma.

Nenhum dos métodos apresentados exibem uma linguagem de **descrição da arquitetura** ou ADL (*Architectural Description Language)*, porém visões arquiteturais são utilizadas pelos métodos apresentados. De acordo com Babar e Gorton (2004, p.7, tradução nossa) “O tipo e a quantidade de visões podem variar, pois dependem do método que foi aplicado”.

O ADD apresenta um esboço de uma visão arquitetural ao final das iterações necessárias para definir a arquitetura.

O AMDAQ, não informa um detalhamento da visão, porém é ressaltado por Zayaraz e Thambidurai (2005) que a visão utilizada no método é a visão das partes interessadas, ou seja, as expectativas e a seleção da arquitetura é realizada com base no que as partes interessadas esperam, com base nas visões que as mesmas tem em relação ao sistema.

O CA, não informa um detalhamento da visão, porém é mencionado que diferentes pontos de vistas, oriundos da visão do usuário e do arquiteto de software, são utilizados na seleção da arquitetura.

As **abordagens de avaliação** utilizadas pelos métodos apresentados exibem diferenças na forma de selecionar a arquitetura mais adequada, estas diferenças correspondem à maneira que a avaliação é realizada, por exemplo, o ADD, realiza a definição e seleção da arquitetura, a partir do momento que os ASRs estão definidos, neste momento os cenários, táticas e listas de verificação já foram criados.

O AMDAQ realiza cálculos atribuindo pesos aos atributos escolhidos pelas partes interessadas, os atributos de qualidade escolhidos devem estar presentes na arquitetura selecionada.

O CA utiliza um modelo baseado na norma ISO/IEC 9126-1 (1998) que específica os atributos de qualidade, a seleção da arquitetura é realizada com base nos atributos informados pela norma, ou seja, a arquitetura que atende de forma mais adequada os atributos de qualidade definidos pela norma é escolhida.

Em relação ao elemento **ferramenta de apoio**, os métodos apresentados não oferecem uma ferramenta de apoio que possa ser utilizada durante a execução das atividades propostas.

O **detalhamento** refere-se às características individuais dos métodos, ou as particularidades que o distinguem dos outros métodos, por exemplo, o AMDAQ, e o CA, precisam de arquiteturas candidatas como entrada para o início da execução das atividades dos métodos, ou seja, as arquiteturas precisam existir e estarem prontas, para consequentemente serem avaliadas e depois selecionada a mais adequada. No ADD isto não ocorre da mesma forma, no ADD, são definidos os ASRs, selecionadas as táticas, definidos os cenários de qualidade e executada uma lista de verificação para compor uma arquitetura que atenda aos atributos de qualidade requeridos. No ADD a seleção não é da arquitetura em si, mais existem técnicas que permitem tomar a decisão de seleção mais cedo que as outras, não é preciso ter a arquitetura definida para esta finalidade.

As **técnicas utilizadas** pelos métodos são distintas, conforme apresentado abaixo:

- ADD - (Cenários, táticas, listas de verificação).
- AMDAQ – (Determinação do peso de preferência, conversão dos valores dos atributos de qualidade para as arquiteturas candidatas, cálculo das pontuações cumulativas).
- CA – (Modelo de qualidade apresentado pela ISO/IEC-9126-1 e opcionalmente a criação de cenários apresentados por Kazman; Klein e Clements (2000).

Um item a ser observado é que a criação de cenários é uma técnica que faz parte da definição e seleção da arquitetura utilizada pelo ADD, contudo, o CA apresenta a técnica de criação de cenários como opcional, embora as técnicas de criação de cenários apresentadas no ADD e no CA serem distintas, a criação de cenários auxilia na identificação dos atributos de qualidade que deverão ser considerados na arquitetura, todavia, o AMDAQ, não apresenta a técnica de criação de cenários, a definição dos atributos é baseada em cálculos e na opinião das partes interessadas.

4.5.4 Confiabilidade

De acordo com Jeffery; Babar e Zhu (2004, p.3, 2004, tradução nossa) “a quantidade de publicações em revistas acadêmicas, estudos de casos, livros sobre o assunto e a aplicação prática em diferentes contextos de sistemas de software”, auxiliam na validação da maturidade do método.

O critério utilizado para definir a maturidade e validação dos métodos, foi baseado nas publicações e aplicações dos métodos por outros autores.

O ADD é um método maduro e validado, pois foi aplicado por outros autores e os resultados obtidos foram coerentes com as saídas esperadas. Na mesma direção o AMDAQ, é um método maduro e validado, o contexto do método foi utilizado por outros autores em avaliações de arquiteturas empresariais.

O CA é um método validado e maduro, o método foi utilizado por outros autores durante o processo de refinamento dos atributos de qualidade que serão inseridos na arquitetura de software.

4.5.5 Recomendações Para Uso dos Métodos Avaliados

A Tabela 15 apresenta as diferenças dos métodos, a partir dos questionamentos representados na Tabela 14, porém a Tabela 15 apresenta a diferença de forma macro.

A partir da análise dos elementos e respostas oriundas da aplicação do *framework* na comparação dos métodos ADD, AMDAQ e CA é possível obter um direcionamento sobre qual método escolher, ou seja, a avaliação individual dos métodos apresenta as diferenças que influenciam a decisão por um método ou outro.

Ao analisar-se o ADD após a aplicação do *framework* de comparação é possível perceber que o ADD é um método que é direcionado para aqueles que vão criar a arquitetura a partir do ínicio, pois no ADD a arquitetura é criada a partir do ínicio, ou seja, a arquitetura é definida dentro do método, através de uma sequência definida de atividades que devem ser seguidas para alcançar uma visão arquitetural, a partir de um esboço que será criado na finalização dos passos definidos pelo método. Ao analisar-se a Tabela 15 é possível perceber que o ADD é o método que

apresenta maior completude em relação aos outros, ou seja, ele atende a maioria dos elementos que foram definidos na Tabela 14.

Ao analisar-se o AMDAQ após a aplicação do *framework*, é possível perceber que este método é direcionado para aqueles que precisam tomar a decisão de escolher entre uma arquitetura de software ou outra, e dependem da opinião das partes interessadas para tomar esta decisão, porém a opinião das partes interessadas é baseada na intuição das mesmas, ou seja, os atributos de qualidade que serão eleitos para compor a arquitetura escolhida serão ponderados com base na intuição das partes interessadas sobre a arquitetura de software, que deve estar mais aderente às necessidades apresentadas pelo sistema.

Entretanto ao analisar-se o CA após a aplicação do *framework* é possível perceber que este método é direcionado para aqueles que preferem seguir uma regra ou um padrão que define os atributos de qualidade, ou seja, o CA pode auxiliar o arquiteto a selecionar a arquitetura de software mais adequada a partir de atributos de qualidade documentados e reconhecidos pela norma ISO/IEC 9126-1 (1998). Vale ressaltar que o CA permite selecionar a arquitetura de software mais adequada com base nos atributos e sub atributos de qualidade definidos pela norma ISO/IEC 9126-1 (1998), nesta direção, é possível compreender que arquiteturas candidatas deverão estar disponíveis para eleição, e consequentemente a avaliação é realizada, o resultado final está baseado na escolha da arquitetura de software mais aderente aos atributos de qualidade definidos pela norma ISO/IEC 9126-1 (1998).

A aplicação do NIMSAD customizado por nós e Babar e Gorton (2004) na Tabela 14, auxilia na tomada de decisão entre um método ou outro a partir das necessidades que precisam ser atendidas pelo método proposto, ou seja, a avaliação está baseada em quatro componentes que são: Contexto, Partes interessadas, Conteúdo e Confiabilidade. Cada componente contém elementos coerentes ao domínio do problema que o *framework* deve responder, e cada elemento representa um questionamento que nada mais é do que uma necessidade que deve ser respondida pelo método que está sendo avaliado.

O *framework* customizado por nós e Babar e Gorton (2004) auxilia na identificação das vantagens e desvantagens dos métodos que foram analisados, ou seja, dado que os elementos representam os questionamentos oriundos das necessidades definidas pelo avaliador, o método pode contemplar a necessidade definida ou não. Nesta direção percebe-se que ao realizar a análise dos métodos é

possível identificar diferentes respostas para o mesmo elemento, pois as respostas são oriundas de métodos distintos, desta forma, um elemento definido na Tabela 14, pode conter diferentes respostas.

A identificação das diferentes respostas para os mesmos elementos auxilia na identificação das vantagens e desvantagens de cada método, conforme o que se espera de cada um, ou seja, as diferenças identificadas são claras e permite que a escolha seja realizada com base nas respostas apresentadas por cada método. Um detalhe a ser ressaltado neste contexto refere-se à decisão de escolha, o *framework* apesentado na Tabela 14, auxilia na escolha a partir da comparação dos métodos na identificação das vantagens e desvantagens de cada um a partir da perspectiva do avaliador, com base nos elementos (questionamentos) que foram definidos para os mesmos, porém, não é informado qual o método deve ser escolhido, pois este tipo de decisão não está no contexto do *framework*, ou seja, o *framework* apresentado na Tabela 14 auxilia a tomar esta decisão, porém esta decisão depende das preferências das partes interessadas, das preferências do avaliador, do domínio do problema em questão.

4.6 Considerações do Capítulo

A utilização do *framework* apresentado na Tabela 14 apresenta perguntas que fazem parte do contexto, cobertura, e conteúdo, dos métodos que foram avaliados nas seções 4.2, 4.3 e 4.4, as perguntas apresentadas no *framework* que é apresentado na Tabela 14 permitem que o avaliador compare as diferenças, vantagens e desvantagens entre os métodos que são avaliados.

A aplicação do método apresentada a partir da seção 4.5 tem o objetivo de mostrar as diferenças entre os métodos que estão sendo avaliados, porém, as vantagens e as desvantagens entre os métodos comparados não são apresentadas, pois o objetivo da aplicação do *framework* apresentado na Tabela 14 é apresentar de forma prática como a comparação dos métodos é realizada, entretanto após a aplicação do *framework,* é possível saber as vantagens e desvantagens dos métodos, a partir do que é esperado de cada um, com base nos elementos (questionamentos) definidos. Nesta direção Kitchenham e Babar (2007) realizam *tradeoffs* após a aplicação dos métodos, de forma a permitir que o método mais adequado seja selecionado para um determinado fim.

Vale ressaltar que a possibilidade de identificar as vantagens e desvantagens dos métodos apresentados e consequentemente a escolha do melhor método vai depender dos interesses do avaliador e das partes interessadas.

A aplicação do *framework* não apresenta diferenças com níveis de detalhes amplamente técnicos para os métodos apresentados, ele apresenta as diferenças a nível conceitual, por este motivo uma avaliação baseada no NIMSAD pode ser aplicada para diferentes métodos em diferentes contextos da engenharia de software, pois a comparação é realizada com base em conceitos e as perguntas apresentadas são definidas com base no que é esperado do método que está sendo avaliado, ou seja, as perguntas que são elaboradas devem ser perguntas que auxiliam o avaliador identificar diferenças, semelhanças, vantagens e desvantagens nos métodos que estão sendo comparados com o *framework*.

É possível perceber que entre os métodos que foram avaliados, o método que apresenta o maior número de recursos e vantagens é o ADD, ele é um método bem estruturado com técnicas de apoio que auxiliam no refinamento dos requisitos que vão compor a arquitetura. Portanto o *framework* de avaliação apresentado na Tabela 14 pode auxiliar os avaliadores ou as partes interessadas a identificar o método mais adequado, a partir da comparação das respostas apresentadas para cada método.

Nesta direção é possível identificar o método mais adequado para um domínio.

5. CONSIDERAÇÕES FINAIS

Este capítulo apresenta os resultados obtidos com o trabalho, pontos positivos e negativos da aplicação do NIMSAD, as contribuições e trabalhos futuros.

5.1 Pontos Positivos da Aplicação do Método

O *framework* adaptado do NIMSAD apresentado, auxilia na identificação do método mais adequado, possibilitando analisar vantagens e desvantagens que possam existir entre os métodos, contudo, é importante ressaltar que o *framework* apresentado é abrangente na avaliação dos métodos.

Conforme Jayaratna apud Matinlassi (2004) o NIMSAD é um *framework* genérico e pode ser utilizado para avaliar diferentes métodos na engenharia de software, o *framework* pode ser customizado segundo o contexto em que ele será aplicado, de tal maneira, é possível criar evoluções, ou seja, customizações podem realizadas no NIMSAD, esta customização auxilia o executor a adaptar o *framework* ao seu domínio.

Com a aplicação do método foi possível identificar as diferenças entre os seguintes métodos:

- ADD;
- AMDAQ;
- CA;

Estes três métodos apresentam objetivos com características semelhantes, a saída final dos três métodos é uma arquitetura de software que atenda aos atributos de qualidade que foram definidos pelas partes interessadas.

O ADD deve ser utilizado quando existe a necessidade de definir a arquitetura a partir do ínicio através de um conjunto de técnicas que permitem validar os atributos de qualidade e consequentemente apoiam a seleção da arquitetura do software.

O AMDAQ deve ser utilizado quando a decisão pela escolha da arquitetura depende das partes interessadas no projeto do software, ou seja, as partes interessadas informam os atributos de qualidade que devem compor a arquitetura, e consequentemente pesos são atribuídos aos atributos de qualidade que foram

definidos pelas partes interessadas e consequentemente, os atributos de qualidade mais significativos com base nos pesos, são escolhidos pelo arquiteto de software.

O CA deve ser utilizado quando os atributos de qualidade ainda não foram definidos com eficácia, ou seja, a necessidade de um guia, ou uma norma pode dirigir a escolha da arquitetura mais adequada, o CA possibilita o uso de um modelo de qualidade definido para a escolha da arquitetura mais adequada com base nos atributos de qualidade que apresentam maior aderências ao modelo de qualidade padrão.

A aplicação do NIMSAD customizado apresentado na Tabela 14 auxiliou na identificação das diferenças entre os métodos. A identificação das diferenças permite que o avaliador identifique as vantagens e as desvantagens nos métodos que estão sendo comparados, possibilitando que a escolha seja realizada com base em uma avaliação formal, com um *framework* de avaliação aceito, testado e validado em outros contextos de avaliação de métodos. A aplicação do NIMSAD customizado apresentado na Tabela 14, orienta na tomada de decisões, auxiliando os avaliadores (aplicadores do método) a escolher o método que mais satisfaça os objetivos e as metas a serem alcançadas.

5.2 As Limitações da Aplicação do Método

O NIMSAD pode ser customizado, sendo que a adaptação apresentada na Tabela 14 é um exemplo desta customização. Um problema que pode surgir ao customizar o método de avaliação é a apresentação de possíveis imprecisões, pois as perguntas não foram avaliadas em diferentes contextos.

Ao aplicar o método adaptado é possível perceber que as perguntas são elaboradas com base no contexto do problema que prentende-se solucionar, contudo, o método não apresenta uma forma de avaliar se as perguntas são precisas, o avaliador precisa compreender de forma única e objetiva o que realmente precisa ser comparado e avaliado. Somente desta forma as perguntas corretas podem ser elaboradas. Caso o avaliador não tenha precisão na elaboração das perguntas, falhas podem influenciar de forma negativa a decisão de escolha de um melhor método. Desta forma, é possível compreender que a execução do método depende de um claro entendimento do que deve ser comparado.

5.3 Contribuições do Trabalho

Durante a realização do trabalho foi assumido o desafio de apresentar como uma comparação de métodos de seleção de arquitetura pode ser realizada utilizando um *framework* customizado a partir do NIMSAD.

Neste trabalho foram apresentadas as definições de atributos de qualidade, arquitetura de software, padrões arquiteturais, visões arquiteturais e consequentemente métodos que permitem selecionar a arquitetura candidata mais adequada foram apresentados.

Em um projeto de software, diferentes necessidades são oriundas das partes interessadas comprometidas com o sistema que está sendo elaborado, pois existem expectativas que precisam ser atendidas para que o software que está sendo construído possa contemplar os pontos de vistas das partes interessadas no projeto. Entretanto, surge a necessidade de apresentar um modelo que possa atender aos requisitos e as necessidades oriundas das partes interessadas, este modelo é a arquitetura do software, nela devem estar contemplados os requisitos e as expectativas das partes interessadas. A arquitetura pode ser criada, a partir de um desenho, esboço, rascunho, porém, a seguinte pergunta pode surgir para um arquiteto, após o mesmo identificar as arquiteturas iniciais de um sistema:

- Como criar/identificar/selecionar a melhor arquitetura com base nos requisitos, restrições e atributos de qualidade?
- Existe um método ou *framework* que possa auxiliar na identificação das vantagens e desvantagens entre um método de seleção de arquitetura e outro?

Este trabalho pretendeu solucionar estas questões. O capítulo 3 apresentou os métodos que permitem selecionar uma arquitetura de software que satisfaça os atributos de qualidade, definidos pelas partes interessadas. O capítulo 4 apresentou um *framework* que auxilia o arquiteto a identificar as vantagens e as desvantagens dos métodos apresentados no capítulo 3.

O *framework* apresentado permite identificar o método de seleção de arquitetura software mais adequado, a partir dos questionamentos definidos na avaliação. Uma característica que deve ser ressaltada é que o *framework* apresentado é abrangente e pode ser utilizado como referência para avaliação de

outros métodos em diferentes domínios, porém, as perguntas podem ser alteradas conforme o domínio do que será avaliado.

Conforme Jayaratna apud Matinlassi (2004) o *framework* NIMSAD pode ser utilizado na comparação de métodos em diferentes domínios.

Portanto, é possível compreender que a customização do NIMSAD apresentada neste trabalho também pode ser aplicada em métodos de categorias diferentes, desta forma é possível identificar que o *framework* customizado do NIMSAD utilizado por nós na Tabela 14, herda as mesmas características do método original, o NIMSAD. Nesta direção é possível ressaltar que o método é adaptável a diferentes domínios, tal situação pode auxiliar na seleção de métodos diferentes dos apresentados no capítulo 3, ou dos métodos avaliados por Dobrica e Niemelã (2002).

Em virtude dos fatos mencionados pode-se concluir que o *framework* apresentado neste trabalho auxilia o arquiteto, avaliador, aplicador do método, ou qualquer outra parte interessada a realizar a comparação de métodos com finalidades semelhantes, desta forma, é possível identificar quais são os pontos fortes e fracos de cada método, esta situação, direciona o executor do *framework* a selecionar o método mais adequado, com base nas vantagens identificadas que corresponderem a melhor abordagem para o domínio que está sendo avaliado.

5.4 Trabalhos Futuros

Com base nas leituras e estudos realizados durante a elaboração deste trabalho, novas propostas de estudos surgiram para trabalhos futuros, tais propostas seguem abaixo:

- Um estudo com aplicação prática do ADD em um sistema de dispositivo móvel;
- Um estudo de avaliação e seleção de padrões arquiteturais baseado em atributos de qualidade para um sistema de software;
- Aplicação real de uma arquitetura de software no processo de desenvolvimento de software em um sistema de dispositivo móvel de localização de vagas.

REFERÊNCIAS

AVISON, D.; FITZGERALD, G. **Information Systems Development: Methodologies, Techniques and Tools**. 2.ed. London: McGraw-Hill International, 1995, p.505.

BABAR, A.B.; GORTON, I. **Comparison of Scenario-Based Software Architecture Evaluation Methods**. In: PROCEEDINGS OF THE 11TH ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, p.600-607, Busan, 2004. Disponível em: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1371976&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1371976 Acesso em 05 jan. 2015.

BASS; L.; CLEMENTS, P.; KAZMAN, R. **Sofware Architecture in Pratice**. 3. ed. Massachusetts: Pearson Education, 2012. 589 p.

BRASIL. Instituto Tecnológico da Aeronaútica. **Construção de gráficos**. [entre 1996 e 2015]. Disponível em: http://www.fis.ita.br/labfis24/grafic/textos_graf/graf_texto1.htm>. Acesso em 19 jun.2015.

CHUNG, L.; BRIAN, A. N.; ERIC, Y.; MYLOPOULOS, J. **Non-Functional requirements in software enginnering**. 1. ed. New York: Springer Science+Business Media, 2000.

DOBRICA, L.; NIEMELÃ, E. **A Survey on Software Architecture Analysis Methods**. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, v.28, n.7, p.638-653. [S.L]: IEEE, 2002. Disponível em: < http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1019479&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F32%2F21927%2F01019479.pdf%3Farnumber%3D1019479> Acesso em: 12 mai. 2015.

ESTADOS UNIDOS. Software Engineering Institute. **Attribute-Driven Design Method**. Carnegie Mellon University, [entre 2009 e 2012]. Disponível em: < http://www.sei.cmu.edu/architecture/tools/define/add.cfm>. Acesso em 02 jun. 2015.

FORSELL, M.; HALTTUNEN, V.; AHONEN, J.; PENJAM, J. **Evaluation of component-based software development methodologies**. In: PROCEEDINGS 57 OF THE FENNO-UGRIC SYMPOSIUM ON SOFTWARE TECHNOLOGY. Tallin: Tallin Technical University, 1999.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, INTERNATIONAL ELECTRONICAL COMMISION. **ISO/IEC: FCD 9126-1.2**: Information Technology - Software Product Quality. Part 1: Quality Model, 1998.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, INTERNATIONAL ELECTRONICAL COMMISION. **ISO/IEC FDIS 25010.** System and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality. Geneva, 2010. 34 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, INTERNATIONAL ELECTRONICAL COMISSION, INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS. **ISO/IEC/IEEE 42010**. Systems and software enginnering – Architecture description. Geneva: 2011. 37 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, INTERNATIONAL ELECTRONICAL COMISSION, INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS. **ISO/IEC/IEEE 24765**. Systems and software engineering – Vocabulary. Geneva: 2010. 401 p.

JAYARATNA, N. **Understanding and evaluating methodologies: NIMSAD: A Systematic Framework (The Mcgraw-Hill Information Systems, Management and Strategy Series)**. McGraw-Hill, Berkshire, 1994. p.288.

JEFFERY, R.; BABAR, M.A.; ZHU, L. **A Framework for Classifying and Comparing Software Architecture Evaluation Methods**. PROCEEDINGS OF THE 2004 AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, p.309-318, Melbourne, 2004. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1290484&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1290484> Acesso em 03 mar. 2015.

KAZMAN, R. KLEIN, M. CLEMENTS, P. "**ATAM: Method for Architecture Evaluation**". Hanscom: Software Engineering Institute, August 2000. (Technical Report CMU/SEI-2000-TR-004, ESC-TR-2000-004). Disponível em: < https://www.sei.cmu.edu/reports/00tr004.pdf> Acesso em 04 jun. 2015.

KHEONG, L.S.; JAYARATNA, N. **Framework for Structuring Learning in Problem-Based Learning**. Australia: Curtin University, [entre 1995 e 2014]. Não paginado. Disponível em: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.231&rep=rep1&type=pdf> Acesso em: 10 mai. 2015.

KITCHENHAM, B.; BABAR, A.B. **Assesment of a Framework for Comparing Software Architecture Analysis Methods**. Ireland: University of Limerick, 2007. Não paginado. Disponível em: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.6470&rep=rep1&type=pdf> Acesso em: 08 apr. 2015.

KOSKINEN, J.; LINTINEN, H.; SIVULA, H.; TILUS, T. **Evaluation of Software Modernization Estimation Methods Using NIMSAD Meta Framework**. Information Technology Research Institute. Jyvaskyla : University of Jyvaskyla, 2004. Disponível em:<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.198.585&rep=rep1&type=pdf> Acesso em: 17 mai. 2015.

KRUCHTEN, P. **Rational Unified Process, The: An Introduction**. 3.ed. Boston: Pearson Education, 2003. 336 p.

LAMSWEERDE. A, VAN. **Goal-oriented requirements engineering: a guided tour**. In: PROCEEDINGS. FIFTH IEEE INTERNATIONAL SYMPOSIUM ON,

REQUIREMENTS ENGINEERING, 2001. Toronto: IEEE, 2001, p. 249-262. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=948567&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D948567> Acesso em 04 jan. 2015.

LINDA, M et al. **A Framework for Software Product Line, Version 5.0**. 5 ed. Software Engineering Institute, [entre 2009 e 2012]. Disponível em: < http://www.sei.cmu.edu/productlines/frame_report/what.is.a.PL.htm>. Acesso em: 27 mai. 2015.

LOSAVIO, F et al. **Quality Characteristics for Software Architecture**. Journal of Object Technology: v2, n.2, p.133-150, 2003. Chair of Software Engineering. Zurich, 2003. Disponível em: < http://www.jot.fm/issues/issue_2003_03/article2.pdf> Acesso em: 03 jan. 2015.

LOSAVIO, F. "**Quality Models to Design Software Architecture**".Journal Object Technology: v1, n.4, p.165-178. Chair of Software Engineering. Zurich, 2002. Disponível em: <http://www.jot.fm/issues/issue_2002_09/article4.pdf> Acesso em: 10 mar. 2015.

MATINLASSI, M. **Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KOBRA and QADA**. In: PROCEEDINGS OF THE 26TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, p. 127-136. Edinburg, 2004. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1317435&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1317435> Acesso em 05 jan. 2015.

NORD, R.L. et al. **Integrating Software-Architecture-Centric Methods into the Rational Unified Process**. Hanscom: Software Engineering Institute, July 2004. (Technical Report CMU/SEI-2004-TR-011 ESC-TR-2004-011). Disponível em: < https://resources.sei.cmu.edu/asset_files/TechnicalReport/2004_005_001_14396.pdf > Acesso em 04 jun. 2015.

OSTERLIND, M. et al. **Enterprise Architecture Evaluation Using Utility Theory**. ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE WORKSHOPS (EDOCW), 2013 17TH IEEE INTERNATIONAL, p.347-351.Vancouver, 2013. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6690571&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D6690571> Acesso em 10 jun. 2015.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional**. Tradução de Mario Moro Fechio e Ariovaldo Griesi. 7.ed. Porto Alegre: AMGH Editora Ltda, 2011. 776 p.

ROZANSKI, N.; WOODS, E. **Software system architecture: working with stakeholders using viewpoints and perspectives**. 2. ed. Massachusetts: Pearson Education, 2012. Não paginado.

SEE, T.K, GURNANI.A, LEWIS.K. **Multi-Attribute Decision Making Using Hypothetical Equivalents and Inequivalents**. Journal of Mechanical Design, v.126. Buffalo: University of Buffalo, 2004. Disponível em: < http://does.eng.buffalo.edu/administrator/components/com_jresearch/files/publications/See.JMD.2004.pdf> Acesso em 08 apr. 2015.

SEE, T.K, LEWIS.K. **Multi attribute Decision Making Using Hypothetical Equivalents**. In: PROCEEDINGS OF DETEC'02 ASME 2002 DESIGN ENGINEERING TECHNICAL CONFERENCES AND COMPUTERS AND INFORMATION IN ENGINEERING CONFERENCE, 2002, Montreal. Montreal: ASME, 2002, p. 10. Disponível em: < http://does.eng.buffalo.edu/administrator/components/com_jresearch/files/publications/2002.See.DETC.pdf> Acesso em 16. jun. 2015.

SOMMERVILLE, I. **Engenharia de software**. Tradução de Kalinka Oliveira e Ivan Bosnic. 9. ed. São Paulo: Pearson Prentice Hall, 2011. 529 p.

WOJCIK, R. et al. **Attribute-Driven Design (ADD), Version 2.0**. Hanscom: Software Engineering Institute, 2006. (Technical Report CMU/SEI-2006-TR-023 ESC-TR-2006-023). Disponível em: < http://www.sei.cmu.edu/reports/06tr023.pdf> Acesso em: 08 jan. 2015.

WOOD, WG. **A Pratical Example of Applying Attribute-Driven Design (ADD), Version 2.0**. Hanscom: Software Engineering Institute, February 2007. (Technical Report CMU/SEI-2007-TR-005 ESC-TR-2007-05). Disponível em: <http://www.sei.cmu.edu/reports/07tr005.pdf> Acesso em 04 jun. 2015.

ZAYARAZ, G.; THAMBIDURAI. P. **Software Architecture Selection Framework Based on Quality Attributes**. In: IEEE INDICON 2005 CONFERENCE., 11-13 dec, 2005, Chennai. Chennai: Departament of Computer Science & Engineering, 2005. p. 167-170. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1590147&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1590147> Acesso em 02 jan. 2015.