Um Padrão de Qualidade para Componentes de Software

Régis Patrick Silva Simão
regis.simao@serpro.gov.br

Arnaldo Dias Belchior
belchior@unifor.br

Mestrado de Informática Aplicada, Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321 CEP 60.811.341 – Fortaleza – CE – Brasil

Resumo

O sucesso de aplicações baseadas em componentes de software depende de vários fatores, dentre eles, a qualidade desses componentes é um dos mais importantes. As características de qualidade, segundo a ISO/IEC 9126, podem ser utilizadas como metas a serem atingidas no desenvolvimento, na seleção e na aquisição de componentes e, também, como fatores na predição de propriedades de aplicações finais. Este trabalho identifica e organiza as características e subcaracterísticas de qualidade de maior relevância para componentes de software e define dimensões de qualidade que influenciam no grau de importância das mesmas, gerando um padrão de qualidade para os mesmos, segundo uma das dimensões apresentadas. Esse padrão foi elaborado através de uma pesquisa de campo, realizada com desenvolvedores de componentes e de aplicações baseadas em componentes, cujos resultados foram obtidos através da utilização de um método de avaliação de qualidade de software.

Palavras-chave: avaliação da qualidade, qualidade de componentes, medição de software.

Abstract

The success of component-based applications depends on many factors. Software component quality is one of the most important. Quality characteristics, in conformity to ISO/IEC 9126, may be used as goals in development, selection and acquisition of components, as well as elements for predicting the properties of final applications. This work identifies and organizes quality characteristics and subcharacteristics of major relevance to software components and defines dimensions of quality that change the degree of characteristic's and subcharacteristic's importance, so as to come out with a quality standard for them considering one of the dimensions of quality. This standard is the result of a field survey, carried out with developers of components and component-based applications. The survey's data were analyzed through a software quality evaluation method.

Keywords: quality evaluation, component quality, software metrics.

1. Introdução

Com a globalização dos mercados, as empresas passaram a ter maior preocupação com os problemas apresentados nas formas tradicionais de desenvolvimento de software. As técnicas de desenvolvimento, normalmente usadas, focam a construção de uma aplicação por vez. Possuem uma forte atenção nos prazos de entrega dos sistemas, mas pequeno é o percentual deles que chegam a seus clientes no tempo previsto. Freqüentemente, essas aplicações não são fáceis de manter e evoluir, enfatizando a entrega dos mesmos, mas não a

continuidade da relação cliente/fornecedor. Outro aspecto importante é o não cumprimento dos custos orçados; muitas das aplicações gastam mais do que o inicialmente previsto [5].

A competitividade exige das empresas uma nova postura em relação à construção de sistemas. Essa postura deve buscar a qualidade do processo de desenvolvimento, que permita o aumento da produtividade dos desenvolvedores e, conseqüentemente, da organização. Desta forma, desenvolve-se sistemas que sejam entregues no prazo e no orçamento estimados, e garantindo a produção de aplicações com níveis adequados de qualidade.

O *Desenvolvimento baseado em Componentes* (DBC) aparece como uma técnica promissora para a resolução dos problemas supra citados. Essa técnica consiste na divisão de blocos monolíticos em módulos interoperáveis. Esta granularização das aplicações reduz a complexidade. Os desenvolvedores passam a construir módulos independentes e combiná-los nas aplicações finais. A reutilização de módulos pré-fabricados pode substituir ou, pelo menos, diminuir a atividade de construção desses módulos, reduzindo o tempo e o custo do desenvolvimento, e aumentando a produtividade. A estes módulos, dá-se o nome de componentes [14, 25, 26].

Com atenção voltada para a construção de partes menores e pontuais, e para a combinação destes componentes em aplicações ou componentes maiores, acredita-se que a qualidade de produtos de software possa ser melhorada. A reutilização de componentes em diversos sistemas permite uma melhoria continua da qualidade desses componentes, pois as utilizações anteriores podem ser encaradas como testes práticos dos mesmos e, conseqüentemente, fornecendo subsídios para estimativas da qualidade do software a ser construído.

Este trabalho objetiva contribuir para a construção de componentes de software de qualidade, apresentando uma hierarquia de características e subcaracterísticas de qualidade para os mesmos, identificando aspectos que influenciam no grau de importância dessas características e elaborando um padrão de qualidade.

Este trabalho está organizado da seguinte forma: a seção 2 descreve em linha gerais a qualidade em aplicações baseadas em componentes. A seção 3 apresenta aspectos relacionados com a qualidade de componentes de software. A seção 4 mostra e analisa os resultados obtidos em uma pesquisa de campo realizada. A seção 5 exibe as principais conclusões desta pesquisa.

2. Qualidade de Sistemas Baseados em Componentes

Vários fatores influenciam na qualidade de uma aplicação baseada em componentes [1, 3, 11, 14, 19, 26]. Bachman [1] define que a qualidade do software depende da qualidade dos componentes e do *framework* de componentes utilizados. Woodman [26] acrescenta que o processo de desenvolvimento e o grau de maturidade da organização também influenciam na qualidade dos produtos de software baseados em componentes.

A qualidade do produto de software está fortemente relacionada à qualidade do processo de construção [10, 21]. DBC aparece como uma promessa na construção de aplicações, onde se enfatiza sistematicamente características de qualidade. Entretanto, os processos de DBC atualmente não tratam explicitamente esta questão [18].

A seleção e a definição das atividades a serem utilizadas no processo são muito importantes para o sucesso de aplicações baseadas em componentes [11]. A inclusão e o bom detalhamento de atividades como busca, seleção, adaptação e composição de componentes

são de fundamental importância, para a definição de um bom processo de desenvolvimento baseado em componentes [11, 25]. Obviamente, a inclusão de atividades já validadas nas técnicas tradicionais de construção de software, como modelagem de negócio, levantamento de requisitos e testes, também são relevantes.

Muitas características de qualidade podem ser medidas em tempo de projeto (por exemplo: manutenibilidade, clareza, etc.) [3, 19]. Desta forma, atividades de garantia de qualidade [17, 21] devem ser incluídas no processo, para avaliar e controlar continuamente a qualidade dos artefatos gerados, garantindo, como conseqüência, a qualidade das características da aplicação.

Considerando-se que reutilização é uma atividade a ser estimulada e maximizada para se atingir altos graus de produtividade e qualidade, e baixos custos, a velocidade na localização e recuperação dos componentes são aspectos que dão agilidade ao processo. Alguns mecanismos de busca estão sendo definidos e, principalmente, incluídos em processos de desenvolvimento, como atividades-chave [22, 25].

A seleção de componentes é uma atividade de extrema importância em DBC. É através dela que os componentes são avaliados e escolhidos para uso na aplicação. Como exemplos de subatividades, tem-se a verificação das informações apresentadas e dos serviços oferecidos pelos componentes, e a predição dos resultados a serem alcançados com a combinação dos mesmos. A teoria da predição de combinações consiste em prever as características de qualidade da aplicação final, com base nas características dos componentes selecionados e de suas combinações [1, 8]. Nessa subatividade, está concentrada a maioria dos trabalhos desenvolvidos atualmente em DBC [6, 8, 23].

As características não funcionais normalmente são introduzidas em uma aplicação através da arquitetura [5, 18]. D'Souza [9] afirma que "a arquitetura de um sistema consiste da estrutura de suas partes (em tempo de projeto, execução, distribuição etc.), da natureza e das propriedades relevantes, externamente visíveis daquelas partes (interfaces, módulos, objetos etc.), e os relacionamentos e restrições entre elas."

Preiss [18] afirma que o resultado das decisões arquiteturais manifesta-se nos estilos arquiteturais. Bachman [1] define estilo arquitetural como "um padrão de projeto de alto nível, descrito através de tipos de componentes e de seus padrões de interação". Preiss complementa que componentes de software e suas combinações são as realizações das decisões arquiteturais [18].

Segundo Bosch [5], existem três motivos para a representação explícita da arquitetura de software. Primeiramente, a comunicação com os desenvolvedores é facilitada através do estilo arquitetural definido. Em segundo, o projeto arquitetural é uma atividade de múltiplos objetivos, onde o engenheiro de software deve balancear vários requisitos durante o desenvolvimento, procurando alcançar os melhores valores, para as diversas características de qualidade. E, por fim, a criação de uma linha de produto de software, que é um conjunto de sistemas, que compartilham uma arquitetura de software comum e um conjunto de componentes reutilizáveis. Logo, é fundamental a representação explícita da arquitetura, no sentido de guiar a reutilização dos componentes e a construção de uma aplicação, que atinja as características de qualidade desejadas. Bosch [5] apresenta um método para projeto de arquiteturas de software, onde as características de qualidade são enfatizadas e perseguidas continuamente.

As aplicações baseadas em componentes são implementadas através de tecnologias de componentes (por exemplo: COM, EJB, CORBA). Essas tecnologias possuem *frameworks*,

que seguem estilos arquiteturais previamente definidos. As características de qualidades derivadas desses estilos são repassadas para os componentes e, conseqüentemente, para as aplicações [1].

Os componentes de software recebem influência das experiências dos desenvolvedores e da organização através do processo de desenvolvimento, da arquitetura estabelecida para o software e da tecnologia utilizada na sua implementação [1, 14, 18]. Através dos estudos sobre teorias de predição de combinações [6, 8, 23], fica clara a influência das características de qualidade dos componentes sobre as aplicações, senão diretamente, através das combinações dos mesmos.

Normalmente, um componente de software tem sido visto como código executável [24]. Entretanto, ainda não há um consenso entre os pesquisadores. Uma outra visão, que está sendo bem aceita, coloca um componente como uma unidade de solução, para um determinado problema de um domínio de aplicação [25]. Essa nova visão estende o conceito de componente. Isto implica que um componente pode existir em diversas formas (especificação, projeto e código), podendo ser utilizado em várias fases do desenvolvimento [9, 13, 19, 25].

Um componente pode ser visto como um software em miniatura, possuindo uma arquitetura, sendo constituído por vários artefatos, podendo ser construído a partir de outros componentes, e contendo características de um software completo [5, 9, 14].

Assim sendo, as características dos componentes influenciam na qualidade do produto de software final. Desta forma, duas áreas de estudo importantes sobre componentes são a qualidade e a certificação dos mesmos. Estes dois tópicos são fundamentais para o sucesso de um processo de DBC e de uma aplicação baseada em componentes [3, 16, 18, 19].

Num processo de DBC, os componentes podem ser construídos, quando ainda não existirem, ou reutilizados, quando pré-fabricados. Neste último caso, devem ser localizados, recuperados e selecionados. Características de qualidade podem ser empregadas para a localização do componente, se forem utilizados mecanismos de classificação baseados nessas características. Para a seleção, quando se compara as propriedades dos componentes com as levantadas, como requisitos que os mesmos devam atender. Na seleção, pode-se ganhar tempo, não sendo necessário inspecionar a veracidade das informações, quando esses componentes são certificados e acredita-se na idoneidade da unidade certificadora.

Neste contexto, evidencia-se a relevância de pesquisas mais específicas sobre em qualidade de componentes de software. Alguns tópicos importantes de estudo nesta área são: a identificação de características de qualidade para componentes, de métricas para a medição das mesmas, e a definição de padrões de qualidade, que os componentes devam possuir.

3. Qualidade de Componentes de Software

As características de qualidade de componentes e suas respectivas métricas podem ser usadas nas atividades de busca e seleção desses componentes, como também na predição dos níveis de qualidade das características de uma aplicação baseada em componentes.

Muitos trabalhos sobre componentes relatam atributos de qualidade que os componentes devem ter [9, 12, 13, 19, 25, 26]. Entretanto, a maioria desses trabalhos não trata especificamente deste tema, ou não o aborda de uma forma extensiva e abrangente. Portanto, ainda não existe um consenso em relação a que características de qualidade um componente deva possuir.

Recentemente, alguns trabalhos têm mostrado uma preocupação com este tema [3, 18]. Preiss [18] apresenta um pequeno conjunto de características de qualidade, organizado através de um modelo de classificação simples. Bertoa [3] apresenta uma lista de características e suas possíveis métricas para componentes COTS (*Commercial-Off-The-Shelf*). Ele organiza essas características, segundo a norma ISO/IEC 9126 [12]. Contudo, percebeu-se, em alguns casos, a utilização diferenciada das subcaracterísticas de qualidade, em relação aos conceitos apresentados pela norma ISO, além de considerar algumas métricas como subcaracterísticas.

A definição de padrões de qualidade é importante na atividade de elaboração de produtos de software, notadamente, na construção de componentes. Os padrões podem ser usados como parâmetros, para que se possa dar mais atenção aos aspectos que levem os componentes a níveis de qualidade esperados, com maior rapidez e menor custo. Segundo Boegh [4], alcançar graus elevados em características de qualidade custa caro, devido a grande quantidade de recursos necessários.

Estes padrões visam identificar a importância das características de qualidade nos produtos de software. Podem ser obtidos através de pesquisas de campo realizadas com especialistas no domínio de aplicação em questão. Essas pesquisas usualmente utilizam-se de questionários de avaliação, que coletam o grau de importância das características de qualidade do produto de software em avaliação [20].

Este trabalho cobre dois dos aspectos em qualidade de componentes: a identificação e a organização de características de qualidade, e a elaboração de um padrão de qualidade para componentes de software.

Para a identificação e a organização das características de qualidade, foi realizada uma pesquisa bibliográfica na área, sendo essas características organizadas segundo o que propõe a norma ISO/IEC 9126 [12]. Uma característica de qualidade é uma propriedade do componente de software, através da qual a qualidade é descrita e avaliada. Uma característica pode ser detalhada em múltiplos níveis de subcaracterísticas.

A norma ISO/IEC 9126 está subdividida em duas partes: modelos de qualidade para características externas e internas, e o modelo de qualidade para qualidade em uso. Este trabalho consiste somente do modelo de qualidade para características externas e internas, que definem seis características (funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade) e várias subcaracterísticas de qualidade.

Durante o levantamento das características e subcaracterísticas de qualidade de componentes de software, observou-se que o grau de importância dada a algumas dessas características pode variar em três dimensões distintas:

- *o domínio da aplicação*: correspondo ao domínio de aplicação, onde o componente é utilizado;
- *a função arquitetural*: o componente pode ser distribuído pelas camadas de interface, negócio, dados, e infra-estrutura; e
- *o nível de abstração*: o componente pode ser utilizado em suas formas de especificação, projeto e código.

Na elaboração do padrão de qualidade de componentes de software, desejava-se descobrir o grau de importância das características de qualidade para as três dimensões citadas anteriormente. Entretanto, percebeu-se que investigar a dimensão dos domínios de aplicação

existentes seria de alta complexidade ou inviabilizaria esta pesquisa, dada a sobrecarga de trabalho de se tratar todos os domínios de aplicação. Quanto à dimensão dos níveis de abstração, as características estão muito relacionadas com a forma (especificação, projeto e código) e com o paradigma adotado, e não com o conceito propriamente dito de componentes de software.

Assim, foi desenvolvido um questionário com as características e subcaracterísticas de qualidade, considerando-se somente a dimensão das funções arquiteturais dos componentes. Isto significa que, para cada subcaracterística a ser analisada, os especialistas deveriam avaliar o grau de importância dessa subcaracterística, considerando componentes de interfaces, de negócio, de dados e de infra-estrutura. Esse questionário foi aplicado a desenvolvedores de aplicações baseadas em componentes (reutilização de componentes) e /ou desenvolvedores de componentes propriamente dito.

Contudo, algumas análises ainda podem ser realizadas sobre as outras dimensões. Para a dimensão de domínios de aplicações, pode-se identificar o domínio de aplicação, em que o especialista atuou mais fortemente, através do Questionário de identificação do perfil do especialista (*QIPE*), que também faz parte desta pesquisa e, com isto, gerar padrões de qualidade para o domínio desejado.

Para a dimensão de níveis de abstrações, pode-se considerar as características de qualidade e o grau de importância das mesmas já levantadas anterior para etapas do processo de desenvolvimento de software em geral, como exemplo, o padrão de qualidade levantado para especificações orientadas a objetos [7].

Os dados dos questionários, coletados na pesquisa de campo, foram tratados através do Modelo *Fuzzy* para Avaliação da Qualidade de Software (Modelo Rocha Estendido) [2]. Na representação de modelos de raciocínio imprecisos, como os de qualidade, tem-se utilizado a teoria dos conjuntos *fuzzy*, que vem sendo utilizada em várias áreas do conhecimento humano, sendo o elo de ligação entre modelos imprecisos (subjetivos) do mundo real e sua representação matemática [2, 27].

Um conjunto *fuzzy* é caracterizado por uma função de pertinência, que mapeia os elementos de um domínio ou universo de discurso X para um número real em [0, 1]. Formalmente, $\tilde{A} : X \rightarrow [0, 1]$. Desta forma, um conjunto *fuzzy* $\tilde{A}$, por exemplo, apresenta-se como um conjunto de pares ordenados, em que o primeiro elemento é $x \in X$, o segundo, $\mu_{\tilde{A}}(x)$, é o grau de pertinência ou a função de pertinência de x em $\tilde{A}$, que mapeia x no intervalo [0, 1], ou seja, $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$. A pertinência de um elemento a um determinado conjunto passa a ser uma questão de gradação. Em casos extremos, o grau de pertinência é 0, indicando que o elemento não pertence ao conjunto, ou o grau de pertinência é 1, se o elemento pertence 100% ao conjunto [2].

Na avaliação das características e subcaracterísticas de qualidade, os especialistas atribuíram notas de 0 (zero) a 4 (quatro) para cada uma delas, que passou pelo processo de "fuzificação", isto é, a transformação dessas notas em números *fuzzy* normais triangulares, conforme a *Tabela 1*.

Um número *fuzzy* normal triangular pode ser representado por $\tilde{N}$ (a, m, b), onde os valores a e b identificam, respectivamente, os limites inferior e superior da base do triângulo, onde o grau de pertinência $\mu_{\tilde{A}}(x) = 0$. O valor de m corresponde à altura do triângulo, onde $\mu_{\tilde{A}}(x) = 1$, que é o valor desejável de qualidade. Na seção seguinte, serão exibidos os resultados dessa pesquisa, segundo o modelo *fuzzy* de avaliação da qualidade [2].

Tabela 1: Números *fuzzy* normais triangulares para a avaliação da qualidade [2]

Nota	Número *Fuzzy* Triangular Normal	Termo Lingüístico	Sigla	Interpretação
0	$\tilde{N}_1 = (0,0; 0,0; 1,0)$	Sem importância	SI	Indica de maneira absoluta que a existência do atributo de qualidade não tem importância.
1	$\tilde{N}_2 = (0,0; 1,0; 2,0)$	Pouca importância	PI	Indica de maneira absoluta que a existência do atributo de qualidade tem pouca importância.
2	$\tilde{N}_3 = (1,0; 2,0; 3,0)$	Desejável	D	Indica de maneira absoluta que é desejável a existência do atributo de qualidade.
3	$\tilde{N}_4 = (2,0; 3,0; 4,0)$	Muito importante	MI	Indica de maneira absoluta que a existência do atributo de qualidade é muito importante.
4	$\tilde{N}_5 = (3,0; 4,0; 4,0)$	Imprescindível	I	Indica de maneira absoluta que a existência do atributo de qualidade é imprescindível.

Como neste trabalho será utilizado o conjunto de termos lingüísticos da *Tabela 1*, apresenta-se, na *Figura 1* abaixo, o gráfico de suas funções de pertinência.

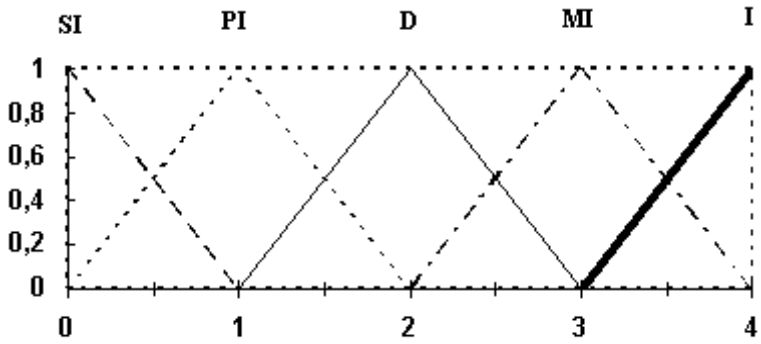

Figura 1: Funções de pertinência de números *fuzzy*, para termos lingüísticos [2]

Os perfis dos avaliadores, através do Questionário de Identificação do Perfil do Especialista, é utilizado pelo modelo *fuzzy* para dar peso aos questionários e influenciar no resultado final da avaliação, dando pesos maiores aos especialistas com mais conhecimentos e experiência. A pesquisa de campo colheu a opinião de 10 (dez) especialistas do estado do Ceará.

A *Tabela 2* apresenta o conjunto completo das seis características, segundo a ISO/IEC 9126, e 124 subcaracterísticas de qualidade, levantadas para componentes de software, baseadas, principalmente, em [3, 5, 7, 9, 11, 12, 13, 14, 15, 16, 18, 19, 21, 22, 24, 25, 26] e seus respectivos números *fuzzy* normais triangulares, que correspondam aos resultados da pesquisa de campo realizada para o obter o padrão de qualidade para componentes de software. Na seção seguinte, são comentados alguns dos resultados dessa pesquisa.

4. Avaliação dos Resultados

A *Tabela 3* apresenta os resultados iniciais obtidos para as características de qualidade, segundo a ISO/IEC 9126 [12], levantadas para componentes de software. Esses resultados estão dispostos em ordem decrescente de seu grau de importância, através de números *fuzzy* normais triangulares e suas respectivas interpretações.

Tabela 2: Características e Subcaracterísticas da Qualidade de Componentes de Software

Característica de Qualidade	Números *Fuzzy*
1. Funcionalidade	**(2,19; 3,19; 3,73)**
1.1. Adequação	(2,61; 3,61; 3,90)
1.1.1. Adequação tecnológica	(1,92; 2,92; 3,72)
1.1.1.1. Adequação da metodologia	(1,97; 2,97; 3,79)
1.1.1.2. Infra-estrutura tecnológica adequada	(2,11; 3,11; 3,83)
1.1.1.3. Adequação ambiente de desenvolvimento	(1,57; 2,55; 3,46)
1.1.2. Adequação Arquitetural	(2,57; 3,57; 3,98)
1.1.3. Cobertura	(2,63; 3,63; 3,84)
1.1.4. Completitude	(2,50; 3,50; 3,79)
1.1.5. Corretitude	(2,97; 3,97; 4,00)
1.1.6. Consistência	(2,83; 3,83; 4,00)
1.2. Acurácia	(2,91; 3,91; 4,00)
1.3. Autocontido	(1,88; 2,88; 3,65)
1.4. Coesão funcional	(1,34; 2,34; 3,32)
1.5. Interoperabilidade	(1,81; 2,81; 3,57)
1.5.1. Compatibilidade de dados	(1,69; 2,69; 3,54)
1.5.2. Interatividade	(2,31; 3,31; 3,78)
1.5.3. Interoperabilidade interna	(1,71; 2,71; 3,50)
1.6. Segurança de acesso	(2,49; 3,49; 3,81)
1.6.1. Confidenciabilidade	(2,78; 3,78; 3,94)
1.6.2. Imputabilidade	(1,84; 2,84; 3,46)
1.6.3. Privacidade	(2,14; 3,14; 3,72)
1.6.4. Vulnerabilidade	(2,85; 3,85; 3,97)
1.7. Conformidade com a funcionalidade	(2,26; 3,26; 3,81)
2. Confiabilidade	**(2,05; 3,04; 3,64)**
2.1. Maturidade	(2,04; 3,04; 3,58)
2.1.1. Memória de estado	(2,19; 3,18; 3,59)
2.1.2. Ausência de erros	(2,73; 3,73; 3,95)
2.1.3. Sinalização	(2,14; 3,13; 3,74)
2.1.4. Integridade	(2,60; 3,60; 3,89)
2.1.5. Não volatilidade de versão	(1,36; 2,36; 3,28)
2.1.6. Degradabilidade	(1,37; 2,37; 3,12)
2.2. Tolerância a falhas	(2,33; 3,33; 3,90)
2.2.1. Robustez	(2,55; 3,55; 3,94)
2.2.2. Prevenção a contingência	(2,12; 3,12; 3,86)
2.3. Recuperabilidade	(1,93; 2,88; 3,44)
2.3.1. Facilidade de desfazer serviços	(1,73; 2,69; 3,44)
2.3.2. Auxílio em situações de erro	(2,00; 3,00; 3,54)
2.3.3. Auditabilidade	(1,88; 2,73; 3,19)
2.3.4. Tratamento de erros	(2,66; 3,66; 3,99)
2.4. Avaliabilidade	(1,66; 2,66; 3,41)
2.4.1. Verificabilidade	(1,62; 2,62; 3,46)
2.4.2. Validabilidade	(1,69; 2,69; 3,36)
2.5. Conformidade com a confiabilidade	(2,25; 3,25; 3,83)
3. Usabilidade	**(1,48; 2,45; 3,31)**
3.1. Acessibilidade	(1,40; 2,40; 3,21)
3.1.1. Identificação do componente	(1,47; 2,47; 3,15)
3.1.2. Classificação do componente	(1,32; 2,32; 3,28)
3.2. Legibilidade	(1,63; 2,63; 3,47)
3.2.1. Clareza	(1,66; 2,66; 3,48)
3.2.2. Concisão	(1,41; 2,41; 3,35)
3.2.3. Estilo	(1,21; 2,21; 3,16)
3.2.4. Correção no uso do método	(1,87; 2,87; 3,87)
3.2.5. Uniformidade de Terminologia	(1,66; 2,65; 3,48)
3.2.6. Uniformidade no grau de abstração	(1,57; 2,57; 3,37)
3.3. Inteligibilidade	(1,54; 2,53; 3,36)
3.3.1. Contextualização	(2,06; 3,06; 3,65)
3.3.1.1. Estrutura básica do componente	(1,97; 2,97; 3,56)
3.3.1.2. Variabilidade	(2,12; 3,12; 3,67)
3.3.1.3. Dependências	(2,07; 3,07; 3,71)
3.3.1.3.1. Dependências de Contexto	(2,40; 3,40; 3,90)
3.3.1.3.2. Dependências Arquiteturais	(1,64; 2,64; 3,42)
3.3.1.3.3. Dependências Tecnológicas	(2,09; 3,09; 3,75)
3.3.2. Justificabilidade	(1,26; 2,26; 3,22)
3.3.3. Utilidade	(1,48; 2,47; 3,30)
3.3.4. Relevância	(1,56; 2,56; 3,40)
3.4. Facilidade de uso	(2,32; 3,32; 3,97)
3.4.1. Uniformidade de comportamento	(2,52; 3,52; 3,98)

Característica de Qualidade	Números *Fuzzy*
3.4.2. Uniformidade de interface	(2,12; 3,12; 3,96)
3.5. Apreensibilidade	(1,19; 2,18; 3,15)
3.5.1. Conteúdo orientado ao usuário	(0,81; 1,80; 2,78)
3.5.2. Disponibilidade de auxílios	(1,34; 2,32; 3,28)
3.5.3. Suporte a usuários	(1,26; 2,26; 3,24)
3.6. Operacionalidade	(2,04; 3,04; 3,73)
3.6.1. Manipulabilidade	(2,13; 3,13; 3,76)
3.6.1.1. Disponibilidade de artefatos	(2,11; 3,11; 3,84)
3.6.1.2. Estilo Arquitetural	(2,46; 3,46; 3,90)
3.6.1.3. Rastreabilidade	(1,16; 2,16; 3,10)
3.6.2. Controlabilidade	(1,95; 2,95; 3,69)
3.7. Atratividade	(0,34; 0,87; 1,83)
3.8. Conformidade com a usabilidade	(1,19; 1,99; 2,98)
4. Eficiência	**(2,11; 3,10; 3,83)**
4.1.Comportamento em relação ao tempo	(2,36; 3,35; 3,82)
4.1.1. Tempo de ligação da implementação do componente	(1,02; 1,95; 2,86)
4.1.2. Capacidade de recepção e processamento de informações	(2,38; 3,36; 3,85)
4.1.3. Capacidade de geração de informação	(2,42; 3,41; 3,84)
4.2. Comportamento em relação aos recursos	(2,04; 3,04; 3,94)
4.3. Comportamento em relação ao estado	1,55; 2,42; 3,12)
4.3.1. Suporte a manutenção do estado entre as interações	(1,61; 2,53; 3,42)
4.3.2. Capacidade de serialização	(1,13; 2,03; 2,92)
4.3.3. Suporte à persistência	(1,37; 2,06; 2,71)
4.3.4. Suporte a transações	(2,03; 2,95; 3,37)
4.4. Escalabilidade	(2,70; 3,70; 3,90)
4.5. Nível de Granularidade Adequado	(2,12; 3,12; 4,00)

Característica de Qualidade	Números *Fuzzy*
4.6. Conformidade com a eficiência	(1,81; 2,81; 3,79)
5. Manutenibilidade	**(1,92; 2,92; 3,70)**
5.1. Analisabilidade	(2,06; 3,06; 3,81)
5.1.1. Corretitude da Arquitetura	(2,06; 3,06; 3,81)
5.2. Implementabilidade	(1,74; 2,73; 3,65)
5.2.1. Viabilidade econômica	(1,65; 2,64; 3,59)
5.2.2. Viabilidade financeira	(1,56; 2,56; 3,55)
5.2.3. Viabilidade mercadológica	(1,54; 2,54; 3,51)
5.2.4. Viabilidade de cronograma	(1,76; 2,76; 3,65)
5.2.5. Viabilidade de mão de obra	(1,91; 2,90; 3,84)
5.2.6. Viabilidade tecnológica	(2,21; 3,21; 3,92)
5.3. Modificabilidade	(2,27; 3,27; 3,88)
5.3.1. Modificabilidade corretiva	(2,78; 3,78; 3,94)
5.3.2. Evolutibilidade	(1,96; 2,96; 3,84)
5.3.3. Extensibilidade	(2,25; 3,25; 3,87)
5.4. Estabilidade	(2,30; 3,30; 3,86)
5.5. Testabilidade	(1,37; 2,37; 3,36)
5.5.1. Autoteste de inicialização	(1,49; 2,49; 3,49)
5.5.2. Pacote de Testes	(1,44; 2,44; 3,42)
5.5.3. Geração de Relatórios de Testes	(0,43; 1,43; 2,43)
5.6. Conformidade com a manutenibilidade	(1,61; 2,61; 3,55)
6. Portabilidade	**(2,10; 3,10; 3,80)**
6.1. Adaptabilidade	(1,90; 2,90; 3,69)
6.1.1. Generalidade	(1,94; 2,93; 3,76)
6.1.1.1. Tipificação	(1,94; 2,93; 3,76)
6.1.2. Capacidade para ser configurado	(1,86; 2,86; 3,62)
6.2. Capacidade para ser instalado	(2,32; 3,32; 3,98)
6.3. Coexistência	(2,13; 3,13; 3,89)
6.4. Substituibilidade	(2,33; 3,32; 3,83)
6.5. Conformidade com a portabilidade	(1,45; 2,45; 3,36)

Funcionalidade é a característica do componente prover serviços que satisfaçam as necessidades especificadas, quando o componente for usado sob condições específicas [12]. Foi considerada a mais importante pelos avaliadores, obtendo o grau de importância de 3,19, com 81,01% de *muito importante* e 18,99% de *imprescindível*. Isso, em parte, pode ser

atribuído à ênfase que os processos de desenvolvimento dão às atividades de modelagem de negócio e levantamento de requisitos.

Portabilidade é a característica do componente ser transferido de um ambiente para outro. Obteve o segundo maior grau de importância, com valor de 3,10, com 89,87% de *muito importante* e 10,13% de *imprescindível*. Essa característica apresenta muitos fatores técnicos envolvidos na atividade de reuso, como adaptabilidade, substituibilidade etc., evidenciando a importância dos desenvolvedores nos aspectos técnicos de componentes.

Eficiência é a característica do componente realizar suas funções sem desperdício de recursos, sob condições especificadas [12]. Recebeu o grau de importância de 3,10, com 90,36% de *muito importante* e 9,64% de *imprescindível*. Memória, armazenamento, rede, isto é, recursos em geral são sempre itens dispendiosos e sempre exigem a preocupação dos desenvolvedores. Normalmente, a forma como um componente utiliza os recursos disponíveis é observada na aquisição do mesmo.

Tabela 3: Características de qualidade ISO para componentes de software

Características	**Números *Fuzzy***	**Interpretações**
Funcionalidade	(2,19; 3,19; 3,73)	18,99% Imprescindível e 81,01% Muito importante
Portabilidade	(2,10; 3,10; 3,80)	10,13% Imprescindível e 89,87% Muito importante
Eficiência	(2,11; 3,10; 3,83)	9,64% Imprescindível e 90,36% Muito importante
Confiabilidade	(2,05; 3,04; 3,64)	4,43% Imprescindível e 95,57% Muito importante
Manutenibilidade	(1,92; 2,92; 3,70)	91,77% Muito importante e 8,23% Desejável
Usabilidade	(1,48; 2,45; 3,31)	44,51% Muito importante e 55,49% Desejável

Confiabilidade é a característica do componente que o mantém em um determinado nível de desempenho, quando usado sob condições especificadas [12]. Foi conferido o grau de 3,04, com 95,57% de *muito importante* e 4,43% de *imprescindível*. Algumas subcaracterísticas de Confiabilidade normalmente são atribuídas à arquitetura da aplicação e não a componentes em si, como exemplo: Tolerância a Falhas [3].

Manutenibilidade é a característica do componente ser modificado. Modificações podem incluir correção, melhorias ou adaptação do componente para mudanças em ambiente, e em requisitos e especificações funcionais [12]. Obteve o grau de 2,92, com 8,23% de *desejável* e 91,77% de *muito importante*.

Usabilidade é a característica do componente ser entendido, aprendido, usado e atrativo para o usuário, quando usado sob determinadas condições [12]. Ela compreende subcaracterísticas relacionadas à contextualização do uso do componente, à facilidade de localização, ao entendimento, à manipulação e à operação. Foi considerada a menos importante, obtendo grau de 2,45, com 8,23% de *desejável* e 91,77% de *muito importante*. Normalmente, desenvolvedores não dedicam muita atenção aos aspectos não técnicos do desenvolvimento [9, 11, 13]. Aqui também podemos constatar esta afirmação, tendo usabilidade com o menor grau de avaliação.

Do conjunto das subcaracterísticas de qualidade levantadas, avaliou-se apenas as seis de maior relevância e as três menos importantes, para este trabalho, considerando-se o nível

hierárquico imediatamente inferior ao das características de qualidade. A *Figura 2* apresenta, em ordem decrescente do grau de importância, essas 6 subcaracterísticas melhores avaliadas.

A subcaracterística *Acurácia* (da característica *Funcionalidade*) consiste na capacidade do componente prover a geração de resultados ou efeitos, corretos ou esperados, com o grau de precisão exigido [12]. Foi considerada a mais importante pelos avaliadores, obtendo o grau de importância de 3,97, com 91,05% de *imprescindível* e 8,95% de *muito importante*.

A *Escalabilidade* (da característica *Eficiência*) é a capacidade do componente acomodar maiores volumes de processos sem a necessidade de mudanças na implementação [15, 26]. Foi considerada a segunda mais importante e obteve o grau de importância de 3,70, com 70,25% de *imprescindível* e 29,75% de *muito importante*.

A subcaracterística *Adequação* (da característica *Funcionalidade*) é a propriedade do componente prover a presença de um conjunto de serviços (funções) e sua adequação para as tarefas especificadas [12]. Recebeu o grau de importância de 3,61, com 61,48% de *imprescindível* e 38,52% de *muito importante*.

Segurança de Acesso (da característica *Funcionalidade*) é a propriedade do componente evitar acesso não autorizado, acidental ou deliberado, a componentes e dados e de permitir que pessoas autorizadas tenham acesso aos dados e componentes [12]. Foi conferido o grau de 3,49, com 49,02% de *imprescindível* e 50,98% de *muito importante*.

Comportamento em relação ao tempo (da característica *Eficiência*) é a capacidade do componente prover tempo de resposta, tempo de processamento e velocidade na execução de suas funções, sob condições atestadas [12]. Obteve o grau de 3,35, com 34,64% de *imprescindível* e 65,36% de *muito importante*.

A subcaracterística Tolerância a falhas (da característica *Confiabilidade*) é a capacidade do componente manter um nível especificado de desempenho em casos de falhas no componente ou de violação de suas interfaces especificadas [12]. Recebeu o grau de 3,33, com 33,50% de *imprescindível* e 66,50% de *muito importante*.

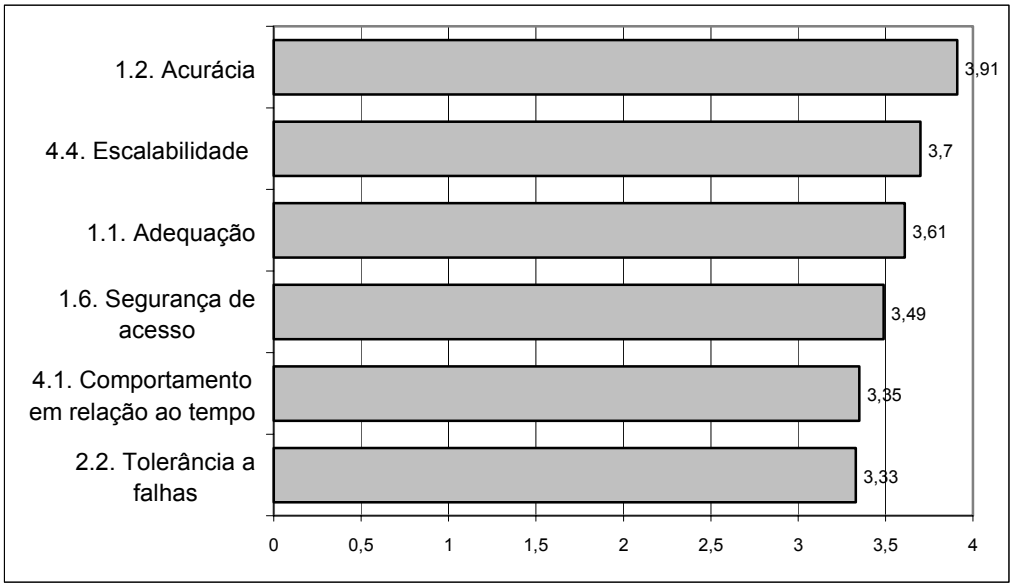

Figura 2: Subcaracterísticas de Qualidade melhores avaliadas

Das seis subcaracterísticas melhores avaliadas, três pertencem a característica Funcionalidade (Adequação, Acurácia e Segurança de acesso), duas, a Eficiência (Comportamento em relação ao tempo e Escalabilidade) e uma, a Confiabilidade (Tolerância a falhas). Cinco das seis subcaracterísticas melhores avaliadas já são definidas pela norma ISO/IEC 9126.

A *Figura 3* apresenta, em ordem decrescente do grau de importância, essas 3 subcaracterísticas de menor grau de importância, segundo a visão dos desenvolvedores.

A subcaracterística *Apreensibilidade* (da característica *Usabilidade*) consiste na capacidade do componente permitir, ao usuário, aprender a utilizá-lo [12]. Obteve o grau 2,18, sendo considerada o terceiro menor grau de importância, com 18,01% de *muito importante* e 81,99% de *desejável*.

Conformidade com a usabilidade (da característica *Usabilidade*) é a característica do componente estar de acordo com os padrões, convenções, guias de estilo e regulamentações relativos à usabilidade [12]. Recebeu o segundo menor o grau de importância, com valore de 1,99, com 99,49% de *desejável* e 0,51% de *pouca importância*.

A subcaracterística *Atratividade* (da característica *Usabilidade*) é a característica do componente ser atrativo para o usuário (exemplo: cores, projetos gráficos, etc.) [12]. Foi avaliada com o menor o grau de importância, recebendo 0,87, com 87,39% de *pouca importância* e 12,61% de *sem importância*.

As três subcaracterísticas de menores graus de importância pertencem à característica Usabilidade e também estão definidas na norma ISO/IEC 9126. Desta forma, ratifica-se a afirmação de que os desenvolvedores normalmente pouco enfatizam os aspectos não técnicos.

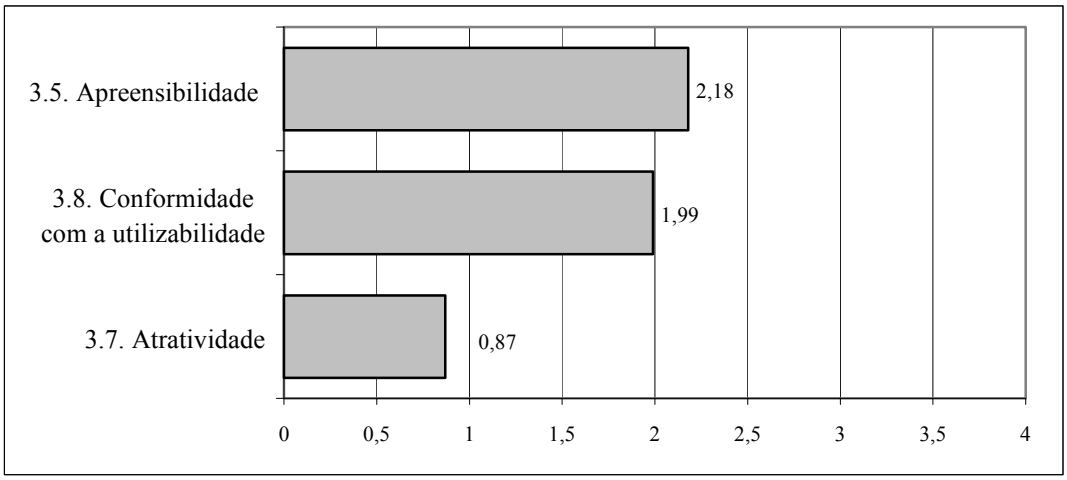

Figura 3: Subcaracterísticas de Qualidade com menores graus de importância

Entretanto, se Atratividade for observada segundo a dimensão da função arquitetural, pode-se observar os seguintes resultados: para componentes de interface, obteve o grau de importância 3,53, com 52,61% de *muito importante* e 47,39% de *desejável*; para componentes de negócio, recebeu o grau de 0,60, com 60,01% de *pouca importância* e 39,99% de *sem importância*; para componentes de dados, obteve a mesma avaliação que componentes de negócio; e para componentes de infra-estrutura, foi conferido o grau de 0,82, com 82,30% de

pouca importância e 17,70% de *sem importância*. A *Tabela 4* apresenta os resultados da avaliação da subcaracterística Atratividade segundo a dimensão da função arquitetural.

Tabela 4: Resultado da avaliação da subcaracterísticas Atratividade, segundo a dimensão da função arquitetural

Função Arquitetural (Componentes de ...)	**Números *Fuzzy***	**Interpretações**
Interface	(2,53; 3,53; 3,89)	52,61% Imprescindível e 47,39% Muito importante
Negócio	(0,17; 0,60; 1,60)	60,01% Pouca importância e 39,99% Sem importância
Dados	(0,17; 0,60; 1,60)	60,01% Pouca importância e 39,99% Sem importância
Infra-estrutura	(0,17; 0,82; 1,82)	82,30% Pouca importância e 17,70% Sem importância

5. Conclusão

Com premência da produtividade e de qualidade, e de diminuição dos custos pretendidos pelo DBC, aumenta a necessidade por parâmetros, que permitam a construção e a seleção de componentes com níveis adequados de qualidade.

Este trabalho propôs um conjunto de características e subcaracterísticas de qualidade para componentes de software, baseado no padrão ISO/IEC 9126. Também foram definidas três dimensões de qualidade que influenciam no grau de importância dessas características e subcaracterísticas (as dimensões dos domínios de aplicação, das funções arquiteturais e dos níveis de abstração).

Estas propriedades de qualidade estão sendo validadas, segundo a dimensão de funções arquiteturais (componentes de interface, negócio, dados e infra-estrutura), através de uma pesquisa de campo, realizada com desenvolvedores de componentes e de aplicações baseadas em componentes.

As características de qualidade avaliadas, em ordem decrescente de importância, é Funcionalidade, Portabilidade, Eficiência, Confiabilidade, Manipulabilidade e Usabilidade. As subcaracterísticas de nível imediatamente inferior às características, que obtiveram as melhores avaliações, em ordem decrescente, foram Acurácia, Escalabilidade, Adequação, Segurança de acesso, Comportamento em relação ao tempo e Tolerância a falhas. As subcaracterísticas que obtiveram os menores graus de importância foram Apreensibilidade, Conformidade com a usabilidade, e Atratividade.

As subcaracterísticas melhores avaliadas estão compreendidas nas características de qualidade: Funcionalidade, Eficiência e Confiabilidade. As de menores graus estão na característica Usabilidade. Considerando este resultado e ordem de importância consideradas pelos desenvolvedores para as características, pode-se observar ainda uma maior preocupação dos mesmo com aspectos técnicos de componentes do que com aspectos como facilidade de utilização.

Referências Bibliográficas

[1] Bachman, F., 2000, *Volume II: Technical Concepts of Component-Based Software Engineering, Software Engineering Institute*, Technical Report.

[2] Belchior, A. D., 1997, *Um Modelo Fuzzy para Avaliação da Qualidade de Software*, Tese de Doutorado, UFRJ-COPPE, Rio de Janeiro.
[3] Bertoa, M. e Vallecillo, A., 2002, *Atributos de Calidad para Componentes COTS*, 5º Workshop Iberoamericano de Engenharia de Requisitos e Ambientes de Software.
[4] Boegh, J *et al.*, 1993, *A practitioners guide to evaluation of software*, Software Engineering Standards Symposium.
[5] Bosch, J., 2000, *Design and Use of Software Architectures: Adopting and evolving a product-line approach*, Editora Addison-Wesley, ACM Press.
[6] Chen, S. *et al.*, 2002, *Performance Prediction of COTS Components-based Enterprise Applications*, V ICSE Workshop on Component-Based Software Engineering.
[7] Clunie, C. E., 1997, *Avaliação da Qualidade de Especificações Orientadas a Objetos*, Tese de Doutorado, UFRJ-COPPE, Rio de Janeiro.
[8] Crnkovic, I. *et al.*, 2002, *Anatomy of a Research Project in Predictable Assembly*, V ICSE Workshop on Component-Based Software Engineering, White Paper.
[9] D'Souza, D. F. e Wills, A. C., 1998, *Object, Components, and Frameworks with UML: The Catalysis Approach*, Editora Addison-Wesley, Massachusetts.
[10] Fuggetta, A., 2000, *Software Process: a roadmap*, International Conference on Software Engineering.
[11] Gómez-Perez, A. e Lozano, A., 2002, *Impact of Software Components Characteristics above Decision-making Factors*, III ICSE Workshop on Component-Based Software Engineering.
[12] ISO/IEC 9126-1, 2001, *Information Technology – Software Product Quality – part 1: Quality Model*.
[13] Jacobson, I., Griss e M., Jonsson, P., 1997, *Software Reuse: Architecture, Process and Organization for Business Success*, Editora Addison-Wesley.
[14] Kallio, P. e Niemelä, E., 2001, *Documented Quality of COTS and COM Components*, IV ICSE Workshop on Component-Based Software Engineering.
[15] Lycett, M., 2001, *Understanding variation in component-based development: case findings from practice*, publicado no *Information and Software Technology Journal*, 43, pp. 203-213.
[16] Mohagheghi, P., 2001, *Experiences with certification of reusable components in the GSN project in Erricsson, Norway*, IV ICSE Workshop on Component-Based Software Engineering.
[17] NBR ISO 9001:1994, *Sistemas da Qualidade, Modelo para garantia da qualidade em projetos, desenvolvimento, produção, instalação e serviços associados*.
[18] Preiss, O., Wegmann, A. e Wong, J., 2001, *On Quality Attribute Based Software Engineering*, 27th Euromicro Conference.
[19] Preiss, O. e Wegmann, A., 2002, *A System Perspective on the Quality Description of Software Components*, 6th World Multiconference on Systemics, Cybernetics and Informatics.
[20] Rocha, A. R., 1983, *Um Modelo para Avaliação da Qualidade de Especificações*, Tese de Doutorado, PUC-RJ, Rio de Janeiro.
[21] Rocha, A. R. C., Maldonado, J. C. e Weber, K. C., 2001, *Qualidade de Software: Teoria e Prática*, Prentice Hall.
[22] Seacord, R., C., 1999, *Software Engineering Component Repositories*, II ICSE Workshop on Component-Based Software Engineering.

[23] Stafford, J. e McGregor, J. D., 2002, *Issues in Predicting the Reliability of Composed Components*, V ICSE Workshop on Component-Based Software Engineering.
[24] Szyperski, C., 1998, *Component Software – Beyond Object-Oriented Programming*, Addison-Wesley.
[25] Villela, R. M. M. B., 2000, *Busca e Recuperação de Componentes em Ambientes de Reutilização de Software*, Tese de Doutorado, UFRJ-COPPE, Rio de Janeiro.
[26] Woodman, M. *et al.*, 2001, *Issues of CBD Product Quality and Process Quality*, IV ICSE Workshop on Component-Based Software Engineering.
[27] Zadeh, L. A., 1998, *Fuzzy Logic*, IEEE Transaction Compute, vol. 25.