# Um Levantamento de Métodos de Avaliação de Arquiteturas de Software Específicas

**Lucas Bueno Ruas de Oliveira e Elisa Yumi Nakagawa**

[1]Departamento de Sistemas de Computação
Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
São Carlos, SP, Brazil

{buenolro,elisa}@icmc.usp.br

***Abstract.*** *A diversity of methods to software architecture evaluation has been proposed, aiming at the quality of software systems. Efforts to identify, analyse and compare these methods can be also found. However, a more recent study has not been conducted. Thus, in this paper, we aim at identifying methods to evaluate recent and specific architectures — reference architectures, service-oriented architectures and aspect-oriented architectures — since in the last years, these architectures have become topics of research, besides broadly used. As main result, we can pointed out that there is a lack of mature and widely accepted methods to evaluate such architectures.*

***Resumo.*** *Uma diversidade de métodos de avaliação de arquiteturas de software vêm sendo propostos, visando a qualidade dos sistemas de software. Esforços para o levantamento, análise e comparação desses métodos podem também ser encontrados. Contudo, um levantamento mais recente não tem sido conduzido. Assim, este artigo tem como principal objetivo a identificação de métodos que abordem a avaliação de arquiteturas específicas e recentes — a saber arquiteturas de referência, arquiteturas orientadas a serviço e arquiteturas orientadas a aspectos — considerando-se que nos últimos anos essas arquiteturas têm-se tornado alvo tanto de pesquisa quanto de utilização. Como principal resultado, observou-se que há ainda uma grande carência de métodos consolidados e largamente aceitos para a avaliação de tais arquiteturas.*

## 1. Introdução

Nas últimas décadas, a área de Arquitetura de Software tem recebido crescente atenção como um importante ramo da Engenharia de Software [Kruchten et al. 2006, Shaw and Clements 2006]. De acordo com [Shaw and Clements 2006], em um futuro próximo, arquiteturas atingirão o mesmo status das demais tecnologias bem sucedidas. Arquiteturas de software são consideradas como sendo a principal estrutura de sistemas e compreendem elementos de software, as propriedades externamente visíveis desses elementos e o relacionamento entre eles [Bass et al. 2003]. Vale a pena ressaltar então que arquiteturas de software desempenham um papel fundamental na determinação da qualidade de sistemas de software. Decisões arquiteturais podem influenciar diretamente o sistema a ser concebido, podendo habilitar, facilitar, dificultar ou interferir

no alcance das metas referentes aos requisitos funcionais e de qualidade. Nesse contexto, arquiteturas de referência têm surgido como um elemento que agrega conhecimento de específico domínio de aplicação, promovendo o reúso desse conhecimento para o desenvolvimento de novos sistemas. Arquiteturas de referência para diferentes domínios, tais como para sistemas embarcados [Graaf et al. 2005] e sistemas de comércio eletrônico [Angelov et al. 2008], podem ser encontrados. Inclusive, arquiteturas de referência para o domínio de Engenharia de Software podem também ser encontrados [Eickelmann and Richardson 1996, Nakagawa et al. 2007].

Com o advento de novas tecnologias, sistemas de software com base em diferentes formas de organização têm surgido. É o caso de sistemas que possuem arquiteturas orientadas a serviço (do inglês, *Service-Oriented Architecture* (SOA)). Nesses sistemas, recursos de software são empacotados como "serviços" que são módulos bem definidos e auto-contidos que fornecem funcionalidades de negócio e são independentes do estado ou contexto de outros serviços [Papazoglou and Heuvel 2007]. Com isso, a utilização de SOA possibilita o desenvolvimento de sistemas de software de baixo acoplamento, promovendo a produtividade em virtude da possibilidade do reúso de serviços e possibilitando uma melhor compreensão do domínio de negócio.

Em uma outra perspectiva, também recentemente, sistemas orientadas a aspecto têm surgido e suas arquiteturas passaram a ter que ser tratadas. Entende-se por Programação Orientada a Aspectos (POA) [Kiczales et al. 1997] uma abordagem de desenvolvimento de software que suporta uma melhor separação de interesses (do inglês, *Separation of Concerns* (SoC)) e mais adequadamente reflete a forma como os desenvolvedores pensam sobre o sistema [Kiczales et al. 1997]. Essencialmente, a POA introduz uma unidade de implementação modular — o aspecto — que tem sido tipicamente utilizada para encapsular um interesse transversal (isto é, um interesse que se encontra espalhado ou entrelaçado com outros interesses do sistema). Modularidade, manutenibilidade e facilidade de implementação podem ser alcançados com a POA [Laddad 2003]. Aspectos têm também sido explorados nas primeiras fases do ciclo de vida do software, inclusive no projeto arquitetural [Ivers et al. 2004, Navarro et al. 2007]. De acordo com [Baniassad et al. 2006], considerar aspectos nessas primeiras fases facilita a identificação e análise de aspectos nas demais fases e, como uma consequência, uma melhor separação de interesses nos sistemas.

Considerando-se a relevância de arquiteturas de software como base sobre as quais todos os sistemas são construídos, a avaliação da qualidade das arquiteturas é fundamental. A condução de avaliações em arquiteturas de software permite a descoberta e a resolução de potenciais problemas em sistemas a serem ainda desenvolvidos, sendo, segundo [Clements et al. 2002], considerada uma maneira rápida e barata de se evitar posterior insucesso dos sistemas de software. Dessa forma, uma arquitetura que passa por meio de um processo de avaliação pode apresentar melhores perspectivas de desenvolvimento de um sistema com qualidade. Diversos estudos têm sido conduzidos no sentido de estabelecer métodos e critérios para avaliação de arquiteturas de software, sendo que os mais conhecidos são o SAAM (*Scenario-Based Architecture Analysis Method*) [Kazman et al. 1994] e o ATAM (*Architecture Trade-off Analysis Method*) [Kazman et al. 1998]. Observa-se que a utilização de métodos de avaliação permite a análise de riscos relacionados à evolução de software, apresentando potenciais estratégias

para lidar com esses riscos [Slyngstad et al. 2008].

Nesse contexto, são também encontrados esforços no sentido de classificar, comparar e analisar os diversos métodos de avaliação arquitetural, objetivando facilitar o entendimento e a escolha dos métodos a serem adotados para avaliar uma determinada arquitetura [Dobrica and Niemel 2002, Babar and Gorton 2004, Barcelos and Travassos 2006a, Ionita et al. 2002]. Porém, devido a esses estudos não serem tão recentes, alguns métodos importantes não foram abordados, principalmente aqueles voltados à avaliação da qualidade de arquiteturas de software mais específicas ou recentes.

Dessa forma, o objetivo deste artigo é o levantamento e análise de métodos de avaliação que dêem ênfase especificamente na avaliação da qualidade de arquiteturas de referência, arquiteturas orientadas a aspecto e arquiteturas orientadas a serviço. Essas três arquiteturas serão chamadas neste trabalho de arquiteturas específicas[1], apesar de não estarem no mesmo nível, mas que foram consideradas neste trabalho em função de serem recentes ou estarem em evidência no contexto de desenvolvimento de software. Assim, este artigo apresenta-se como um complemento aos estudos anteriores, visando auxiliar pesquisadores e profissionais no entendimento e escolha dos métodos mais adequados e provendo uma cenário atual sobre as capacidades e limitações que métodos para avaliação dessas arquiteturas específicas têm apresentado.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta um panorama geral sobre os métodos de avaliação arquitetural e os trabalhos que foram conduzidos no sentido de analisá-los e classificá-los. A Seção 3 apresenta os métodos de avaliação identificados e pertinentes para avaliação de arquiteturas específicas. Uma breve discussão é apresentada na Seção 4. Na Seção 5 são apresentadas as conclusões e perspectivas de trabalhos futuros.

## 2. Trabalhos Relacionados

Em virtude da importância e da variedade de métodos de avaliação existentes, podem ser encontrados trabalhos na literatura que visam fazer um levantamento e classificação desses métodos utilizando diferentes critérios [Dobrica and Niemel 2002, Babar and Gorton 2004, Barcelos and Travassos 2006a, Ionita et al. 2002]. De um modo geral, tais trabalhos têm por objetivo conduzir análise sobre os métodos, ressaltando as similaridades e diferenças. O trabalho de [Babar and Gorton 2004] apresenta uma discussão sobre os quatro principais métodos baseados em cenários — a saber, SAAM [Kazman et al. 1994], ATAM [Kazman et al. 1998], ALMA (*Architecture Level Modifiability Analysis*) [Bengtsson et al. 2004] e PASA (*Performance Assessment of Software Architecture*) [Williams and Smith 2002]) — buscando ressaltar as principais semelhanças e diferenças entre os métodos com relação a critérios como: momento de aplicação, foco de utilização, insumos, produtos e maturidade. Já o trabalho de [Dobrica and Niemel 2002] apresenta um estudo mais abrangente envolvendo oito métodos, não se restringindo apenas aos métodos baseados em cenários, mas discutindo características como o envolvimento dos *stakeholders* durante o processo de avaliação e a capacidade de se utilizar uma

[1]Apesar da importância e dos diversos trabalhos que abordam a avaliação de arquiteturas de linhas de produto, tais como [Babar 2007, Filho et al. 2008], este artigo objetiva focar particularmente nas arquiteturas específicas.

base de conhecimento pré-existente. Da mesma forma, [Ionita et al. 2002] classificou cinco métodos, dentre eles o FAAM (*Family–Architecture Analysis Method*) [Dolan 2002] e o CBAM (*Cost Benefit Analysis Method*) [SEI ], não incluídos nos trabalhos anteriormente citados (trabalhos de Babar e Dobrica). Por fim, [Barcelos and Travassos 2006a] conduziu uma Revisão Sistemática sobre o tema com a intenção de selecionar e avaliar de forma imparcial e justa os trabalhos que abordassem métodos de avaliação de arquiteturas de software, uma vez que os trabalhos anteriores nessa linha não tinham utilizado critérios mais rígidos de seleção dos métodos.

Como resultados da análise dos quatro trabalhos sobre levantamento e classificação de métodos de avaliação arquitetural, observa-se que eles não têm a preocupação de dar ênfase aos métodos de avaliação de arquiteturas específicas (arquiteturas de referência, arquiteturas orientadas a serviço e arquiteturas orientadas a aspecto), embora abordem métodos de avaliação para arquiteturas com diversas características particulares, como por exemplo para linhas de produto de software. Outra observação importante é o fato de que todos os trabalhos apontam SAAM e ATAM como sendo métodos maduros, amplamente utilizados, e que servem como base para diversos outros métodos presentes na literatura. Assim, esses dois métodos são brevemente apresentados a seguir, uma vez que são também base para os métodos apresentados neste trabalho.

## 2.1. SAAM

O SAAM utiliza-se de cenários com o objetivo de conduzir as avaliações de arquiteturas de software quanto a seus aspectos de qualidade [Kazman et al. 1994]. Um cenários é definido como uma descrição resumida de um determinado uso pretendido de um sistema por um *stakeholder*, podendo ser classificados em: (i) diretos, quando suportados pela arquitetura; ou (ii) indiretos, quando necessitam de ajustes na arquitetura para serem suportados. SAAM é aplicado no início do ciclo de desenvolvimento e provê ao arquiteto a possibilidade de optar por uma arquitetura com um *tradeoff* aceitável entre os atributos de qualidade. Esse método é composto por cinco principais etapas: desenvolvimento dos cenários, descrição da arquitetura de software, classificação, priorização e avaliação individual de cada cenário, interação entre cenários e avaliação global. Na Figura 1, apresentam-se as etapas do SAAM, bem como os relacionamentos entre as etapas.


**Figura 1. Etapas do SAAM**

SAAM tem como principais insumos (artefatos consumidos) a descrição da arquitetura de software, os requisitos de qualidade e as direções de negócio. Os pro-

dutos (artefatos produzidos) da utilização desse método incluem a apresentação das qualidades sensíveis a cenários, mapeamentos entre cenários e os componentes da arquitetura de software e também a previsão do esforço associado à mudança de cada cenário. Sendo assim, os produtos da utilização de SAAM provêem um conjunto de técnicas que auxiliam na realização de várias atividades do processo de desenvolvimento, como por exemplo, a caracterização dos atributos de qualidade, a elicitação de cenários e sua classificação [Babar and Gorton 2004]. Para isso, SAAM utiliza-se de diferentes *stakeholders* como: desenvolvedores, gerentes de projeto e arquitetos. SAAM avalia cada um de seus cenários por meio da investigação e descrição dos *cenários diretos* e *cenários indiretos*. Nesse método, o custo de acomodar os cenários indiretos é estimado por meio da contagem de mudanças necessárias para a adequação do mesmo. Por ser considerado uma abordagem madura, SAAM serve como base ao desenvolvimento de diversos outros métodos de avaliação encontrados na literatura [Bengtsson et al. 2004, Kazman et al. 1998, Tekinerdogan 2004, Kim et al. 2008].

### 2.2. ATAM

O ATAM é uma evolução do SAAM e possui como importante característica a utilização de *tradeoffs* [Kazman et al. 1998]. Em sua aplicação, ATAM aborda características de apoio à capacidade de modificação, confiabilidade, performance, segurança e entre outros atributos de qualidade das arquiteturas [Kazman et al. 1998]. Em arquiteturas de software, algumas características de qualidade são ortogonais, ou seja, ao se modificar uma característica outras também são alteradas. Em virtude disso, é necessária uma análise para produção de um melhor *tradeoff* das características relevantes ao contexto em que a arquitetura de software está inserida. A determinação do *tradeoff* não deve apenas considerar aspectos técnicos mas também todos os demais envolvidos, como os aspectos financeiros e de gerência. O ATAM utiliza como insumos as metas de negócio, descrição da arquitetura de software e a especificação do software, e tem como principais produtos uma lista de cenários, descrição dos *tradeoff points*, pontos de sensibilidade, riscos e abordagens sobre a arquitetura de software [Babar and Gorton 2004]. ad ATAM é um processo constituído de duas fases e nove atividades distribuídas durante essas fases (Figura 2). Na primeira fase, o ATAM é apresentado à equipe que irá utilizá-la, seguido da determinação dos objetivos de negócio, apresentação da arquitetura, identificação das abordagens arquiteturais, geração da árvore de atributos de qualidade e análise das abordagens arquiteturais. Na segunda fase, as atividades realizadas são: um *brainstorm* para a determinação dos cenários prioritários, análise dos cenários e apresentação dos resultados. ATAM não provê qualquer técnica específica de avaliação, no entanto, apresenta heurísticas relacionadas às medidas qualitativas da arquitetura e utiliza modelos teóricos para as análises quantitativas [Babar and Gorton 2004]. O ATAM é um método amplamente utilizado, sendo possível encontrar na literatura outros métodos nele baseados, como por exemplo, o EATAM (*Extended Architecture Trade-off Analysis Method*) voltado a linhas de produto de software [Kim et al. 2008].

## 3. Métodos de Avaliação de Arquiteturas Específicas

A investigação por métodos de avaliação das arquiteturas abordadas neste trabalho — arquiteturas de referência, arquiteturas orientadas a serviço e arquiteturas orientadas a aspecto — foi motivada principalmente pela necessidade do próprio grupo de pesquisa

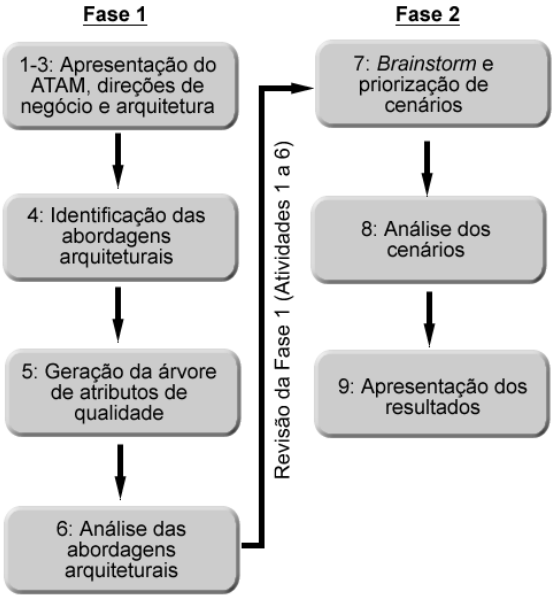


**Figura 2. Fases e atividades do ATAM (Adaptado de [Babar and Gorton 2004])**

em Arquitetura de Software do ICMC/USP em avaliar a qualidade das arquiteturas que têm sido propostas e utilizadas [Nakagawa et al. 2007, Nakagawa and Maldonado 2008].

Para a identificação de trabalhos que envolvem métodos para a avaliação de tais arquiteturas, foi utilizada a ideia da Revisão Sistemática[2]; contudo, sem utilizar todo o formalismo dessa abordagem. Foram utilizadas as bases de dados eletrônicas indexadas (IEEE, ACM e Springer), busca em máquinas de busca eletrônica (Scirus e Google), anais de eventos relacionados e consultas a especialistas do domínio. Para sistematizar a busca, foram utilizadas *strings* de busca formadas pela combinação dos sinônimos das palavras-chave em inglês e português identificadas, tais como "*reference architecture*", "*aspect-oriented*", "*service-oriented*", "*evaluation method*" e "*analysis method*". Como resultado, um conjunto de trabalhos foram identificados; no entanto, uma pequena parcela referia-se exatamente a métodos para avaliação das arquiteturas específicas.

A seguir, são apresentadas, além dos resultados e discussão dos trabalhos identificados mais relevantes, uma descrição das características das arquiteturas em questão, visando inclusive prover um cenário para o qual novos métodos devem ser propostos ou métodos existentes possam ser aprimorados.

### 3.1. Arquiteturas de Referência

Arquiteturas de referência têm-se destacado como um tipo especial de arquitetura, provendo direções mais precisas para a especificação de arquitetura concretas (ou seja, instâncias arquiteturais) de uma classe de sistemas ou sistemas de um dado domínio de aplicação. Por conseguinte, arquiteturas de referência influenciam diretamente a qualidade e o projeto de todo um conjunto de arquiteturas concretas e no conjunto de sistemas

[2]Uma revisão sistemática consiste em um meio de identificação, avaliação e interpretação de todos os trabalhos de pesquisa relevantes e disponíveis sobre uma questão de pesquisa, tópico ou fenômeno de interesse [Kitchenham 2004].

derivadas dessa arquitetura. Contudo, os métodos de avaliação de arquiteturas de software, tais como SAAM e ATAM, não podem ser diretamente aplicados a esse tipo de arquitetura. Tal restrição deve-se ao fato de que existe uma série de diferenças entre arquiteturas concretas e arquiteturas de referência [Bass et al. 2003], sendo que a principal diferença é o fato de arquiteturas de referência ser de natureza genérica e são projetadas para atender atributos e funcionalidades de interesse de todos os *stakeholders* de um domínio específico, como ilustrada na Figura3.

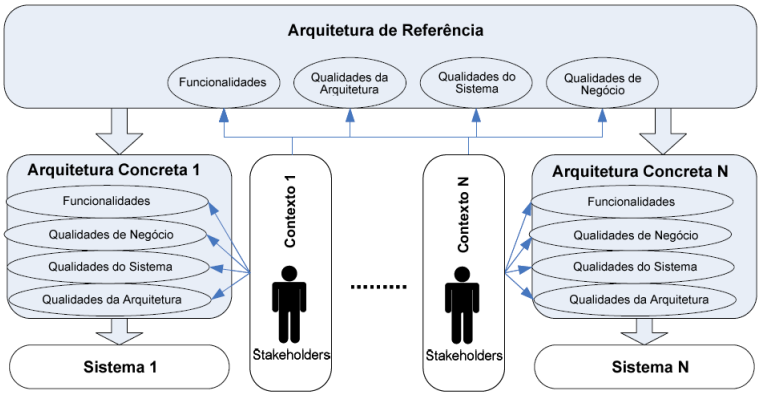


**Figura 3. Interação de *stakeholders* e contextos entre arquiteturas concretas e de referência [Angelov et al. 2008]**

Outras diferenças que podem ser observadas entre arquiteturas de referência com relação às concretas são: (i) a não existência um grupo claramente definido de *stakeholders* como suporte à construção e validação de uma arquitetura de referência; (ii) por questões políticas e de negócio, a composição de um grupo heterogêneo (de diferentes empresas e instituições) contendo todos os interessados na construção de uma arquitetura de referência para um determinado domínio é inviável; (iii) e arquiteturas de referência são definidas em um nível de abstração mais alto, o que dificulta a previsão dos cenários existentes. Tais fatos inviabilizam a aplicação da maioria dos métodos de avaliação existentes, já que esses fazem o uso de cenários e são fortemente dependentes dos *stakeholders* envolvidos.

Em virtude dessas diferenças, há a necessidade de propor métodos específicos e adequados para avaliação de arquiteturas de referência. Conduziu-se então uma investigação por esses métodos e como resultado, foram encontrados trabalhos que propõem adaptações do SAAM [Graaf et al. 2005] e também do ATAM [Angelov et al. 2008, Gallagher 2000]. O trabalho de [Graaf et al. 2005] apresenta o D-SAAM (*Distributed implementation of* SAAM) que tem por objetivo a avaliação de um modelo de arquitetura de referência para sistemas embarcados. O D-SAAM tem como objetivo principal a redução do impacto de uma determinada organização sobre a arquitetura sendo avaliada, pois quando se está avaliando arquiteturas de referência, os *stakeholders* interessados podem pertencer a diferentes organizações e possuir interesses diversos sobre a arquitetura. Já [Gallagher 2000] apresenta um abordagem para avaliação baseada no ATAM, ilustrada por meio do estudo de caso de um sistema de controle do departamento de defesa. No entanto, a utilização dessa abordagem para outras arquiteturas de referência parece ser limitada, pois a formação do grupo de *stakeholders* envolvidos

no estudo de caso compreende apenas membros pertencentes ao governo, não sofrendo assim, restrições ligadas ao aspecto organizacional. Por fim, [Angelov et al. 2008] ilustra uma abordagem um pouco menos específica, exemplificada por meio da avaliação de uma arquitetura de referência para sistemas de *e-commerce*. As modificações inseridas nessa abordagem têm a intenção de contornar os problemas apresentados no parágrafo anterior, tornando possível a avaliação de arquiteturas de referência por meio de um método de propósito geral. Contudo, observa-se que esse e outros trabalhos são pontuais e tratam apenas de relatos de experiências e soluções para casos específicos. Segundo [Angelov et al. 2008], há ainda a necessidade de mais investigação e proposição de métodos de avaliação para essas arquiteturas.

### 3.2. Arquiteturas Orientadas a Aspectos

Com a advento da POA, tem surgido a necessidade de tratar as arquiteturas de sistemas orientadas a aspecto que, dentre outras definições, podem ser entendidas como arquiteturas que possuem elementos arquiteturais, chamados de aspectos arquiteturais, encapsulando interesses transversais, como por exemplo a persistência, coordenação e performance [Cuesta et al. 2006, Dai and Cooper 2005, Rashid and Chitchyan 2003].

Os métodos tradicionais de avaliação de arquiteturas de software não possuem uma distinção explícita entre os elementos arquiteturais convencionais, que podem ser elicitados por meio das atuais abstrações, e os interesses arquiteturais (aspectos arquiteturais) que entrecortam outros elementos arquiteturais [Tekinerdogan 2004]. Sendo assim, existe o risco de que esses possíveis interesses transversais não sejam detectados como interesses em nível arquitetural e, por conseguinte, não sejam incluídos em nível de projeto e implementação, impedindo assim que a aplicação tenham determinados atributos de qualidade. Para solucionar esse problema, foi proposto um método baseado no SAAM, denominado ASAAM (*Aspectual Software Architecture Analysis Method*) [Tekinerdogan 2004]. Esse método provê um conjunto de técnicas adicionais ao SAAM que tem por objetivo a identificação de aspectos arquiteturais, classificando cenários em: diretos, indiretos, de aspectos e de aspectos arquiteturais. De forma complementar, esse método fornece uma análise detalhada da interação entre cenários e também uma caracterização de vários componentes em: componentes coesos, de composição, entrelaçados e mal definidos.

O ASAAM, assim como o SAAM, possui como insumos: a documentação dos requisitos, a descrição da arquitetura e a caracterização do problema. Segundo [Tekinerdogan 2004], a aplicação desse método é realizada em cinco etapas: (i) desenvolvimento de uma candidata a arquitetura que será analisada com relação às características de qualidade e potenciais aspectos; (ii) desenvolvimento dos cenários que representam os principais usos do sistema, de forma similar ao SAAM; (iii) avaliação individual de cenários e identificação dos aspectos, no qual os cenários são classificados de forma análoga ao SAAM e, em seguida, avaliados para identificação dos possíveis aspectos arquiteturais; (iv) avaliação da interação entre cenários e classificação dos componentes, verificando se a arquitetura apoia a separação de interesses e classificando os cenários em categorias; e por fim, a etapa de (v) refatoração da arquitetura, que se baseia na interação entre os cenários e a classificação dos componentes para adequar a arquitetura. A descrição da aplicação do ASAAM é ilustrada na Figura 4.

O principal benefício de ASAAM é o suporte sistemático ao gerenciamento de

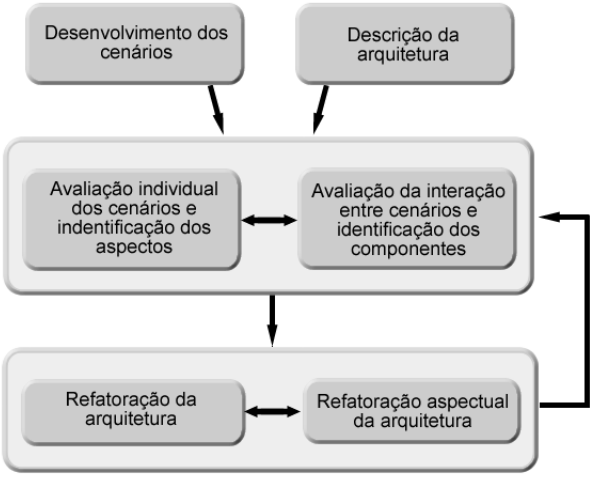


**Figura 4. Atividades do ASAAM (Adaptado de [Tekinerdogan 2004])**

aspectos arquiteturais de uma maneira explícita. ASAAM foi validado em estudos de caso e, segundo [Tekinerdogan 2004], sua aplicação pode beneficiar métodos de análise e projeto orientado a aspecto [Stein et al. 2002, Clarke and Walker 2001].

### 3.3. Arquiteturas Orientadas a Serviço

A adoção da SOA adiciona novos fatores que precisam ser considerados no desenvolvimento de sistemas de software. Distribuição, dados trafegados sobre a Web, integração com outras aplicações são algumas das características pertencentes a essa nova classe de software. Nesse contexto, é essencial que os métodos de avaliação existentes sejam evoluídos para que essas características sejam consideradas e avaliadas mais adequadamente, se for o caso.

Métodos tradicionais de avaliação arquitetural, tais como o SAAM e ATAM, são capazes de analisar e prever uma série de características de qualidade em arquiteturas. Porém, são altamente dependentes dos *stakeholders* envolvidos, não sendo adequados para a avaliação de arquiteturas nas quais os requisitos providos pelo sistema são realizados de maneira distribuída, por diferentes instituições, como ocorre em sistemas que têm SOA como base. Nesse cenário, uma das características de SOA é o alto custo relacionado à evolução de suas características de qualidade. Isso deve-se ao fato que sistemas de software, dentro de seus domínios de aplicação, devem possuir diversos atributos de qualidade que atualmente só podem verificados por meio de tentativa e erro, quando a arquitetura já está implementada [Becker et al. 2008]. Nesse contexto, o Q-ImPrESS possibilita a avaliação da evolução de arquiteturas que contenham características relacionadas a serviços, ainda durante a fase de projeto [Becker et al. 2008]. Com esse método, engenheiros de software podem analisar mudanças em nível arquitetural e prever as consequências de decisões relacionadas a diversos aspectos de qualidade do sistema de software como: segurança, performance e capacidade de modificação. De posse dessas decisões, Q-ImPrESS possibilita a criação de um *tradeoff* entre as principais características de qualidade. A análise desse *tradeoff* torna possível a classificação de características na criação de uma configuração de arquitetura possivelmente melhor antes mesmo de iniciar a fase de codificação. Outros trabalhos têm-se dedicado a investigar características

específicas de arquiteturas orientadas a serviço, tais como a interação entre serviços e análise de desempenho [Iacob and Jonkers 2007, Schroeder and Mayer 2008]. Contudo, observa-se que há uma carência de trabalhos que abordem a avaliação da arquitetura como um todo.

## 4. Uma Breve Discussão

A Tabela 1 apresenta uma síntese dos métodos para avaliação de arquiteturas de software específicas abordados neste trabalho, sendo listados os trabalhos que os propõem (coluna 2), principais características (coluna 3) e se esses podem ser aplicados a qualquer arquitetura com as mesmas características (coluna 4). Por exemplo, enquanto que a abordagem de [Tekinerdogan 2004] foi proposta de modo a ser aplicável a qualquer arquitetura orientada a aspecto, a abordagem de [Gallagher 2000] não parece ser adequada para a avaliação de qualquer arquitetura de referência. Observa-se também que na mesma linha das demais abordagens, a maioria das identificadas neste trabalho são extensões do SAAM e do ATAM.

**Tabela 1. Síntese das abordagens de avaliação**

| Síntese das Abordagens/Métodos | | | |
|---|---|---|---|
| Tipos de Arquitetura | Autor/Ano | Principais características | Aplicável a todas as arquiteturas |
| Arquitetura de Referência | Graaf et al., 2005 | Propõe o D-SAAM, uma adaptação do SAAM para avaliar uma arquitetura de referência para sistemas embarcados. | Não |
| | Gallagher, 2000 | Estende o ATAM para avaliar uma arquitetura de referência para sistema de controle militar. Método de difícil generalização para avaliar outras arquiteturas de referência. | Não |
| | Angelov et al., 2008 | Modifica o ATAM para que seja possível sua utilização em arquiteturas de referência. Dentre as abordagens apresentadas neste trabalho, é a que pode ser generalizada mais facilmente. | Não |
| Arquitetura Orientada a Aspectos | Tekinerdogan, 2004 | Suporte sistemático ao gerenciamento de aspectos arquiteturais. Desenvolvido por meio da adaptação do SAAM. | Sim |
| Arquitetura Orientada a Serviços | Becker et al., 2008 | Apresenta o Q-ImPrESS, um método que possibilita a determinação de um *tradeoff* entre atributos de qualidade em SOAs. | Sim |

Observa-se que, por se tratarem de novas tecnologias (no caso deste artigo, a orientação a serviços e a orientação a aspectos), métodos de avaliação específicos ou extensões de métodos existentes ainda parecem ser necessários. Observa-se que os trabalhos nessa linha são bastante recentes e maturidade nesses métodos devem ainda ser alcançada. No caso de arquiteturas de referência, apesar dessas não serem tão recentes, questões relacionadas a sua devida avaliação têm sido ainda deixados um pouco de lado. Contudo, observa-se que é ainda mais importante a avaliação de arquiteturas de referência, uma vez que ela pode dar origem a várias outras instâncias arquiteturais.

Além dos métodos de avaliação arquitetural, outras abordagens podem ser também encontradas para avaliação da qualidade de arquiteturas de software, tais como o uso de

*checklist*. ArqCheck [Barcelos and Travassos 2006b] é uma abordagem de inspeção de documentos arquiteturais que utiliza uma *checklist* criada a partir do conhecimento utilizado normalmente durante o projeto de arquiteturas, possuindo o objetivo de descobrir defeitos nesses documentos. Tal abordagem tem o objetivo de minimizar limitações existentes nos métodos de avaliação usualmente utilizados. Essas limitações são relativas à dependência demasiada do conhecimento dos avaliadores, elevado custo de aplicação, incompletude da avaliação dos requisitos de qualidade e limitação dos contextos de aplicação [Barcelos and Travassos 2006b]. Em virtude disso, abordagens de inspeção de artefatos de software têm obtido bons resultados, até porque, assim como as demais práticas de inspeção de software, deve ser executada de maneira formal e rigorosa. Apesar disso, observa-se que não são ainda encontrados na literatura *checklist* para inspeção de arquiteturas de referência, arquiteturas orientadas a aspecto e orientadas a serviço.

É importante ressaltar também que a combinação de diferentes métodos, abordagens e técnicas é um prática importante para uma avaliação mais completa de arquiteturas de software, pois cada um possui seus pontos fontes e limitações. Isso leva a crer que há a necessidade de propor ou evoluir, além dos métodos de avaliação, outras abordagens e técnicas para a avaliação também de arquiteturas específicas.

## 5. Conclusão e Trabalhos Futuros

Arquiteturas de software são consideradas estruturas fundamentais na determinação da qualidade de sistemas. A condução de avaliações de arquiteturas é uma importante prática no auxílio à construção dessas arquiteturas, visto que podem contribuir para a qualidade dos sistemas de software resultantes. Diversos métodos de avaliação de arquiteturas de software têm sido propostos na literatura e também classificados e discutidos.

Considerando-se a relevância que tem tomado as arquiteturas chamadas específicas neste trabalho — arquiteturas de referência, arquiteturas orientadas a serviço e arquiteturas orientadas a aspecto — uma observação mais informal e qualitativa mostra que há ainda a necessidade de mais esforços no sentido de estabelecer métodos consolidados e que possam de fato contribuir para a área de Arquitetura de Software. Então, a principal contribuição deste trabalho foi apresentar o cenário atual dos métodos de avaliação de arquiteturas específicas, bem como apresentar as características dessas arquiteturas.

Como trabalhos futuros, pretender-se estender essa pesquisa no sentido de realizar um estudo comparativo e mais crítico-analítico envolvendo tanto os métodos abordados neste trabalho quanto aqueles que venham a surgir. Considerando-se a demanda de métodos para atender ao cenário apresentado, a ideia é também propor métodos de avaliação arquitetural, complementar métodos consolidados ou investigar o uso de métodos já existentes, inclusive com o intuito de conduzir avaliações das arquiteturas que têm sido propostas pelo grupo do qual faz parte este trabalho.

## Referências

Angelov, S., Trienekens, J. J., and Grefen, P. (2008). Towards a method for the evaluation of reference architectures: Experiences from a case. In *ECSA´08: Proc. of the 2nd Eur. Conf. on Software Architecture*, pages 225–240, Paphos, Cyprus.

Babar, M. and Gorton, I. (2004). Comparison of scenario-based software architecture evaluation methods. In *Software Engineering Conference, 2004. Proc. of the 11th Asia-Pacific*, pages 600–607, Busan, Korea.

Babar, M. A. (2007). Evaluating product line architectures: Methods and techniques. In *APSEC '07: Proceedings of the 14th Asia-Pacific Software Engineering Conference*, page 13, Washington, DC, USA. IEEE Computer Society.

Baniassad, E., Clements, P. C., Araujo, J., Moreira, A., Rashid, A., and Tekinerdogan, B. (2006). Discovering early aspects. *IEEE Software*, 23(1):61–70.

Barcelos, R. F. and Travassos, G. (2006a). Evaluation approaches for software architectural documents: a systematic review. In *Proc. of the IX Ibero-American Workshop on Requirements Engineering and Software Environments (IDEAS´06)*, La Plata, Argentina.

Barcelos, R. F. and Travassos, G. H. (2006b). ArqCheck: Uma abordagem para inspeção de documentos arquiteturais baseada em checklist. In *V Simpósio Brasileiro de Qualidade de Software (SBQS'2006)*, pages 175–189.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*. Addison-Wesley.

Becker, S., Trifu, M., and Reussner, R. (2008). Towards supporting evolution of service-oriented architectures through quality impact prediction. In *ASE Workshops 2008, 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 77–81, L'Aquila, Italy.

Bengtsson, P., Lassing, N., Bosch, J., and van Vliet, H. (2004). Architecture-level modifiability analysis (ALMA). *Journal of Systems and Software*, 69(1-2):129–147.

Clarke, S. and Walker, R. J. (2001). Composition patterns: An approach to designing reusable aspects. In *Proc. of the 23rd Int. Conf. on Software Engineering (ICSE´01)*, volume 0, pages 5–14, Toronto, Canada.

Clements, P., Kazman, R., and Klein, M. (2002). *Evaluating Software Architecture*. The SEI Series in Software Engineering. Addison-Wesley, Boston, MA.

Cuesta, C. E., Romayb, M. P., de la Fuente, P., and Barrio-Solórzano, M. (2006). Coordination as an architectural aspect. *Electronic Notes in Theoretical Computer Science*, 154(1):25–41.

Dai, L. and Cooper, K. (2005). Modeling and analysis of non-functional requirements as aspects in a uml based architecture design. In *6th Int. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS Int. Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05)*, pages 178–183, Los Alamitos, CA, USA.

Dobrica, L. and Niemel, E. (2002). A survey on software architecture analysis methods. *IEEE Trans. Software Engineering*, 28(7):638–653.

Dolan, T. J. (February 2002). *Architecture Assessment of Information-System Families*. PhD thesis, Department of Technology Management, Eindhoven University of Technology.

Eickelmann, N. S. and Richardson, D. J. (1996). An evaluation of software test environment architectures. In *Proc. of the 18th Int. Conf. on Software Engineering (ICSE'96)*, pages 353–364, Berlin, Germany.

Filho, E. D. S., Oliveira Cavalcanti, R., Neiva, D. F., Oliveira, T. H., Lisboa, L. B., Almeida, E. S., and Lemos Meira, S. R. (2008). Evaluating domain design approaches using systematic review. In *ECSA '08: Proceedings of the 2nd European conference on Software Architecture*, pages 50–65, Berlin, Heidelberg. Springer-Verlag.

Gallagher, B. P. (2000). Using the architecture tradeoff analysis method to evaluate a reference architecture: A case study. Technical Report CMU/SEI-2000-TN-007.

Graaf, B., van Dijk, H., and van Deursen, A. (2005). Evaluating an embedded software reference architecture – industrial experience report. In *Proc. of the 9th European Conf. on Software Maintenance and Reengineering (CSMR´2005)*, pages 354–363.

Iacob, M.-E. and Jonkers, H. (2007). Quantitative analysis of service-oriented architectures. *Inte. Journal of Enterprise Information Systems*, 3(1):42–60.

Ionita, M., Hammer, D., and Obbink, H. (2002). Scenario-based software architecture evaluation methods: An overview. In *Workshop on Methods and Techniques for Software Architecture Review and Assessment at the ICSE´02*, pages 1–12, Orlando, Florida.

Ivers, J., Clements, P., Garlan, D., Nord, R., Schmerl, B., and Silva, J. R. O. (2004). Documenting component and connector views with UML 2.0. Technical report. CMU/SEI-2004-TR-008.

Kazman, R., Bass, L., Abowd, G., and Webb, M. (1994). SAAM: A method for analyzing the properties of software architectures. In *Proc. of the 16th Int. Conf. on Software Engineering*, pages 81–90, Sorrento, Italy.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. (1998). The architecture tradeoff analysis method. In *Proc. of 4th IEEE Inter. Conf. on Engineering Complex Computer Systems (ICECCS'98)*, pages 68–78, Monterey, CA, USA.

Kiczales, G., Irwin, J., Lamping, J., Loingtier, J., Lopes, C., Maeda, C., and Menhdhekar, A. (1997). Aspect-oriented programming. In *Proc. of the 11th Eur. Conf. on Object-Oriented Programming*, pages 220–242, Jyväskylä, Finland.

Kim, T., Ko, I. Y., Kang, S. W., and Lee, D. H. (2008). Extending ATAM to assess product line architecture. In *Proc. of the 8th IEEE Int. Conf. on Computer and Information Technology (CIT´2008)*, pages 790–797.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical Report TR/SE-0401, Keele University.

Kruchten, P., Obbink, H., and Stafford, J. (2006). The past, present, and future for software architecture. *IEEE Software*, 23(2):22–30.

Laddad, R. (2003). Aspect-oriented programming will improve quality. *IEEE Software*, 20(6):90–91.

Nakagawa, E. Y. and Maldonado, J. C. (2008). Architectural requirements as basis to quality of software engineering environments. *IEEE Latin America Transactions*, 6(3):260–266.

Nakagawa, E. Y., Simão, A. S., Ferrari, F., and Maldonado, J. C. (2007). Towards a reference architecture for software testing tools. In *Proc. of the 19th Int. Conf. on Software Eng. and Knowledge Eng. (SEKE'2007)*, pages 1–6, Boston, USA.

Navarro, E., Letelier, P., and Ramos, I. (2007). Requirements and scenarios: Running aspect-oriented software architectures. In *6th IEEE/IFIP Conf. on Software Architecture (WICSA'07)*, page 23, Washington, DC, USA. IEEE Computer Society.

Papazoglou, M. P. and Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415.

Rashid, A. and Chitchyan, R. (2003). Persistence as an aspect. In *Proc. of the 2nd Int. Conf. on Aspect-oriented Software Development (AOSD'03)*, pages 120–129, New York, NY, USA. ACM Press.

Schroeder, A. and Mayer, P. (2008). Verifying interaction protocol compliance of service orchestrations. In *ICSOC'08: Proceedings of the 6th International Conference on Service-Oriented Computing*, pages 545–550, Berlin, Heidelberg. Springer-Verlag.

SEI. CBAM: Cost benefit analysis method. Technical report, Software Engineering Institute. http://www.sei.cmu.edu/activities/architecture/productsservices/cbam.html.

Shaw, M. and Clements, P. (2006). The golden age of software architecture. *IEEE Software*, 23(2):31–39.

Slyngstad, O. P., Li, J., Conradi, R., and Babar, M. A. (2008). Identifying and understanding architectural risks in software evolution: An empirical study. In *Proc. of the 9th Inter. Conf. on Product-Focused Software Process Improvement (PROFES'08)*, pages 400–414, Berlin, Heidelberg. Springer-Verlag.

Stein, D., Hanenberg, S., and Unland, R. (2002). A UML-based aspect-oriented design notation for aspectj. In *Proc. of the 1st Inter. Conf. on Aspect-oriented Software Development (AOSD'02)*, pages 106–112, New York, NY, USA. ACM.

Tekinerdogan, B. (2004). ASAAM: Aspectual software architecture analysis method. In *Proc. of the 4th Working IEEE/IFIP Conf. on Software Architecture (WICSA'04)*, pages 5–14, Oslo, Norway.

Williams, L. and Smith, C. (2002). PASA: A method for the performance assessment of software architecture. In *Proc. of the 3rd Workshop on Software Performance*, pages 179–189, Rome, Italy.