



Análise da Arquitetura de Software

Essencial para Arquitetura de Software



Antonio Mendes da Silva Filho

antoniom.silvafilho@gmail.com

Professor e consultor em área de tecnologia da informação e comunicação com mais de 20 anos de experiência profissional, é autor do livros *Arquitetura de Software* e *Programando com XML*, ambos pela Editora Campus/Elsevier, tem mais de 30 artigos publicados em eventos nacionais e internacionais, colunista para *Ciência e Tecnologia* pela Revista Espaço Acadêmico com mais de 80 artigos publicados, tendo feito palestras em eventos nacionais e exterior. Foi Professor Visitante da University of Texas at Dallas e da University of Ottawa. Formado em Engenharia Elétrica pela Universidade de Pernambuco, com Mestrado em Engenharia Elétrica pela Universidade Federal da Paraíba (Campina Grande), Mestrado em Engenharia da Computação pela University of Waterloo e Doutor em Ciência da Computação pela Universidade Federal de Pernambuco.

A arquitetura de software de um sistema serve para definir a organização das funcionalidades desse sistema e propriedades ou requisitos não funcionais suportados pelo mesmo. Características peculiares da arquitetura determinarão quais propriedades serão preponderantes. A necessidade da arquitetura de software prover suporte a um conjunto de requisitos, geralmente conflitantes, requer que uma análise arquitetural seja realizada, tema tratado neste artigo.

Desenvolvimento de Sistemas de Software

O processo de desenvolvimento de sistemas de software, similarmente a outros sistemas, compreende três fases genéricas – definição, desenvolvimento e manutenção – conforme ilustrado na **Figura 1**.

De que se trata o artigo?

Apresenta o uso de requisitos arquiteturais na análise da arquitetura de software e a destaca no desenvolvimento de um sistema de software.

Para que serve?

Entender o papel da análise arquitetural dentro do processo de desenvolvimento de software orientado para arquitetura e definição da arquitetura de software a ser adotada.

Em que situação o tema é útil?

Identificação de requisitos arquiteturais importantes a um sistema de software em desenvolvimento e construção de cenários para definição da arquitetura e como esta suporta esses requisitos.

A fase de definição engloba a identificação de informações que deveriam ser processadas, funções e desempenho desejados, tipo de interface a ser utilizada, tarefas que o sistema deveria prover suporte, perfil de usuários do sistema, dentre outras.

A fase de desenvolvimento concentra-se no projeto de estruturas de dados e arquitetura de software do sistema, conversão do projeto para alguma linguagem de

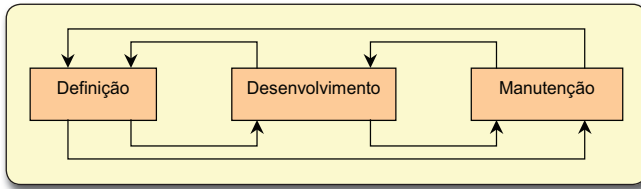


Figura 1. Fases genéricas no processo de desenvolvimento de software

programação, realização de testes e avaliação. Finalmente, a manutenção considera modificações e/ou correções necessárias no sistema a fim de que este atenda aos requisitos do usuário. Perceba que o processo de desenvolvimento de um sistema de software tem duas grandes atividades de interesse: o desenvolvimento da porção de software que implementa as funcionalidades do sistema, e a atividade que a antecede e norteia o desenvolvimento, que é o projeto da arquitetura de software. Essa última atividade é resultado da análise arquitetural que provê informações para decisões arquiteturais, como ilustrado na Figura 2. A análise arquitetural e os critérios para definição de uma arquitetura são apresentadas nas seções subsequentes.

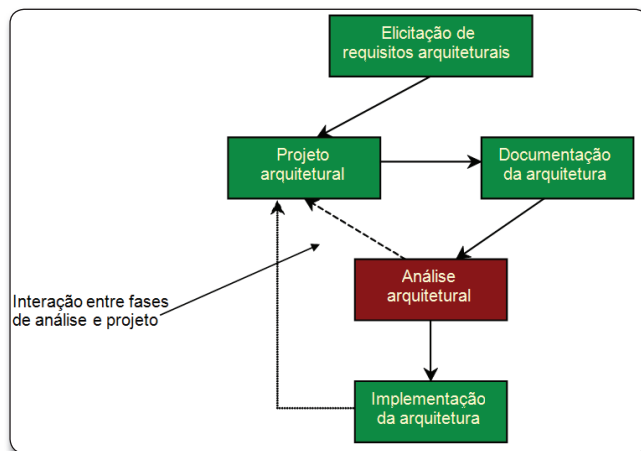


Figura 2. Etapas no projeto da arquitetura de software.

Análise da Arquitetura de Software

O projeto da arquitetura de software é uma etapa essencial no desenvolvimento de sistemas de software de grande porte. Dentro deste contexto, a arquitetura de software é fundamental para o desenvolvimento de software onde se tem funcionalidades distintas, sendo concebidas a partir da mesma arquitetura base. Entretanto, antecedendo a etapa de projeto da arquitetura de software, há a necessidade de fazer o levantamento dos requisitos do sistema, como mostrado na Figura 2.

De um modo geral, a análise e projeto de um sistema geralmente engloba as atividades de:

- **Desenvolvimento de modelo arquitetural** – Os dados são coletados durante a elicitação de requisitos a fim de serem analisados e incorporados ao modelo arquitetural (isto é, o produto resultante). Neste caso, o modelo passa a ser o elemento central da análise.
- **Melhoria e síntese de uma solução** – Incrementam-se as informações descritas no modelo arquitetural (inicial).

- **Análise da solução** – A análise da solução é realizada em termos do modelo arquitetural que se tem em mãos, podendo identificar a necessidade de refinar o modelo.

Perceba que o modelo arquitetural pode ser considerado como um ‘esboço’ inicial da arquitetura de software do sistema em desenvolvimento. É importante ainda observar que a análise de uma arquitetura de software é um processo iterativo no qual são feitos refinamento de um modelo arquitetural inicial, derivado a partir de um conjunto de requisitos arquiteturais. Em cada iteração, uma mini análise ocorre, refinando a arquitetura proposta inicialmente. Esse processo é ilustrado na Figura 3.

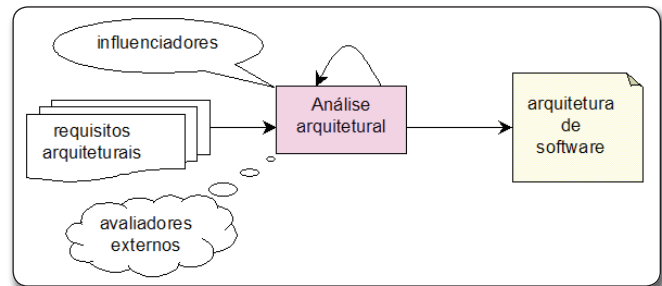


Figura 3. Análise arquitetural.

Entretanto, note que não é tarefa fácil validar uma arquitetura de software quanto ao suporte que oferece a um conjunto de requisitos. É muitas vezes difícil associar novas arquiteturas a atributos de qualidade. Observa-se, geralmente, que os desenvolvedores concentram-se nos aspectos funcionais das arquiteturas. Assim, o principal propósito da análise arquitetural é verificar se os requisitos arquiteturais têm sido levados em conta durante o desenvolvimento de um sistema de software.

Propriedades da Arquitetura de Software

A arquitetura de software é uma parte essencial de um sistema de software complexo. Com a crescente complexidade de sistemas de software, a especificação global do sistema, ou seja, sua arquitetura passa a ser um aspecto de maior significado quando comparado à escolha de algoritmos ou estruturas de dados. Um sistema possui um conjunto de requisitos arquiteturais, envolvendo atributos de qualidade (ou requisitos não funcionais) e atributos de projeto. Esses requisitos constituem propriedades desejadas ou apresentadas por uma arquitetura de software. Dentre esses requisitos, uma arquitetura de software apresenta propriedades importantes que podem ser vistas como intrínsecas a ela. Tais propriedades são: eficiência, integridade e flexibilidade

A eficiência pode ser descrita como a quantidade de recursos computacionais necessários para que um programa realize suas funções. Esses recursos computacionais podem ser vistos em termos da capacidade de armazenamento e processamento. Um dos principais requisitos associados à eficiência é o desempenho. O desempenho é um requisito não funcional essencial para um sistema de software e constitui num enorme desafio

lidar com tal requisito durante o desenvolvimento de um sistema. Por exemplo, seria possível ter os seguintes requisitos de desempenho quando do desenvolvimento de um sistema de software para administração de cartões de crédito:

- Obter um bom tempo de resposta para autorização de vendas com cartão de crédito. O termo bom poderia ser traduzido em 7 segundos.
- Obter um bom uso de memória para armazenar as informações de um cliente. Poderia ser especificado o máximo de 10 Kb para armazenar as informações de cada cliente.

A eficiência é uma propriedade de suma importância nas decisões de projeto e tem forte influência na análise arquitetural de um sistema. Perceba que a forma na qual os componentes de uma arquitetura encontram-se distribuídos, bem como os mecanismos de comunicação entre eles, são determinantes do desempenho e, portanto, da eficiência do sistema. Essa propriedade, às vezes, atua em conflito com outros requisitos do sistema, exigindo que compromissos sejam assumidos em termos de custo e desempenho de um sistema de software.

Outra importante propriedade é a integridade. Aqui, a noção de integridade refere-se à integridade em termos da unificação do projeto em níveis distintos. A arquitetura de software descreve a organização de um sistema de software num nível mais elevado, objetivando a unificação do projeto, ou seja, serve de referência para o projeto, conforme ilustrado na **Figura 4**.

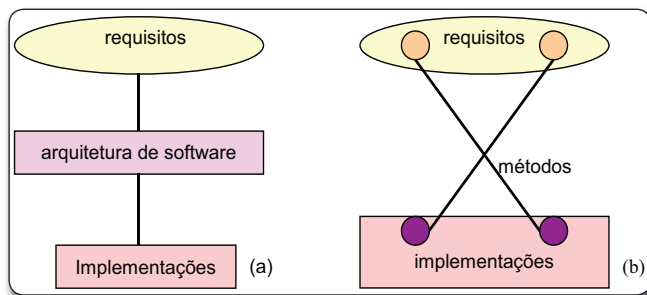


Figura 4. Arquitetura de software como referência de projeto.

Note que a arquitetura de software é central e essencial à qualidade de um sistema, associados um conjunto de atributos de qualidade. Similarmente, a integridade (arquitetural) funciona como elemento coordenador que unifica o projeto do sistema de software em diferentes níveis.

Complementando, outra importante propriedade é a flexibilidade, que diz respeito ao esforço exigido para modificar um sistema de software. Em outras palavras, é a facilidade com a qual um sistema de software pode ser estendido a fim de dar suporte a uma ou mais funcionalidade(s) adicionada(s).

Por exemplo, a criação de interfaces adicionais aumenta flexibilidade, incrementando a facilidade de fazer modificações, uma vez que as interfaces separam partes distintas de funcionalidade em componentes de software diferentes.

SAAM – Método de Análise de Arquitetura de Software

Um exemplo de método de análise de arquitetura de software, denominado SAAM (Software Architecture Analysis Method), foi inicialmente concebido com o objetivo de auxiliar arquitetos de software a compararem soluções arquiteturais, tendo sido o precursor de outros métodos de análise arquitetural. O método SAAM compreende os seguintes objetivos:

- Definir um conjunto de cenários que representem usos importantes do sistema no domínio, envolvendo os pontos de vista de todos aqueles que possuem papéis influenciadores no sistema.
- Utilizar cenários para gerar uma partição funcional do domínio, uma ordenação parcial das funções e um acoplamento dos cenários às várias funções existentes na partição.
- Usar a partição funcional e ordenação parcial, juntamente com cenários de uso, a fim de realizar a análise das arquiteturas propostas.

Assim, para cada cenário, a análise fornece uma pontuação das arquiteturas que estão sendo avaliadas, denominadas de arquiteturas candidatas. Recai, portanto, sobre o avaliador a responsabilidade de determinar os pesos dos cenários considerados, bem como a pontuação global das arquiteturas candidatas.

O método de análise arquitetural SAAM é baseado no uso de cenários. Fazendo uso de cenários, um analista pode usar uma descrição de arquitetura de software para avaliar o potencial do sistema de software prover suporte a atributos de qualidade ou requisitos não funcionais. Entretanto, é importante observar que avaliar uma arquitetura de software para determinar se ela oferece suporte a requisitos não funcionais não é tarefa fácil. Tais requisitos tendem a serem um tanto vagos.

Objetivando realizar a análise, portanto, torna-se necessário considerar o contexto no qual o sistema de software encontra-se inserido, bem como considerar as circunstâncias específicas àquele contexto. Uma forma de atingir esse objetivo é fazendo uso de cenários a fim de entender e avaliar se uma arquitetura oferece suporte ou não a um específico requisito não funcional, e sob quais circunstâncias.

SAAM compreende um conjunto de cinco passos que possuem interdependências entre si, conforme ilustra a **Figura 5** e descritas a seguir.

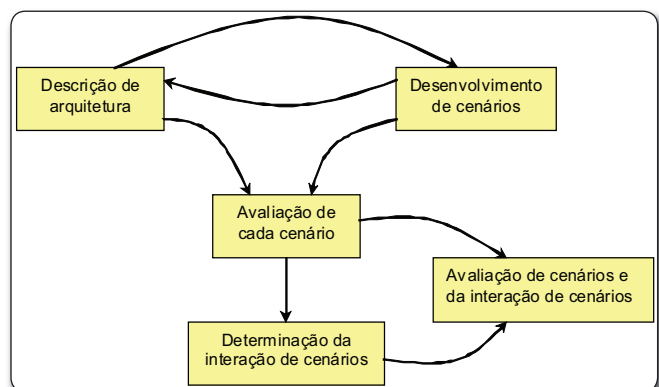


Figura 5. Etapas de SAAM.

1. Desenvolvimento de cenários – Desenvolvimento de cenários de tarefas que ilustram os tipos de atividades que o sistema de software deve prover suporte. Também, o analista deve considerar possíveis modificações que o sistema possa sofrer ao longo do tempo. Ao desenvolver esses cenários, um arquiteto deve capturar todos os usos importantes do sistema, que irão refletir os requisitos não funcionais. Além disso, esses cenários deverão apresentar interações, considerando os principais influenciadores com papéis relevantes ao sistema de software, tais como: usuário e/ou cliente, desenvolvedor, mantenedor e administrador de sistema.

2. Descrição de arquitetura – Para cada análise, deve-se possuir uma representação arquitetural do sistema, fazendo uso de uma notação arquitetural definida, descrevendo componentes e conectores utilizados na especificação da arquitetura. Isto permitirá que o pessoal envolvido na análise tenha um entendimento comum do problema. A especificação arquitetural deve informar os componentes funcionais e de dados do sistema bem como as relações existentes entre eles. Desse modo, uma representação arquitetural fará uma distinção entre componentes ativos (que efetuam transformações de dados) e componentes passivos (que armazenam dados). Adicionalmente, devemos considerar dois tipos de conexões: (a) conexões de dados (onde se tem passagem de dados entre componentes) e (b) conexões de controle (na qual um componente atua sobre um outro o habilitando a executar alguma tarefa). É importante observar que a descrição gráfica nos dá uma representação estática da arquitetura do sistema, que é complementada através de uma representação dinâmica expressa em linguagem natural que fornece uma descrição do comportamento do sistema ao longo do tempo.

3. Avaliação de cada cenário – Para cada cenário, deve-se determinar se a arquitetura candidata oferece suporte diretamente às tarefas associadas ao cenário ou se alguma modificação é necessária e, portanto, a arquitetura oferece suporte ao cenário apenas indiretamente. No primeiro caso, os cenários representam funções que a arquitetura já possui. O segundo caso compreende cenários que incluem funções além das existentes que a arquitetura de prover suporte. Neste caso, uma modificação deve ser feita na arquitetura através da adição ou alteração de um componente ou conexão. Ao final, deveria obter uma tabela sumarizando os cenários onde a arquitetura candidata provê suporte aos requisitos não funcionais associados. Nessa tabela, pode-se adotar a seguinte convenção:

- Se a arquitetura candidata suporta a(s) tarefa(s) contida(s) no cenário considerado, então se pode atribuir um escore +, e não há necessidade de fazer qualquer análise adicional.
- Se a arquitetura candidata não suporta diretamente a(s) tarefa(s) contida(s) no cenário, então se precisa continuar a avaliação a fim de determinar a extensão das modificações que devem ser realizadas. Uma forma de determinar isso é conhecendo o número de componentes e conexões que precisarão ser adicionados ou modificados. Isto requer um conhecimento profundo da arquitetura, o qual pode ser obtido junto ao desenvolvedor ou através do conjunto de especificações do sistema. Se duas arquiteturas estão sendo comparadas, então aquela que exigir menos modificação ou adição de componentes e conexões, receberá o escore +, enquanto aquela requerendo mais adições ou modificações em componentes e conexões, receberá o escore -.

4. Determinação da interação de cenários – Uma interação entre cenários ocorre quando dois ou mais cenários indiretos exigem modificações em algum componente ou conexão. A identificação da interação de cenários permite saber como a funcionalidade do sistema de software é alocada durante o projeto. Em outras palavras, a interação de cenários mostra as tarefas às quais os componentes estão associados. Na realidade, a determinação da interação de cenários avalia a extensão à qual a arquitetura provê suporte a uma separação de interesses apropriada. Por exemplo, um elevado grau de interação pode indicar que a funcionalidade de um componente específico está inadequadamente isolada. Dessa forma, para cada cenário, deve-se determinar como os componentes e conexões são afetadas pelo cenário considerado.

5. Avaliação de cenários e da interação de cenários – Esta é uma etapa subjetiva do processo de análise, envolvendo os principais influenciadores do sistema. Nesta etapa, realiza-se uma avaliação global atribuindo um escore a cada cenário e interação de cenário. Isto é baseado nas implicações que os cenários têm sobre a arquitetura do sistema. Note que, anteriormente, foram desenvolvidos cenários, fez-se o mapeamento deles na descrição arquitetural do sistema, e determinaram-se as interações de cenários existentes. Esta avaliação reflete a influência dos atributos de qualidade (ou requisitos não funcionais) associados aos cenários.

Este método de análise permite um avaliador utilizar um conjunto de métricas associadas com os cenários elaborados para que possa analisar qual arquitetura de software, dentre as arquiteturas candidatas, provê melhor suporte ao conjunto de requisitos não funcionais desejados para o sistema de software.

É importante observar que as duas primeiras etapas possuem elevado grau de dependência entre elas, como mostrado na **Figura 5**. Os tipos de cenários desenvolvidos terão influência sobre o nível de granulosidade da arquitetura. Agora, como se pode determinar um conjunto de cenários? Isto dependerá do tipo de tarefas que o sistema de software suportará. Dentro deste contexto, os cenários elaborados têm um papel de suma importância, uma vez que eles irão ajudar a compreender a descrição arquitetural do sistema.

Considere, por exemplo, uma das arquiteturas de software para interface com usuário, denominada de Serpent, ilustrada na **Figura 6**.

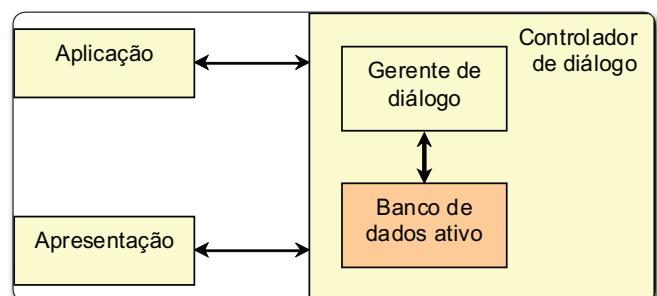


Figura 6. Arquitetura Serpent.

O componente de aplicação dá suporte à semântica computacional da aplicação. A apresentação é um componente que oferece suporte aos mecanismos de interação nos níveis lógico e físico, sendo completamente independente da semântica da aplicação. A coordenação entre estes dois primeiros componentes é feita através do controlador de diálogo. Esse terceiro componente é encarregado de mediar a comunicação da interação do usuário com a aplicação. Toda a comunicação na arquitetura Serpent é mediada por meio de restrições a dados compartilhados de um banco de dados. Esta estrutura é implementada como um banco de dados ativo. Assim, quando os valores no banco de dados são alterados, eles são automaticamente comunicados a qualquer componente registrado que possua interesse sobre o dado modificado.

Uma partição canônica para software de interface com usuário, considerada como um meta-modelo para arquiteturas de sistemas interativos, identifica que as cinco funções básicas em um software de interface compreendem:

- **Núcleo Funcional (NF)** - realiza as manipulações de dados e outras funções orientadas ao domínio. É também chamado de componente específico do domínio ou, simplesmente, aplicação.
- **Adaptador de Núcleo Funcional (ANF)** - agrega os dados específicos do domínio em estruturas de alto nível, realiza verificações semânticas e dispara tarefas de diálogo inicializadas no domínio. Ele é um mediador entre o Diálogo e o Núcleo Funcional.
- **Diálogo (D)** - faz a mediação entre partes específicas do domínio e específicas da apresentação de uma interface com usuário, realizando o mapeamento de dados quando necessário. Assegura consistência e o sequenciamento de tarefas.
- **Componente de Interação Lógica (IL)** - fornece um conjunto de objetos independentes de toolkit para o componente de Diálogo.
- **Componente de Interação Física (IF)** - implementa a interação física entre usuário e computador. Lida com dispositivos de entrada e saída e é, geralmente, conhecido como componente de toolkit de interação.

Agora, se tentarmos fazer o mapeamento dessas cinco funções para sistemas interativos (na arquitetura de software de interface de usuário de Serpent), obteremos uma arquitetura conforme ilustrada na **Figura 7**.

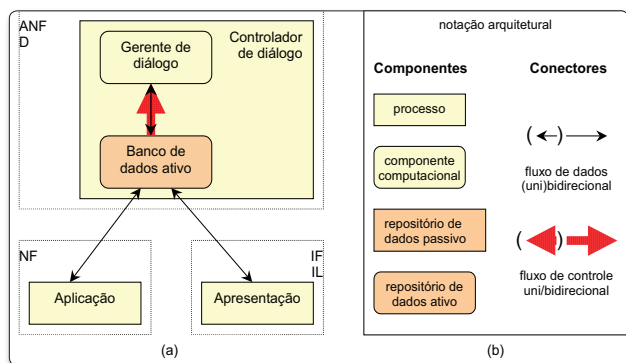


Figura 7. Arquitetura Serpent baseada no meta-modelo de arquitetura.

A arquitetura Serpent foi redesenhada fazendo uso da notação arquitetural apresentada na **Figura 7b**. As funções básicas de um software de interface com usuário foram mapeadas na estrutura inicial da arquitetura Serpent (vide **Figura 6**), resultando a arquitetura ilustrada na **Figura 7a**. Algumas observações podem ser feitas:

- Não há separação arquitetural entre as funções IL e IF no processo de apresentação;
- Também, não existe separação arquitetural entre as funções de ANF e D no processo do controlador de diálogo;
- Exceto o gerenciador de diálogo, todos os componentes de Serpent são monolíticos. Em outras palavras, eles não oferecem suporte arquitetural para subdivisão de funcionalidade dentro de um componente.

Considerando a arquitetura Serpent, vamos supor que se deseja efetuar uma modificação como, por exemplo, adicionar uma nova opção de menu. Vamos avaliar o grau no qual essa arquitetura suporta essa modificação. Perceba que a arquitetura Serpent separou o controlador de diálogo e, portanto, é fácil verificar que a modificação proposta (adicionar uma opção de menu) deverá ser feita nesse processo. O controlador de diálogo possui dois componentes: o gerente de diálogo, responsável por comandar o comportamento do diálogo, e o banco de dados ativo, o qual mantém o estado do diálogo. Esse cenário, no qual se deseja fazer uma modificação (adição de nova opção de menu), está associado à mudança no comportamento do diálogo e, portanto, está isolado no gerente de diálogo. Adicionalmente, o gerente de diálogo é subdividido em componentes menores associados a partes específicas do comportamento do diálogo como, por exemplo, um menu. Desse modo, pode-se concluir que a arquitetura Serpent fornece suporte apropriado a este tipo de cenário envolvendo modificação no aspecto de um sistema interativo.

Uso de Atributos de Qualidade na Análise Arquitetural

Uma das razões de concentrar atenção sobre a arquitetura é para identificar e analisar as decisões iniciais de projeto. Transformar estas decisões de projeto em uma estrutura que ofereça suporte a um raciocínio que permita antecipar tendências a nível arquitetural é essencial para obter os benefícios do foco na arquitetura.

Embora possamos apontar indicativos de qualidade, seria praticamente impossível fazer previsão de qualidade num estágio inicial de projeto. Note que, por exemplo, o método de análise arquitetural SAAM não objetiva precisamente prever o comportamento de atributos de qualidade. A meta desse método de análise dá-se em entender o papel e as conseqüências de decisões arquiteturais em relação aos atributos de qualidade de algum sistema que esteja sendo analisado.

Dessa forma, o método SAAM apresentado serve de ferramenta ao arquiteto ou engenheiro de software para prever suporte a atributos de qualidade. Note que isto é baseado em decisões de projeto que ocorrem no nível arquitetural.

Sabemos que a arquitetura é um elemento essencial para a obtenção de sucesso, em termos tecnológicos e de mercado de

qualquer empresa. Geralmente, as metas de qualidade e conjunto de funcionalidades constituem os principais motivadores de um sistema. Considere, por exemplo, um fabricante de centrais telefônicas que esteja desenvolvendo um projeto de uma nova central. Assim, pode ser especificado que a central deve ser capaz de rotear ligações telefônicas, prover suporte a diversos tipos de tons, redirecionar ligações, gerar contas telefônicas de usuários com discriminação de ligações efetuadas, dentre outras funcionalidades. Entretanto, para que o sistema alcance sucesso na prestação de serviços para seus assinantes, ele deve também operar com elevado nível de desempenho, facilitar quaisquer modificações (em termos de adição ou alteração de características), possuir bom nível de disponibilidade e ter baixo custo.

Como essas metas de qualidade podem ser alcançadas?

A arquitetura do sistema é um meio pelo qual podemos determinar se essas metas de qualidade podem ser realizáveis ou não. SAAM e outros métodos constituem respostas a essa questão. Perceba que essa necessidade de prever o comportamento de atributos de qualidade num sistema requer conhecimento de estilos arquiteturais prováveis de serem empregados no sistema. Um estilo arquitetural inclui:

- Descrição de tipos de componentes e sua topologia;
- Descrição de padrão de interação (no nível de dados e controle) entre os componentes;
- Descrição informal das vantagens e desvantagens na adoção do estilo.

Estilos arquiteturais permitem ao arquiteto de software distinguir classes de projeto através de evidências existentes de como cada classe tem sido utilizada juntamente com o raciocínio qualitativo para explicar porque certas classes têm determinadas propriedades. Assim, por exemplo, pode-se adotar o estilo arquitetural de pipes e filtros quando se deseja obter reuso e não há restrição quanto ao desempenho.

Agora, pode ser feito um mapeamento de um estilo arquitetural em um conjunto de modelos de atributos de qualidade. Estes modelos de atributos de qualidade ajudam a prever o comportamento desses atributos. Esta noção de mapeamento de decisões e propriedades arquiteturais em modelos de atributos de qualidade é ilustrada na **Figura 8**.

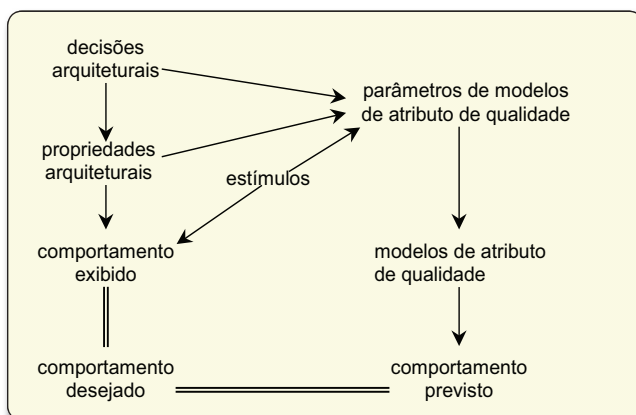


Figura 8. Mapeamento de modelos arquiteturais em modelos de atributos.

É importante observar que decisões arquiteturais influenciam direta e indiretamente no comportamento apresentado pela arquitetura. Como poderíamos caracterizar esse comportamento? Considere, por exemplo, a alocação de funcionalidade a um conjunto de processos. Isto requer que o arquiteto ou engenheiro de software tome decisões arquiteturais. Disto resulta que tais processos precisam ser alocados a processadores, requerendo ainda mais decisões arquiteturais. Note que, associado a eles, temos um conjunto de tempos de execução de processo, os quais constituem propriedades arquiteturais. Dessa forma, as propriedades arquiteturais, em conjunto com estímulos (tais como taxa de chegada de mensagens), nos leva a um comportamento de desempenho exibido pelo sistema.

É importante observar que, para avaliar uma arquitetura em função de requisitos não funcionais, é necessário expressar esses requisitos ou atributos de qualidade em termos mensuráveis, denominado de medidas de atributos de qualidade. Tais medidas dependem de propriedades da arquitetura ou parâmetros de atributos de qualidade e dos estímulos. Considere como atributo de qualidade o desempenho. Esse requisito pode ser expresso em termos das seguintes medidas de atributo de qualidade:

- Latência - tempo da ocorrência de um evento até a resposta àquele evento, dado em unidades de tempo;
- Throughput - taxa na qual o sistema pode responder a eventos, dada em transações por unidades de tempo.

Os estímulos podem ser vistos como mudanças de estado ao qual a arquitetura deve responder. Se, novamente, considerarmos o desempenho, temos o padrão de chegada de um evento que pode ser periódico, esporádico ou probabilístico. Se um estímulo ocorre, então o sistema deve responder fazendo uso de seus recursos para realizar alguma computação.

Note que analisar uma arquitetura é uma atividade investigativa na qual o avaliador busca saber quão bem uma arquitetura tem sido projetada em relação aos requisitos não funcionais desejados para o sistema. Tais requisitos estão intrinsecamente associados à qualidade do sistema. Um arquiteto de software, ao tomar decisões de projeto, busca maximizar os benefícios obtidos com o sistema e, ao mesmo tempo, minimizar os custos de implementação do projeto. Note que as metas de negócios de uma empresa também têm influência sobre decisões arquiteturais tomadas pelo projetista. Essas decisões têm implicações técnicas e econômicas.

As implicações técnicas compreendem as características do sistema de software e os atributos de qualidade desejados. As implicações econômicas referem-se ao custo de implementar o sistema. A **Figura 9** ilustra o contexto das decisões arquiteturais e implicações associadas.

Considerando a **Figura 9**, tem-se que a determinação dos custos e benefícios associados às decisões arquiteturais pode ser obtida através de:

- Avaliação da importância dos requisitos não funcionais;
- Quantificação dos escores de benefícios dos requisitos não funcionais;
- Quantificação dos custos dos requisitos não funcionais;
- Escolha de cenários e estratégias de projeto arquitetural.

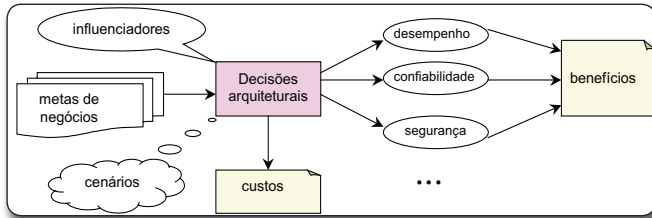


Figura 9. Contexto de decisões arquiteturais e implicações associadas.

É importante observar que, para quantificar e compreender de uma forma consistente o impacto que as estratégias de projeto arquitetural têm sobre os atributos de qualidade, os influenciadores necessitam de uma representação mais concreta dos atributos de qualidade. Para este propósito, cenários devem ser utilizados. Durante a análise arquitetural, a elicitação e análise de cenários permitem a caracterização de atributos de qualidade. Esses cenários especificam características envolvendo estímulos e respostas concretizando os atributos de qualidade. Isto permite que os influenciadores possam avaliar os efeitos desses atributos. Esta avaliação é centrada na premissa de maximizar os benefícios obtidos com a arquitetura de um sistema e, ao mesmo tempo, minimizar os custos associados com sua implementação.

Conclusão

Neste artigo, um conjunto de propriedades de arquitetura de software consideradas intrínsecas à estrutura do sistema foi apresentado. Estas propriedades, juntamente com os requisitos arquiteturais, envolvendo requisitos não funcionais e atributos de projeto, servem de base para a análise arquitetural.

O método de análise arquitetural SAAM foi apresentado, ilustrando-se os passos de como fazer a análise. Adicionalmente, foi discutido o papel dos requisitos não funcionais, ou atributos de qualidade, como mecanismo para auxiliar nas decisões arquiteturais. Cabe destacar que a análise arquitetural é realizada sob a óptica de minimizar custos e agregar benefícios, isto é, prover suporte aos requisitos não funcionais. ●

Links

Scenario-Based Analysis of Software Architecture

http://www.sei.cmu.edu/architecture/scenario_paper/

SAAM: A Method for Analyzing the Properties of Software Architectures

<http://www.sei.cmu.edu/publications/articles/saam-metho-propert-sas.html>

SEI's Software Architecture Technology Initiative

www.sei.cmu.edu/architecture/sat_init.html

The Serpent Runtime Architecture and Dialogue Model

<http://www.sei.cmu.edu/pub/documents/88.reports/pdf/tr06.88.pdf>

The Software Architecture Portal

<http://www.softwarearchitectureportal.org/>

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto.

Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback

