Evolução de software baseada em avaliação de Arquitetura de Software.

Danielle P. Noronha Pontes[1], Reginaldo Arakaki [2]

1 Escola Superior de Tecnologia – Universidade Estadual do Amazonas (UEA) / MINTER(UEA/USP), Manaus – AM – Brasil; 2 Engenharia da Computação e Sistemas Digitais - Universidade de São Paulo,
São Paulo – SP – Brasil

dani@doctortech.com.br,reginaldo.arakaki@poli.usp.br

Abstract. Este trabalho discorre sobre o estudo da utilização do método de avaliação ATAM como referência para evolução arquitetural de um sistema legado. O estudo apresentado está dividido em duas partes: a elaboração de um roteiro para evolução de software e a aplicação do roteiro em um ambiente real de um sistema para automação de linhas aéreas. Um dos objetivos é aplicar a correlação entre a evolução arquitetural e os requisitos não-funcionais. As decisões arquiteturais para a evolução do software são tomadas com base nas evidências obtidas na avaliação arquitetural realizada a partir dos atributos de qualidade estabelecidos como meta.

Keywords: Evolução de Software, Arquitetura de Software, Engenharia de Software, RM-ODP, ATAM.

1 Introdução.

O mercado de desenvolvimento de software tem apresentado mudanças significativas no que diz respeito à construção de sistemas. Atualmente encontram-se sistemas complexos instanciados nos mais diversos ramos de atividade o que dificulta as tentativas de substituição de plataforma, sistema ou fornecedor. Entretanto o processo de envelhecimento de um software é natural e inevitável, o que gera uma necessidade constante de evolução dos sistemas que esperam e precisam se manter ativos por um período longo de tempo [1]. Outro fator relevante a ser apontado é que parte dos sistemas corporativos instanciados no mercado apresenta problemas estruturais que afetam negativamente alguns requisitos fundamentais de qualidade, o que demanda uma evolução estrutural para adequar a novas exigências do usuário [1]. Mudar sistemas sem técnica pode prejudicar alguns aspectos de qualidade da arquitetura. Para decidir como implementar as mudanças, é preciso usar um método que permita o controle da qualidade da arquitetura.

Adaptar os sistemas para as novas necessidades do mercado tais como: integração, flexibilidade, portabilidade, qualidade do serviço e segurança é a maneira de aumentar o tempo de vida de um sistema. Esta área de estudo tem evoluído significativamente.

Um método estruturado ajuda a garantir que questões importantes serão tratadas antecipadamente, durante os estágios de análise de requisitos e projeto quando os problemas podem ser resolvidos de forma mais barata. Dentro do processo proposto para evolução de software optou-se por utilizar o método de avaliação *Architecture Tradeoff Analysis Method* (ATAM) como referência para apoiar a evolução arquitetural. Este método foi escolhido porque o fundamento da avaliação é estabelecer os atributos de qualidade desejados a partir dos pontos críticos do processo de negócio (referenciados como *business drivers*) [2]. Durante a tentativa de atender as necessidades de mercado situações nas quais há conflito de escolha (*trade-off*) sempre surgem.

O objetivo do trabalho é estudar a utilização do método de avaliação ATAM no apoio da evolução arquitetural de um sistema de software, decorrente da correlação entre a evolução arquitetural e os requisitos funcionais.

Para equacionar essas questões são apresentados um roteiro de avaliação e os resultados de um exemplo prático. O roteiro para avaliação de arquitetura de software terá como referência o método de avaliação de arquitetura de software ATAM e como objetivo, gerar um plano de evolução a partir de sua arquitetura, assegurando a longevidade do sistema no mercado. O exemplo prático da aplicação deste roteiro será em um ambiente real de uma empresa de aviação.

2 Evolução de Software e Arquitetura de Software.

Bass Graaf em [3] distingue claramente dois tipos de transformações durante a vida de um software. Segundo o autor os modelos e processos existentes envolvem tipicamente transformações verticais, do abstrato para o concreto, como acontece no ciclo de desenvolvimento de software. Por outro lado atividades como manutenção e evolução, típicas em qualquer software, envolvem transformações horizontais como a migração do sistema de uma plataforma para outra. Este trabalho propõe o planejamento para execução de transformações horizontais que envolvem aspectos arquiteturais do sistema.

Como uma das soluções para a problemática da evolução do software o autor faz a seguinte colocação: ” O software tornou-se mais e mais difícil de entender, manter ou adaptar, e difícil de reusar e evoluir. Isso acontece devido ao aumento do tamanho e complexidade dos softwares e pela rápida evolução das tecnologias de processamento de dados. A arquitetura de Software esta emergindo como uma solução para esta problemática [4].”

Na mesma linha de argumentação usada por Sadou, Chávez (2009) em seu trabalho[5], cita os autores Krutchen, Obbink e Staford (2006)[6] que afirmam que do ponto de vista prático é possível controlar e supervisionar o desenvolvimento e evolução de sistemas através da arquitetura de software.

De acordo com Garlan e Perry em [7] a arquitetura de software pode expor as dimensões através das quais um sistema deve evoluir. Evolução subtende a idéia de mudança essencial que não esta clara no termo manutenção. A evolução sugere novos designs evoluindo de sistemas antigos. Finalmente pode ser argumentado que manutenção e evolução oferecem diferentes perspectivas da natureza da mudança.[8].

É a inovação e não a preservação, que direciona a mudança de software: um sistema moderno adaptado a novos ambientes evolui de um sistema antigo. [8].

Em [9] os autores acreditam que um dos desafios na pesquisa em evolução de software é como analisar as mudanças estruturais dos sistemas. O autor utiliza o termo arquitetura de software para se referir a estrutura de um sistema, enfatizando a organização dos componentes e os relacionamentos entre eles. Para obter o sucesso neste processo, a evolução da arquitetura deve ser identificada e gerenciada para manter a coerência da arquitetura. [4]. Desta forma, enquanto a manutenção degrada a vida e a confiabilidade do software, a evolução planeja mudanças que permitirão um tempo de vida longo ao software.

Bennett e Rajlich apresentam um modelo descritivo de evolução de software chamado de *StagedModel* de manutenção e evolução que sumarizam muitas dessas idéias. O modelo é dividido em quatro estágios: Desenvolvimento Inicial, Evolução ativa, *Servicing* e *Phase out.* [8].

Considerando o objeto dessa investigação, cumpre reportar-se ao estágio do *Phase out.* Neste existe claramente a decisão de substituir ou eliminar o sistema inteiramente por causa do alto custo de manutenção ou pelo surgimento de novas soluções tecnológicas. Para tanto o autor da ênfase a elaboração de uma estratégia de saída. Diferentemente dessa perspectiva, esta pesquisa aponta, por fim, outra direção: pensar em uma estratégia que planeje a criação de uma nova versão para possibilitar a sobrevivência do software no mercado.

3 Roteiro para avaliação de arquitetura

O processo utilizado para criação do plano de evolução tem como referência os 9 passos do método de avaliação de arquiteturas – ATAM . A Tabela 1 apresenta as fases do processo e os passos envolvidos em cada uma.

Tabela 1. Etapas do Método ATAM

	ETAPAS	PONTOS IMPORTANTES
APRESEN-TAÇÃO	1 - Apresentar o ATAM 2 - Apresentar os objetivos de negócio 3 - Apresentação da arquitetura	Nesta fase são conhecidas as necessidades do cliente e a realidade do sistema avaliado.
INVESTIGAÇÃO E ANÁLISE	4 - Identificando métodos arquiteturais 5 - Gerar a árvore de utilidade. 6 – Análise dos métodos arquiteturais	Nesta fase os participantes sugerem agrupamentos de cenários. Depois que a votação esta completa, os avaliadores determinam um ponto de corte com até 15 cenários.
TESTES	7 – Brainstorm e priorização de cenários 8 – Análise dos métodos arquiteturais	Nesta fase são levantados os pontos de conflitos. É necessário localizar todos os elementos arquiteturais importantes onde existem múltiplos pontos de sensibilidades.
ENTREGA	9 – Consolidar os resultados	Esse plano é um conjunto de recomendações para reestruturar a arquitetura sob a luz dos resultados da analise.

Na etapa de **definição das diretrizes** (Fig. 1) é realizado o processo de avaliação do sistema baseado no método ATAM. Os 9 passos do método foram adaptados no sentido de direcionar a avaliação para tratar questões ligadas à evolução do software.

A fase de entrega, aonde se encontra o passo 9 (Tabela 1), resume os resultados que serão utilizados na consolidação das diretrizes.

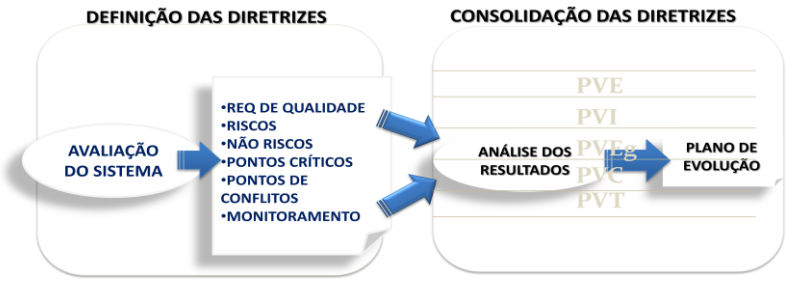

Fig. 1. Estrutura do roteiro utilizado na pesquisa

A etapa de **consolidação das diretrizes** consiste na análise dos dados e classificação das táticas geradas pelos avaliadores de acordo com os pontos de vista do RM-ODP. De posse dos resultados da avaliação é realizada uma análise dos resultados no sentido de elaborar o plano de evolução. Informações como analise arquitetural dos cenários prioritários, pontos de sensibilidades e *trade-offs* são consolidados em um documento. Os artefatos resultantes do processo de avaliação através do ATAM são adequados para pautar a estratégia da evolução do software. Neste documento estão: os Requisitos de Qualidade; os Riscos e não riscos; os Pontos de sensibilidades e críticos; os Pontos de Conflitos e o Plano de Monitoramento de Riscos.

As táticas resultantes do processo irão direcionar as ações que devem ser tomadas para evolução do software de acordo com os requisitos colocados como meta. Os *trade-offs* irão indicar quais as perdas e ganhos entre as táticas conflitantes. E por fim os riscos apontam os fatores e variáveis que devem ser observados, monitorados e controlados durante o processo de evolução. Este conjunto de resultados forma as decisões arquiteturais que serão analisadas com o objetivo de traçar as diretivas para evolução de cada ponto de vista do RM-ODP observando o seu nível de abstração, (*Reference Model - Open Distributed Processing*) [ISO/IEC 10746-3 *apud* [10]]. De acordo com o padrão ISO 42010 [ISO/IEC 2007 apud 5] um ponto de vista fornecem um padrão ou template, a partir do qual é possível desenvolver visões individuais, estabelecer os objetivos e a audiência para uma visão e as técnicas para sua criação e análise. [5].

O agrupamento das decisões arquiteturais a partir das visões definidas no RM-ODP possibilitará agilidade e eficiência ao processo de evolução, uma vez que pode ser direcionado para as áreas e pessoas competentes. Desta forma o equipe de desenvolvimento poderá elaborar um planejamento estratégico para o desenvolvimento da nova versão do produto direcionado a partir dos pontos de vista do RM-ODP, que são:

- Ponto de Vista de Empresa (PVE): Processo de negócio
- Ponto de vista de Informação (PVI): Informações e estruturas estratégicas

- Ponto de Vista de Computação (PVC): Funcionalidades, cálculos e ações comportamentais.
- Ponto de Vista de Engenharia (PVEg): Mecânismos de engenharias.
- Ponto de Vista Tecnologia (PVT): Plataforma tecnológica.

Estas informações irão compor a estratégia da empresa na elaboração do plano para evolução do software.

4 Aplicação Prática

O sistema avaliado tem como contexto a Gestão de Companhias Aéreas, consiste em um sistema integrado para gestão de linhas áreas que contempla todos os setores da companhia. O sistema atende ao mercado atual, mas apresenta algumas características que começa a colocá-lo em desvantagem em relação à concorrência, uma vez que o sistema não foi desenvolvido para web, possui um processo de manutenção lento e com muitos efeitos colaterais. Outro ponto relevante e a falta de documentação arquitetural do sistema.

O objetivo da avaliação é gerar informação que possa guiar a evolução do sistema para o lançamento de uma nova versão do software que atenda as necessidades estratégicas das gerencias e dos atuais clientes no sentido de alcançar a continuidade do software no mercado atual e futuro.

4.1 Consolidação dos Resultados

Para este exemplo de aplicação, as decisões arquiteturais resultantes do processo, foram adequadas. A seguir expõem-se os pontos a comentar sobre os resultados obtidos.

A aplicação do método começou com 7 cenários de usuários e 14 cenários de negócios que foram levantados na identificação dos objetivos do negócio junto ao cliente. Os cenários dos usuários foi resultado das demandas dos usuários do sistema que possuíam informações acerca das deficiências do sistema e os cenários de negócio vieram da leitura da visão estratégica do cliente. Na tabela 2 estão resumidos o número de cenários levantados para cada atributos de qualidade.

Tabela 2. Número de requisitos por atributo de qualidade.

Categorias	Atributos de Qualidade	Qtd
Funcionalidade	Adequação	1
	Precisão	1
Confiabilidade	Tolerância a falhas	1
	Recuperabilidade,	1
Usabilidade	Compreensibilidade	1
Eficiência	Desempenho	1
Manutenibilidade	Analisabilidade	1
	Modificabilidade	4
	Testabilidade	1
Portabilidade	Adaptabilidade	2
	Instalabilidade	1
	Co-existência	4

Atributos de Negócio	Time-to-market	2
	Custo e benefício	1

Durante a fase de análise das abordagens existentes para atender os 22 novos requisitos foram detectados que dentre as abordagens arquiteturais existentes atende totalmente somente 1 cenário, atende parcialmente a 11 cenários e não atende a 8 cenários. Esses números comprovam a necessidade da reestruturação da arquitetura para adequar as novas necessidades.

No passo 7 foi realizado um *Brainstorm* e priorização de cenários para levantar dentre o grupo cenários prioritários, mas foi observado durante a reunião que alguns cenários eram semelhantes e poderiam ser agrupados. De posse da árvore de utilidade foi elaborado uma matriz identificada por Mapa de Cenários. O objetivo deste mapa é visualizar a princípio os cenários semelhantes, facilitando um possível agrupamento. Este passo de agrupamento foi incorporado pelos avaliadores para facilitar a resolução de conflitos que poderiam surgir. Com o agrupamento o número de cenários caiu de 22 para 16.

Na priorização os principais cenários de um sistema são classificados em função de sua importância e complexidade, considerando a percepção de negócio e arquitetura. As duas variáveis de priorização Importância e Complexidade foram classificadas em alta (A), média (M) e baixa (B) de acordo com as características dos requisitos. Cinco cenários foram classificados como alta prioridade.

No passo seguinte, análise dos mecanismos arquiteturais foi realizada a análise arquitetural de cada cenário prioritário e então traçadas as táticas para soluções das questões de cada cenário. Também foram identificados: Ambiente, Estímulo, Resposta e Abordagem Arquitetural Existente.

Neste ponto é importante observar que somente acatar as decisões arquiteturais resultantes da avaliação não irá resolver as questões da evolução. Muitas vezes uma decisão arquitetural afeta o atendimento de outro requisito. Na maioria das vezes é necessário fazer escolhas entre as decisões a serem aceitas. Em casos de arquiteturas complexas onde estão envolvidos vários sistemas e subsistemas ou uma linha de produto se faz necessária uma análise mais consistente nas escolhas utilizadas e nos efeitos que elas provocam nos sistemas envolvidos. Dentro deste trabalho foi possível destacar dois pontos de sensibilidades:

- S.1.2. Estratégias de normalização podem degradar a *performance*.
- S.1.1. O aumento do número de componentes pode degradar a *performance*.

Levando em consideração todos os requisitos da árvore de utilidade e analisando as táticas sugeridas, foram identificados 13 *trade-off* dos quais 5 podem ser considerados pontos de sensibilidades pois afetam requisitos que foram considerados com alta prioridade pelos *stakeholders*. Esses conflitos envolveram cenários das mais diversas classificações tais como: Eficiência x Confiabilidade, Manutenibilidade x Confiabilidade, Funcionalidade e Atributo de negócio x Eficiência, Portabilidade x Eficiência, Manutenibilidade x Portabilidade.

A aplicação do método mostrou sua eficiência quando se trata de evolução do software, pois permite nortear a avaliação a partir da arquitetura atual do software e dos cenários que o requisitante deseja alcançar. A aplicação foi bem sucedida na avaliação da arquitetura atual mesmo quando ela não possui documentação e na

questão de traçar estratégias de atuação considerando a opinião e a necessidade de todos os envolvidos (*stakholders*). Em relação às praticas sem método, fica claro a vantagem do uso de um método estruturado que conduza a avaliação. Os resultados são concretos e passiveis de serem utilizados pela equipe do cliente.

4.2 Aspectos do Plano de Ação para Evolução

Os artefatos resultantes do processo de avaliação através do ATAM se mostraram adequados para pautar a estratégia da evolução do software. O plano de evolução terá como base os seguintes itens:

- Conjunto de Requisitos de Qualidade;
- Conjunto de Riscos e não riscos;
- Conjunto de Pontos de sensibilidades e críticos;
- Pontos de Conflitos;
- Plano de Monitoramento de Riscos.

As táticas sugeridas direcionam as ações que devem ser tomadas para evolução do software de acordo com os requisitos colocados como meta. Os *trade-offs* indicam quais são as perdas e ganhos entre táticas conflitantes. E por fim os riscos apontam os fatores e variáveis que devem ser observados, monitorados e controlados durante o processo de evolução. Estas decisões arquiteturais são analisadas com o objetivo de traçar as diretivas para evolução de cada ponto de vista RM-ODP, de acordo com o seu nível de abstração. Desta forma o arquiteto e sua equipe poderão elaborar um planejamento estratégico para o desenvolvimento da nova versão do produto. Estas informações irão compor a estratégia da empresa na elaboração do plano para evolução do software.

Dentro do contexto da aplicação exemplo, as táticas são classificadas de acordo com os pontos de vista RM-ODP. Esta classificação facilitará a ação das diversas competências da empresa, cada uma na sua área específica. No exemplo em questão, essa divisão facilitou a separação das tarefas e monitoramento da execução de cada uma pelos seus responsáveis.

Os responsáveis pelos pontos de vistas devem traçar uma estratégia para cada tática sugerida e acompanhar sua evolução. Nesta aplicação exemplo não surgiram táticas conflitantes, pois os conflitos foram solucionados em uma etapa anterior. Mas ainda é necessária uma fase de *brainstorm* e divulgação das estratégias selecionadas para contemplar as táticas sugeridas. Na Tabela 3 estão listadas as táticas e algumas estratégias sugeridas pela equipe de avaliadores.

Na análise dos cenários, o atributo de qualidade manutenibilidade mostrou ser um requisito prioritário. A principal táticas sugeridas para resolver esta questão é o baixo acoplamento e a alta coesão. O primeiro diz respeito à dependência entre classe e método, quando existem no código elementos muito acoplados, surgirão problemas com manutenção do código. A coesão trata da quantidade de tarefas especificas que são realizadas dentro de uma classe, ou seja, sobre um mesmo conceito. Aplicar este conceito dentro de projetos exige um bom conhecimento do negocio e domínio da tecnologia utilizada. Uma sugestão de estratégia é utilizar design patterns. O padrão Façade ou Facade ajuda a resolver parte deste problema, existem ainda, alguns

frameworks não intrusos como Spring que auxiliam na redução do acoplamento entre as classes.

Tabela 3. Táticas Sugeridas classificadas de acordo com os pontos de vistas ODP.

PV	Tática Arquitetural	Estratégia
Empresa	-Utilização do processo certificado. -Treinamento da equipe . -Manter checklist de teste dos componentes utilizados ou novos. -Executar teste e homologação da nova funcionalidade. -Realizar controle de versão.	-Deve ser elaborado um cronograma que considere como tempo máximo o período determinado pela avaliação que diz que o lançamento de uma nova versão não deve ultrapassar 3 anos. - A equipe da empresa é responsável por analisar cada uma das táticas sugeridas e criar um cronograma de ação de acordo com a ordem de prioridade das atividades, observando as prioridades e relacionamentos do tipo dependência entre as táticas. - São responsáveis pelas ações de gerenciamento acompanhando o cumprimento das decisões tomadas
Informação	-Realizar e manter a documentação do sistema. -Modelagem da base de dados flexível através de normalização.	- Os envolvidos devem manter a documentação do sistema atualizada - O administrador de banco de dados deve utilizar técinicas de *tuning,* técnicas de modelagem de dados para otimizar o tempo de resposta
Computação	- Padronização de interface através de *template*. -Implementar telas limpas e simplificadas. -Criar componentes/classes para desenvolver interface padrão. -Carregar apenas informações necessárias, e conforme forem sendo solicitadas.	- Definição de interface padrão. - Uso de padrões que otimizem tempo de resposta do produto final
Engenharia	-Manter Baixo acoplamento. -Manter Alta coesão do software. -Controle e tratamento de falhas (banco, código, entrada de dados).	- Utilizar o padrão Façade ou Facade para resolver parte deste problema, existe ainda, alguns frameworks não intrusos como Spring que auxiliam na redução do acomplamento entre as classes.
Tecnologia	-Iniciar pesquisa por linguagens mais adequadas e gerador de código. -Utilizar tecnologias robustas como Java. -Utilizar SQL padrão. -Utilizar paradigma OO. - Programação no banco de dados para agilizar o tempo de resposta.	- Definir framework padrão que atenda às táticas sugeridas

5 Conclusão.

O roteiro apresentado mostrou-se capaz de expandir as capacidades da arquitetura resultante, diferente dos procedimentos tradicionais centrado somente em ajustes funcionais A utilização do roteiro definido nesta pesquisa permitiu a análise dos pontos fracos da arquitetura do sistema e o estudo de uma abordagem.

Para reduzir os riscos associados com a evolução de software, o método de avaliação baseada em cenários pode ser usado para melhorar a arquitetura de sistema durante todo o ciclo de vida do sistema. A vantagem de usá-lo no início das atividades de reestruturação do sistema ou até no início do desenvolvimento é a possibilidade de descobrir problemas de projeto numa fase que ainda é possível tomar as decisões adequadas.

A aplicação do exemplo deixou algumas lições importantes no uso do método ATAM como base para avaliação de arquiteturas, como: Melhorias de Documentação, Coleta de Cenário e foco da avaliação, Problema descobertos durante todo o ciclo de vida de software, Melhorias para a família de produtos de software.

O processo de avaliação descrita não é específico para famílias de produtos de software e pode ser aplicada a qualquer arquitetura de software. No entanto considerando o contexto de uma arquitetura de software de uma família de produtos, a avaliação considerar a evolução para o conjunto dos produtos da família. Nas avaliações com base no cenário de cada etapa do processo são afetados por esta dimensão.

Referências

1. Svahnberg. M.; Supporting software architecture evolution. 2003. Tese (doutorado). 2003 Blekinge Institute of Technology - Suécia, 2003.
2. Clements, P; Kazman R; Klein, M.: Evaluating software architecture: Methods and case studies. SEI, 8 edição. (2009)
3. Graaf. B.: Model-Driven Evolution of Software Architectures,"Software Maintenance and Reengineering, European Conference on. v. 0; p. 357-360. Los Alamitos, CA, USA (2007)
4. Sadou, N.: Tamzalit D.; Oussalah M. A unified Approach for Software Architecture Evolution at different abstraction levels, In: Proceedings of the 2005 Eighth International Workshop on Principles of Software Evolution, p. 65-70, IEEE Computer Society Washington, DC, USA. (2005)
5. Chávez, M.: Um Processo para o controle da evolução da Arquitetura de Software Baseado em ODP. 2009. Dissertação (Mestrado) - Universidade de São Paulo. São Paulo. (2009)
6. Kruchten , P.; Obbink, H.; Stanford, J.: The past, present, and future for software architecture. Software, IEEE, v. 23, n.2, p. 22-30. (2006)
7. Garlan. D.; Perry. D. Introduction to the Special Issue on Software Architecture," IEEE Transactions on Software Engineering, vol. 21, no. 4, p. 269-274, Abril. (1995)
8. GODFREY, M.W.; GERMAN, D.M.; The past, present, and future of software evolution, In: Frontiers of Software Maintenance. p. 129-138; Beijing. Setembro 2008
9. Tu. Q, Godfrey. M. W.: "An Integrated Approach for Studying Architectural Evolution", In: 10th International Workshop on Program Comprehension, p.127, Paris, France. (2002)
10. Becerra, J.: Aplicabilidade do padrão de processamento distribuído e aberto nos projetos de sistemas de automação. Tese (Doutorado), Universidade de São Paulo, São Paulo. (1998)