MAC5715 - Tópicos Avançados em POO
Professor: Fábio Kon

Concern Organizer
(Versão refatorada do padrão VisaoDeArquiteto)

Michelet del Carpio Chávez

1 Anedota

Rui (Gerente de Projeto): E ae Rafael, tenho dois grupos de desenvolvedores que já decoraram os casos de uso e estão prontos para programar. Cadê os diagramas UML para eles implementarem?
Rafael (Arquiteto): Pois é Rafael, eu acho que devem começar pelo banco de dados, mas nosso administrador de banco de dados está aprendendo Postgresql. Você sabe que ele só manja Oracle...
Rui: Então vou mandar que façam as telas de IHM... Isso não vai mudar, né?
Rafael: Pois é... mas o cliente não sabe se será uma aplicação para web ou standalone ,o departamento de TI deles está avaliando ambas tecnologias. Eu sugeri uma aplicação web, mas ele está com um pé atrás porque diz que é insegura.
Rui: Então o que eu faço com esse pessoal, não vou pagar salário de graça não... Vou tentar alocar neles noutro projeto.
(dias mais tarde...)
Carlos (Gerente de comercial): Rafael, o sistema vai ser web mesmo. Os usuários vão entrar numa página web e rodarão o aplicativo desde lá. Aqui o relatório do pessoal de IT.
Rafael: Ótimo. Então vamos usar JSP com Applets e tudo isso no Linux.
Carlos: Muito bom cara. O cliente vai ficar contente com o novo orçamento.
Rafael: Uai, lógico! E como o pessoal já sabe Java, fica mais fácil pra gente. Pode falar pro Rui que na próxima semana tem seu pessoal programando.
(meses depois...)
Desenvolvedor 1: Ce viu cara, o pessoal do projeto X se lascou e demitiram um gerente!
Desenvolvedor 2: Ué, que aconteceu?
Desenvolvedor 1: Eles fizeram um sistema Java no Linux, quando o cliente era parceiro da Microsoft...
Desenvolvedor 2: E o arquiteto, ele não sabia de nada?
Desenvolvedor 1: não, ELE NÃO SABIA!

2 Objetivo

Organizar e avaliar *concerns*(1) de um sistema de software.

3 Motivação

Existe um conjunto de atividades, executadas entre a captura de requisitos e a fase de projeto(2) no qual os *stakeholders* (3) são ouvidos para avaliar sua importância no projeto, seus anseios e interesses, e o grau de influência destes no sistema(4). Na maioria das vezes, este processo é nebuloso, não-racional e orientado a resolver um problema específico. Sendo assim, o arquiteto de software, ou o encarregado de cumprir este papel, defronta-se com conjunto de *concerns* que nem sempre estão relacionados e muitas vezes são antagônicos, inclusive, com as funcionalidades do sistema em si. Por isso, é importante para o arquiteto de software a existência de um mecanismo que ajude à organização e ao balanceamento dos *concerns.*

4 Contexto

A arquitetura de software determina a estrutura real do sistema, assim como as interações entre as subunidades envolvidas em diferentes níveis de granularidade [1], ela é o resultado de um conjunto de decisões técnicas e de negócio no qual estão envolvidos todos os *stakeholders*.

Para tomar estas decisões, o arquiteto possui, além da sua experiência, um conjunto de estilos arquiteturais e padrões de projeto que, além de serem guias valiosos, facilitam a compreensão e descrição do sistema [1].

5 Problema

Como representar os *concerns* de um sistema na descrição de software e lhes atribuir importância para tomar decisões de arquitetura?

6 Forças

Existem varias forças que agem na aplicação deste padrão. Estas estão relacionadas com os *stakeholders.* A continuação apresentamos alguns deles:

- O cliente quer baixo custo e o software entregue à tempo.
- O usuário final deseja bom funcionamento, desempenho, segurança, disponibilidade e usabilidade.
- O cliente não tem tempo para reuniões a respeito do desenvolvimento do software.
- O gerente do projeto quer um baixo custo e precisa que os desenvolvedores sejam utilizados.
- O pessoal de marketing quer o produto o mais rápido possível nas prateleiras e que o produto seja competitivo com os concorrentes.
- O arquiteto precisa conhecer todos os fatores que influenciam o software.
- E custoso para a empresa treinar seus programadores numa nova linguagem de programação e/ou metodologia de desenvolvimento.

De forma paralela a estas forças existem também as restrições, as quais estão associadas a:

- Capacidade da organização de desenvolver o software
- Capacidade do cliente de transmitir o que deseja
- Experiência do arquiteto em projetos de software

7 Solução

7.1 Estrutura

O padrão ConcernOrganizer está contextualizado no diagrama de classes de análise da Figura 1. A Figura 2 apresenta o detalhe das classes que fazem parte do padrão. Seus principais elementos são descritos a seguir:

Ambiente: É o lugar onde o sistema habita no transcorrer do seu ciclo de vida. Este pode ser, por exemplo: a empresa desenvolvedora do software, o ambiente de implantação do software.

Arquiteto: A pessoa, grupo, ou organização responsável pela arquitetura do sistema.

Arquitetura: A organização fundamental do sistema, representando seus componentes, as relações entre eles e com o ambiente e os princípios que guiam seu projeto e evolução

Concern: Interesse, preocupação ou assunto sobre o sistema que afete ao *stakeholder*.

Descrição de Arquitetura: Um conjunto de produtos para documentar a arquitetura. Estes documentos podem ser representados como: diagramas UML, desenhos, documentos de texto, diagramas de processos, etc.

Sistema: Um conjunto de componentes organizados para cumprir uma função específica ou um conjunto de funções.

Stakeholder: Um indivíduo, equipe, ou organização com interesses e *concerns* relativos ao sistema. Os *stakeholders* podem ser clientes, desenvolvedores, arquitetos, etc.

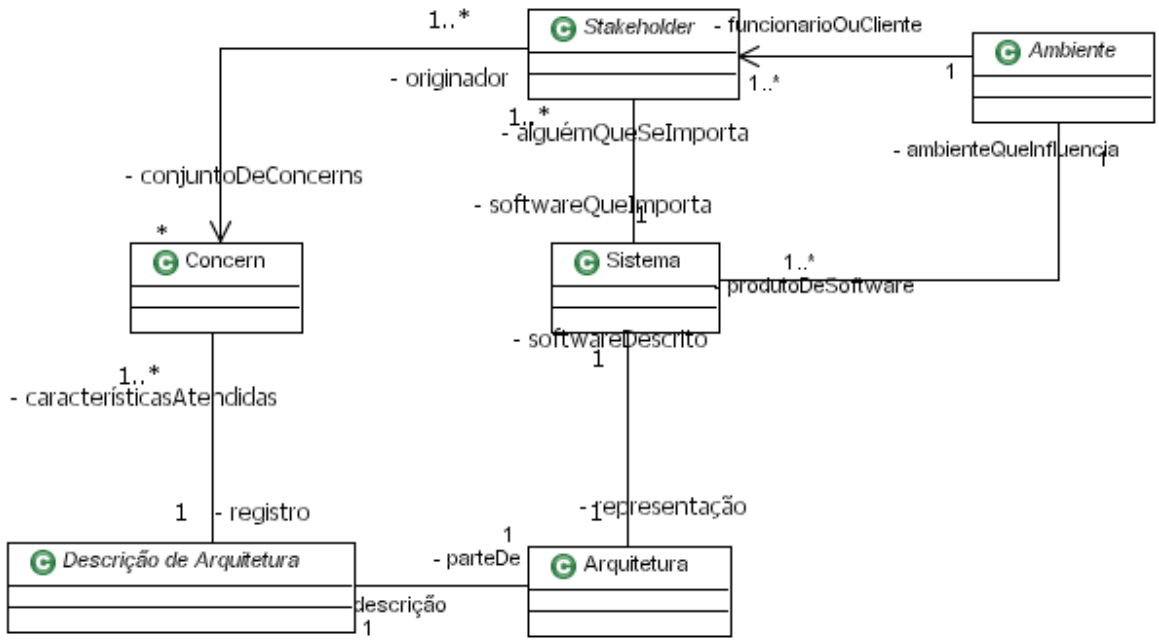

Figura 1Diagrama de classes UML[2] que apresenta o contexto do padrão ConcernOrganizer

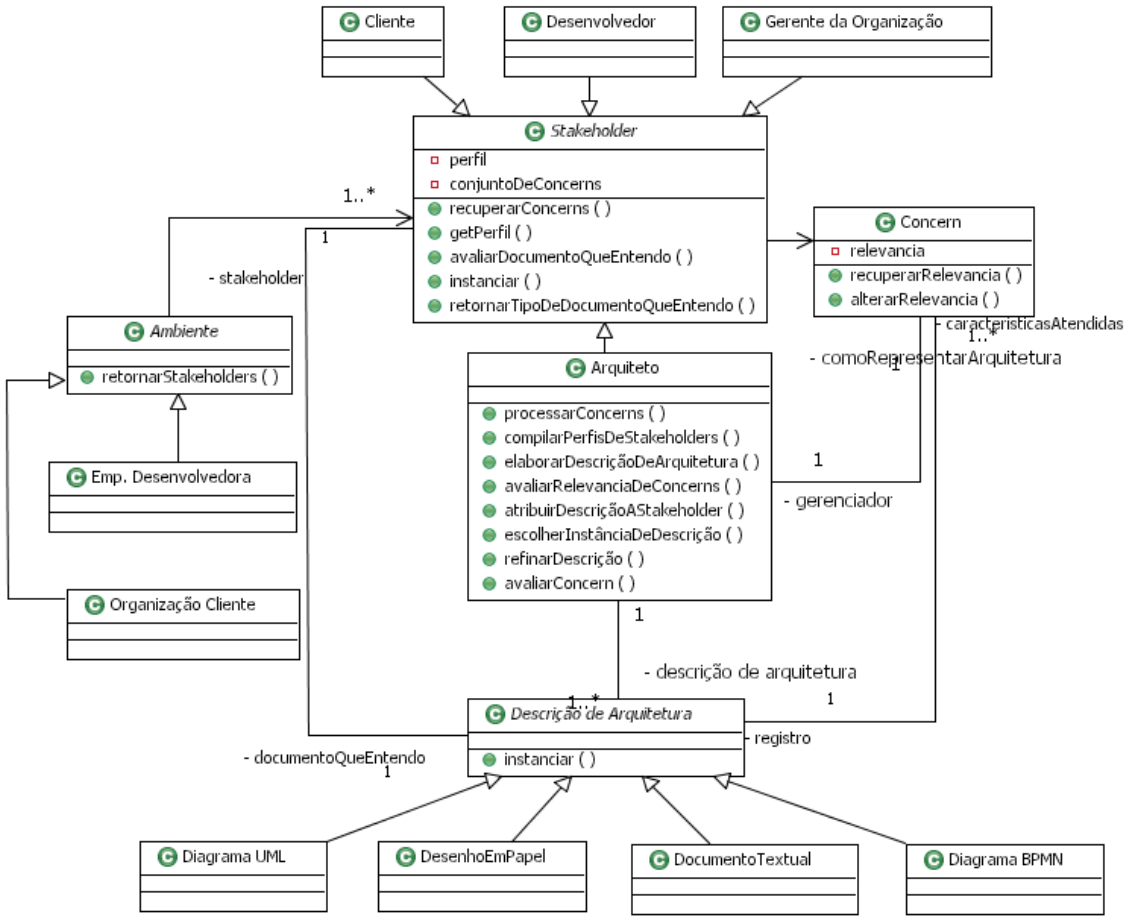

Figura 2 Detalhe das classes do padrão ConcerOrganizer

7.2 Dinâmica

O funcionamento do padrão está descrito através de 4 iterações:

7.2.1 Iteração 1: Compilar perfis dos *Stakeholders*

Nesta iteração, um perfil de *stakeholder* é compilado para cada *stakeholder* que é relevante para a descrição da arquitetura. Um perfil é um conjunto de atributos textuais que tem informação sobre: o *stakeholder*, seus objetivos, suas tarefas e seus *concerns*[3]
O objetivo desta atividade é tornar explícita a posição do *stakeholder* e permitir analisar suas necessidades de informação.
A Figura 3 apresenta o diagrama de seqüência associado a esta iteração.

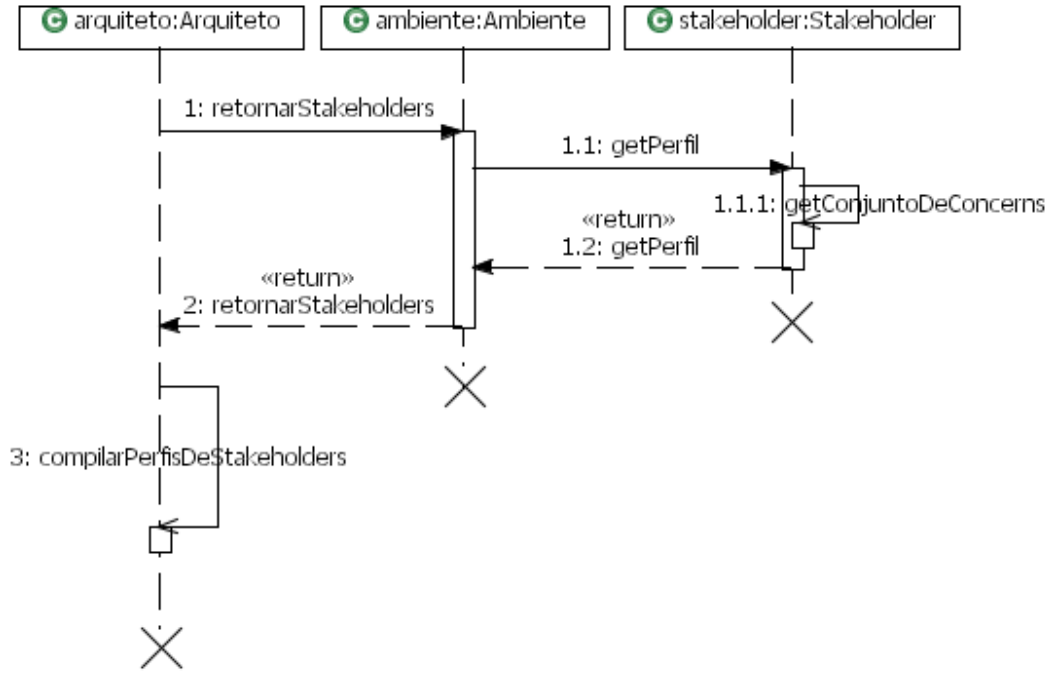

Figura 3 Iteração 1 do padrão: Compilar Perfis dos *Stakeholders*

7.2.2 Iteração 2: Avaliar *concerns*

Nesta iteração, uma vez compilados os perfis dos *stakeholders*, procede-se à avaliação dos *concerns* relacionados a cada *stakeholder*. Primeiro tenta-se compreender o *concern* de acordo com a ordem de relevância atribuída pelo próprio *stakeholder* após esta avaliação, o arquiteto altera a relevância do *concern* utilizando sua própria visão do sistema e avaliando o papel do *stakeholder* dentro do processo de definição de arquitetura. Este processo é feito da seguinte maneira:

```
Para cada stakeholder
     Recuperar Concerns
     Para cada Concern
          Recuperar Relevância
          Avaliar concern
          Alterar relevância
```

A descrição através de um diagrama de seqüência é apresentado na Figura 4.

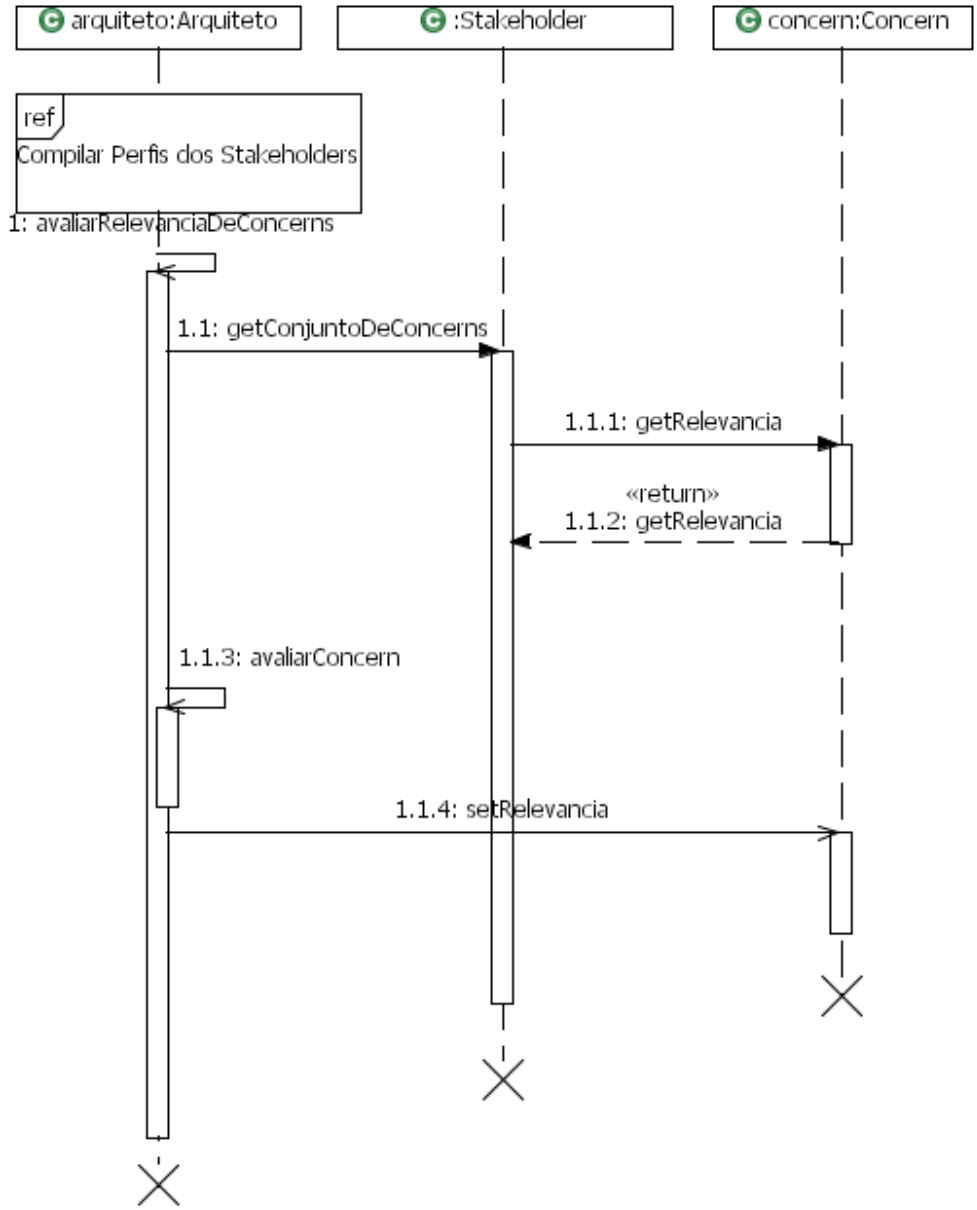

Figura 4 Iteração 2 do Padrão: Avaliar Concerns

7.2.3 Iteração 3: Elaborar descrição adequada

O objetivo desta iteração é relacionar as informações arquiteturais e os *concerns* dos *stakeholders* através de um documento de descrição de arquitetura.

Esta iteração é muito importante porque seu resultado estará refletido na compreensão do sistema por parte dos *stakeholders.*

A primeira parte da iteração diz respeito à selecionar os tipos de descrição de arquitetura mais adequados a cada *stakeholder*. Pois, por exemplo, o *concern* segurança pode ser diferente para um desenvolvedor que para o usuário final, e sua descrição também pode ser diferente. Para isso é preciso selecionar dentre os tipos de documentos de definição de

arquitetura que o *stakeholder* é capaz de compreender, o mais adequado para a representação do *concern.* Uma vez realizada esta tarefa, procede-se à elaboração do documento de descrição de arquitetura em si.

A Figura 5 apresenta o diagrama de seqüência associado a esta iteração:

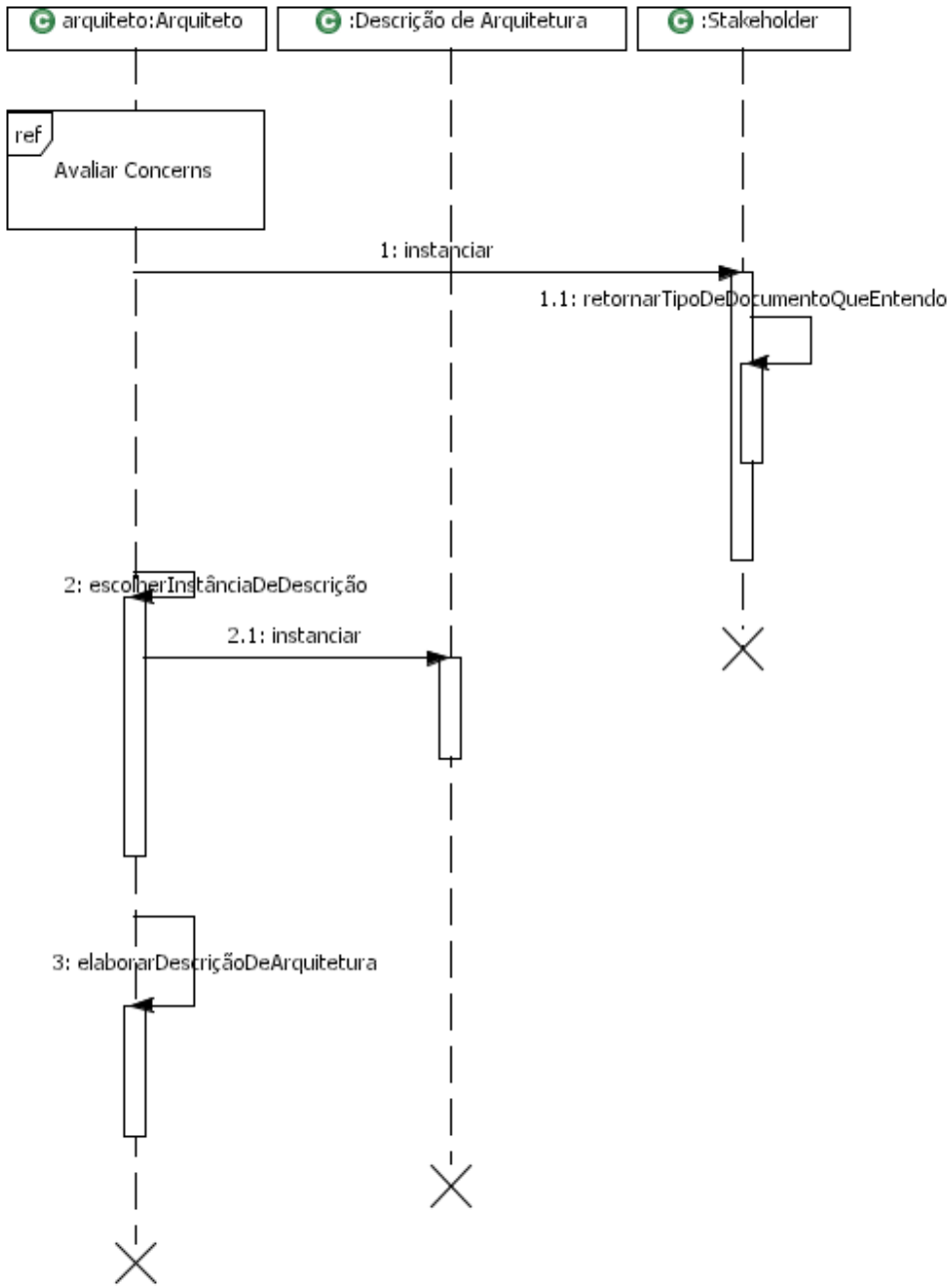

Figura 5 Iteração 3 do padrão: Elaborar descrição Adequada

7.2.4 Iteração 4: Refinar descrição

Esta iteração representa o retorno do usuário relacionado à compreensão e à representação dos seus *concerns* no documento de arquitetura. Nesta iteração é solicitada uma avaliação do documento por parte do s*takeholder* para proceder a um refinamento do documento. Quando o *stakeholder* se mostra satisfeito, a iteração pára.

A Figura 6 apresenta o diagrama de seqüência desta iteração.

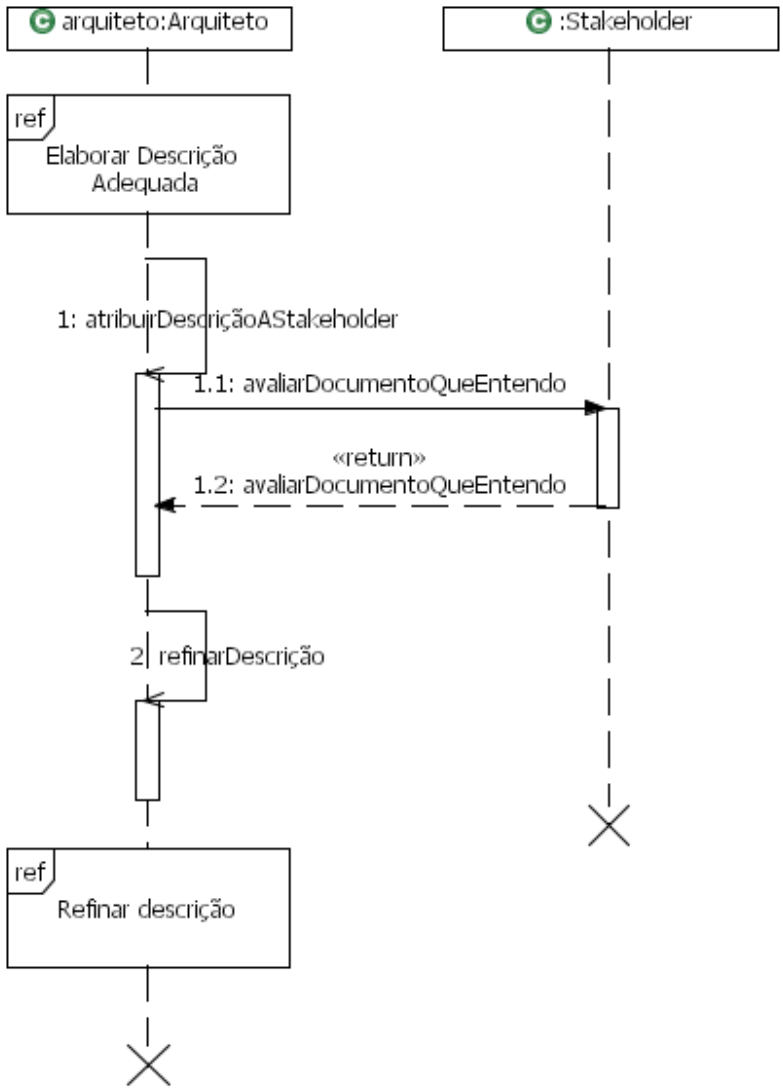

Figura 6 Iteração 4 do padrão: Refinar descrição

8 Exemplos de utilização

Este padrão pode ser utilizado em ambientes de desenvolvimento onde os *stakeholders* precisem compreender a arquitetura do sistema.

9 Conseqüências

+ Compreensão detalhada da arquitetura por parte dos *stakeholders.*
+ O arquiteto ganha conhecimento dos *stakeholders* e seus *concerns.*
+ A organização que desenvolve o software armazena a experiência do desenvolvimento do software.
+ Cada grupo de *stakeholders* tem sua própria visão do sistema (representado pelo documento de descrição de arquitetura).
+ Facilita a gerência de projeto, simplificando a organizando a atribuição de responsabilidades.
- As atividades necessárias para utilização do padrão podem ser dispendiosas e ocupar o tempo do arquiteto como de todos os *stakeholders.*
- Devido à estratificação dos *concerns,* que são apresentados em diferentes documentos de descrição de arquitetura, perde-se a visão holística do sistema de software e o gerenciamento desses documentos pode tornar-se dispendioso.

10Usos conhecidos

10.1 Processo Unificado

O processo unificado é um processo iterativo-incremental [4] o qual está focado fortemente na arquitetura. O uso deste padrão aparece na fase de elaboração, abrangendo as disciplinas de modelagem de negócios e análise.

10.2 Component off-the-shelf (COTS)

Cada vez mais, o desenvolvimento de software se torna um processo de identificar e selecionar pacotes de software existente. A escolha de pacotes requer a identificação de requisitos e *concerns* desejáveis (usualmente expressados como características ou serviços) e sua relação com o que está disponível no mercado[5]. Desta forma, o ConcernOrganizer, é o mecanismo pelo qual a empresa desenvolvedora melhora sua capacidade de adquirir COTS dado que organiza seus requisitos e *concerns* de forma a identificar suas necessidades e as do seu projeto.
De forma similar, uma empresa desenvolvedora de COTS direciona o seu desenvolvimento a um conjunto de funcionalidades e *concerns* que são identificados, organizados e agrupados para desenvolver um COT.

10.3 Ensino de Engenharia de Software

Recentes artigos [6][7] apresentam a problemática relacionada ao ensino de engenharia de software. Esses relatos mostram uma série de características relacionadas ao uso do ConcernOrganizer, como separação de interesses, relacionamento com o cliente e o uso adequado da arquitetura de software como ferramenta de engenharia.

11Glossário

Este glossário especifica as acepções utilizadas para certos termos que possuem vários significados ou que não têm tradução direta no português. A seguir apresentam-se as locuções, na ordem de aparição no texto, e as acepções empregadas pelo autor.

1 Concern:

1. A matter that relates to or affects one. See Synonyms at affair.
2. Regard for or interest in someone or something.

Fonte: *The American Heritage® Dictionary of the English Language, Fourth Edition Copyright © 2000 by Houghton Mifflin Company. Published by Houghton Mifflin Company. All rights reserved.*

2 Design phase.

1. The period of time in the software life cycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements. See also: detailed design; preliminary design.

Fonte: IEEE Standard Glossary of Software Engineering Terminology IEEE Standard Glossary of Software Engineering Terminology IEEE Std 610.121990

3 System

1. A collection of components organized to accomplish a specific function or set of functions.

Fonte: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems IEEE Std 1471-2000*

4 Stakeholder

1. One who has a share or an interest, as in an enterprise.

Fonte: *The American Heritage® Dictionary of the English Language, Fourth Edition Copyright © 2000 by Houghton Mifflin Company. Published by Houghton Mifflin Company. All rights reserved.*

2. An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.

Fonte: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems IEEE Std 1471-2000*

12Bibliografia

[1] Shaw, M., Garlan, D. : Software Architecture: perspectives on an emerging discipline. Prentice Hall (1996)

[2] Object Management Group; UML 2.0 Superstructure Specification, 2004

[3] H. Koning and H. van Vliet. A Method for Defining IEEE Std 1471 Viewpoints, The Journal of Systems & Software, 2006 - Elsevier

[4] I. Jacobson, G. Booch, J. Rumbaugh. The Unified Process. Addison-Wesley, Reading, Mass., 2001.

[5] Bashar Nuseibeh. Weaving the software development process between requirements and architecture. In Proceedings of ICSE-2001 International Workshop: From Software Requirements to Architectures (STRAW-01) Toronto, Canada,2001.

[6] P. Lago and H. van Vliet, "Teaching a Course on Software Architecture," Proc. 18th Conf. Software Eng. Education and Training, IEEE CS Press, 2005, pp. 35–42.

[7] Van Vliet, H. 2006. Reflections on Software Engineering Education. *IEEE Softw.* 23, 3 (May. 2006), 55-61. DOI= http://dx.doi.org/10.1109/MS.2006.80