Integração de Características Transversais Durante a Modelagem de Requisitos

Lyrene Fernandes da Silva[1,2], Julio Cesar Sampaio do Prado Leite[1]

[1]Departamento de Informática - PUC-Rio - Rua Marquês de São Vicente 225, Gávea - 22453-900 - Rio de Janeiro-RJ, Brasil

{lyrene,julio}@inf.puc-rio.br

Abstract. *The principles of crosscutting concern separation and composition have been used by the Aspect-Oriented Development Community in order to solve the tangling and scattering problems. In this work we present a method for integrating crosscutting concerns in the requirements engineering process. This approach uses goal models and the concepts defined in aspect-oriented languages to provide separation, composition and visualization of crosscutting concerns in order to facilitate the modeling and tracing between them.*

Keywords. *Requirement modeling, crosscutting concerns, early-aspects, goal models.*

Resumo. *O princípio de separação e composição de características transversais[3] vem sendo usado pela comunidade de Desenvolvimento Orientado a Aspectos para tratar os problemas de espalhamento e entrelaçamento, no nível de implementação. Neste trabalho apresentamos uma proposta de integração de características transversais na fase de definição de requisitos. Utilizamos modelos de metas e os conceitos do desenvolvimento orientado a aspectos para prover a separação, composição e visualização de características transversais, facilitando a modelagem e rastreabilidade delas.*

Palavras-chave. *Modelagem de requisitos, características transversais, early-aspects, modelos de metas.*

1. Introdução

Para prover maior facilidade de compreensão, manutenibilidade e reusabilidade pesquisadores em engenharia de software têm investido na separação de *concerns* através do uso de diferentes abstrações e modelos, realçando diferentes visões do sistema [Tarr 99]. Não há uma clara definição para *concerns*. Algumas definições possíveis são: "é uma propriedade observável desejada", "é uma *feature*", "é alguma responsabilidade", "é um subproblema" [Tekinerdoðan 04]. Neste trabalho chamamos

[2] Este trabalho foi realizado com o apoio do CNPq.

[3] Do inglês crosscutting concerns. Em [Piveta 04] há sugestões de tradução para os termos utilizados em orientação a aspectos. Entretanto, ainda não existe um consenso a este respeito. Neste trabalho adotamos o termo características transversais como tradução para crosscutting concerns.

concerns de características; elas podem ser qualquer característica do sistema ou do domínio que seja interessante analisar quer isoladamente quer em conjunto com outras.

O princípio de separação é importante para diminuir a complexidade em certos momentos, permitindo o entendimento e análise de uma única característica por vez. Entretanto, um sistema é naturalmente composto de diferentes características, e entendê-las em conjunto é igualmente importante. Se por um lado, precisamos separar as características, por outro precisamos compô-las. Muitas vezes estas características são fortemente relacionadas, entrelaçadas e/ou sobrepostas, influenciando ou restringido umas às outras, sendo chamadas de características transversais [Tekinerdoðan 04]. Isto dificulta a separação e análise das partes do sistema bem como a análise do impacto que umas exercem sobre as outras.

Para facilitar a separação e composição de características transversais, nos últimos anos houve um investimento no paradigma de orientação a aspectos [Kiczales 97]. Este novo paradigma de programação tem por objetivo tornar mais fácil a manutenibilidade e evolução de software. Entretanto, trata de aspectos ou características transversais de baixo nível de abstração quando os requisitos já foram congelados e muitas decisões de gerência e desenho já foram tomadas.

Com o intuito de possibilitar o tratamento de aspectos em todas as etapas do desenvolvimento, diminuindo o *gap* entre os diferentes paradigmas e promovendo o desenvolvimento orientado a aspectos, têm surgido linguagens e métodos de modelagem considerando estes novos elementos e relacionamentos de desenho de maneira a permitir a separação e composição de características transversais no nível de desenho [Chavez 04]. Esta abordagem tem se estendido também, à fase de definição de requisitos, denominada por alguns de *early-aspects* [Rashid 02]. As abordagens em *early-aspects* têm definido métodos para identificar, modelar e mapear candidatos a aspectos para aspectos em níveis mais baixos de abstração [Rashid 03][Baniassad 04][Brito 04].

Nosso trabalho está relacionado a estes por darmos foco à modelagem de características transversais. Para isto, propomos um método de integração de características transversais fundamentado na idéia implementada em linguagens de programação orientadas a aspectos. Assim como nestas linguagens, nosso método de integração aborda a "separação" e "composição". A separação é realizada pela modelagem de características através de um modelo de metas [Mylopoulos 92]. A composição é realizada por um mecanismo (*weaver*) que gera um modelo único do sistema sendo definido, facilitando a inclusão e exclusão de características deste modelo. Além disto, o método aborda também a extração de diferentes visões dos modelos separados e compostos. Nesta etapa, isto é essencial, visto que esta é a etapa de elaboração da solução, então ora é importante visualizar as características juntas, ora separadas.

Nossa premissa é que facilitar a modelagem de características transversais, nesta fase, pode ajudar não só nas tarefas de priorizar, validar e gerenciar requisitos, mas também prover reuso da especificação e conhecimento para desenhar a arquitetura do sistema e manter a rastreabilidade entre características transversais e entre as fases de definição de requisitos e arquitetura.

As seções seguintes deste artigo estão organizadas da seguinte maneira: na Seção 2 resumimos os principais conceitos que fundamentam este trabalho; na Seção 3 apresentamos o método de separação, composição e visualização de características transversais e os benefícios que esta abordagem traz para a modelagem de requisitos através de um exemplo ilustrativo; na Seção 4 apresentamos os principais trabalhos que estão relacionados a este; por fim, na Seção 5 relatamos nossas conclusões e futuros trabalhos.

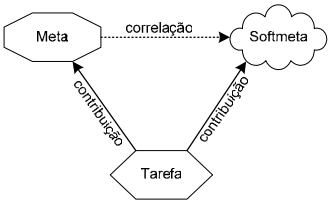
2. Estado da Arte

Nossa abordagem se aplica à modelagem de requisitos. Para isto utilizamos modelos de metas e alguns conceitos utilizados em linguagens de programação orientadas por aspectos. Estes modelos e conceitos são apresentados a seguir nas seções 2.1 e 2.2.

2.1. Modelos de metas

Modelos de metas representam requisitos funcionais ou não funcionais através da decomposição de metas [Giorgini 02][Yu 04]. Esta decomposição indica como satisfazer uma determinada meta e por outro lado indica a razão pela qual as submetas são necessárias. Na literatura há algumas variações de modelos de metas [Yu 04] [Lamsweerde 01]. Em geral, eles usam árvores *and/or* para representar a decomposição de metas e definir um espaço de soluções alternativas para satisfazer a meta raiz. Em [Mylopoulos 92], *softmetas* são propostas como meio para modelar e analisar requisitos não funcionais.

Há alguns trabalhos que propõem técnicas de análise de metas, por exemplo: análise de obstáculos, explora os possíveis obstáculos para a satisfação de uma meta [Lamsweerde 00]; análise qualitativa de metas, permite atribuir valores qualitativos aos relacionamentos entre metas e ajuda a formalizar e pensar sobre estes relacionamentos [Giorgini 02]; propagação de rótulo, propaga os rótulos em uma árvore de metas de maneira a tornar visível conflitos de interesses e objetivos [Giorgini 02]; análise de variabilidade [Gonzáles 04], possibilita a análise de soluções alternativas.



Tipo de elo	and	or	make	help	unknow	hurt	break
Contribuicao	S	S	S	S	S	S	S
Correlação	N	N	S	S	S	S	S

Contribuições e correlações

Figura 1. O modelo de metas V-graph [Leite 04]

Para os nossos propósitos, estamos interessados no *V-graph*, um tipo de modelo de metas [Yu 04]. Este modelo é composto de 3 elementos: *softmetas*, metas e tarefas. Os possíveis relacionamentos entre eles são elos de contribuição ou de correlação, contendo um dos seguintes rótulos: *and*, *or*, *make, help, unknow, hurt* e *break* (veja Figura 1). Este modelo permite a descrição de nós intencionais (metas e *softmetas*) e nós operacionais (tarefas). Cada elemento do *V-graph* é composto de duas partes: um tipo e um tópico. O tipo descreve uma função genérica ou um requisito não funcional

genérico. O tópico captura a informação contextual para o elemento. Esta informação contextual é similar ao *subject* definido em [Zave 97].

Nós escolhemos este modelo porque com ele temos três níveis de abstração, *softmetas*, metas e tarefas. Estes diferentes níveis nos permitem pensar em três níveis de variabilidade e configurabilidade, nos indicam características diferentes, nos permitem separar características de maneiras diferentes e enriquecem os relacionamentos transversais entre diferentes árvores de metas. Além disto, o tipo e tópico dos elementos nos permitem ter uma visão de dados e de funções no mesmo modelo, sem tornar a visibilidade afetada, sendo outra maneira de ver as características.

Em [Yu 04] define-se um algoritmo para identificar meta-aspectos em modelos V-graph. Este algoritmo é baseado nos relacionamentos existentes entre tarefas. Tarefas com alto fan-in são candidatas a aspectos. Em [Leite 04], os autores propõem um processo para viabilizar o reuso de requisitos não funcionais. Esta abordagem define uma linguagem baseada no framework 5W2H [Leite 04] para armazenar e recuperar requisitos da biblioteca. Além disto, eles sugerem que um mecanismo de composição poderia ser utilizado para juntar um modelo de uma aplicação ao requisito não funcional recuperado da biblioteca.

O trabalho [Leite 04] motiva nossa idéia de definir um mecanismo automatizado de separação e composição de características transversais. Com o modelo de metas nós podemos utilizar as técnicas de análise de metas para derivar novas visões do modelo de metas separado ou integrado. Desta forma, adicionamos os benefícios dos modelos de metas às técnicas de separação e composição de características transversais e vice-versa.

2.2. Programação orientada a aspectos

A programação orientada a aspectos provê a separação e composição de características transversais no nível de implementação [Kiczales 97]. No caso da linguagem AspectJ [Kiczales 01], a separação é realizada através da inserção de uma nova abstração denominada aspecto à programação orientada a objetos. A composição é realizada através de um combinador denominado *weaver*, que é responsável por pré-processar o código orientado a objetos, inserindo ou modificando os objetos com o comportamento dos aspectos.

Além dos métodos e atributos, um aspecto é composto de *pointcuts*, *advices* e *inter-type declarations*. Eles podem alterar a estrutura estática ou dinâmica de um programa. A estrutura estática é alterada adicionando, por meio das *inter-types declarations*, membros (atributos, métodos ou construtores) a uma classe, modificando assim a hierarquia do sistema. Já a alteração na estrutura dinâmica de um programa ocorre em tempo de execução por meio dos *joinpoints*, que são selecionados por *pointcuts*, e através da adição de comportamentos (*advices*) antes ou depois dos *joinpoints*.

Um *pointcut* indica os pontos onde um determinado comportamento será inserido. Ele é descrito por um nome e um corpo. O corpo indica os *joinpoints* onde serão aplicados os *advices* associados, podendo haver vários *joinpoints* aninhados através dos operadores *or*, *and* e *not*. Cada *pointcut* está associado a um ou mais *advices*. *Advices* descrevem o comportamento (trecho de código) a ser inserido nos *joinpoints*. Existem três formas de *advice*, são elas: *before*, *around* e *after*. Como seus

nomes sugerem, *before* executa antes do *joinpoint*, *after* executa depois e *around* executa antes e depois, alternativamente, substituindo o comportamento do *joinpoint*.

Nossa abordagem utiliza os conceitos de *pointcut*, *advice* e *intertype declaration* na modelagem de requisitos para representar como características transversais afetam umas as outras. Para os nossos propósitos, criamos um novo tipo de relacionamento denominado "relacionamento transversal" ao invés de um novo elemento, como definido em AspectJ. Desta forma, nossa abordagem é não intrusiva, tal como na linguagem HyperJ [Tarr 00]. O relacionamento transversal contém as informações de *pointcuts*, *advices* e *inter-types declarations* e representam relacionamentos de rastreabilidade entre os diferentes modelos de metas. Na Seção 3 detalhamos nossa abordagem e apresentamos um pequeno exemplo.

3. Um Método de Integração de Características Transversais

Nossa proposta é um método para integrar características transversais na fase de definição de requisitos, utilizando modelos de metas e os conceitos definidos pelo desenvolvimento orientado a aspectos. Denominamos o núcleo deste método de "composição". A composição é um processo automatizado que a partir de um conjunto de modelos de metas e informações de como combiná-los gera um modelo integrado do sistema contemplando um conjunto de visões que pode ser derivado dele.

Este método de integração é composto de duas outras atividades: a separação que provê técnicas e linguagem para representação das características de um sistema; e visualização, que provê regras para recuperar as diferentes visões do modelo integrado. Na Figura 2, ilustramos as principais atividades do método de integração de características transversais.

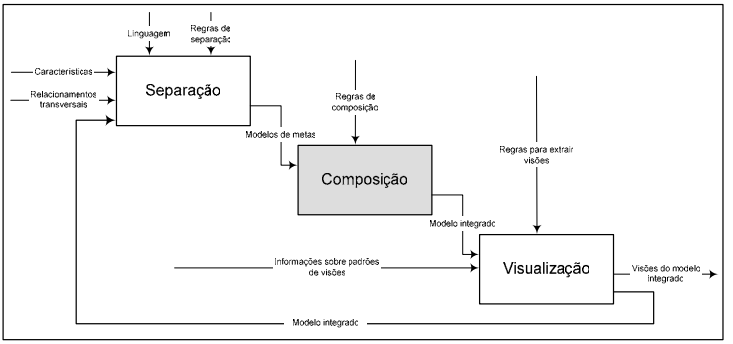


Figura 2. Método de integração de características transversais

A separação de características tem o objetivo de fornecer apoio à modelagem de requisitos através de modelos de metas. Cada modelo de metas é utilizado para retratar um grupo de características do sistema tais como, as funcionalidades específicas da aplicação, requisitos de segurança, tratamento de exceções, dentre outros. Para possibilitar a modelagem do relacionamento entre os diferentes modelos de metas nós definimos um novo tipo de relacionamento que registra como unir estes modelos, denominado relacionamento transversal [Silva 05]. Tanto requisitos funcionais quanto

não-funcionais são modelados como modelos de metas, podendo ser considerados como características transversais se estiverem relacionados a outros de maneira a modificá-los ou restringi-los.

Representando características transversais e não-transversais da mesma forma, através de modelos de metas, facilitamos o reuso de modelos de metas em diferentes sistemas, levando em consideração que a mesma característica pode ser vista como transversal em um sistema, mas não-transversal em outro. Outra vantagem é que não alteramos a essência do modelo de requisitos escolhido, apenas acrescentamos um novo tipo de relacionamento a ser interpretado pelo mecanismo de composição. Desta forma, as técnicas de análise aplicadas a modelos de metas podem continuar sendo aplicadas aos modelos gerados pelo mecanismo de composição.

Os modelos de metas são entradas para o mecanismo de composição. O mecanismo de composição interpreta as informações contidas nos relacionamentos transversais e realiza a composição propriamente dita. Este processo é controlado por regras de composição gerando um modelo com todas as informações integradas, ou seja, gera modelos de metas contendo o comportamento (metas e tarefas) adicionado pelos relacionamentos transversais. As informações contidas nos relacionamentos transversais podem ser dependentes do domínio, e desta maneira não reutilizáveis. Entretanto, nossa experiência mostrou que algumas destas relações sempre ocorrem e poderiam ser igualmente reutilizadas, este é um ponto que queremos pesquisar mais profundamente.

O modelo resultante da composição e informações sobre algumas visões comuns são as entradas para a atividade de visualização. Esta atividade provê alguns modelos parciais do modelo integrado de maneira a facilitar o entendimento das características do sistema e da composição delas. Nós consideramos que o mais interessante para o desenvolvedor não é ter apenas a representação do sistema completo, mas sim diferentes visões deste modelo. Cada visão retrata o sistema por um certo ângulo, por exemplo: uma visão de quais partes do sistema utilizam um mecanismo de autenticação ou sofrem impacto dele, uma visão que mostre quais usuários tem autorização para utilizar quais partes do sistema, o impacto que o sistema sofre se uma outra característica transversal for adicionada, matrizes de rastreabilidade, dentre outras.

Este método de integração de características transversais utiliza os conceitos definidos pela linguagem AspectJ como um mecanismo para ajudar a modelagem de requisitos, ou seja, a escrever modelos de metas facilitando a inclusão e exclusão de características no modelo. Esta facilidade é essencial enquanto o conjunto final de requisitos não foi concluído e para evolução de requisitos. Além disto, esta abordagem provê uma nova forma de ver os requisitos, possibilitando a análise de que requisitos podem ser mapeados para aspectos nos níveis subseqüentes do desenvolvimento, caso se esteja utilizando uma abordagem de desenvolvimento orientada a aspectos.

3.1. Estudo de Caso

Este estudo de caso é composto de três modelos de metas: um retratando os requisitos intrínsecos da aplicação; um retratando alguns requisitos de segurança, neste caso, autenticação e criptografia para o sistema em questão; e um retratando o requisito de persistência. As Figuras 2, 3 e 4 ilustram, estes modelos de metas. Devido às limitações

da ferramenta gráfica utilizada, as elipses indicam *softmetas*; os hexágonos representam metas; e os retângulos, tarefas; os relacionamentos indicam decomposição.

O modelo de metas na Figura 3 é referente a um sistema de informação para edição do léxico e conjunto de cenários de um projeto de software [Felicíssimo 04]. A meta "Gerenciar [projeto]" é realizada pelas tarefas e submetas "Criar [projeto]", "Excluir [projeto]", "Salvar [projeto] como", "Editar [projeto]" dentre outros. Modelamos as principais metas do sistema e suas operacionalizações mostrando como atingir as metas. Entretanto, para possibilitar estas tarefas o ambiente computacional impõe algumas outras características tais como persistência, confiabilidade e confidencialidade.

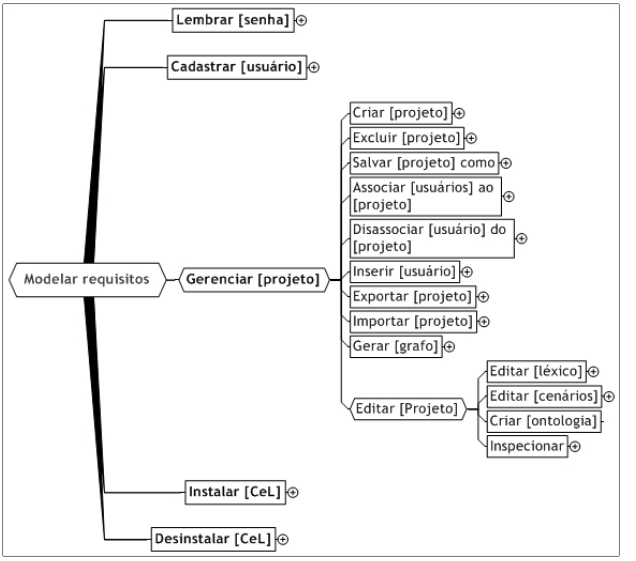


Figura 3. Modelo de metas dos requisitos do domínio

Na Figura 4 ilustramos o que se entende por segurança neste sistema. A "Segurança" é realizada através de outra *soft*-meta, denominada "Confidencialidade". Por sua vez, "Confidencialidade" é alcançada por um mecanismo de "Autenticação" e um de "Criptografia". "Criptografia" é constituída das tarefas "Criptografar [dados]", "Transmitir [dados]", "Descriptografar [dados]". "Autenticação" é composta das tarefas "Pedir [dados]" para autenticação, "Validar [dados]" e "Controlar [acesso]". A linha tracejada entre "Criptografia" e "Autenticação" indica um relacionamento transversal, na Figura 7 detalhamos esta relação.

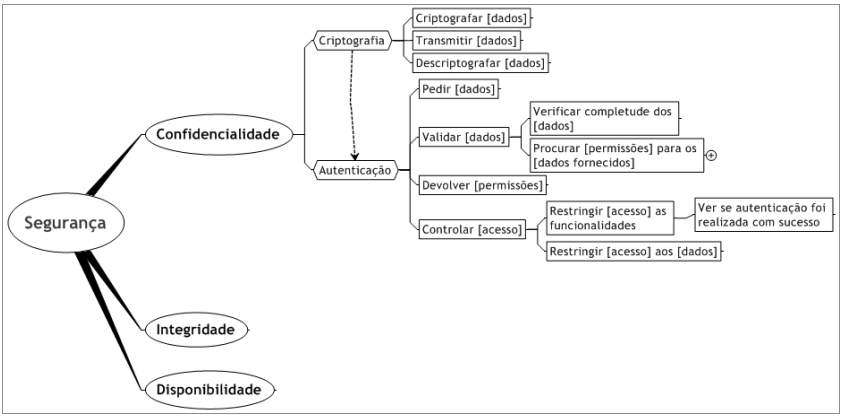


Figura 4. Modelo de metas dos requisitos de segurança

Na Figura 5 ilustramos o modelo de metas para "Persistência". Neste caso, a persistência pode ser realizada em arquivo ou em banco de dados (BD). A operacionalização para o caso do BD é composta das tarefas: "Verificar se [BD] está conectado", "Iniciar [BD]", "Conectar [BD]" e "Desconectar [BD]". Tanto no caso da persistência em BD quanto em arquivo são necessárias algumas operações de registro que devem ser restringidas pela tecnologia de BD ou de arquivo utilizada, tais como, "Consultar [dados]", "Incluir [dados]", "Modificar [dados]" e "Apagar [dados]".

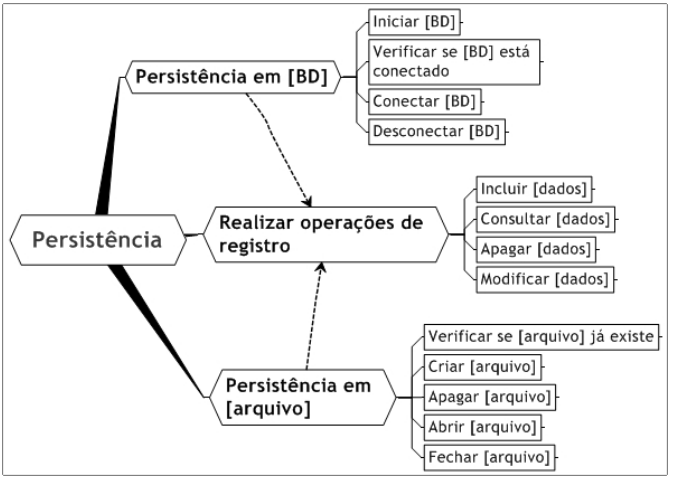


Figura 5. Modelo de metas dos requisitos de "Persistência"

Consideramos que o conjunto de requisitos sobre a aplicação, persistência, criptografia e autenticação são características do sistema sendo desenvolvido. Separamos cada uma delas de maneira a facilitar a análise delas isoladamente.

Entretanto, há relacionamentos não só entre a primeira e as outras três características, mas também entre as três últimas características.

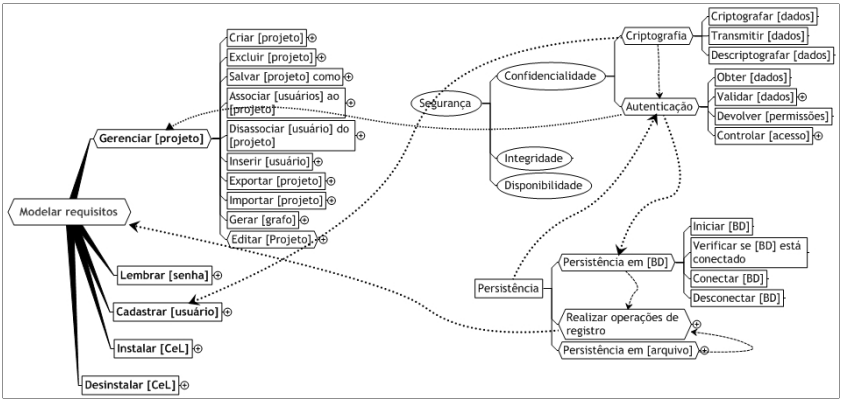


Figura 6. Relações transversais entre os modelos de metas

Na Figura 6, ilustramos algumas relações transversais entre estas características. Como exemplo, a linha tracejada entre criptografia e autenticação indica que a meta criptografia corta (*crosscuts*) a meta autenticação. Saber da existência desta relação é importante e pode nos alertar sobre o acoplamento entre elas. Entretanto, o que isto significa? que todas as tarefas de criptografia podem ser realizadas quando se faz a autenticação, ou apenas uma delas é realizada? E em que momento dentro de autenticação é necessário realizar tarefas de criptografia?

Para representar como estes tipos de relacionamento ocorrem, nos baseamos na linguagem AspectJ: *pointcuts* indicam o ponto-destino da relação; *advices* descrevem qual comportamento do ponto-origem (seqüência de tarefas) é incluído no ponto-destino e se será antes, durante ou depois dele; *inter-type declarations* descrevem elementos (tarefas ou tópicos) a serem adicionados no ponto destino. Assim, cada relacionamento transversal pode estar associado a vários *pointcuts, advices* e *intertype declarations*.

Nas Figuras 7 e 8 mostramos estas informações para os relacionamentos transversais entre as características “Criptografia” e “Autenticação”, e entre “Autenticação” e “Persistência em [BD]”, respectivamente. Na Figura 7 indicamos que após a tarefa “Obter [dados]” de “Autenticação”, as tarefas “Criptografar [dados]” e “Transmitir [dados]” são realizadas; e após “Validar [dados]” da meta “Autenticação”, é realizada a tarefa “Descriptografar [dados]”. Por sua vez, na Figura 8, descrevemos que “Autenticação” é realizada antes de qualquer tarefa que compõe “Gerenciar [projeto]” de “Modelar requisitos” (neste caso, através do controle de sessão), e é realizada também para que a tarefa “Conectar [BD]” de “Persistência em [BD]” seja atingida.

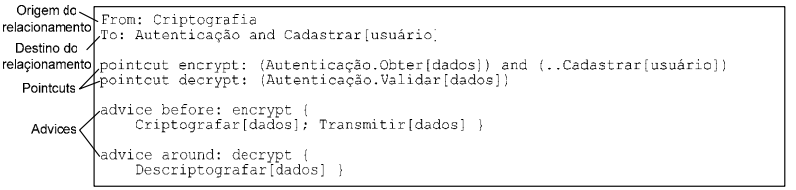


Figura 7. Detalhes do relacionamento transversal entre Criptografia e Autenticação

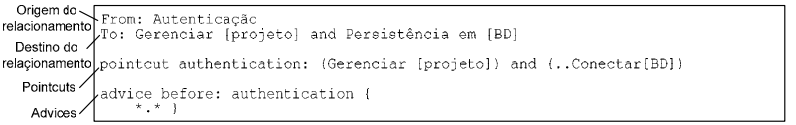


Figura 8. Detalhes do relacionamento transversal Autenticação e Persistência em [BD]

O mecanismo de composição processa estas informações e gera um modelo único com todas os elementos e seus relacionamentos. Na Figura 9 ilustramos uma visão do resultado da composição dos modelos de "Persistência em [BD]" e "Autenticação".

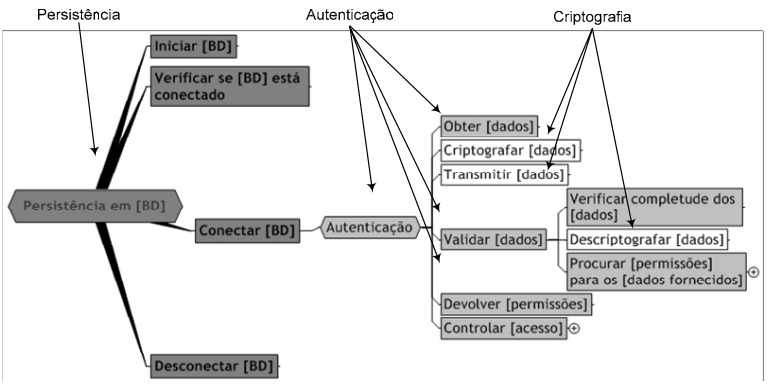


Figura 9. Composição das metas "Persistência em [BD]" e "Autenticação"

- O relacionamento entre "Persistência" e "Autenticação" (na Figura 6) resulta na inclusão das tarefas definidas em "Autenticação" para que se atinja a tarefa "Conectar [BD]" da meta "Persistência em [BD]", veja na Figura 9.
- O relacionamento entre "Autenticação" e "Criptografia" (na Figura 6) indica que algumas tarefas definidas em "Criptografia" são necessárias para que a meta "Autenticação" seja atingida, veja na Figura 9.

O modelo na Figura 9 é uma composição das tarefas definidas nos diferentes modelos de metas da Figura 6. Os diferentes tons de cinza representam a diferente origem que cada tarefa tem. Modelar estas propriedades separadamente nos permite pensar sobre elas isoladamente e facilita a remoção e/ou adição de novas características.

Por exemplo, caso decida-se que para conectar o BD não é necessário realizar nenhum mecanismo de autenticação, o trabalho desempenhado pelo modelador é somente de excluir o relacionamento transversal entre "Persistência em [BD]" e "Autenticação". Da mesma maneira, caso decida-se que a persistência em uma certa aplicação é feita em arquivo e não em BD simplesmente se removeria a parte de "Persistência em [BD]" ou se manteriam as duas possibilidades sendo apenas uma implementada.

Além de criar o modelo único, o mecanismo de composição possibilita a geração de diferentes visões, por exemplo o modelo ilustrado na Figura 9 . Algumas possíveis visões são: a visão de todas estas características juntas; a visão das tarefas do domínio que são modificadas por uma determinada característica; a visão de quais usuários possuem permissão para quais tarefas. Estas visões podem dar apoio ao levantamento de novos requisitos, à análise de impacto de mudanças, à priorização de requisitos, a estimar preço, tempo e esforço para implementar novos requisitos, dentre outras.

Estamos desenvolvendo o conjunto de ferramentas para tornar nosso método efetivo. Este conjunto de ferramentas utiliza: um modelo de componentes, um modelo de composição e um modelo de visões. Estes modelos dão suporte as três atividades definidas na Figura 2, respectivamente. Estaremos utilizando o padrão XML para representarmos estes modelos internamente.

O modelo de componentes define a linguagem de modelagem, ou seja, seus elementos e regras [Silva 05]. Como definimos em [Silva 05], a linguagem é uma extensão para o modelo de metas V-graph [Yu 04] e é composta de modelos de metas e relacionamentos transversais. Os modelos de composição e de visão estão sendo elaborados. O primeiro define as regras utilizadas para interpretar cada um dos relacionamentos transversais contidos no modelo de metas. O segundo define, em XSLT, o conjunto de visões que pode ser obtido do modelo integrado.

4. Trabalhos Relacionados

Há várias propostas para tratar características transversais, mas o foco tem sido dado à identificação de candidatos a aspectos [Rashid 02] e [Baniassad 04]. O primeiro utiliza os pontos de vista dos interessados no sistema de maneira a tentar encontrar a interseção de subconjuntos de requisitos. Este trabalho é fortemente baseados na rastreabilidade entre requisitos e pontos de vista. O segundo utiliza análise léxica das sentenças de requisitos e uma representação gráfica para identificar interseções entre as ações dos requisitos. Os trabalhos [Brito 04] e [Sousa 04] utilizam catálogos de requisitos não funcionais para dar apoio à tarefa de identificação de candidatos a aspectos, visto que requisitos não funcionais restringem o comportamento do sistema, espalhando-se por vários módulos.

Para representação dos aspectos candidatos, [Moreira 02], [Brito 04], [Rashid 03] e [Sousa 04] utilizam templates descrevendo como e onde cada aspecto exercerá impacto. [Sousa 04] utiliza o modelo de casos de uso para modelar os requisitos funcionais e usa as relações *extend* para representar os aspectos candidatos no mesmo modelo. Para integrar os aspectos candidatos à especificação dos requisitos funcionais, estes trabalhos definem regras de composição. Entretanto, com exceção de [Rashid 03], a composição realizada é manual, baseada na interpretação das informações contidas nos *templates*. Em [Rashid 03], os requisitos funcionais e as características transversais

são representadas através de modelos em XML e as regras de composição são utilizadas para criação de um novo modelo em XML com as informações integradas.

Todos estes trabalhos inspiraram a idéia para nossa proposta. Entretanto, eles se distinguem da nossa por focarmos a modelagem e composição de características transversais com o propósito de fornecer diferentes visões do modelo integrado, facilitando a inclusão e exclusão de diferentes características da especificação do sistema. Não é nosso objetivo criar um processo, nem identificar características transversais, apenas modelá-las de maneira a facilitar sua análise.

Dos trabalhos acima mencionados o que mais se assemelha ao nosso é [Rashid 03], entretanto, este provê apenas uma visão do modelo composto e não várias como é o nosso objetivo. Consideramos que nas fases anteriores a implementação, a composição de características transversais não tem seus reais benefícios se não prover maneiras de extrair visões das características integradas, pois o modelo integrado pode ser tão complexo que não proverá sua principal vantagem que é facilitar o entendimento do sistema. Outra diferença é que ele trata sentenças de requisitos, enquanto nós estamos tratando modelos de metas que possuem elementos e relações com uma semântica que possibilita inferências.

Apesar dos trabalhos [Sousa 04] e [Baniassad 04] tratarem da modelagem de características transversais elas não oferecem apoio automatizado para composição, mas somente uma extensão para alguns modelos da UML. Outra diferença é que estes trabalhos tentam identificar, separar e compor candidatos a aspectos, tentando fazer um mapeamento direto deles para aspectos no nível de implementação. Nós não temos esta pretensão, mas sim possibilitar a modelagem e entendimento de características que possuem esta propriedade transversal, mapeando a rastreabilidade entre elas, apesar de pretendermos com isto dar apoio aos desenvolvedores nas decisões de desenho.

5. Conclusões e Trabalhos Futuros

Levando em consideração os princípios de separação e composição de características transversais, definimos um método que objetiva dar suporte a modelagem de requisitos. Este método é constituído por três atividades principais: Separação, Composição e Visualização. A separação dá suporte a modelagem de requisitos por oferecer um novo tipo de relacionamento que possibilita o registro de como os diferentes elementos restringem uns aos outros. A composição oferece um mecanismo, similar ao *weaver* da linguagem AspectJ, que interpreta os relacionamentos transversais e gera um modelo com todas as informações unidas. A atividade de visualização oferece um mecanismo para extrair diferentes visões parciais ou totais do modelo integrado.

Consideramos que o impacto de realizar estas três atividades juntas é muito maior do que realizá-las separadamente. Juntas, estas atividades ajudam o engenheiro de requisitos a tratar cada conjunto de requisitos isoladamente ou em conjunto com os demais. Por facilitar a modelagem acreditamos que estamos provendo um impacto positivo em todo o processo de desenvolvimento de software, facilitando a evolução de requisitos, o reuso de características transversais, a análise de variabilidade, a análise de conflitos, a estimativa de custos e prazos, as decisões de desenho e a evolução de software.

Para tornar esta abordagem eficaz estamos trabalhando: na implementação do conjunto de ferramentas necessário para realizar a integração de modelos de metas; na extensão do modelo de metas para representar as relações transversais [Silva 05]; no desenvolvimento de regras de composição e regras para extração de um conjunto de visões comuns que podem ser derivadas do modelo de metas; na flexibilização da abordagem para que seja aplicada a outros modelos de requisitos; e na modelagem de um conjunto preliminar de características transversais reusáveis.

6. Referências Bibliográficas

[Bakker 05] J. Bakker, B. Tekinerdogan and M. Aksit, "Characterization of early aspects approaches", Proceedings of the Early Aspects Workshop at AOSD, 2005.

[Baniassad 04] E. Baniassad and S. Clarke, "Theme: An approach for aspect-oriented analysis and design", 26th International Conference on Software Engineering (ICSE'04), Scotland, 2004, pp. 158-167.

[Brito 04] I. Brito and A. Moreira, "Integrating the NFR framework in a RE model", Proceedings of the Early Aspects Workshop at AOSD, England, 2004.

[Chavez 04] C. Chavez, "A Model-Driven approach to aspect-oriented design", PhD Thesis, Computer Science Department, PUC-Rio, Rio de Janeiro, Brazil, 2004.

[Felicíssimo 04] C.H. Felicíssimo, J.C.S.P Leite, K.K. Breitman e L.F. Silva, "C&L: Um Ambiente para Edicao e Visualizao de Cenarios e Léxicos", Publicantions in CD., Workshop de Ferramentas do XVIII Simposio Brasileiro de Engenharia de Software. Brasilia, Brasil, ISBN 85-7669-004-7, 2004, pp. 43-48.

[Giorgini 02] P. Giorgini, J. Mylopoulos, E. Nicchiarelli and R. Sebastián, "Reasoning with goal models", Proceedings of the 21st International Conference on Conceptual Modeling, 2002, pp. 167-181.

[Gonzáles 04] B. Gonzáles, M. Laguna and J. Leite, "Visual variability analysis with goal models", Proceedings of IEEE International Symposium on Requirements Engineering (RE'04), Japan, 2004, pp. 38-47.

[Kiczales 97] G. Kiczales et al., "Aspect-oriented programming", Proceedings of the European Conference on Object-Oriented Programming (ECOOP'97), LNCS (1241), Springer-Verlag, Finland, 1997.

[Kiczales 01] G. Kiczales et al., "An overview of aspectJ", Proceedings of the European Conference on Object-Oriented Programming (ECOOP'01), Hungary, 2001.

[Lamsweerde 00] A. Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering", IEEE Transaction Software Engineering, 26(10):978–1005, 2000.

[Lamsweerde 01] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering, Toronto, 2001, pp. 249-263.

[Leite 04] Julio Leite, Yijun Yu, Lin Liu, Eric Yu e John Mylopoulos, "Quality-Based Software Reuse", Proceedings of the CAiSE 2005-LNCS 3520, 2005, pp. 535-550.

[Moreira 02] A. Moreira, J. Araújo and I. Brito, "Crosscutting quality attributes for requirements engineering", Proceeding of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002), ACM Press, Italy, 2002.

[Mylopoulos 92] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach", IEEE Transactions on Software Engineering, 18(6):483–497, June 1992.

[Piveta 04] Eduardo K. Piveta , Vinicius C. Garcia, Luiz C. Zancanella, Antonio F. do Prado, "Termos em português para Desenvolvimento de Software Orientado a Aspectos", 1º Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos (WASP'04), 2004.

[Rashid 02] A. Rashid, P. Sawyer, A. Moreira and J. Araújo, "Early aspects: a model for aspect-oriented requirements engineering", Proceedings of IEEE Joint Conference on Requirements Engineering, Germany, 2002, pp. 199-202.

[Rashid 03] A. Rashid, A. Moreira and J. Araújo, "Modularization and composition of aspectual requirements", Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, ACM, 2003, pp. 11-20.

[Silva 05] L. Silva, J. Leite, "Uma linguagem de modelagem de requisitos orientada a aspectos", Proceedings of the Requirement Engineering Workshop at CAiSE 2005, Porto-Portugal, 2005.

[Sousa 04] G. Sousa, S. Soares, P. Borba and J. Castro, "Separation of crosscutting concerns from requirements to design: Adapting the use case driven approach", Proceedings of the Early Aspects Workshop at AOSD, England, 2004.

[Sousa 03] G. Sousa, I. Silva, J. Castro, "Adapting the NFR framework to aspect-oriented requirements engineering", Proceedings of the 17th Brazilian Symposium on Software Engineering - SBES'2003, Brazil, 2003.

[Tarr 99] P. Tarr, H. Ossher, W. Harrison, S. Sutton, "N degrees of separation: multi-dimensional separation of concerns", Proceedings of the International Conference on Software Engineering (ICSE'99), Los Angeles-CA, 1999.

[Tarr 00] P. Tarr, H. Ossher; Hyper/J User and Installation Manual; 2000. http://www.alphaworks.ibm.com/tech/hyperj

[Tekinerdoðan 04] B. Tekinerdoðan, A. Moreira, J. Araújo, P. Clements, "Early aspects: aspect-oriented requirements engineering and architecture design", Report Early Aspects Workshop at AOSD, England, 2004.

[Yu 04] Y. Yu, J. Leite and J. Mylopoulos, "From goals to aspects: discovering aspects from requirements goal models", Proceedings of IEEE International Symposium on Requirements Engineering (RE'04), Japan, 2004, pp. 38-47.

[Zave 97] P. Zave, M. Jackson, "Four dark corners of requirements engineering", ACM Transaction Software Engineering Methodologics, 6(1):1–30, 1997.