5
Estudos de Caso

Neste Capítulo, apresentamos dois estudos de caso realizados para demonstrar a abordagem descrita no Capítulo 4. O primeiro estudo de caso é um sistema colaborativo para edição de léxico e cenários (também utilizado como exemplo nos capítulos anteriores) e o segundo é um sistema para divulgação de eventos na Universidade da Califórnia.

Ambos os estudos de caso referem-se a sistemas simples (com pequeno número de funcionalidades). Ainda assim, a modelagem em AOV-graph apresenta um número excessivo de relacionamentos, dificultando a leitura, manipulação e análise dos modelos. Esta dificuldade é decorrente não apenas da decomposição dominante imposta pelo modelo de componentes (V-graph), mas principalmente do espalhamento e entrelaçamento de características. Desta forma, aplicar nossa abordagem trouxe alternativas para visualizar os modelos e representar a interação de elementos que se espalham e/ou se entrelaçam em muitas partes dos grafos (modelos de metas).

Na Seção 5.1, descrevemos o contexto e processo utilizado para elaboração destes estudos de caso. Nas Seções 5.2 e 5.3 detalhamos cada estudo de caso por meio da descrição do problema, modelagem e análise dos resultados. Na Seção 5.4, resumimos e mostramos como tais resultados confirmam as contribuições de nossa estratégia para modelagem, rastreabilidade, evolução e reuso de requisitos.

5.1.
Contextualização

A modelagem dos estudos de caso, em AOV-graph, foi realizada com base em documentos disponíveis: no primeiro caso não há um documento de requisitos, mas há uma descrição de suas funcionalidades e arquitetura em (Silva, 2005a), e o sistema, bem como seus cenários encontram-se disponível em (C&L, 2005); e no segundo caso, nos baseamos exclusivamente no documento de requisitos disponível em (Unical, 2005).

Estes documentos foram utilizados na atividade de separação de nossa abordagem. O processo de separação ocorreu da seguinte maneira:

- Leitura da documentação – a leitura ocorreu inicialmente de maneira integral para identificarmos a estruturação e organização dos requisitos, bem como conhecer qual o propósito geral do sistema. Em seguida, a leitura foi realizada de maneira pontual para reescrevermos os requisitos através de metas, *softmetas* e tarefas.
- Identificação dos principais grupos de requisitos – Com base no propósito geral, estruturação e organização dos documentos inferimos os principais requisitos (macro-requisitos ou características) do sistema. A separação mais visível entre estas características é com relação ao que pertence exclusivamente ao objetivo do sistema e ao que é decorrente da automação deste objetivo. Este segundo grupo está, normalmente associado a RNFs e suas operacionalizações, e pode muitas vezes ser modelado de maneira que possa ser reutilizado em outras aplicações. Desta forma, modelamos as características que constituem o núcleo do sistema como um modelo, e cada característica pertencente ao segundo grupo, como um modelo separado dos demais.
- Modelagem dos requisitos como tarefa, meta e *softmeta* – para cada característica, identificamos os requisitos associados e os modelamos como meta, *softmeta* ou tarefa. Sendo que a relação entre requisito (ou cenário, no caso do C&L) para elemento no AOV-graph nem sempre é de um para um. Isto acontece porque, muitas vezes, os requisitos possuem mais de uma função ou dado (entrelaçamento de informações), ou requisitos com o mesmo significado são escritos de maneiras diferentes (espalhamento de informações).
- Relacionando metas, *softmetas* e tarefas – para relacionar os requisitos utilizamos os elos de contribuição, correlação e transversal. Os elos de correlação e contribuição são escritos à medida que modelamos metas, *softmetas* e tarefas, de maneira a identificar suas decomposições. Os relacionamentos transversais são escritos quando metas, *softmetas* e tarefas afetam diferentes pontos nos modelos de maneira a tornar as informações espalhadas e entrelaçadas (veja algumas heurísticas no Capítulo 4, Seção 4.2.2).

- Modificação dos modelos criados – a modelagem das informações contidas na documentação não ocorre em uma única iteração. À medida que a decomposição de metas, *softmetas* e tarefas está sendo modelada, podemos identificar outros grupos de requisitos a serem separados, elementos a serem detalhados e relacionamentos a serem agrupados em um elo transversal.

Estes modelos foram escritos em XML e todos os diagramas foram gerados por meio de XSLT para o formato de entrada do software Visigraph (Visigraph, 2006). Este software é responsável por esquematizar e gerar figuras no formato JPG. Desta forma, a implementação não oferece suporte à edição gráfica dos modelos; para realizar qualquer alteração é necessário modificar o modelo em XML.

As atividades de composição e visualização que complementam este processo de separação ocorrem em seguida, apoiadas pela implementação descrita no Capítulo 4, Seção 4.3.4. Estes mecanismos dão apoio à atividade de separação porque geram diferentes visões do modelo original, facilitando sua análise e modificação.

Para avaliar os ganhos obtidos com o uso do relacionamento transversal, criamos uma nova visão do modelo AOV-graph que mostra todos os modelos e seus relacionamentos sem mostrar a separação entre eles. Esta visão representa como seriam os modelos em V-graph se não utilizássemos nossa abordagem. Assim, podemos mostrar as diferenças gráficas entre estas duas abordagens.

A legenda dos elementos de V-graph e AOV-graph é ilustrada na Figura 76. Esta legenda é válida para todos os modelos de metas dos estudos de caso. Os relacionamentos em negrito são os relacionamentos transversais ou relacionamentos gerados a partir de um relacionamento transversal.

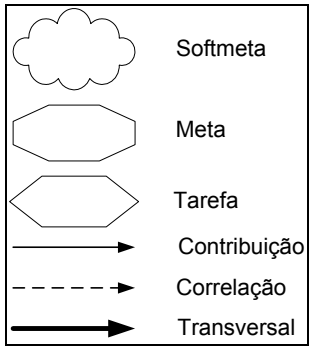


Figura 76. Legenda dos elementos de V-graph e AOV-graph

5.2.
Sistema Colaborativo para Edição de Léxico e Cenários – C&L

C&L é um sistema WEB para edição de cenários e léxico segundo a definição de Leite (1997). Este sistema provê um ambiente colaborativo para que participantes do mesmo projeto possam navegar entre léxico e cenários, e sugerir mudanças para os mesmos. Este sistema foi desenvolvido no meio acadêmico incrementalmente por diferentes equipes, a descrição de suas funcionalidades e alguns detalhes de sua arquitetura estão disponíveis em (Silva, 2005; Felicíssimo, 2004). A modelagem de C&L, apresentada nesta tese, foi realizada tendo como base, também, o sistema disponível em (C&L, 2005).

5.2.1.
Modelagem

A meta principal do sistema C&L é promover a definição de cenários e léxico por membros de equipes que trabalham colaborativamente, mesmo que estejam geograficamente distribuídos. Para realizar esta meta, opta-se por desenvolver uma ferramenta para WEB, um ambiente aberto que promove a divulgação e compartilhamento de informações. Por outro lado, este ambiente exige alguns mecanismos para prover confidencialidade às informações de cada projeto. Além dos requisitos específicos de confidencialidade e de modelagem de requisitos, características de persistência e confiabilidade também estão presentes. Denominamos a característica referente à modelagem de requisitos de CEL para facilitar a citação.

Na Figura 77, ilustramos a modelagem de C&L em V-graph; sem utilizar nossa abordagem. Os quatro grupos de requisitos (CEL, Confidencialidade, Segurança e Persistência) são modelados formando um único grafo de *softmetas*, metas e tarefas. Neste modelo não está explícito quais elementos são referentes à quais grupos de requisitos (quais os requisitos referentes à confidencialidade ou específicos do CEL?). A complexidade decorrente disto e do grande número de relacionamentos e elementos, dificulta a manipulação, análise e modelagem do modelo inteiro ou referente a cada grupo de requisitos. Isto afeta, também, a reusabilidade, porque é difícil identificar partes da modelagem que poderiam ser reutilizadas em outros projetos.

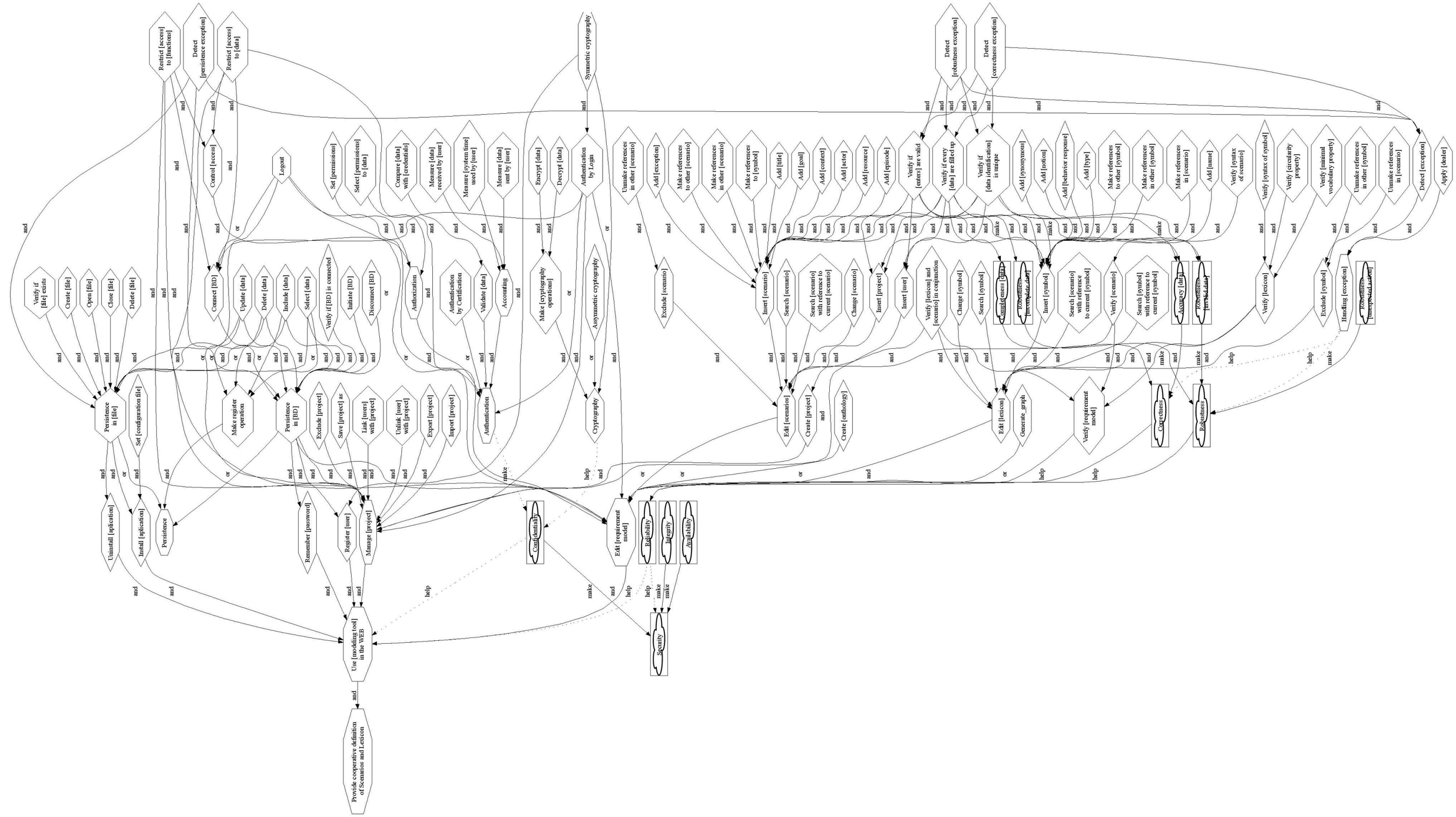


Figura 77. Modelagem de C&L em V-graph

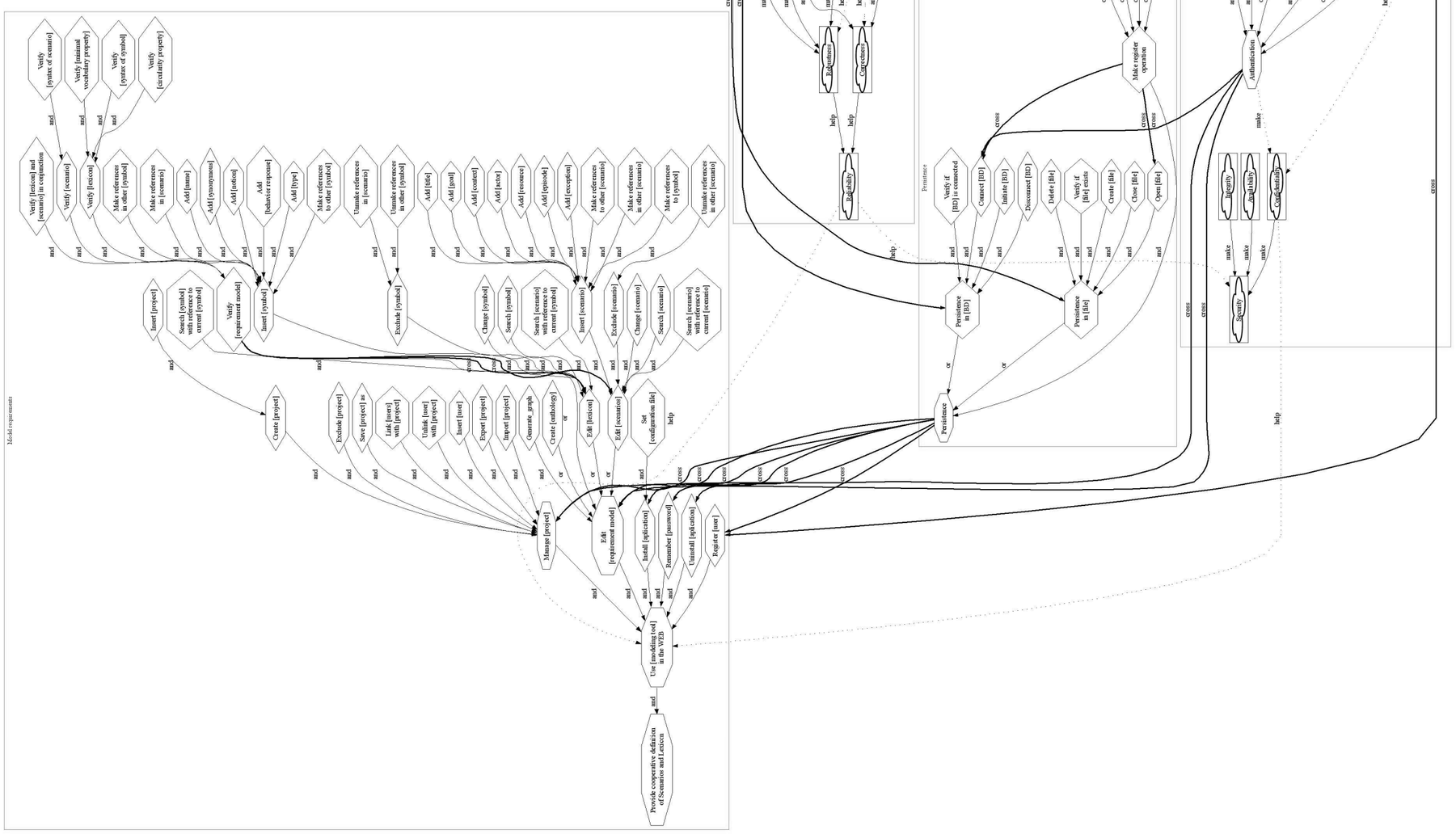


Figura 78. Modelagem de C&L em AOV-graph

Como estes grupos de requisitos são referentes a características distintas e não precisam coexistir no mesmo sistema, os separamos em quatro modelos. Na Figura 78, ilustramos a modelagem de C&L em AOV-graph. Neste modelo ilustramos cada grupo de requisitos com escopo delimitado e as interações entre eles. Esta nova disposição dos grupos esclarece quais *softmetas*, metas e tarefas dizem respeito a cada característica.

Utilizando o relacionamento transversal (em negrito na Figura 78), explicitamos que algumas características afetam outras de maneira que tornam os elementos mais acoplados e pouco coesos. Centralizamos a informação de como cada elemento se espalha pelos demais, e assim diminuímos a quantidade de relacionamentos no modelo. O relacionamento transversal tem um significado além de simplesmente relacionar diferentes características; com ele podemos acrescentar novos elementos que surgem da junção de modelos, i.e., a composição de características resulta em um sistema que é maior que a soma de suas partes. Utilizando, nosso mecanismo de visualização, a seguir, explicamos cada um dos modelos do sistema C&L e seus relacionamentos transversais.

Modelagem de Requisitos (CEL)

Na Figura 79, ilustramos uma visão parcial do sistema C&L, a característica Modelagem de requisitos (CEL). Neste modelo, a tarefa "Verify [requirements model]" representa uma atividade de inspeção do modelo de cenário, do léxico e de ambos os modelos em conjunto. Como esta tarefa poderia ser modelada independentemente do modelo de edição de léxico e cenários e aparece como sub-tarefa em três momentos diferentes na funcionalidade do sistema (quando editando o léxico, ou editando os cenários ou navegando pelas informações dos modelos), a consideramos uma característica que corta as tarefas "Edit [lexicon]" e "Edit [scenario]". Isto significa que "Verify [requirements model]" modifica as tarefas de CEL, incluindo ou substituindo outras tarefas.

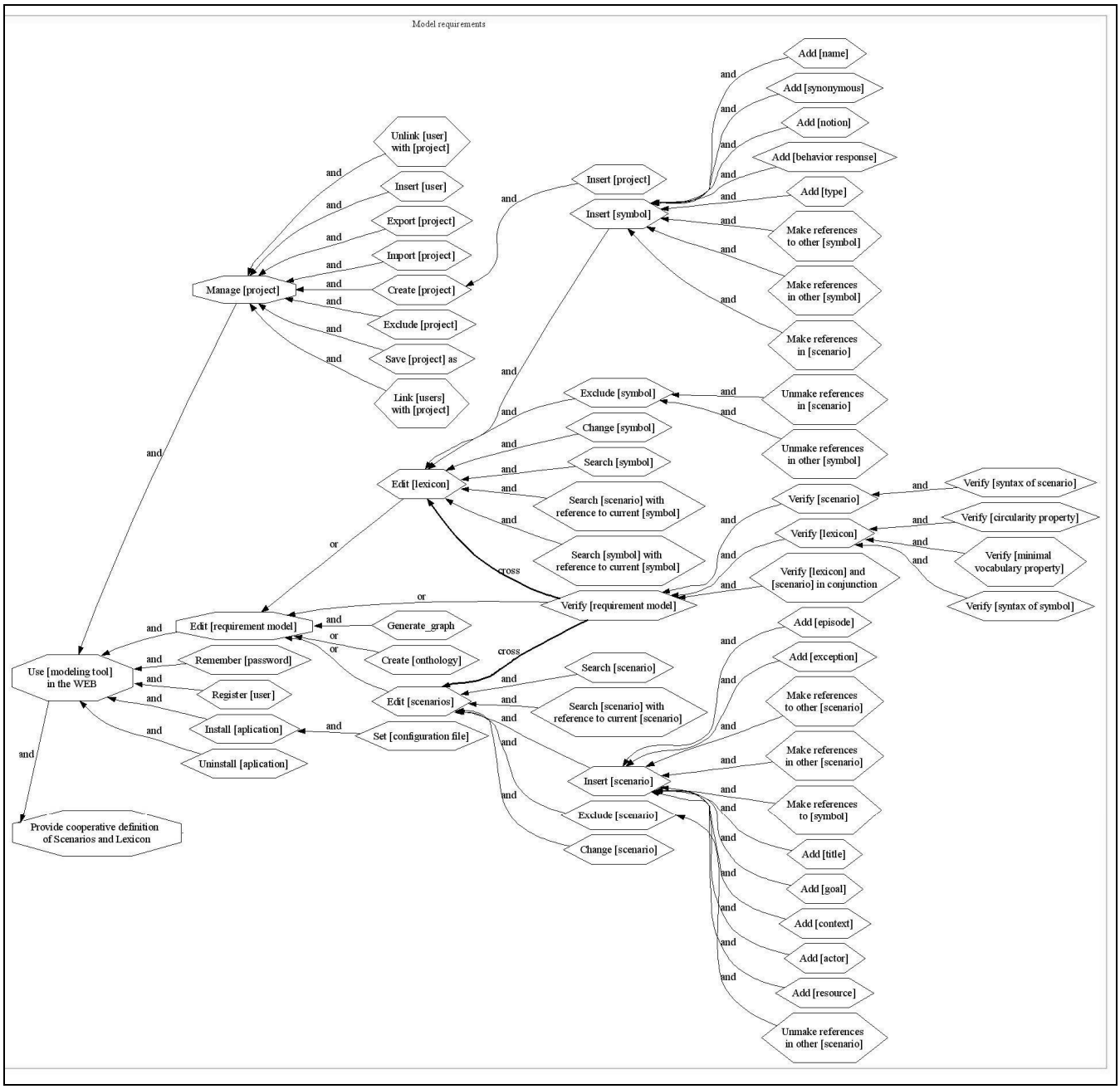


Figura 79. Modelo CEL

```
1.  Crosscutting {
2.    source: Verify [requirement model]
3.    pointcut PC1: include( Edit [scenarios] )
4.    pointcut PC2: include( Edit [lexicon] )
5.    advice around: PC1 {Verify [scenario] }
6.    advice around: PC2 {Verify [lexicon] }
7.    advice around: PC1 and PC2 {Verify [scenario] and [lexicon] in conjunction}
8.  }
```

Figura 80. Relacionamento transversal do modelo CEL

Na Figura 80, descrevemos o relacionamento transversal existente entre "Verify [requirements model]", e "Edit [lexicon]" e "Edit [scenario]". Este relacionamento indica que a atividade "Verify [lexicon]" se entrelaça à tarefa

"Edit [lexicon]", a tarefa "Verify [scenario]" se entrelaça à tarefa "Edit [scenario]", e a tarefa "Verify [scenario] and [lexicon] in conjuction" se entrelaça às tarefas "Edit [scenario]" e "Edit [lexicon]" – ou seja, as atividades "Edit [lexicon]" e "Edit [scenario]" incluem as atividades de verificação. Por outro lado, sob o ponto de vista da tarefa "Verify [requirements model]", este relacionamento transversal indica que suas sub-tarefas estão espalhadas por outras sub-árvores do modelo.

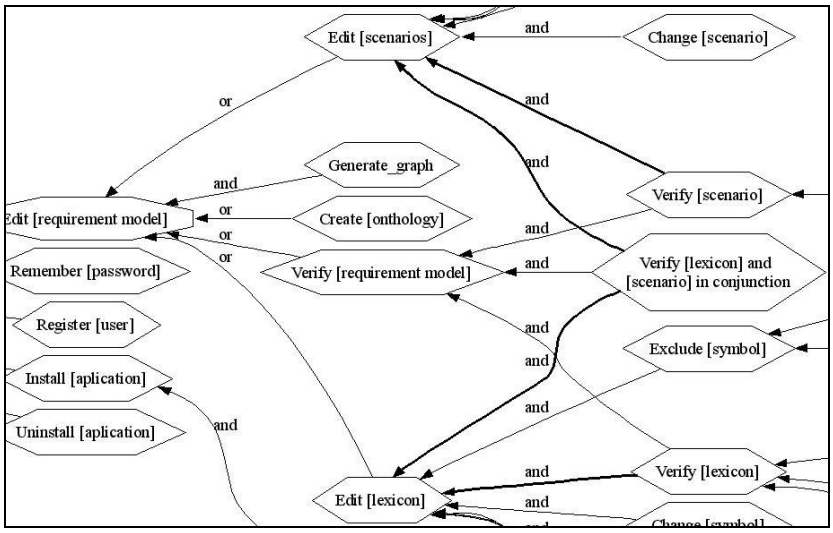


Figura 81. Modelo CEL após a composição

O mecanismo de composição é responsável por propagar estas informações, i.e., por criar os relacionamentos de contribuição entre estas tarefas, veja a Figura 81. Desta forma: 1) podemos trabalhar nestes grupos de requisitos separadamente, sendo necessário definir por meio de um relacionamento transversal como o segundo grupo (Verify [requirements model]) está relacionado ao primeiro (Edit [requirement model]), i.e., neste caso, apenas o segundo modelo afeta o primeiro; 2) quando o foco não é o relacionamento transversal (no modelo antes da composição), ele é representado graficamente por apenas dois elos, aumentando a visibilidade do modelo; 3) quando a composição é realizada o modelo pode ser visualizado com os relacionamentos transversais expandidos, i.e., no lugar de dois elos podemos ver quatro elos.

Segurança

Na Figura 82, ilustramos os requisitos de segurança e suas interações com os modelos de persistência e CEL (elementos em cinza). Neste caso apenas Confidencialidade é decomposto em tarefas. Neste modelo, as metas

"Authenthication" e "Crypthography" são inícios de relacionamentos transversais. Eles afetam o elemento "Authentication by login" do próprio modelo de segurança, e afetam também os elementos "Connect [BD]", "Manage [project]" e Edit [requirement model] dos modelos de persistência e CEL.

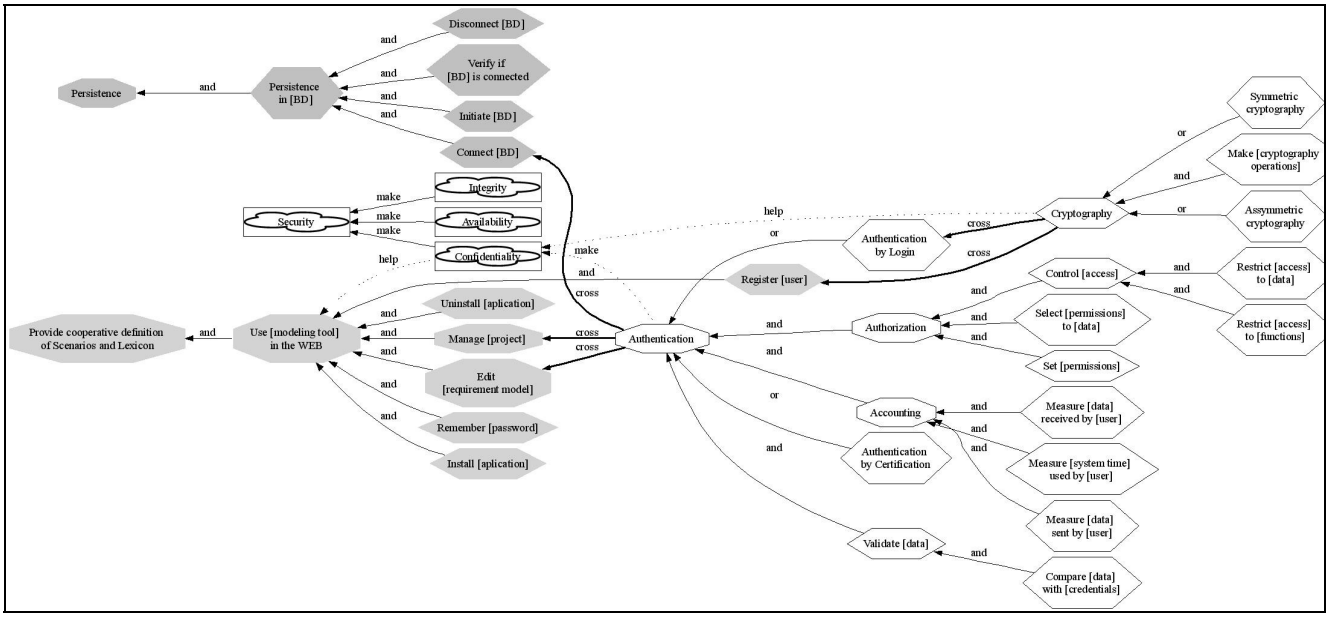


Figura 82. Modelo de segurança

Na Figura 83, apresentamos a definição dos relacionamentos transversais que têm origem no modelo de segurança. O primeiro deles indica que "Authentication" inclui as tarefas "Authentication by login", "Restrict [access] to [data]" e "Restrict [access] to [functions]" nos elementos "Edit [requirement model]" e "Manage [project]". Além disto, *intertype declarations* são definidas para incluir nestes mesmos elementos um atributo denominado "constraint" e valor "confidential data", e uma tarefa denominada "Logout".

1	1. Crosscutting{ 2. source: Authentication 3. pointcut PC3: include(Edit [requirement model]) and include(Manage [project]) 4. and include(Connect [BD]) 5. advice around: PC3 { Authentication by login ; Restrict [access] to [data]; 6. Restrict [access] to [functions] } 7. intertype declaration element: PC3 { Logout } 8. intertype declaration attribute: PC3 { constraint = confidential data } 9. }
2	1. Crosscutting { 2. source: Cryptography 3. pointcut PC4: include(Authentication by login) and include(Register [user]) 4. advice around: PC4 {Symmetric cryptography } 5. }

Figura 83. Relacionamentos transversais do modelo de segurança

O segundo relacionamento transversal, descrito na Figura 83, indica que a tarefa “Crypthography” afeta as tarefas “Register [user]” e “Authentication by login” adicionando a tarefa “Symmetric crypthography”.

Na Figura 84, apresentamos estes relacionamentos transversais após a composição. Nela podemos observar a criação da nova tarefa “Logout” e os novos relacionamentos. Esta visão está focada no serviço segurança, os elementos em cinza são elementos dos modelos CEL e de persistência.

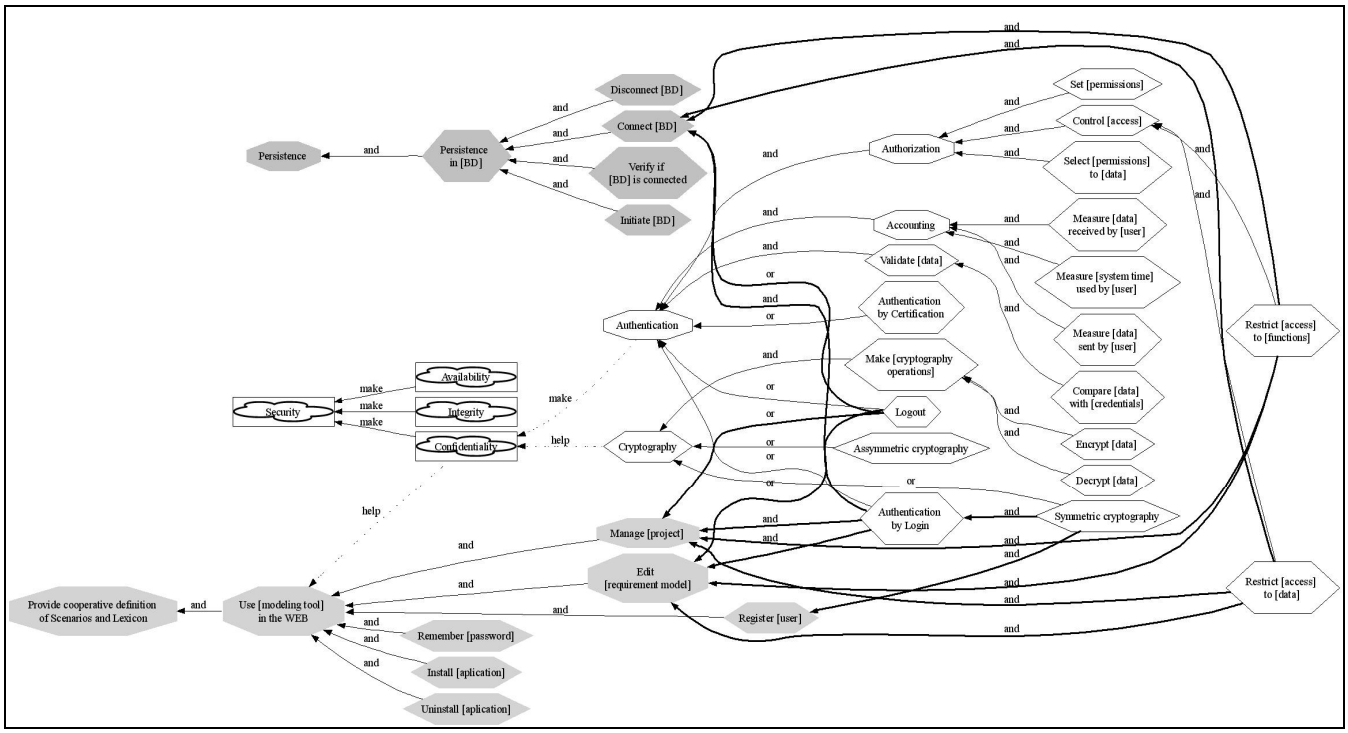


Figura 84. Modelo de segurança após a composição

Persistência

A característica Persistência é fundamental no sistema C&L, estando presente em muitas das tarefas de edição. Na Figura 85, apresentamos o modelo de persistência e os relacionamentos transversais dele para o modelo CEL.

Na Figura 86: o primeiro relacionamento indica que a tarefa “Make register operation” inclui as tarefas “Include [data]”, “Select [data]”, “Delete [data]” e “Update [data]” depois das tarefas “Connect [BD]” e “Open [file]”, i.e., estas tarefas são necessárias para realizar persistência em arquivo ou em banco de dados, mas é necessário que o arquivo tenha sido aberto e/ou o banco de dados esteja conectado; o segundo relacionamento transversal indica que as metas “Manage [project]” e “Edit [requirements model]” são afetadas pela meta

"Persistence in [BD]" e as tarefas "Install [aplication]" e "Uninstall [application]" fazem persistência em arquivo ("Persistence in [file]"). O resultado da composição é ilustrado na Figura 87.

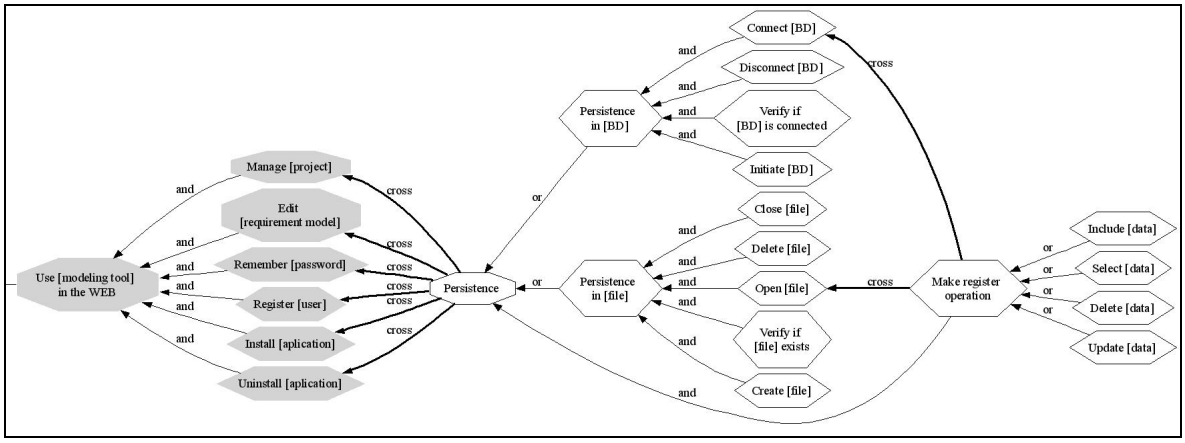


Figura 85. Modelo de persistência

1	1. Crosscutting { 2. source: Make register operation 3. pointcut PC5: include(Connect [BD]) and include(Open [file]) 4. advice after: PC5 {Include [data]; Select [data]; Delete [data]; Update [data] } 5. }
2	1. Crosscutting { 2. source: Persistence 3. pointcut PC6: include(Manage [project]) and include(Edit [requirement model]) 4. pointcut PC7: include(Install [aplication]) and include(Uninstall [aplication]) 5. advice around: PC6 {Persistence in [BD] } 6. advice around: PC7 {Persistence in [file] } 7. }

Figura 86. Relacionamentos transversais do modelo de persistência

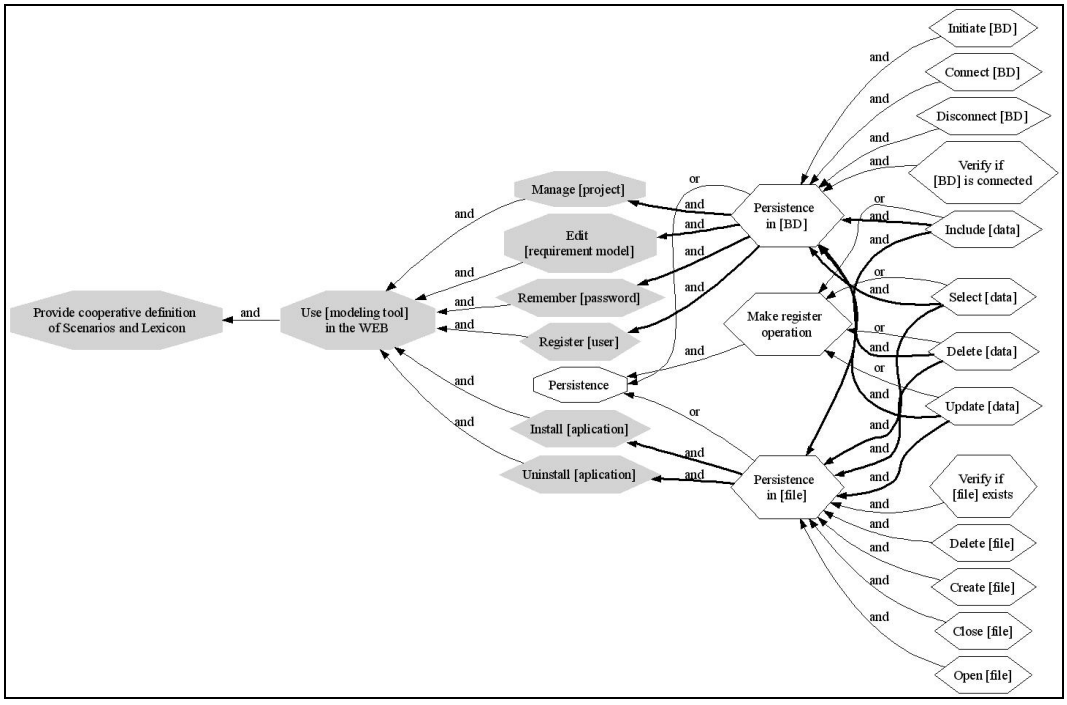


Figura 87. Modelo de persistência após a composição

Confiabilidade

No sistema C&L, apesar de não haver uma forte preocupação com confiabilidade, há algumas tarefas simples para garantir que os dados fornecidos são corretos e que o sistema não pára com o fornecimento de dados errados. Isto é realizado de maneira preventiva, realizando as seguintes verificações durante a entrada de dados: “Verify if [enters] are valid”, “Verify if every [data] are filled up” e “Verify if [data identification] is unique”, veja Figura 88.

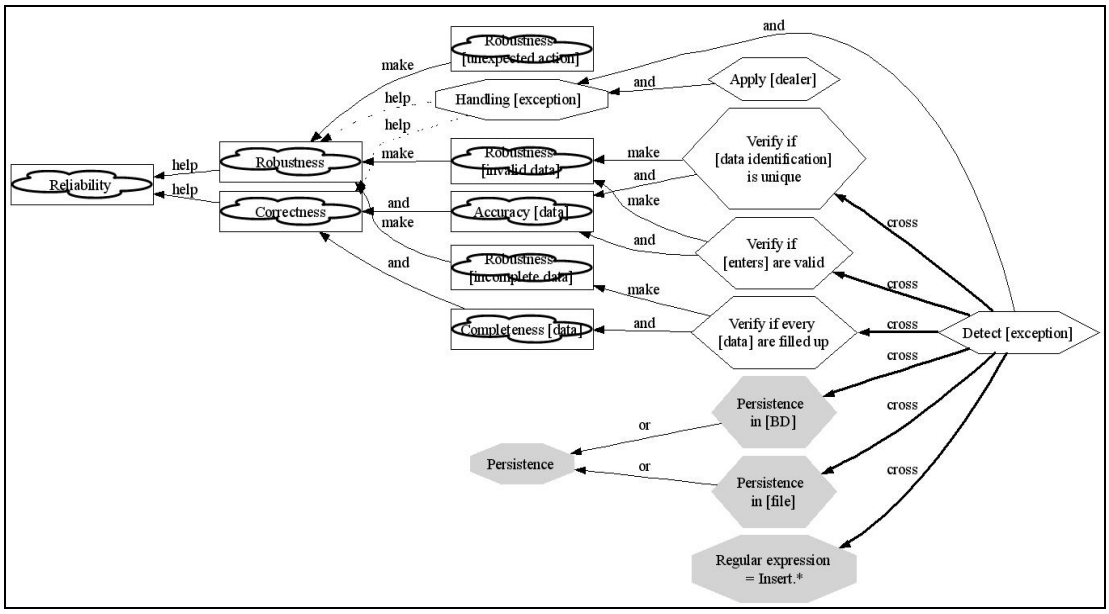


Figura 88. Modelo de confiabilidade

```
1.  Crosscutting {
2.    source: Detect [exception]
3.    pointcut PC8: include( Verify if [data identification] is unique ) and
4.          include( Verify if [enters] are valid )
5.    pointcut PC9: include( Verify if every [data] are filled up )
6.    pointcut PC10: include( Persistence in [BD] ) and include( Persistence in [file] )
7.    pointcut PC11: include( Insert.*={ (Insert [project]) (Insert [user])
8.          (Insert [scenario]) (Insert [symbol]) } )
9.    advice around: PC11 { Verify if [data identification] is unique ;
10.         Verify if [enters] are valid ; Verify if every [data] are filled up }
11.   intertype declaration element: PC10 { Detect [persistence exception] }
12.   intertype declaration attribute: PC8 { exception = Invalid data }
13.   intertype declaration attribute: PC9 { exception = Incomplete data }
14.   intertype declaration element: PC8 and PC9 { Detect [robustness exception] ;
15.         Detect [correctness exception] }
16. }
```

Figura 89. Relacionamentos transversais do modelo de confiabilidade

Na Figura 89, descrevemos o relacionamento transversal existente no modelo de confiabilidade. Este relacionamento transversal adiciona três novas tarefas à “Detect [exception]”, denominadas “Detect [persistence exception]”, “Detect [robustness exception]” e “Detect [correctness exception]”. A primeira

destas novas tarefas é adicionada também em “Persistence in [BD]” e “Persistence in [file]” do modelo de persistência, e as duas últimas tarefas são adicionadas nas tarefas “Verify if [data identification] is unique”, “Verify if [enters] are valid” e “Verify if every [data] are filled up” do sub-modelo de robustez. As tarefas “Verify if [data identification] is unique”, “Verify if [enters] are valid” e “Verify if every [data] are filled up” são incluídas em todas as tarefas cujo nome inicia com “Insert”. Além disto, um atributo denominado “exception” é criado nos elementos “Verify if [data identification] is unique” e “Verify if [enters] are valid” com valor “invalid data”, e em “Verify if every [data] are filled up” com valor “incomplete data”.

Na Figura 90, ilustramos o modelo de confiabilidade após a composição. Os elementos em cinza são os elementos afetados do modelo CEL e de persistência. Podemos observar três novos elementos pertencentes à junção destes modelos: “Detect [persistence exception]”, “Detect [correctness exception]” e “Detect [robustness exception]”.

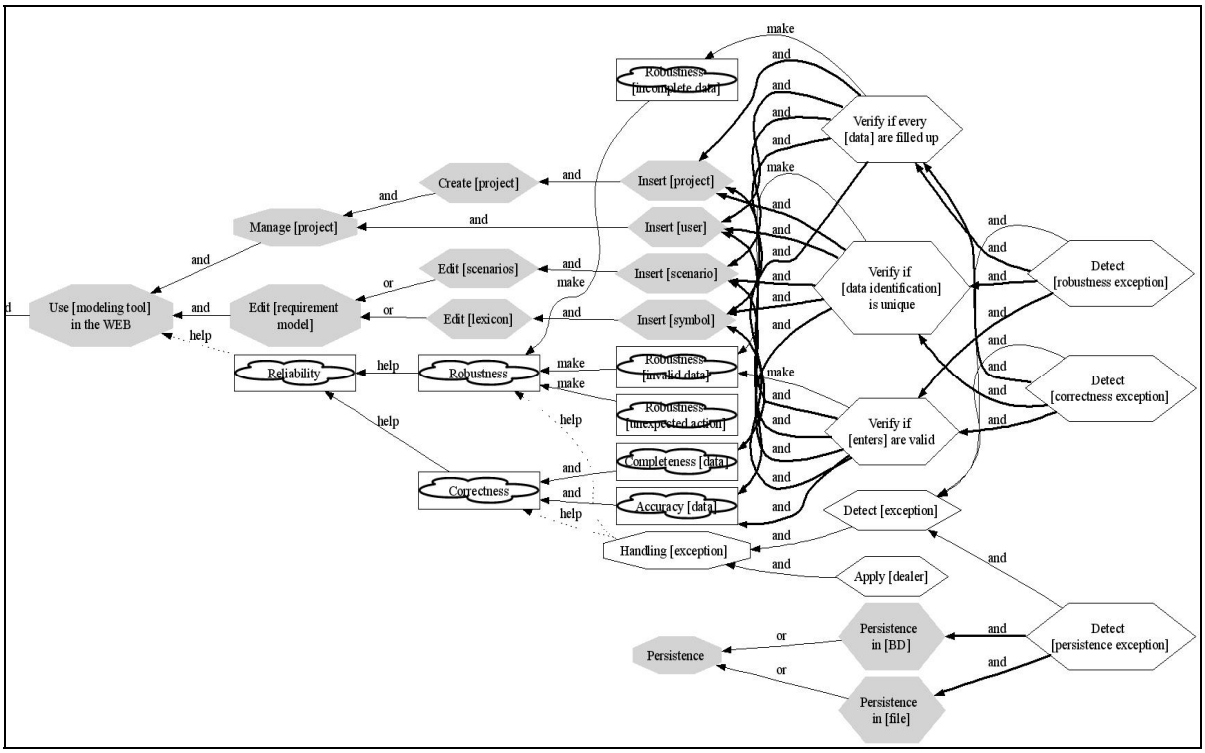


Figura 90. Modelo de confiabilidade após a composição

O mecanismo de visualização é responsável por gerar cada uma das visões anteriores, separando os grupos de requisitos ou juntando somente aqueles que afetam o grupo em foco, ou mesmo mostrando todos os grupos juntos e suas

interações. Na Figura 91, apresentamos a modelagem em AOV-graph do sistema C&L com seus relacionamentos transversais expandidos.

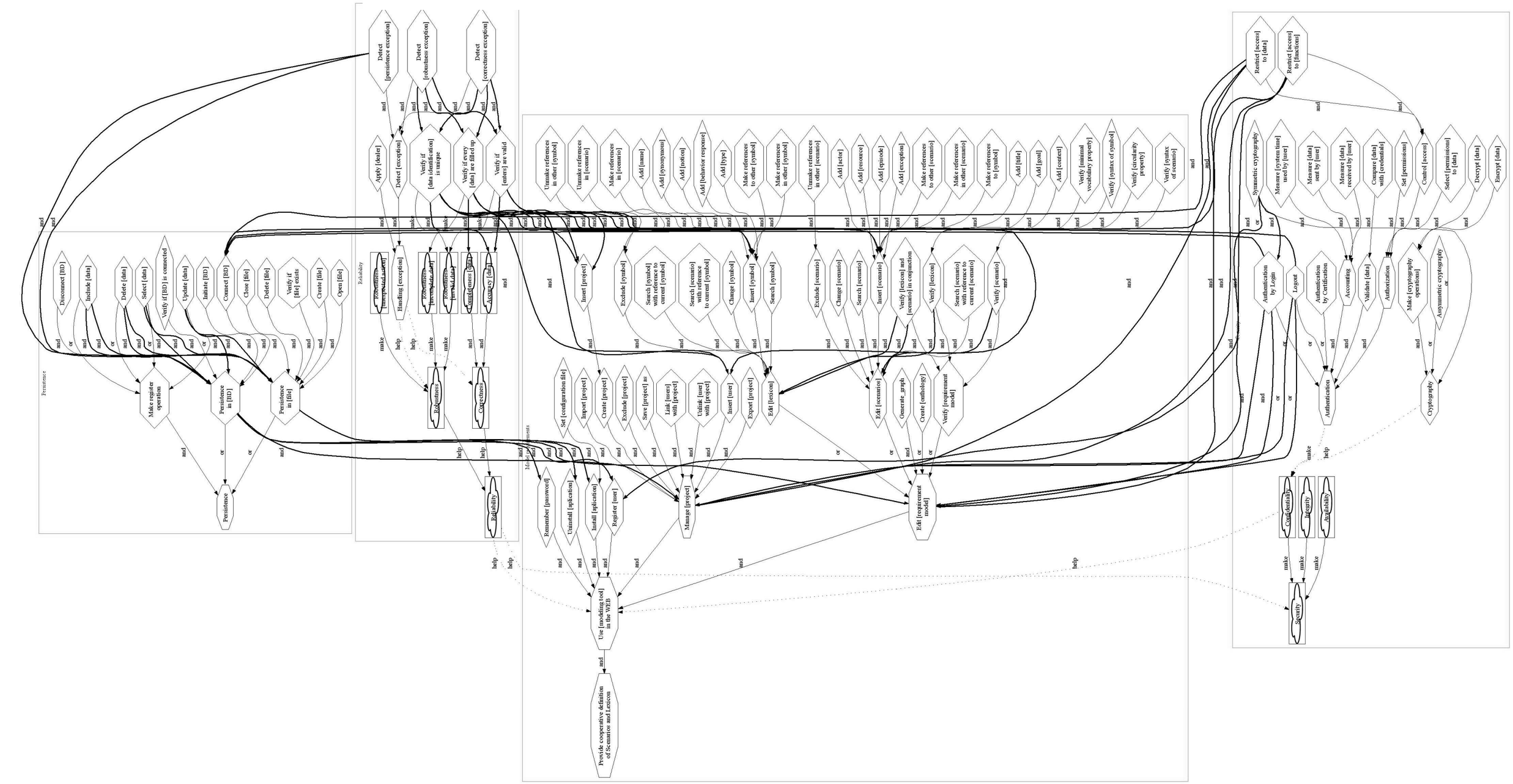


Figura 91. AOV-graph do sistema C&L com os relacionamentos transversais expandidos

O V-graph é uma visão intencional do sistema, i.e., os modelos em V-graph focam a interação entre seus elementos para satisfação de metas, *softmetas* e tarefas, esta é sua decomposição dominante. Entretanto, com a estruturação de cada meta, *softmeta* e tarefa em tipos e tópicos, podemos obter também uma visão de dados (como apresentamos no Capítulo 4, Seção 4.3.3).

Na Figura 92, ilustramos as informações modeladas em AOV-graph na forma de diagrama de classes. Neste diagrama, focamos apenas os objetos relacionados ao modelo CEL. Podemos observar que a maior quantidade de funções está relacionada aos objetos "project", "scenarios" e "symbol". Semanticamente os objetos "modeling tool" e "aplication" dizem respeito ao mesmo elemento, mas foram modelados como objetos distintos porque esta geração se baseia em uma análise léxica dos tópicos.

Os relacionamentos são derivados com base nos relacionamentos existentes entre metas, tarefas e *softmetas* cujos tópicos geram objetos, entretanto não conseguimos identificar o tipo de relacionamento nem a cardinalidade deles. Melhorias nas regras de transformação podem ser feitas para esta identificação. Apesar de não podermos garantir que este modelo é completo e correto, ele representa uma boa visão de quais os principais objetos da modelagem, quais as ações onde eles estão envolvidos e qual a interação entre eles.

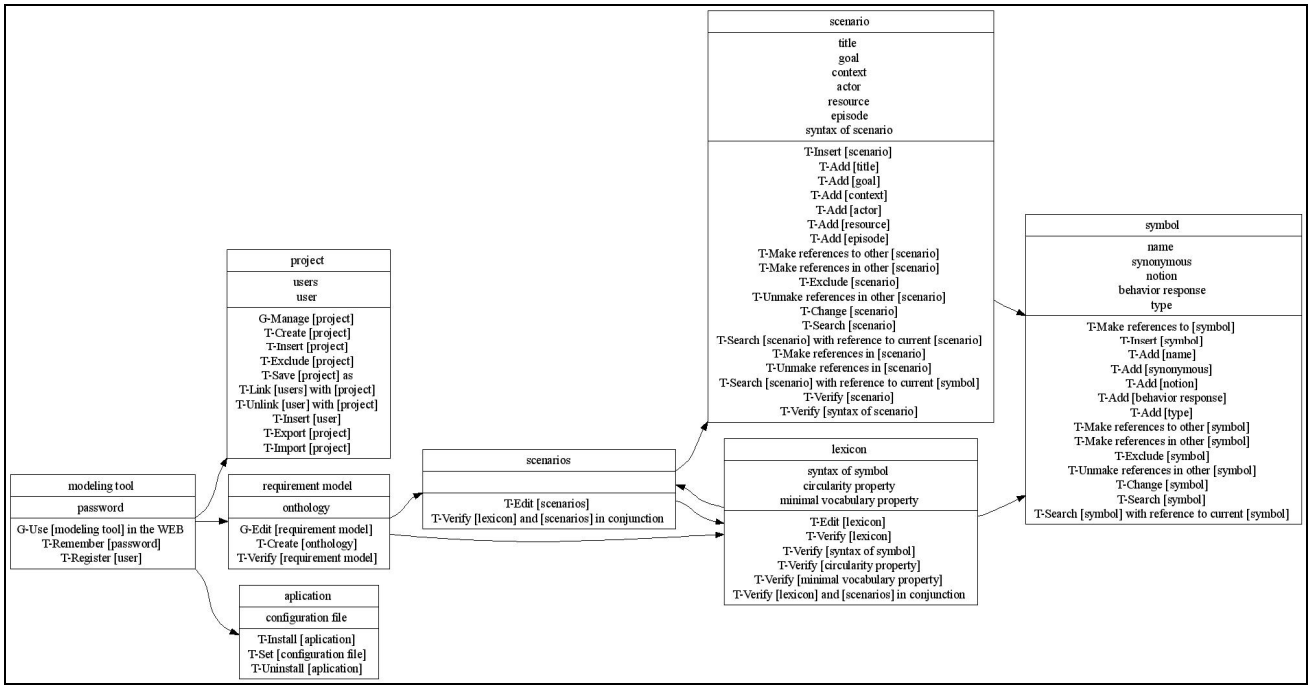


Figura 92. Diagrama de classes do modelo CEL

5.2.2. Análise dos Resultados

A modelagem de C&L inclui 12 *softmetas*, 15 metas e 110 tarefas, veja na Tabela 7. Separamos estes elementos em quatro grandes grupos de requisitos, sendo referentes ao CEL, à segurança, à persistência e à confiabilidade. Dentro de cada um deles ainda separamos em grupos menores tais como "Authentication", "Verify [requirements model]", "Handling [exception]", dentre outros. Esta separação foi realizada por avaliar se cada grupo ou sub-grupo pode existir sem os demais, pode ser reutilizado ou se ele encontra-se repetido em diferentes pontos da modelagem.

Tabela 7. Estatísticas do estudo de caso – C&L

Elementos	Quantidade antes da composição	Quantidade após a composição
Metas	9	9
Tarefas	101	105
Softmetas	12	12
Relacionamentos transversais	6	6
Elos transversais	20+4(expressão regular)=24	56
Correlações	7	7
Contribuições	117	117+56=173
Tipos	52	55
Tópicos	44	47
Atributos	0	3

Ao realizar esta separação tornamos melhor a visibilidade de cada modelo e facilitamos a manipulação deles, possibilitando também que cada grupo de requisitos possa ser modelado por equipes diferentes ou em momentos diferentes. Os modelos de persistência, segurança e confiabilidade podem ser reutilizados em outros sistemas e podem evoluir independentemente um do outro, mesmo depois que os relacionamentos transversais tenham sido definidos.

A definição dos relacionamentos transversais é realizada quando já se tem em vista o que de cada característica é requerido no sistema. Quando os modelos evoluem ou são reutilizados, os relacionamentos transversais contidos neles devem ser redefinidos para o contexto em questão, mas muitas vezes é possível redefinir apenas os *pointcuts*, sendo as *intertype declarations* e *advice* não modificados. Isto significa que é possível que certa característica sempre interaja com outras da mesma maneira, mudando apenas o local da interação, por exemplo, para reutilizar o modelo de persistência é necessário redefinir apenas seu segundo *pointcut*, veja na Seção 5.2.

Neste estudo de caso, definimos 6 relacionamentos transversais, representados no V-graph por 24 elos. Ao processar estes relacionamentos transversais o mecanismo de composição gerou 56 elos. Houve, assim, um crescimento de mais de 130% no número destes elos.

Além de diminuir o número de relacionamentos que o engenheiro de requisitos teria que criar manualmente, a semântica dos relacionamentos transversais torna explícitas quais características têm tendência a se espalhar ou entrelaçar às demais, i.e., o relacionamento transversal não é uma simples interação entre metas, *softmetas* e tarefas. Cada relacionamento transversal representa uma matriz de interações, demonstrando parte da rastreabilidade entre os elementos. Esta parcela de rastreabilidade pode representar elementos críticos no sistema, pois inclui aqueles que afetam ou são afetados por muitas características, ou aqueles que possuem alto acoplamento.

Tabela 8. Matriz de rastreabilidade - relacionamento transversal do modelo de confiabilidade

	Source	***Advice***			***Intertype declaration***		
	Detect [exception]	**Verify if [data identification] is unique**	**Verify if [enters] are valid**	**Verify if every [data] are filled up**	**Detect [robustness exception]**	**Detect [correctness exception]**	**Detect [persistence exception]**
Detect [exception]					And	And	And
Verify if [data identification] is unique	Cross (exception = Invalid data)				And	And	
Verify if [enters] are valid	Cross (exception = Invalid data)				And	And	
Verify if every [data] are filled up	Cross (exception = Incomplete data)				And	And	
Persistence in [BD]	Cross						And
Persistence in [file]	Cross						And
Insert [project]	Cross	And	And	And			
Insert [user]	Cross	And	And	And			
Insert [scenario]	Cross	And	And	And			
Insert [symbol]	Cross	And	And	And			

Na Tabela 8, ilustramos um exemplo de matriz de rastreabilidade com as informações do relacionamento transversal do modelo de confiabilidade. Nesta matriz, podemos observar qual a interação dos elementos da coluna para os elementos das linhas. O rótulo "cross" indica que "Detect [exception]" afeta todos os *pointcuts* indicados nas linhas, e nos três primeiros adiciona um atributo denominado "exception". Os rótulos "and" indicam relacionamentos de contribuição, ou seja, os elementos das colunas são necessários para satisfazer os elementos das linhas. A relação coluna→linha rotuladas com "cross" representam os relacionamentos transversais resumidos, enquanto o rótulo "and" representa os

relacionamentos transversais expandidos. Por exemplo: de maneira resumida, “Detect [exception]” afeta (cross) “Insert [symbol]”; de maneira expandida, “Verify if [data identification] is unique”, “Verify if [enters] are valid” e “Verify if every [data] are filled up” contribuem para “Insert [symbol]”.

5.3. Sistema para Divulgação de Eventos – Unical

Unical é um sistema para divulgação de eventos sociais e acadêmicos na Universidade da Califórnia (UCI). Devido haver uma grande quantidade de eventos acontecendo na universidade e muitas vezes as pessoas não ficarem sabendo de um evento de interesse a tempo, foi desenvolvido o sistema Unical. Sua meta é fazer com que todos o adotem para divulgar e compartilhar eventos, registrando-os em suas agendas de maneira que sejam lembrados sobre eles no momento escolhido. O documento de requisitos deste sistema encontra-se disponível em (Unical, 2005), e é utilizado em estudos de caso desta mesma universidade. O Anexo A contém uma cópia deste documento de requisitos e foi utilizado como fonte de informação para modelagem do Unical nesta tese.

5.3.1. Modelagem

No documento de requisitos do Unical contabilizamos 140 sentenças classificadas como requisitos funcionais e menção a 16 requisitos classificados como não funcionais. Estes últimos são apenas citados com definições abstratas do que se espera sobre eles. Porém, a maioria dos requisitos funcionais traz em sua descrição, restrições referentes a muitos destes RNFs, indicando o entrelaçamento e espalhamento destas características.

Para modelar o Unical separamos cada requisito não funcional daqueles específicos da divulgação de eventos. Na Figura 93, mostramos uma visão geral dos RNFs envolvidos na modelagem e suas correlações com o modelo do Unical. Nesta visão podemos observar contribuições e conflitos existentes entre RNFs. Por exemplo, “Profitable” e “Timeleness” são essenciais para atingir a meta principal do Unical, mas restringem todos os outros RNFs. Esta visão é essencial para a modelagem e tomada de decisões porque explicita a necessidade de avaliar

o quanto de cada requisito é suficiente para atingir a meta principal do sistema, sem o risco de torná-lo inviável.

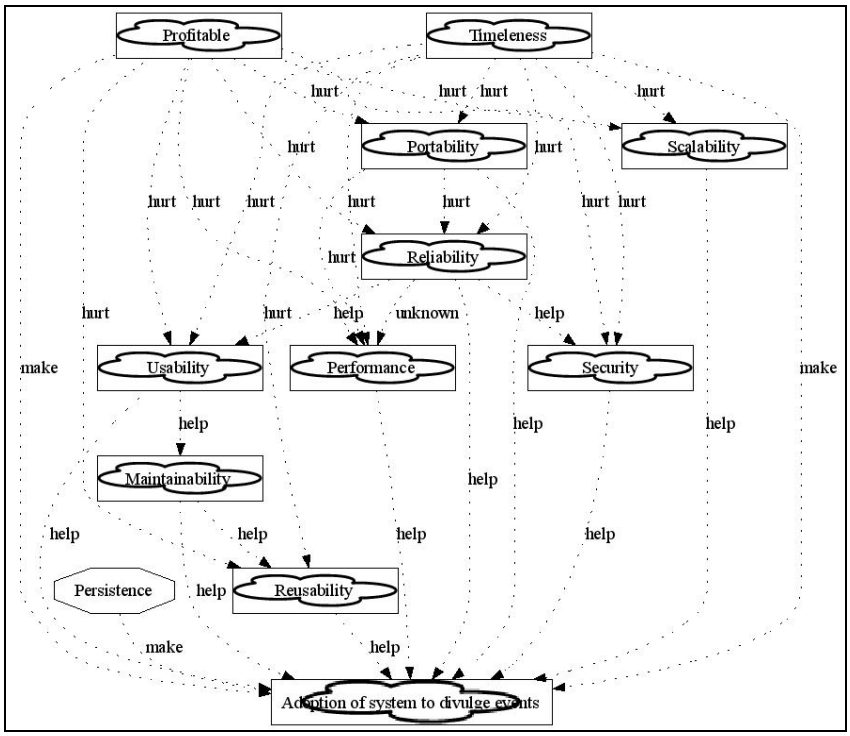


Figura 93. Visão geral do Unical e seus RNFs

Utilizando a abordagem tradicional (V-graph) estes RNFs e suas operacionalizações compõem um único grafo (veja a Figura 94). Assim como no estudo de caso anterior, este modelo apresenta uma grande quantidade de elementos e relacionamentos, tornando sua modelagem, análise, modificação e reutilização difíceis.

Em nossa modelagem utilizando AOV-graph, cada *softmeta* e meta na Figura 93 representa um modelo separado. Além disto, utilizamos relacionamentos transversais para representar a interação entre eles quando há entrelaçamento e espalhamento de seus elementos (veja a Figura 95).

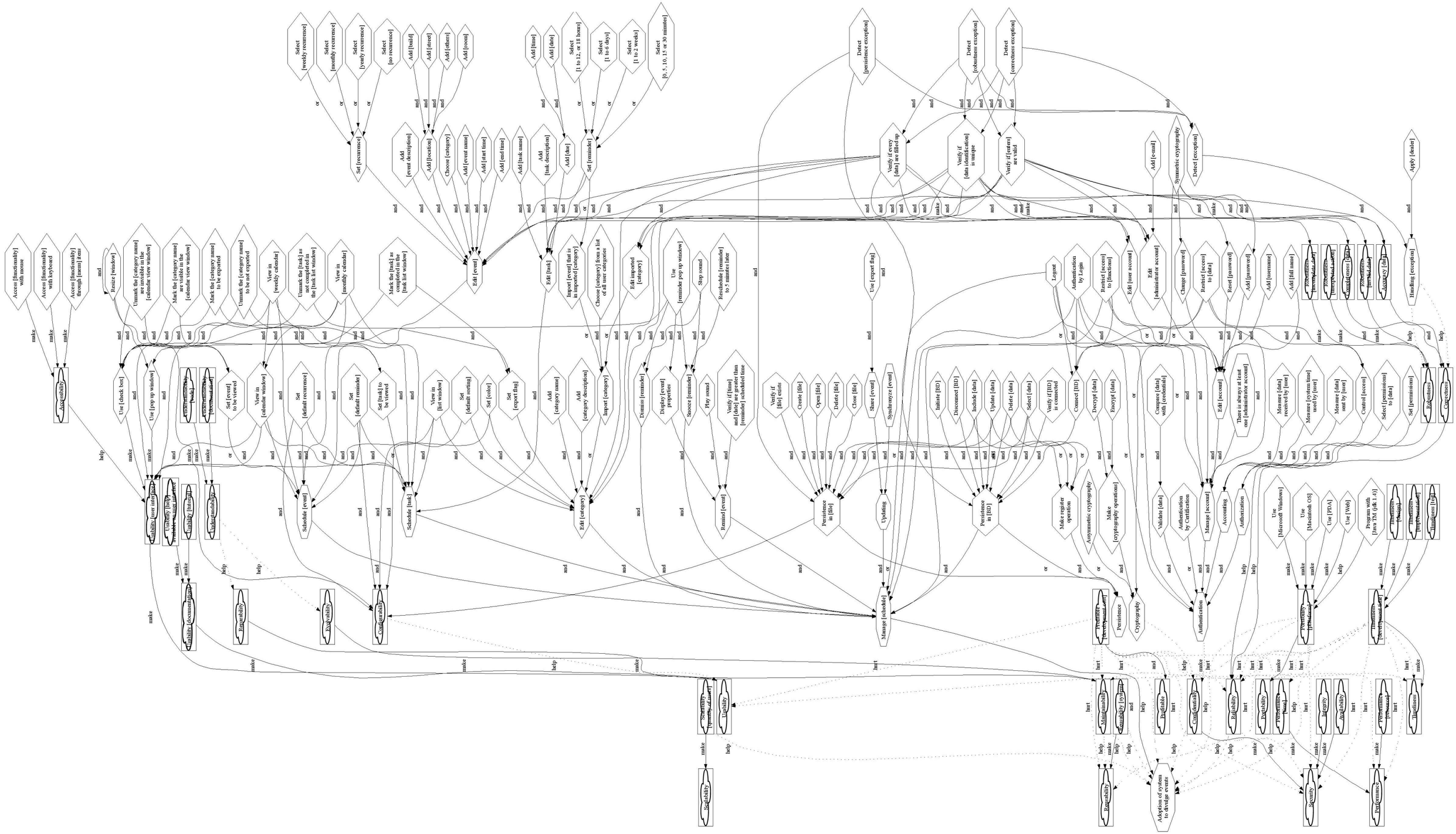


Figura 94. V-graph do sistema Unical

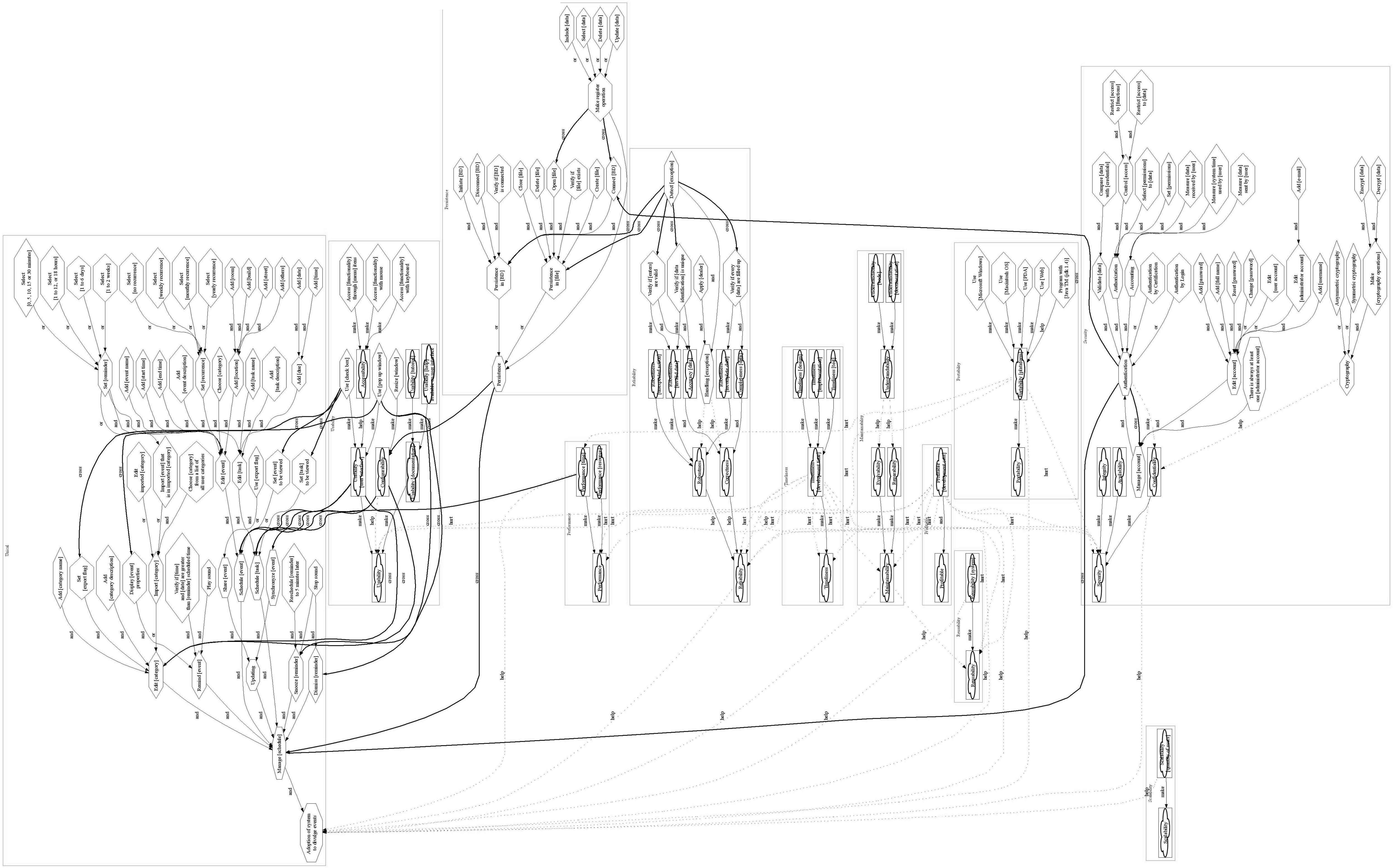


Figura 95. AOV-graph do sistema Unical

Na Figura 95, ilustramos a modelagem do sistema Unical em AOV-graph. Como mencionamos anteriormente, a modelagem consiste dos modelos Unical, segurança, confiabilidade, persistência, usabilidade, performance, portabilidade, escalabilidade, manutenibilidade, reusabilidade e os modelos referentes ao tempo (timeleness) e custo (profitable) de desenvolvimento.

Os modelos referentes à portabilidade e escalabilidade não apresentam suas operacionalizações porque o documento de requisitos não descreve nenhum detalhe sobre como realizar estes RNFs. Os modelos referentes à manutenibilidade, à reusabilidade, ao tempo e custo de desenvolvimento não representam funcionalidades do sistema Unical. Eles representam características de projeto, que também são transversais e representam uma maneira de decomposição diferente dos outros RNFs. Quanto aos modelos relacionados à segurança, confiabilidade e persistência, nós reutilizamos do estudo de caso anterior (C&L) e realizamos algumas modificações para atender novos requisitos do Unical. A seguir, detalhamos os modelos referentes ao Unical, usabilidade e performance, e como realizamos a reutilização dos modelos de segurança, confiabilidade e persistência.

Divulgação de Eventos – Unical

Na Figura 96, ilustramos o AOV-graph referente aos requisitos específicos de divulgação de eventos. Neste modelo não definimos nenhum relacionamento transversal porque consideramos que os elementos que se espalham (“Stop sound” e “Set [reminder]”) não representam maior complexidade para o modelo.

Na Figura 97, ilustramos o MER gerado com base nesta parte do modelo AOV-graph do sistema Unical. Podemos observar as principais entidades (retângulos) e seus atributos (elipses), bem como os relacionamentos entre entidades (elos direcionados), gerados a partir dos relacionamentos de contribuição de AOV-graph.

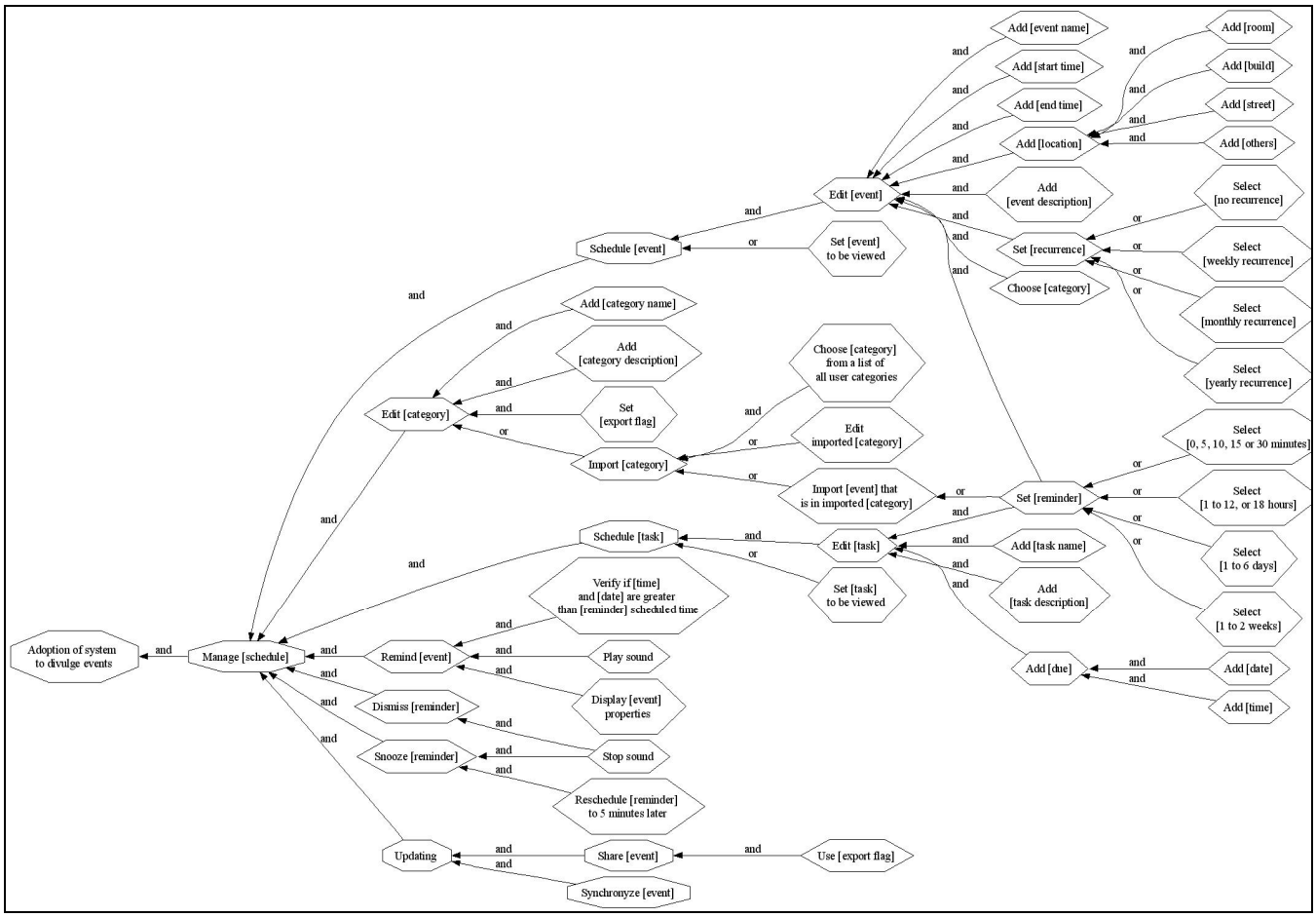


Figura 96. Modelo Unical – divulgação de eventos

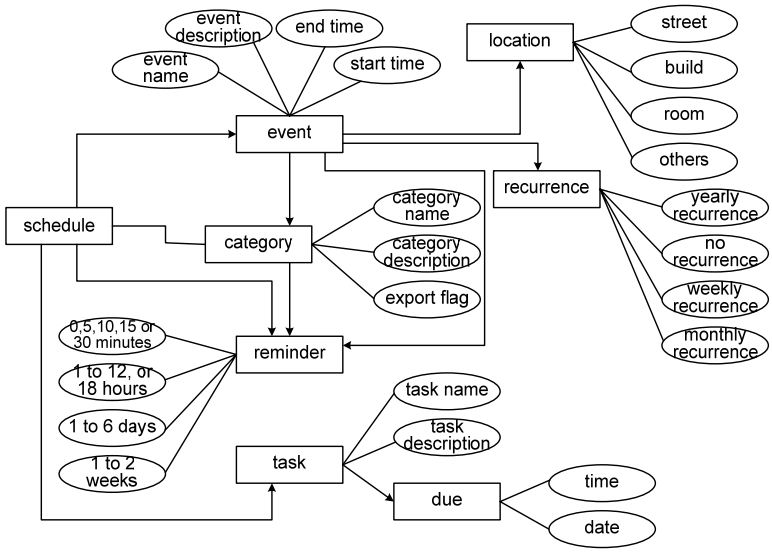


Figura 97. MER para o modelo Unical

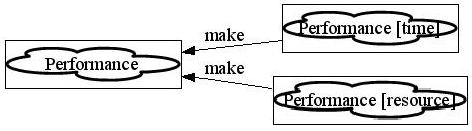
A seguir apresentamos como os modelos de performance, usabilidade e segurança afetam o modelo Unical.

Performance

O documento de requisitos do Unical apresenta explicitamente a preocupação com a característica performance no sentido tempo de resposta. Esta característica é descrita somente pela sentença a seguir:

> "Performance is crucial to UniCal – if the system is too slow, customers may revert to sending e-mails and using word of mouth. Synchronizing events with UniCal should take no more than one second for every fifty events. All other operations must be performed nearly instantaneously."

Entretanto, esta sentença restringe cada um dos requisitos funcionais, i.e., todas as tarefas de Unical estão relacionadas à restrição de performance. Mesmo não havendo tarefas que indiquem como este tempo de resposta é atingido, decidimos modelá-lo como um novo tipo de atributo denominado "performance" que tem os valores "instantaneously" e "one second for 50 events". Na Figura 98a, ilustramos o modelo de performance e na Figura 98b apresentamos a descrição de seu relacionamento transversal.




```
Crosscutting {
  source: Performance [time]
  pointcut PC12: include( .*) and
       not include ( Synchronize [event] )
  pointcut PC13: include (Synchronize [event])
  intertype declaration attribute: PC12 {
       performance = instantaneously }
  intertype declaration attribute: PC13 {
       performance=one second for 50 events }
}
```

(a) (b)

Figura 98. a) Modelo de performance e b) relacionamento transversal do modelo de performance

O atributo adicionado por este relacionamento transversal não é ilustrado no AOV-graph, mas pode ser analisado quando outras visões deste modelo são geradas. Por exemplo, na Figura 99, ilustramos cenários do Unical em que esta restrição de performance é representada.

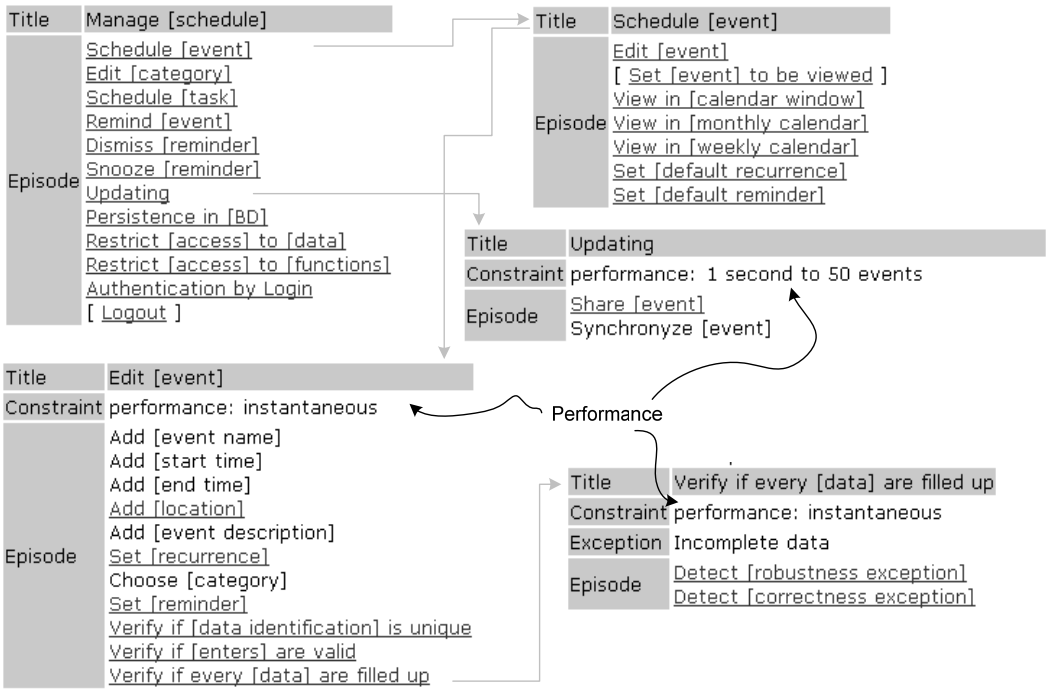


Figura 99. Visão de cenários do modelo Unical

Usabilidade

Na descrição de RFs do documento de requisitos do Unical são mencionados vários detalhes da interface com o usuário. Com base nestes detalhes, modelamos algumas características genéricas relacionadas à usabilidade. Na Figura 100, apresentamos o modelo de usabilidade utilizado neste estudo de caso, e na Figura 101 apresentamos sua interação com o modelo Unical (os elementos em cinza pertencem ao modelo Unical).

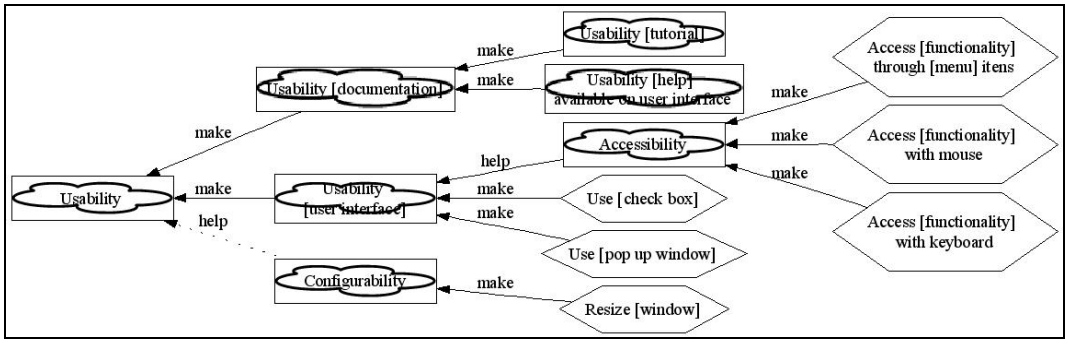


Figura 100. Modelo de usabilidade

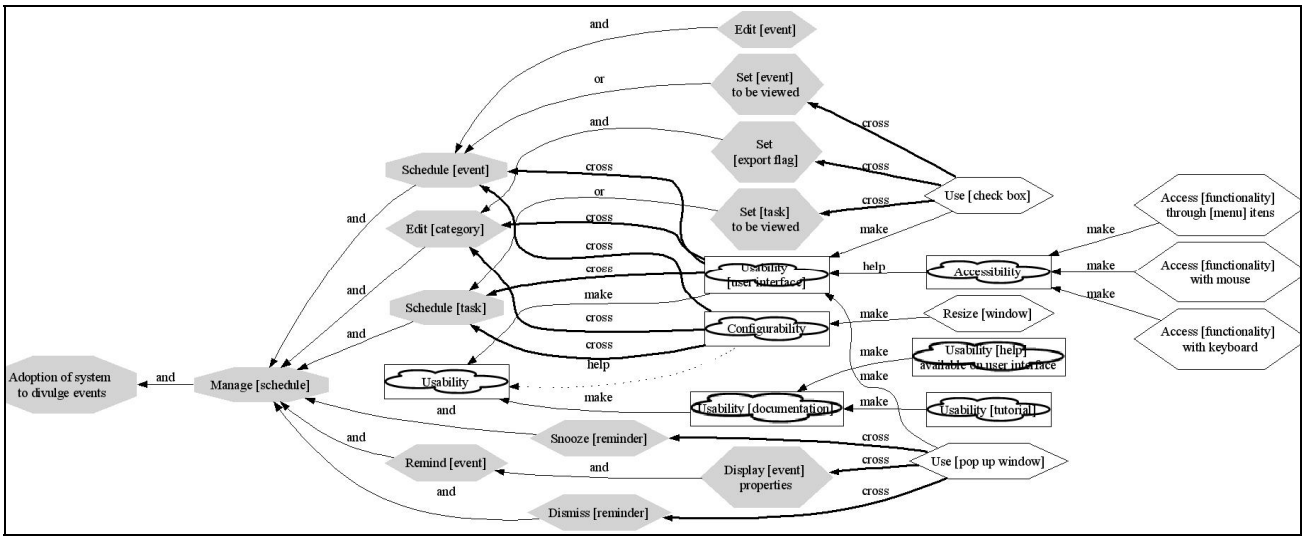


Figura 101. Interação do modelo de usabilidade com o modelo Unical

Novas tarefas e metas surgem com a integração destes dois modelos. Estes novos elementos nem fazem parte do modelo Unical, nem do modelo de usabilidade isoladamente. Na Figura 102, apresentamos a descrição dos relacionamentos transversais existentes entre o modelo de usabilidade e o Unical.

Com o primeiro relacionamento transversal é incluída a nova tarefa "Use [reminder pop up window]" em "Use [pop up window]", "Display [event] properties", "Snooze [reminder]" e "Dismiss [reminder]". Da mesma maneira, no segundo e terceiro relacionamento transversal, as tarefas indicadas nos *intertype declarations* são incluídas no ponto de origem do relacionamento ("Use [check box]" e Usability [user interface]) e nos seus respectivos *pointcuts*. No quarto relacionamento transversal, temos um exemplo de *pointcut* definido por expressão regular. Neste caso todas as tarefas que têm a palavra "window" é um *pointcut*, porém o operador "not" é utilizado para excluir a tarefa "Resize [window]" deste *pointcut*, i.e., o *advice* que atinge o *pointcut* PC11 inclui a tarefa "Resize [window]" em todas as tarefas que contenham a palavra "window" com exceção dela mesma.

1	1. Crosscutting { 2. source: Use [pop up window] 3. pointcut PC1: include(Display [event] properties) and 4. include(Snooze [reminder]) and include(Dismiss [reminder]) 5. intertype declaration element: PC1 { Use [reminder pop up window] } 6. }
2	1. Crosscutting { 2. source: Use [check box] 3. pointcut PC2: include(Set [event] to be viewed) 4. pointcut PC3: include(Set [task] to be viewed) 5. pointcut PC4: include(Set [export flag]) 6. intertype declaration element: PC3 { Mark the [category name] to be exported ; 7. Unmark the [category name] to be not exported } 8. intertype declaration element: PC2 { Mark the [category name] are visible in the 9. [calendar view window] ; Unmark the [category name] are invisible in the 10. [calendar view window] } 11. intertype declaration element: PC4 { Mark the [task] as completed in the [task 12. list window]; Unmark the [task] as not completed in the [task list window] } 13. }

3	1. Crosscutting {
	2. source: Usability [user interface]
	3. pointcut PC5: include(Schedule [event])
	4. pointcut PC6: include(Edit [category])
	5. pointcut PC7: include(Schedule [task])
	6. intertype declaration element: PC5 and PC7 { View in [calendar window] {
	7. View in [monthly calendar] ; View in [weekly calendar] } }
	8. intertype declaration element: PC6 and PC7 { View in [list window] }
	9. }

4	1. Crosscutting { 2. source: Configurability 3. pointcut PC8: include(Schedule [event]) 4. pointcut PC9: include(Edit [category]) 5. pointcut PC10: include(Schedule [task]) 6. pointcut PC11: include(.*window={ (Use [pop up window]) 7. (Use [reminder pop up window]) (Mark the [category name] are visible in the 8. [calendar view window]) (Unmark the [category name] are invisible in the 9. [calendar view window]) (Mark the [task] as completed in the [task list 10. window]) (Unmark the [task] as not completed in the [task list window]) 11. (View in [calendar window]) (View in [list window]) (Resize [window]) }) 12. and not include (Resize [window]) 13. advice around: PC11 { Resize [window] } 14. intertype declaration element: PC9 { Set [color] ; Set [default sorting] } 15. intertype declaration element: PC8 and PC10 { Set [default recurrence] ; 16. Set [default reminder] }}

Figura 102. Relacionamentos transversais do modelo de Usabilidade

Na Figura 103, apresentamos o resultado da composição, com foco no modelo Unical. Os elementos em cinza são aqueles que surgem com o interesse em trazer usabilidade para o sistema de divulgação de eventos. Por outro lado, na Figura 104, ilustramos as mudanças no modelo de usabilidade, decorrentes de sua instanciação para o contexto do Unical.

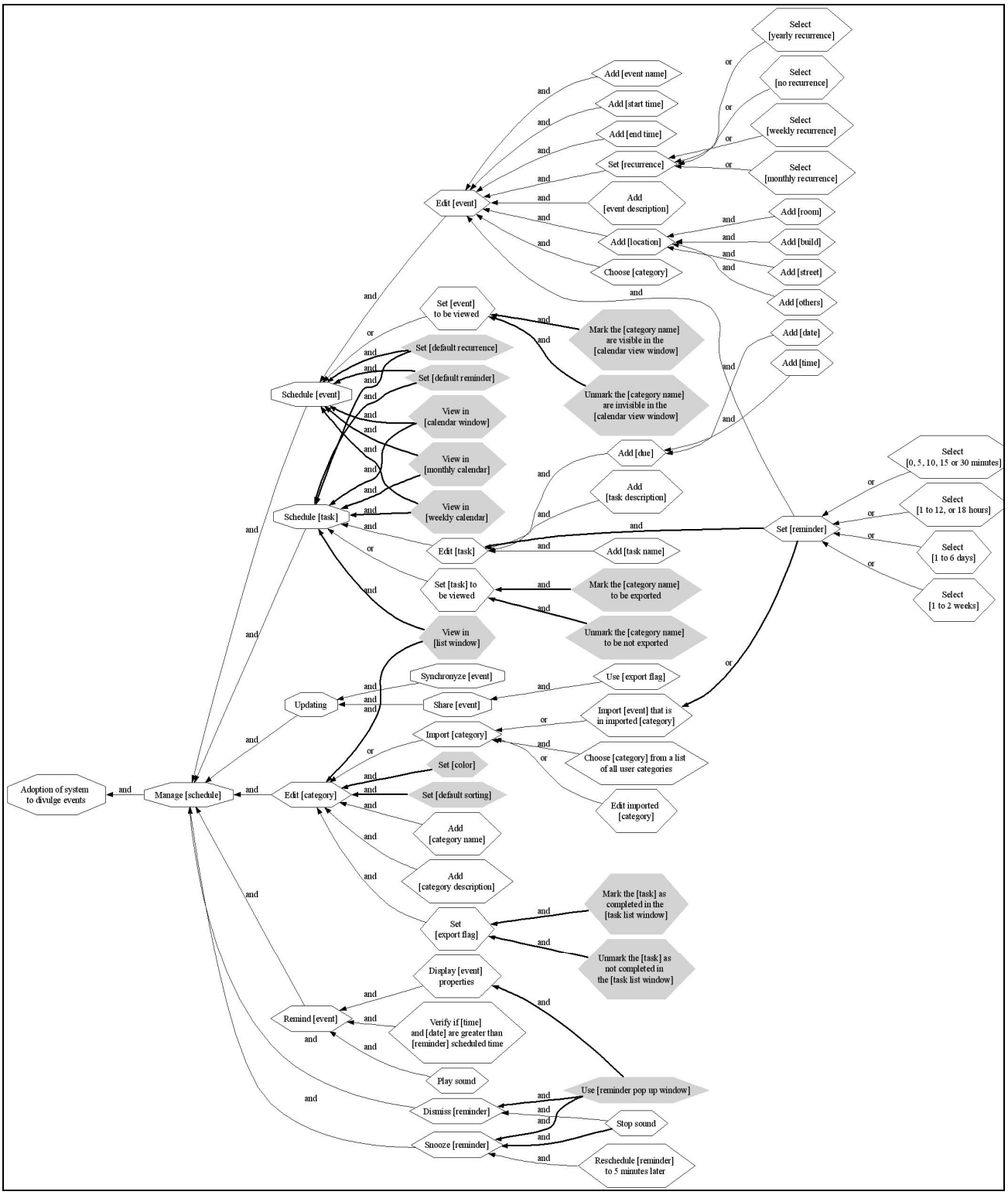


Figura 103. Modelo Unical após a composição

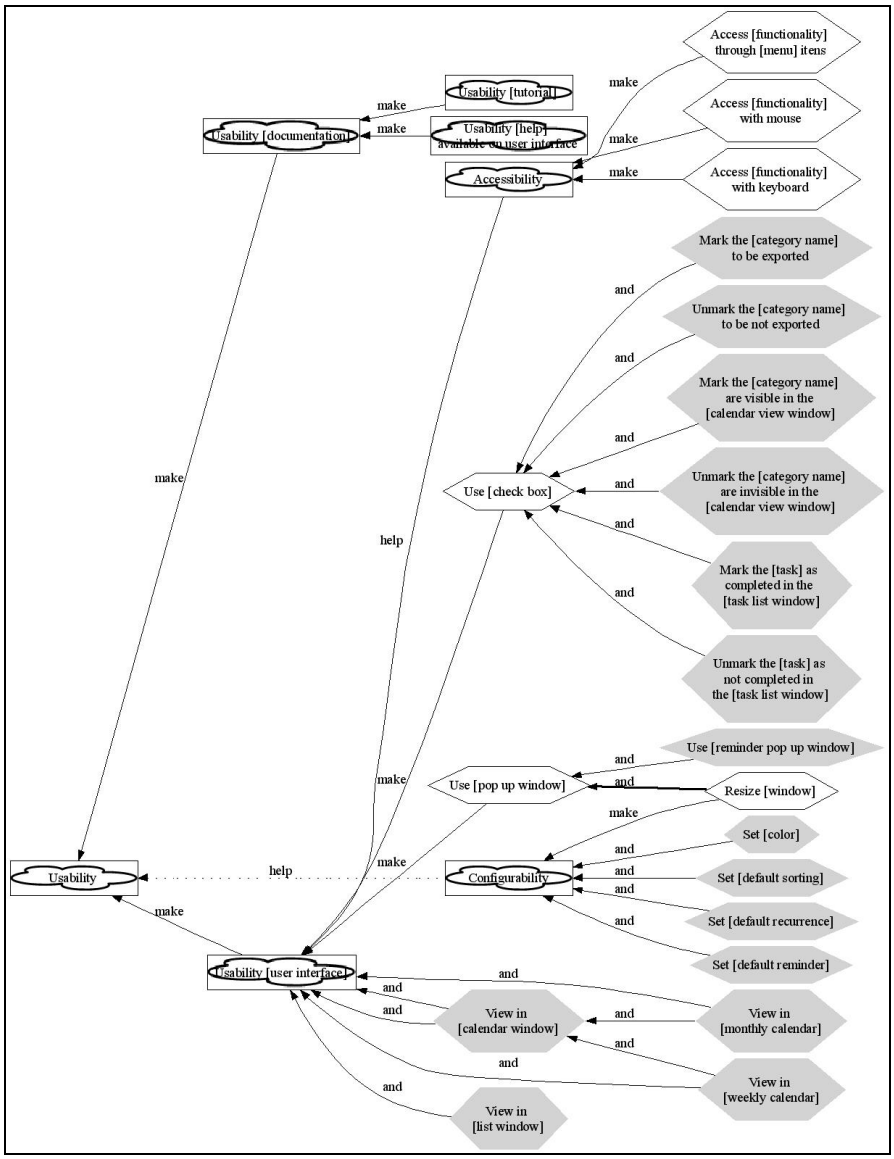


Figura 104. Modelo de usabilidade após a composição

Segurança, Persistência e Confiabilidade

Na Figura 105, ilustramos o modelo de segurança para o sistema Unical. Em cinza realçamos as modificações realizadas no modelo de segurança reutilizado do estudo de caso anterior. Neste modelo incluímos tarefas referentes à gerência das contas dos usuários “Manage [account]” (em cinza).

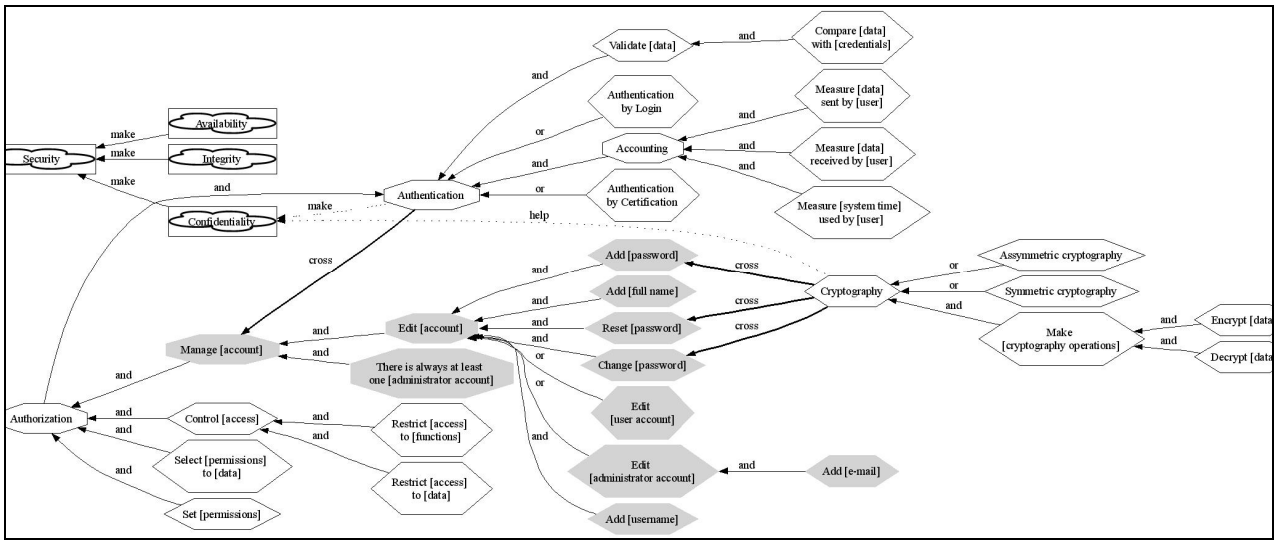


Figura 105. Modelo de segurança

Além disto, modificamos os dois relacionamentos transversais existentes neste modelo para refletir os elementos afetados do Unical. Na Figura 106, ilustramos a descrição destes relacionamentos instanciados para o Unical. As modificações em relação ao estudo de caso anterior estão em cinza.

1	1. Crosscutting{ 2. source: Authentication 3. *pointcut* PC3: include(Manage [account]) and include(Manage [schedule]) 4. and include(Connect [BD] 5. *advice* around: PC3 { Authentication by login ; Restrict [access] to [data]; 6. Restrict [access] to [functions] } 7. *intertype declaration* element: PC3 { Logout } 8. *intertype declaration* attribute: PC3 { constraint = confidential data } 9. }

2	1. Crosscutting {
	2. source: Cryptography
	3. *pointcut* PC4: include(.*password={ Add [password]; Reset [password]; Change
	4. [password] })
	5. *advice* around: PC4 {Symmetric cryptography }
	}

Figura 106. Relacionamentos transversais do modelo de segurança para o Unical

A separação do modelo de segurança permitiu-nos a fácil identificação de seus elementos, evolução e reutilização em dois estudos de caso. Como mostrado na Figura 106, parte dos relacionamentos transversais, também, é reaproveitada; modificamos apenas seus *pointcuts*. Na Figura 107, ilustramos as modificações realizadas nos relacionamentos transversais dos modelos de persistência e confiabilidade (em cinza).

Persistência		
1	1.	Crosscutting {
	2.	source: Make register operation
	3.	pointcut PC5: include(Connect [BD]) and include(Open [file])
	4.	advice after: PC5 {Include [data]; Select [data]; Delete [data]; Update [data] }
	5.	}
2	1.	Crosscutting {
	2.	source: Persistence
	3.	pointcut PC6: include(Manage [schedule])
	4.	pointcut PC7: include(Configurability)
	5.	advice around: PC6 {Persistence in [BD] }
	6.	advice around: PC7 {Persistence in [file] }
	7.	}
Confiabilidade		
3	1.	Crosscutting {
	2.	source: Detect [exception]
	3.	pointcut PC8: include(Verify if [data identification] is unique) and
	4.	include(Verify if [enters] are valid)
	5.	pointcut PC9: include(Verify if every [data] are filled up)
	6.	pointcut PC10: include(Persistence in [BD]) and include(Persistence in [file])
	7.	pointcut PC11: include(Edit.*={ (Edit [event]) (Edit [category]) (Edit imported
	8.	[category]) (Edit [task]) (Edit [user account]) (Edit [administrator account]) })
	9.	advice around: PC11 { Verify if [data identification] is unique ;
	10.	Verify if [enters] are valid ; Verify if every [data] are filled up }
	11.	intertype declaration element: PC10 { Detect [persistence exception] }
	12.	intertype declaration attribute: PC8 { exception = Invalid data }
	13.	intertype declaration attribute: PC9 { exception = Incomplete data }
	14.	intertype declaration element: PC8 and PC9 { Detect [robustness exception] ;
	15.	Detect [correctness exception] }
	16.	}

Figura 107. Relacionamentos transversais do modelo de persistência e confiabilidade

Com a definição de novos tipos (atributos) utilizando *intertype declarations* é possível modelar características transversais além de metas, *softmetas* e tarefas, tais como, interessados por cada parte da modelagem ou custo e tempo de desenvolvimento. A seguir, apresentamos como modelar e visualizar elementos para os quais o modelo de componentes V-graph não dá suporte (não fazem parte da decomposição dominante do V-graph).

Modelagem de interessados

Adicionamos este novo modelo à modelagem do sistema Unical, que consiste em apenas uma tarefa denominada "Model [stakeholders]". Esta tarefa está relacionada aos elementos dos outros modelos por acrescentar o atributo "actor". Desta forma modelamos os interessados em cada grupo de requisitos. Na Figura 108, ilustramos seu relacionamento transversal.

```
1.  Crosscutting {
2.    Source: Model [stakeholder]
3.    Pointcut PC12: include (Manage [schedule]) and include(Usability) and
4.          include (Reliability) and include (Confidentiality) and include (Portability)
5.          and include(Scalability) and include(Performance)
6.    Pointcut PC13: include(Manage [account])
7.    Pointcut PC14: include(Profitable) and include(Timeleness)
8.    Pointcut PC15: include (Maintanability) and include(Reusability)
9.    Intertype declaration attribute: PC12 and PC13 {actor = user}
10.   Intertype declaration attribute: PC13 {actor = administrator user}
11.   Intertype declaration attribute: PC14 and PC15 {actor = project manager }
12.   Intertype declaration attribute: PC15 {actor = development team}}
```

Figura 108. Relacionamento transversal do modelo de cenários

Com este relacionamento transversal inserimos os atores interessados em cada grupo de requisitos. A composição espalha estes atributos na modelagem em AOV-graph. Como o AOV-graph não possui elementos gráficos para sua visualização então geramos uma outra visão para mostrá-los, veja a Figura 109.

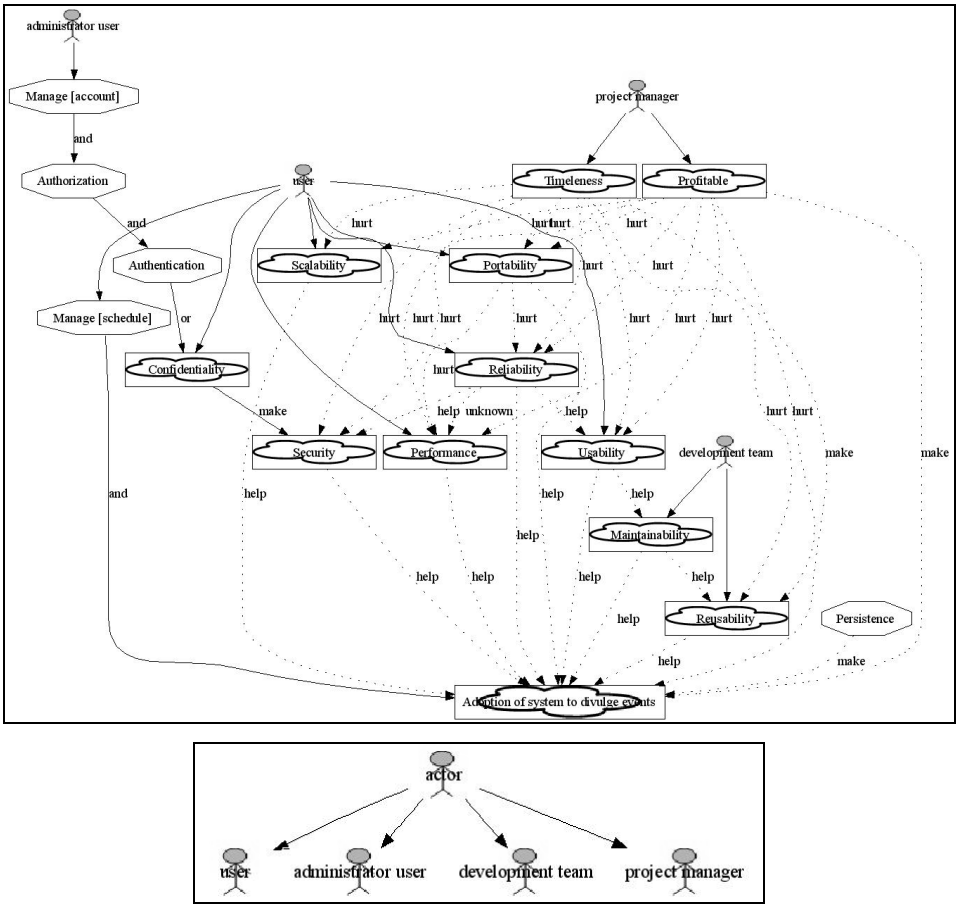


Figura 109. Visão dos interessados do sistema

Assim, a utilização de *intertype declarations* associada ao mecanismo de visualização permite a separação de características transversais, tanto utilizando os elementos da decomposição dominante, quanto outros elementos.

5.3.2. Análise dos Resultados

A modelagem do Unical inclui 41 *softmetas*, 15 metas e 123 tarefas, veja Tabela 9. Separamos estes elementos em doze grupos de requisitos, sendo referentes à divulgação de eventos, usabilidade, performance, persistência, segurança, confiabilidade, manutenibilidade, reusabilidade, portabilidade, escalabilidade e referentes ao custo e tempo de desenvolvimento.

Tabela 9. Estatísticas do estudo de caso – Unical

Elementos	Quantidade antes da composição	Quantidade após a composição
Metas	15	15
Tarefas	123	138
Softmetas	41	41
Relacionamentos transversais	10	10
Elos transversais	40	91
Correlações	35	35
Contribuições	157	157+91=248
Tipos	57	60
Tópicos	79	89
Atributos	0	5

Com a composição 15 novas tarefas foram adicionadas. Além disto, tivemos a adição de 51 elos transversais, representando um crescimento de 127% em relação à quantidade de elos transversais antes da composição. Este crescimento de elos indica:

- Espalhamento de alguns elementos, pois eles estão envolvidos na realização de muitas, metas, *softmetas* e tarefas. Por exemplo, as tarefas "Verify if [data identification] is unique", "Verify if [enters] are valid" e "Verify if every [data] are filled up" contribuem para "Edit [event]", "Edit [category]", "Edit imported [category]", "Edit [task]", "Edit [user account]" e "Edit [administrator account]", com o intuito de prover robustez ao sistema Unical.
- Entrelaçamento de algumas características, pois para realização de cada uma delas é necessária a contribuição de metas, *softmetas* e tarefas específicas de outras características. Por exemplo, para dar suporte a configurabilidade, a persistência em arquivo é necessária para registrar localmente as opções de configuração de cada usuário.

Se considerarmos uma única visão com os doze modelos juntos (mostrado na Figura 110), este espalhamento e entrelaçamento causam dificuldades à

visualização, modelagem e análise dos modelos. Além disto, se nenhum mecanismo de visualização é utilizado, é árduo o retrabalho em escrever e atualizar estes elementos que se repetem em várias visões parciais. Assim, o mecanismo de visualização é essencial para:

- Oferecer visões mais apropriadas para cada tarefa do processo de definição de requisitos, ou para cada interessado no sistema. Por exemplo a visão ilustrada na Figura 96 concentra boa parte das funcionalidades de interesse do usuário final do sistema Unical;
- Oferecer visões parciais e, desta forma, diminuir a complexidade do modelo e permitir melhor definição de cada parte do modelo, tais como cada uma das visões ilustradas na Figura 100 e Figura 105, o modelo de usabilidade e o modelo de segurança, respectivamente;
- Diminuir o retrabalho em escrever e atualizar elementos que estão repetidos em várias visões parciais. Por exemplo, a Figura 103 e a Figura 104 contém os mesmos novos elementos criados pelo relacionamento transversal do modelo de usabilidade;
- Oferecer "visões transversais", visões que mostram as informações do modelo seguindo uma outra forma de decomposição. Por exemplo, os diagramas de classes, MER e cenários criados com as informações modeladas em AOV-graph.

Na Figura 110, ilustramos o resultado da composição dos doze modelos que compõem a modelagem do sistema Unical. Neste modelo tanto quanto no modelo ilustrado na Figura 94 (página 133), é difícil analisar o emaranhado de relacionamentos existentes. Entretanto, na Figura 110 está explícito a que sub-modelo cada elemento pertence (ou está mais associado) e há maior potencial para extrair visões.

Neste estudo de caso, ilustramos exemplos de reuso quando utilizamos o mesmo modelo de segurança, persistência e confiabilidade desenvolvido para o C&L. Neste caso modificamos apenas os *pointcuts* de seus relacionamentos transversais, como mostrado na Figura 106 e Figura 107 (páginas 143 e 144, respectivamente). Desta forma, a separação e o uso do relacionamento transversal facilitam o reuso, pois a composição dos dois modelos fica centralizada no relacionamento transversal.

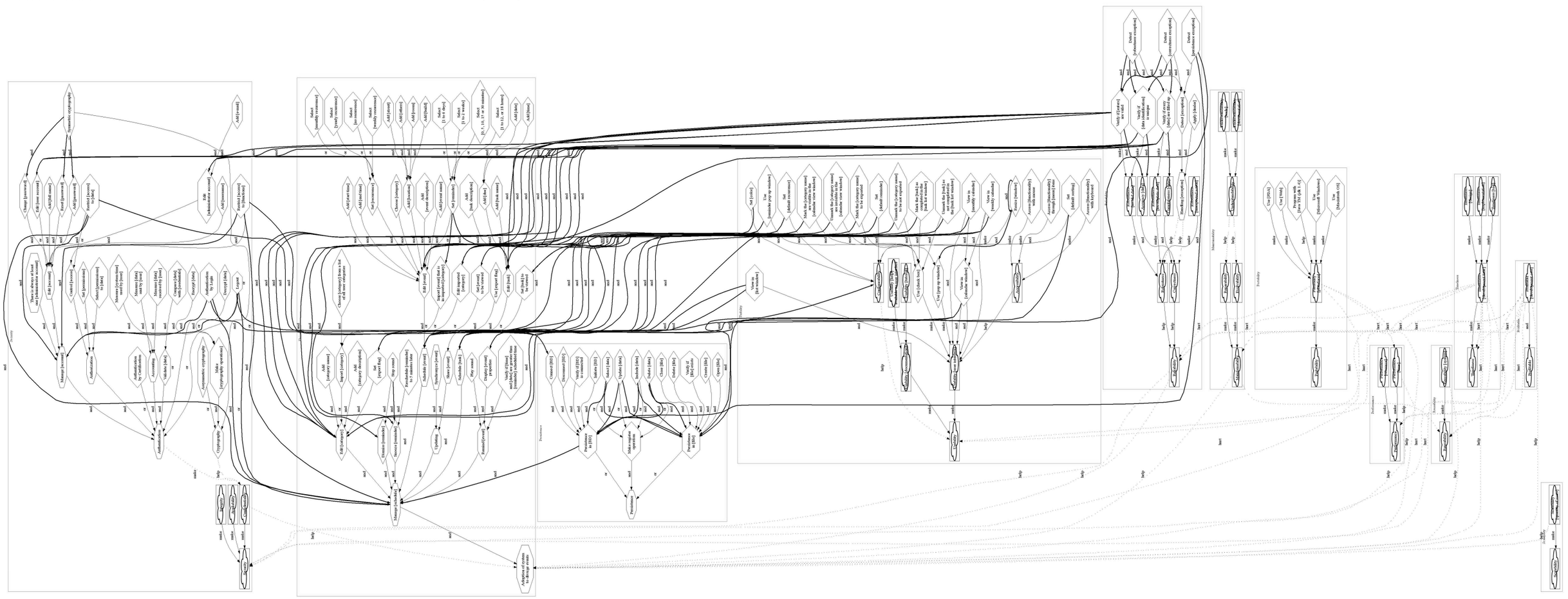


Figura 110. AOV-graph do sistema Unical com os relacionamentos transversais expandidos

5.4. Resumo

Neste Capítulo apresentamos dois estudos de caso referentes aos sistemas C&L e Unical. Apesar de serem sistemas simples e razoavelmente pequenos, a aplicação de nossa estratégia trouxe os seguintes benefícios quanto à modelagem, ao reuso, à rastreabilidade e à evolução:

- Facilidade de modelagem – diminuição no número de relacionamentos que o engenheiro de requisitos deve escrever; centralização das interações partindo do mesmo elemento em um único relacionamento transversal; geração de visões com o relacionamento transversal resumido (não processado) ou expandido (processado pelo mecanismo de composição); geração das visões, matriz de rastreabilidade (Tabela 8, página 130), cenários (Figura 99, página 138), MER (Figura 97, página 136) e diagrama de classes (Figura 92, página 128). Além disso, com o relacionamento transversal resumido explicitamos a natureza transversal de algumas características.
- Reuso – conseguimos reutilizar os modelos de segurança, confiabilidade, persistência e usabilidade em contextos diferentes. Toda a interação entre estes modelos está centralizada nos relacionamentos transversais, eles podem ser reescritos ou, simplesmente, apagados. Os modelos de performance, manutenibilidade e portabilidade podem ser utilizados como ponto de partida para o detalhamento destas características.
- Rastreabilidade – cada relacionamento transversal é uma matriz parcial de rastreabilidade, que pode ser consultada quando necessário (veja o exemplo ilustrado na Tabela 8, página 130). Além disto, a definição dos relacionamentos transversais induz o engenheiro de requisitos a pensar sobre o impacto que as características exercem umas sobre as outras e o induz a analisar tais características enquanto realiza a modelagem. Desta maneira, a Rastreabilidade deixa de ser postergada ou esquecida, como geralmente ocorre durante a definição de requisitos.
- Evolução – mudanças na definição de relacionamentos transversais são propagadas automaticamente. Cada modelo também pode sofrer modificações independentemente dos demais e se os relacionamentos

transversais também precisam ser modificados, é fácil identificar os pontos afetados por meio da matriz de rastreabilidade fornecida pelos próprios relacionamentos transversais. As facilidades providas pela separação dos modelos, geração de visões e reuso também podem contribuir para análise dos modelos sendo criados e sua melhoria, i.e., evolução.

As facilidades de modelagem providas pela separação e pelo relacionamento transversal facilitam a manipulação de modelos grandes. Estimamos que quanto mais elementos a modelagem contém, maior a necessidade de separação, e assim, se tornam essenciais mecanismos que ajudem na composição e visualização das partes separadas. O mecanismo de composição, mesmo utilizando XSLT (tecnologia muito simples de implementação), foi realizado em apenas 2,08 e 3,16 segundos para o primeiro e segundo estudo de caso, respectivamente.