4
Modelagem Comportamental no SHDM

A Modelagem Comportamental é a proposta de uma nova etapa no método SHDM (*Semantic Hypermedia Design Model*) [Lima, 2003]. Durante esta etapa, ocorrerá a descrição das operações que causam transformação no estado da aplicação. Para dar suporte à especificação dessas operações, foi criado o Modelo de Operações que fornece as primitivas de modelagem de comportamento. Este modelo foi apresentado no Capítulo 3.

As características básicas inerentes aos modelos existentes no SHDM, mencionadas no Capítulo 2, foram mantidas. No entanto, algumas alterações tiveram que ser incluídas para que o Modelo de Operações fosse incorporado ao processo de desenvolvimento de aplicações hipermídia.

A construção de uma aplicação *web* é realizada em cinco etapas de acordo com as diretivas do método SHDM, sendo estas: Levantamento de Requisitos, Modelagem Conceitual, Modelagem Navegacional, Projeto da Interface Abstrata e Implementação. Após a extensão do método, este passou a contar com seis fases, tendo sido acrescentada a Modelagem Comportamental. A Modelagem Comportamental será a terceira etapa no fluxo de desenvolvimento, vindo logo após a Modelagem Conceitual.

A seguir, será apresentado com mais detalhes cada uma das fases e as modificações sofridas.

4.1.
Levantamento de Requisitos

O primeiro passo consiste em identificar os atores (*stakeholders*) e as tarefas que eles precisam realizar, com o objetivo de identificar as necessidades dos usuários.

Em uma aplicação *web* de submissão de artefatos para uma conferência, podemos identificar como um ator, o avaliador. O avaliador é alguém que poderá avaliar artefatos da conferência. Outros possíveis atores desta aplicação são o autor do artefato e o *Chair* da Conferência.

Para cada ator identificado listamos as possíveis tarefas que estes podem desempenhar. O ator avaliador pode ter as seguintes tarefas:

- Aceitar/rejeitar convite para avaliação do artefato;
- Avaliar artefato;
- Consultar avaliações por artefato;
- Definir preferência de avaliação para cada artefato a ser avaliado.

Após definidos atores e tarefas, devemos descrever os cenários de uso. Os cenários representam o conjunto de subtarefas que devem ser completadas para que uma tarefa maior possa ser cumprida. São especificados textualmente focando o ponto de vista do usuário. É desejável que seja obtido a partir dos usuários reais, e quando não for possível, este papel pode ser desempenhado por um membro da equipe de *design* ou outro *stakeholder*.

Um possível cenário para a tarefa "Avaliar artefato" é ilustrado abaixo:

"Informo meu e-mail e minha senha e a aplicação apresenta uma relação dos artefatos que foram alocados para eu avaliar. Nesta relação aparecem o título e o resumo de cada artefato. Seleciono um dos artefatos apresentados e a aplicação me retorna um questionário de avaliação do artefato. Neste questionário constam os seguintes campos: originalidade, no qual escolho uma nota de 1 a 5; conteúdo, no qual escolho uma nota de 1 a 5; apresentação, no qual escolho uma nota de 1 a 5; recomendação, no qual escolho uma nota de 1 a 4; e grau de confiança do avaliador, no qual escolho uma nota de 1 a 5. Após entrar com todos os dados seleciono a opção registrar avaliação e a aplicação notifica o *Chair* da Conferência sobre a avaliação."

Depois da análise de vários cenários relacionados com a mesma tarefa, é possível construir uma generalização, denominada *use case* (caso de uso), utilizando algumas diretrizes [Rossi, Pastor, Schwabe e Olsina, 2007]:

1. Identificar os cenários relacionados com a tarefa que está sendo analisada.
2. Para cada cenário, devem ser identificados os itens de dados trocados durante a interação entre usuário e aplicação.
3. Para cada cenário, identificar quais itens de dados estão associados entre si. Aparecem, comumente, juntos no texto do *use case*.

4. Para cada cenário, os itens de dados que estão organizados como conjuntos devem ser identificados. São referenciados explicitamente no texto do *use case* como conjuntos.
5. As sequências de ações existentes nos cenários devem aparecer no *use case*.
6. Todas as operações executadas sobre itens de dados que aparecem nos cenários, geralmente representadas por verbos, devem ser incluídas no *use case*. Podemos citar como exemplo de operação a ser extraída do cenário acima e incluída no *use case*: "Após entrar com todos os dados seleciono a opção registrar avaliação".

Partindo do cenário exemplo e aplicando as diretrizes, chegamos ao seguinte *use case*:

Use case: Avaliar artefato

1. O usuário informa e-mail e senha.
2. Se a senha for válida, a aplicação retorna uma lista dos artefatos que foram alocados para avaliação do usuário, contendo o título e o resumo de cada artefato. Senão, retorna a mensagem de *login* inválido.
3. O usuário seleciona um artefato.
4. A aplicação retorna um questionário de avaliação, contendo os campos: originalidade, cujo valor compreende notas de 1 a 5; conteúdo, cujo valor compreende notas de 1 a 5; apresentação, cujo valor compreende notas de 1 a 5; recomendação, cujo valor compreende notas de 1 a 4; e grau de confiança do avaliador, cujo valor compreende notas de 1 a 5.
5. O usuário informa todos os valores do questionário de avaliação e escolhe e a opção registrar avaliação.
6. A aplicação notifica o *Chair* da Conferência sobre a avaliação.

O último passo do Levantamento de Requisitos consiste na construção do Diagrama de Interação do Usuário ou UID (*User Interaction Diagram*) [Vilain, 2002], uma notação gráfica para descrição dos *use cases*. Para se obter o fluxo de informações trocadas entre o usuário e o sistema e organizá-las em interações, algumas orientações devem ser seguidas [Vilain, 2002]:

1. Primeiramente, analisar o *use case* para identificar as informações trocadas entre o usuário e o sistema. É importante identificar quais

itens de dados são fornecidos pelo usuário e quais são retornados pelo sistema. São representados por nomes.

2. Os itens identificados são divididos dentro dos UIDs. As informações providas pelo usuário e as providas pelo sistema devem ficar em estados separados, assim como, as informações computadas daquelas utilizadas como entrada para a computação. A ordem dos estados de interação depende das dependências entre os dados fornecidos pelo usuário e os retornados pela aplicação.
3. Devem-se distinguir os dados fornecidos pelo usuário dos retornados pela aplicação. Os primeiros devem ser colocados dentro de retângulos; se forem obrigatórios, a borda deve ser contínua; se forem opcionais, a borda deve ser tracejada. Já os segundos são colocados diretamente no estado de interação, sem estarem dentro de um delimitador.
4. Os estados de interação devem ser conectados por transições. Pode haver múltiplos caminhos entre os estados de interação. Rótulos entre colchetes indicam condições e um rótulo indicando cardinalidade representa uma escolha.
5. E por último, as operações executadas por usuários são representadas por linhas com um círculo preenchido na ponta. Normalmente aparecem como verbos nos *use cases*. O nome da operação aparece entre parênteses.

Seguindo as orientações de criação de UID, detalhamos um UID para o *use case* “Avaliar artefato”:

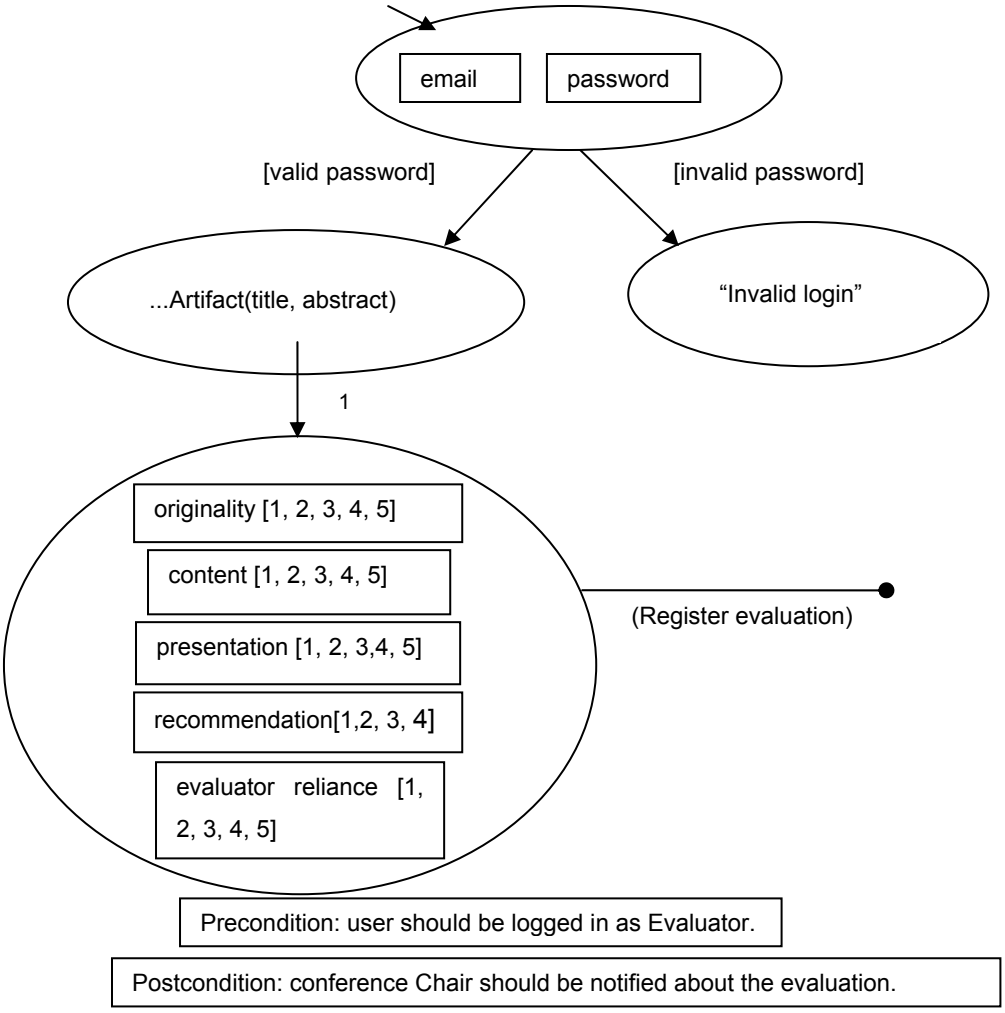

Figura 17 – UID do *use case* “Avaliar artefato”

4.2. Modelagem Conceitual

Nesta etapa do SHDM são descritos todos os itens de informação (“objetos”) do domínio e os relacionamentos entre eles. É produzido um Modelo Conceitual, que é representado por um diagrama de classes, segundo a notação UML. O diagrama de classes é composto por classes, com seus atributos e métodos, e por associações entre estas classes. O esquema conceitual é construído com base nas informações obtidas dos *use cases* e UIDs.

O termo Modelagem Conceitual não é mais usado para denominar a segunda etapa do método SHDM, visto que todos os artefatos produzidos pelo método são conceituais, e não somente o produto desta fase. A única exceção advém da última etapa do SHDM, a Implementação, que não gera um artefato conceitual e sim uma aplicação executável. A denominação adequada para esta fase é Modelagem de Domínio, pois a mesma descreve os dados do domínio de

uma aplicação, representando completamente sua semântica. Dessa forma, o produto dessa fase é o Modelo de Domínio (antigo Modelo Conceitual).

Outra modificação ocorrida nessa fase se refere à especificação de operações. No diagrama de classes utilizado para representar o Modelo de Domínio, uma classe é descrita em termos de seus atributos e métodos, conforme figura 18:

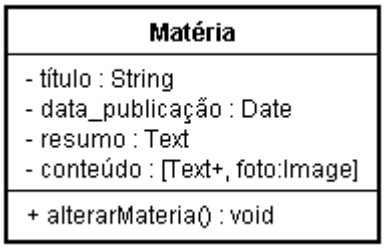

Figura 18 – Classe Matéria com atributos e métodos

As classes de domínio não têm mais a descrição de método, pois isso é de competência da próxima fase do SHDM, a Modelagem Comportamental. Dessa forma, a classe Matéria passou a não mais conter o método “alterarMateria()” em sua composição.

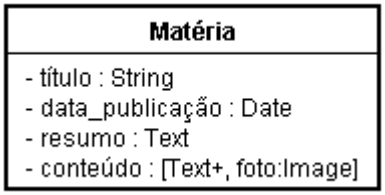

Figura 19 – Classe Matéria sem descrição do método “alterarMateria()”

Os outros processos para a construção do Modelo de Domínio não foram alterados, dentre eles podemos citar [Rossi, Pastor, Schwabe e Olsina, 2007]:

1. Definição de classes: definir uma classe para cada estrutura de dados no UID.
2. Definição de atributos: definir um atributo para cada item de informação (provido pelo usuário ou retornado pelo sistema) que aparecer no UID.
3. Definição de associações: para cada UID, incluir associações para atributos que aparecem em uma estrutura que não correspondem à sua classe, caso exista um relacionamento entre sua classe e a classe que representa a estrutura.

4. Definição de associações: para cada UID, se existir uma estrutura s1 que contem outra estrutura s2, criar uma associação entre as classes correspondentes às estruturas s1 e s2.
5. Definição de associações: para cada transição de estado de interação em um UID, se diferentes classes representam o estado de interação fonte e o estado de interação alvo, definir uma associação entre as classes correspondentes.

Aplicando as orientações 3,4 e 5 ao UID “Avaliar Artefato”, podemos identificar as seguintes associações: Avaliador-Artefato, Avaliador-Avaliação, Artefato-Avaliação. Na figura 20 é ilustrado um Modelo de Domínio inicial derivado do UID “Avaliar Artefato”, já com as alterações propostas.

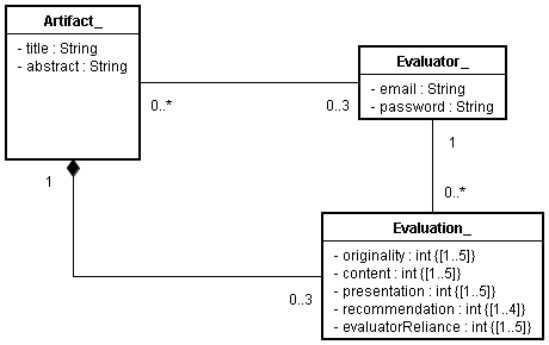

Figura 20 – Modelo de Domínio inicial

Após a análise de todos os UIDs foi possível obter o Modelo de Domínio para uma aplicação de submissão de artefatos em uma conferência, ilustrado na figura 21.

Este Modelo de Domínio é constituído de algumas classes, dentre elas estão as que definem os atores da aplicação: Author e Evaluator, que herdam os atributos da classe Participant. As classes que definem os atores se relacionam com as classes que especificam os artefatos da aplicação. A classe Artifact , que define os artefatos, possui subclasses: Paper e LectureArtifact, que tem como especializações Workshop e Talk. Um avaliador (Evaluator) pode fazer avaliações dos artefatos (Artifact), porém quando possui o papel de Chair pode aceitar ou rejeitar um artefato. Cada artefato (Artifact) deve ser avaliado por três avaliadores (Evaluator). Um artefato é submetido a um tópico (Topic), que está vinculado a uma trilha (Track). Um avaliador (Evaluator) tem como preferência avaliar artefatos de determinado tópico. A classe ConferenceEdition contém os atributos de uma conferência.

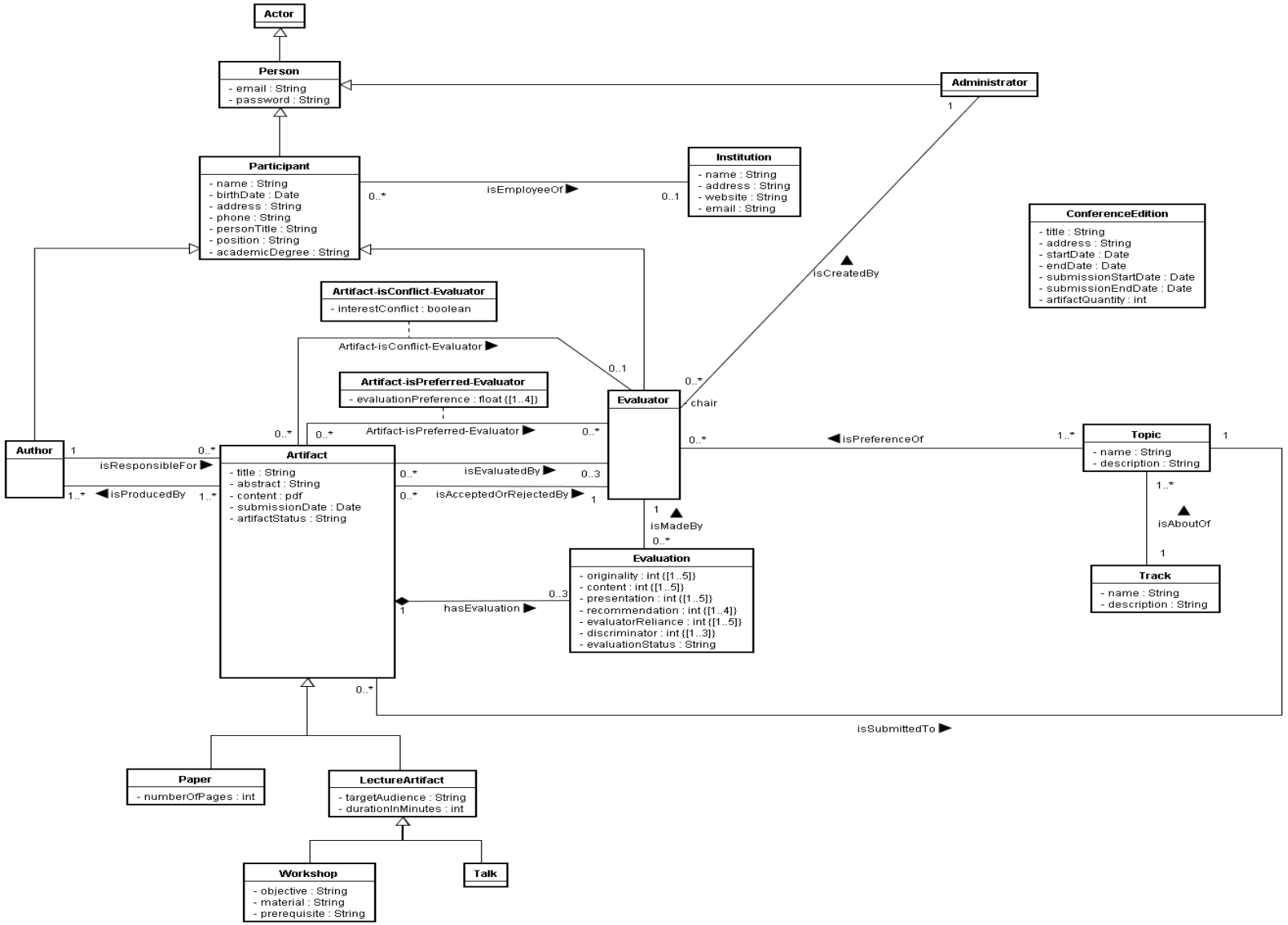

Figura 21 – Modelo de Domínio para a Aplicação *Web* de Submissão de Artefatos para uma Conferência

4.3.
Modelagem Comportamental

É nesta etapa que ocorre a descrição da lógica de negócio da aplicação, através da especificação das operações. As operações são descritas seguindo as primitivas propostas pelo Modelo de Operações. Anteriormente, as operações eram expressas no Modelo de Domínio (antigo Modelo Conceitual) como métodos de classe do metamodelo do SHDM. Estas classes são representadas através de instâncias da metaclasse NavClass, que modela os nós navegacionais. NavClass possui três agregações:

- Objetos da metaclasse NavAttribute, para os atributos;
- Objetos da metaclasse NavOperation, para as operações;
- Objetos da metaclasse Link, para elos.

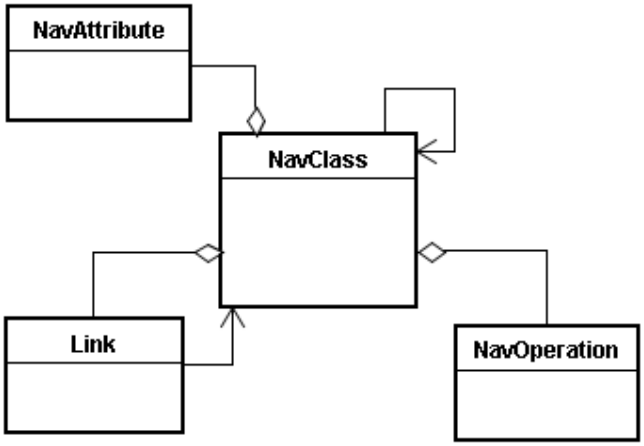

Figura 22 – Parte do Metamodelo do SHDM

As operações definidas pelas instâncias da metaclasse NavOperation podem ser ativadas em instâncias de nós das classes navegacionais da aplicação. NavOperation possui os seguintes atributos [Nunes, 2005]:

- Name: nome da operação;
- Params: nome dos parâmetros que devem ser passados para a operação;
- Code: trecho de código em Ruby que é avaliado sob o contexto de execução do nó corrente;
- Position: indicador opcional da posição da operação que pode ser utilizado em critérios de ordenação pelo ambiente de desenvolvimento.

A metaclasse NavOperation foi substituída pelo Modelo de Operações, no qual as operações passaram a ser descritas. O Modelo de Operações contém a

representação de todas as operações identificadas no Levantamento de Requisitos, durante a especificação dos *use cases* e dos UIDs. Existem quatro formas para identificar e especificar operações a partir dos UIDs:

1. Criar uma operação para cada opção anexada a uma transição de estado do UID que conectar uma ou mais estruturas ou um ou mais itens de dados a um estado de interação de destino. Os parâmetros de entrada para essa operação serão as estruturas ou os itens de dados que estão conectados pela transição e as classes correspondentes a essas informações presentes no estado de interação de origem. As classes correspondentes são aquelas que foram definidas a partir das estruturas ou que têm os itens de dados como atributos. Os parâmetros de saída serão as estruturas ou os itens de dados que estão conectados pela transição e as classes correspondentes a essas informações presentes no estado de interação de destino. As pré e pós-condições podem estar definidas no UID. Um exemplo é apresentado na figura 23, no qual teríamos como parâmetros de entrada as classes Artigo e Revisor para a operação “associar_artigo_revisor”.

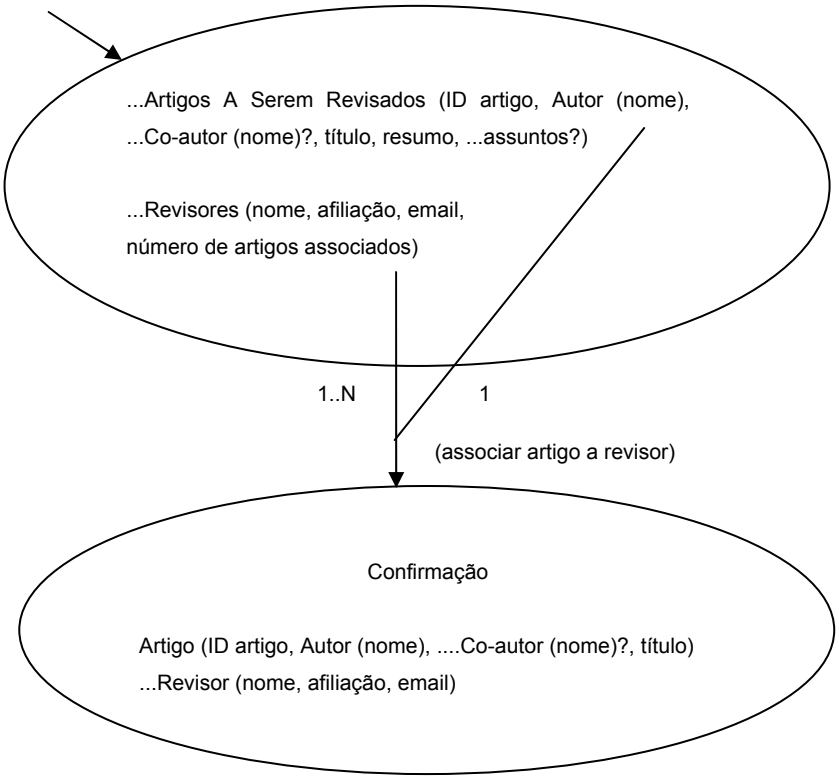

Figura 23 – UID Associar um Artigo a um Revisor [Vilain, 2002]

2. Criar uma operação para cada opção anexada a uma transição de estado do UID que conectar todo um estado de interação a outro estado de interação de destino. Os parâmetros de entrada para essa operação serão as estruturas ou os itens de dados presentes no estado de interação de origem e/ou a classe que representa este estado de interação. Uma classe representa um estado de interação se a estrutura correspondente a ela ou se os itens de dados correspondentes aos seus atributos são as informações mais significativas sendo apresentadas. Os parâmetros de saída serão as estruturas ou os itens de dados que estão conectados pela transição e/ou as classes correspondentes a essas informações presentes no estado de interação de destino. As pré e pós-condições podem estar definidas no UID. Um exemplo é apresentado na figura 24, no qual teríamos como parâmetro de entrada a classe CD para a operação “incluir_lista_compras”.

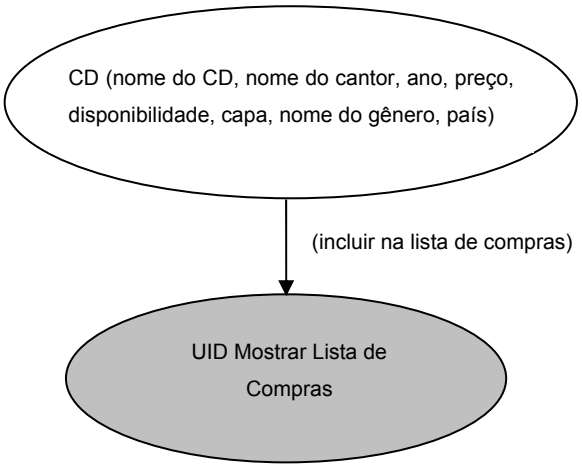

Figura 24 – UID Incluir CD na Lista de Compras [Vilain, 2002]

3. Criar uma operação para cada opção anexada a uma transição de estado do UID que conectar uma estrutura ou item de dados ao mesmo estado de interação, ou seja, os estados de interação origem e destino são os mesmos. Os parâmetros de entrada para essa operação serão as estruturas ou os itens de dados que estão conectados pela transição e as classes correspondentes a essas informações. As classes correspondentes são as classes que foram definidas a partir das estruturas ou que têm os itens de dados como atributos. As pré e pós-condições podem estar definidas no UID. Um exemplo é apresentado na figura 25, no qual teríamos como

parâmetros de entrada a classe Artigo e o item de dados conteúdo para a operação “fazer_upload_conteudo_artigo”.

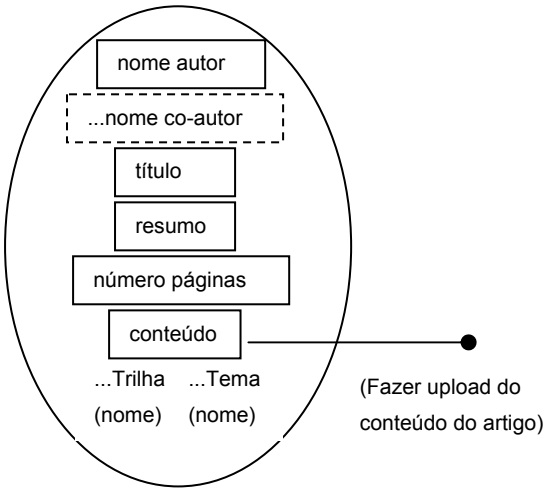

Figura 25 – Parte do UID Submeter Artigo à Conferência

4. Criar uma operação para cada opção anexada a uma transição de estado do UID que conectar todo um estado de interação a ele mesmo, ou seja, os estados de interação origem e destino são os mesmos. Os parâmetros de entrada para essa operação serão as estruturas ou os itens de dados presentes no estado de interação e/ou a classe que representa este estado de interação. Uma classe representa um estado de interação se a estrutura correspondente a ela ou se os itens de dados correspondentes aos seus atributos são as informações mais significativas sendo apresentadas. As pré e pós-condições podem estar definidas no UID. Um exemplo é apresentado na figura 26, no qual teríamos como parâmetros de entrada os itens de dados nome, data de nascimento, endereço, telefone, e-mail, título pessoal, cargo, titulação e senha para a operação “criar_autor”.

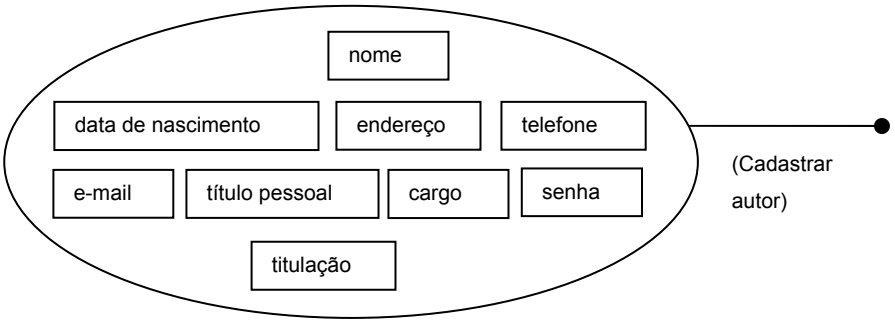

Figura 26 – UID Cadastrar Autor

Nas quatro diretrizes apresentadas para especificação de operações a partir dos UIDs, o projetista pode decidir eliminar alguma operação identificada se concluir que a mesma constitui apenas navegação sobre os dados.

Operações devem sempre ser definidas quando o estado corrente do objeto for alterado ou uma ação for executada. Em determinadas operações, pode não ocorrer o registro da alteração, como por exemplo, no UID da figura 27, no qual temos as operações “enviar e-mail” e “imprimir matéria”. Estas operações definem ações que são executadas sem que o usuário perceba alterações no estado do objeto.

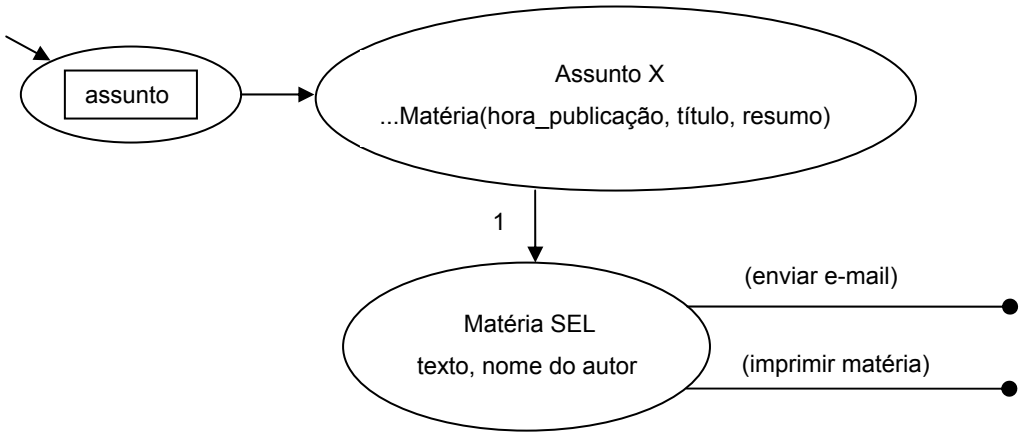

Figura 27 – UID Ler matérias sobre um assunto

Outro exemplo pode ser visto no UID da figura 28, no qual operações também seriam criadas. A opção “criar conta” altera o estado de um objeto, ao criar um novo objeto; e a opção “login”, não é apenas navegação, ela altera o estado do objeto “usuário”, passando o mesmo do estado não logado para o estado logado.

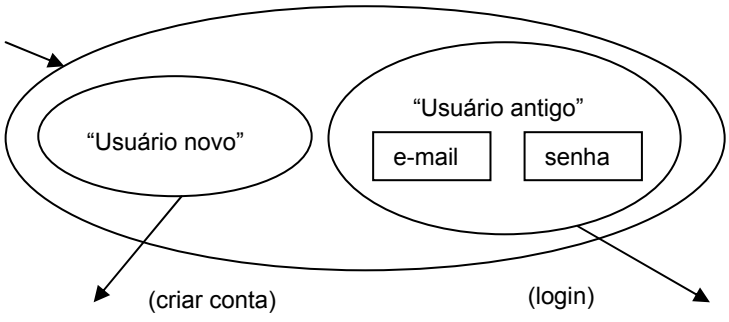

Figura 28 – Parte de um UID com as operações “criar conta” e “login”

Aplicando as orientações de identificação de operações a partir dos UIDs, podemos extrair uma operação do UID “Avaliar artefato” da figura 17, conforme Cartão de Operação ilustrado a seguir. Nesta seção foi apresentado apenas um

exemplo de operação da aplicação *web* de submissão de artefatos para uma conferência. Os outros exemplos serão apresentados no Capítulo 6.

Operation: register_evaluation	
Language: Ruby	**Type:** internal_operation
Precondition:	**Failure_Handling:**
type(user(Evaluator))	¬type(user(Evaluator)) execute Print_UserType_Error()
Parameter In: artifact:Artifact evaluator:Evaluator originality:int content:int presentation:int recommendation:int evaluatorReliance:int	
Parameter Out:	
Postcondition: Send(notifiedEvaluation, Chair)	

Figura 29 – Cartão de Operação de “register_evaluation”

4.4. Modelagem Navegacional

O objetivo desta fase é identificar os objetos navegacionais e a forma como eles são organizados no espaço navegacional [Rossi, Pastor, Schwabe e Olsina, 2007]. Os artefatos que traduzem a realização destas tarefas são, respectivamente, o Modelo de Classes Navegacionais e o Modelo de Contextos Navegacionais.

O Modelo de Classes Navegacionais é descrito como uma visão navegacional sobre o Modelo de Domínio. É neste modelo que ocorre o mapeamento das classes de domínio em classes navegacionais e das relações entre as classes em elos. As relações entre as classes podem se transformar ainda em outros atributos navegacionais, como âncora e índices.

As operações são especificadas nas classes navegacionais de forma análoga à antiga definição de métodos nas classes de domínio. Após a criação de uma etapa específica para modelagem das operações da aplicação, a Modelagem Comportamental, as classes navegacionais passaram a não mais conter a descrição de operações em sua estrutura. No entanto, as operações de navegação sobre os nós da aplicação continuam a ser providas pelo Modelo

Navegacional. Na figura 30 é apresentado um diagrama de classes navegacionais para uma aplicação de submissão de artefatos para uma conferência. Neste modelo, temos algumas classes e relacionamentos:

- Classes navegacionais para atores: autor (Author) e avaliador (Evaluator), que herdam atributos da classe Participant.
    - A classe Author possui um índice para acessar os artefatos do autor.
    - A classe Evaluator possui índices para artefatos avaliados pelo avaliador, artefatos em conflito com os interesses do avaliador e artefatos que o avaliador tem preferência em avaliar. Além de uma lista de tópicos por avaliador.
- Classes navegacionais para artefatos (Artifact): sendo estas Paper e LectureArtifact, subdividida em Workshop e Talk. Possui índices para autores por artefato, avaliações por artefato, avaliadores com conflito para avaliar o artefato e avaliadores com preferência em avaliar o artefato. Possui âncoras para autor responsável por artefato, trilhas por artefato e *Chair* responsável por artefato, e uma lista de tópicos do artefato.
- Classe navegacional para avaliações (Evaluation): com âncoras para o artefato da avaliação e avaliadores por avaliação.
- Classe navegacional para tópicos (Topic): índices para artefatos por tópico e trilha por tópico e âncora para avaliador por tópico.
- Classe navegacional para trilhas (Track): com índice de artefato por trilha e lista dos tópicos da trilha.
- Relacionamentos que geram navegação sobre os dados: autor responsável por artefato, artefato produzido por autor, artefato com conflito e preferência por avaliador, artefato aceito/rejeitado por *Chair*, artefato avaliado por avaliador, avaliação por artefato, avaliação feita por avaliador, artefato sobre um determinado tópico, avaliador com preferência por um tópico, trilha possuir tópico, e ator membro de uma instituição.

O Modelo de Contextos Navegacionais descreve a estrutura navegacional de uma aplicação *web*, por meio de contextos navegacionais. Este modelo tem como primitivas os contextos de navegação, as estruturas de acesso e os elos. Um Modelo de Contextos Navegacionais para uma aplicação de submissão de artefatos para uma conferência é ilustrado na figura 31.

O contexto navegacional é um conjunto de nós relacionados que possuem opções de navegação similares. A estrutura de acesso é um índice que auxilia o usuário na tarefa de encontrar a informação desejada. O elo representa a relação entre os nós e é expresso por meio de âncoras que definem as possibilidades de navegação entre as estruturas de acesso e os contextos.

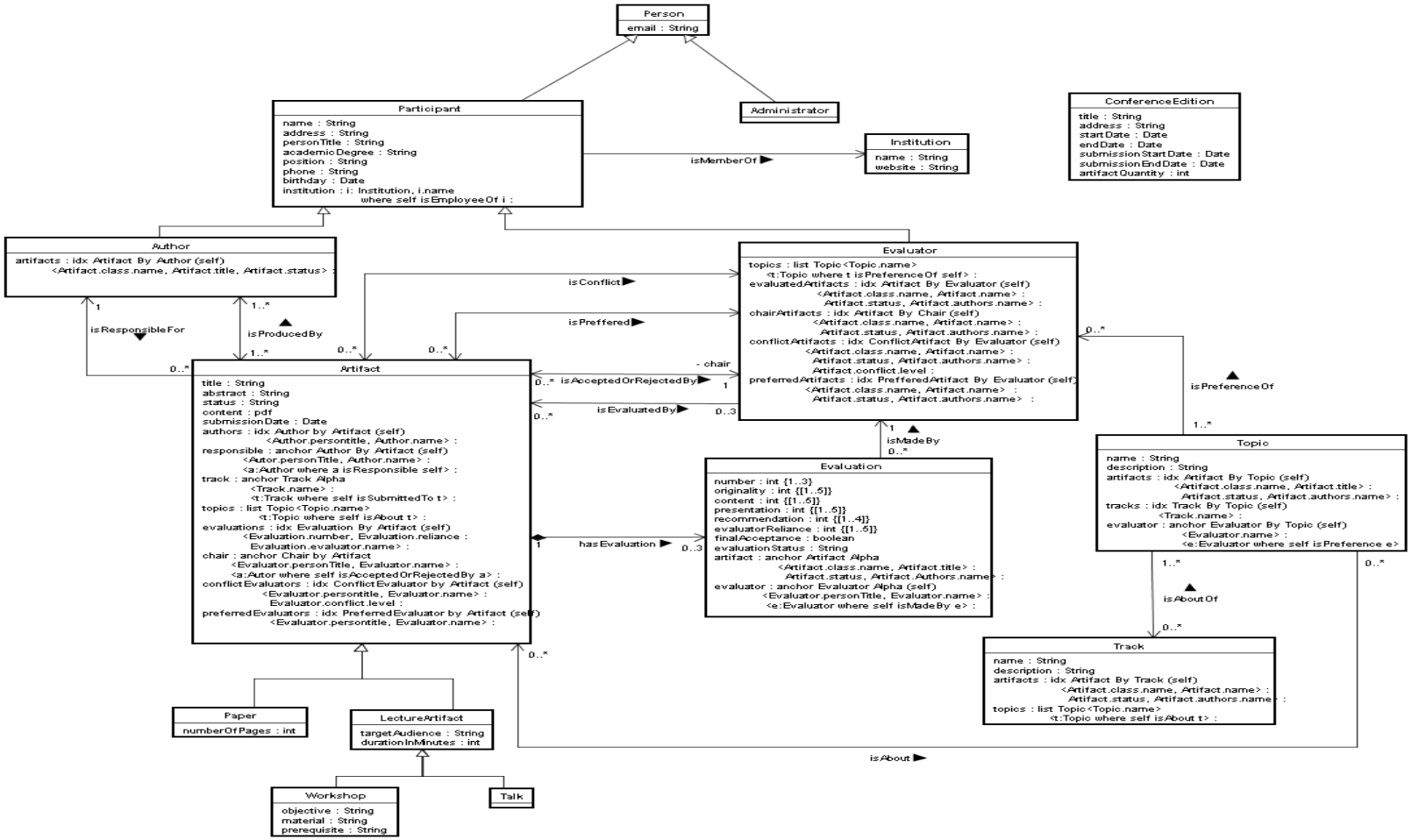

Figura 30 – Diagrama de classes navegacionais para a Aplicação Web de Submissão de Artefatos para uma Conferência

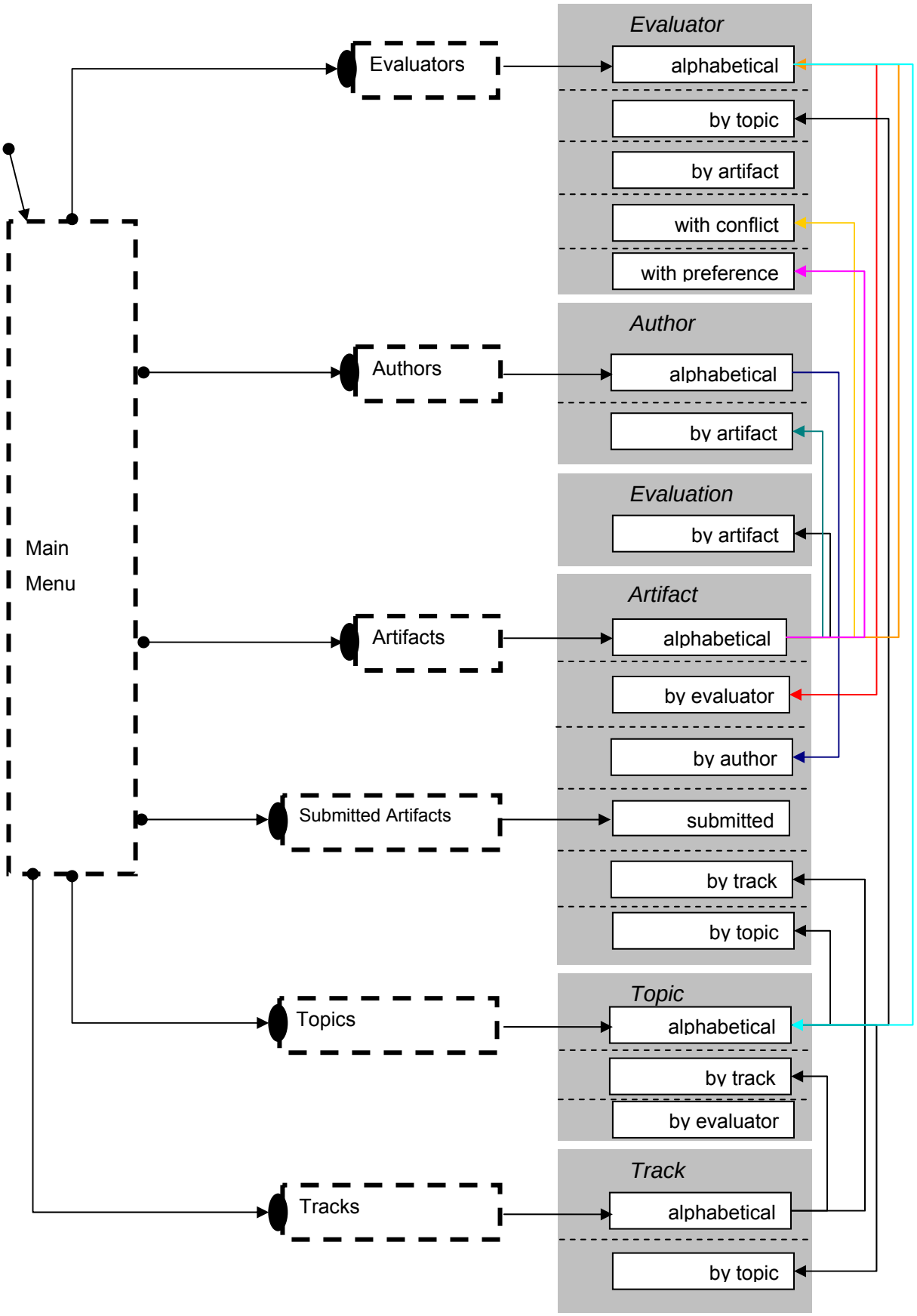

Figura 31 – Modelo de Contextos Navegacionais

4.5.
Projeto da Interface Abstrata

O Projeto da Interface abstrata tem como objetivo tornar perceptível para o usuário os objetos navegacionais e as funcionalidades da aplicação [Rossi, Pastor, Schwabe e Olsina, 2007].

As alterações no Projeto da Interface Abstrata fazem parte do escopo de um trabalho paralelo a este e estão em desenvolvimento [Luna, 2010]. Portanto, não teremos a descrição detalhada dessa fase no presente trabalho.