4
Representando Design Rationale com Kuaba

Normalmente, o primeiro passo realizado pelo projetista no design de um artefato de software é a escolha do método ou processo que será usado para guiar o design. Ao escolher um método ou processo de design, o projetista indiretamente determina o modelo formal (linguagem de especificação) que será usado para descrever o artefato. Por exemplo, quando o projetista escolhe o Processo Unificado (Jacobson et al., 1999) para guiar o seu design, ele indiretamente determina o modelo formal definido pela linguagem UML para especificar seus artefatos.

A existência de um modelo formal para o artefato que está sendo projetado determina grande parte das questões de design e das idéias de solução que o projetista pode propor, uma vez que elas são pré-definidas por este modelo. Neste sentido, o design de um artefato de software seguindo um método de design específico pode ser visto como um processo de instanciação passo a passo do modelo formal que este método prescreve.

Geralmente, o modelo formal de um artefato é descrito pelo metamodelo do método de design utilizado ou da linguagem de especificação que este método usa para descrever artefatos. Este metamodelo é definido como uma das quatro camadas da arquitetura de metamodelagem que define a semântica precisa requerida por modelos complexos (OMG, 2003). Estas camadas estão sumarizadas na tabela abaixo.

Camada	Descrição	Exemplos
Meta-meta modelo	A infra-estrutura para uma arquitetura de meta modelagem. Define a linguagem para especificar meta modelos.	*MetaClasse*, *MetaAtributo*, *MetaOperação*
Metamodelo	Uma instância de um meta-metamodelo. Define a linguagem para especificar um modelo.	Classe, Atributo, Operação, Componente
Modelo	Uma instância de um metamodelo. Define uma linguagem para descrever um domínio de informação.	Lista, estadoAtual, obterValor()
Objetos	Uma instância de um modelo. Define um domínio de informação específico.	[Pessoa: nome = "José", altura=1,80"]

Tabela 1 – Camadas da Arquitetura de Metamodelagem

Como o domínio de design tratado nesta tese é o domínio de designs baseados em modelo, as opções de design que podem ser consideradas durante a representação do *rationale* são definidas pela camada de metamodelo da arquitetura acima, que nós chamamos de modelo formal.

Neste capítulo apresentamos dois exemplos de uso do vocabulário definido pela ontologia Kuaba para representar o *rationale* do design de artefatos de software distintos: o design de um esquema conceitual (seção 4.1) e o design de um modelo de navegação (seção 4.2), ambos considerando o domínio de catálogo de CDs. Em cada exemplo, apresentamos o metamodelo do método de design ou da linguagem de especificação utilizada, que representa o modelo formal do artefato que será usado na instanciação dos elementos da ontologia Kuaba.

4.1. Design Rationale de um Esquema Conceitual

Ao iniciar o design de seu esquema conceitual para o domínio de catálogo de CDs, o projetista define o método de design que será utilizado para especificar seu artefato e qual atividade deste método será realizada para obter o artefato final. Um outro passo importante é o registro do papel que ele vai exercer neste design e do papel que outras pessoas envolvidas no design poderão estar exercendo. O registro dessas informações é importante para facilitar o processamento e a interpretação do *rationale* representado, pois nos fornece parte do contexto no qual o artefato foi projetado. Com base no método de design escolhido é possível obter o modelo formal do artefato que será projetado. O quadro abaixo mostra um trecho da instância da ontologia Kuaba, expresso em F-logic, que representa esta parte do *rationale* para o design que estamos exemplificando.

```
/* Facts */
//Design Activicty Information ----------
up:method[hasName->'Unified Software Development Process';
            prescribes->>{uml}].
uml:formal_model[hasName->'Unified Modeling Language (UML)';
                    hasLocalization->'http://www.omg.org/technology/documents/formal/uml.htm';
                    isPrescribedBy->>{up}].
domainModelActivity:activity[hasName->'Designing the domain Model';
                                hasStartDate->'2004-08-25T09:30:00';
                                hasFinishDate->'2004-08-28T10:10:00';
                                hasExpectedDuration->duration1;
                                requires->>{role1};
                                isExecutedBy->>{ana,carlos}].
duration1:expected_duration[hasAmount->3;
                                 hasUnitTime->'day'].
ana:person[hasName->'Ana Soares'].
carlos:person[hasName->'Carlos Silva'].
role1:role[hasName->'designer'].
```

Quadro 1 – Exemplo de representação do *rationale* sobre a Atividade de Design

Neste trecho da representação de *design rationale*, podemos observar que o projetista definiu o "Processo Unificado" como método de design e selecionou a atividade "Construir o Modelo de Domínio" definida por este processo. Ao escolher o Processo Unificado para guiar o seu design, o projetista define, indiretamente, o modelo formal da UML para especificar seu artefato. As informações sobre este modelo formal, representadas para o objeto "*uml*" no Quadro 1, incluem o nome dado ao modelo formal, o nome do método que o prescreve e o endereço *Web* no qual a especificação deste modelo pode ser encontrada. Além das informações sobre o método de design e o modelo formal que ele prescreve, este trecho da representação também mostra informações específicas sobre a atividade de design, como data de início e duração, e sobre os projetistas que estão participando desse design (nome e papel exercido).

Para construir o esquema conceitual do catálogo de CDs, primeiramente o projetista precisa identificar os itens de informação[7] do domínio da aplicação que são relevantes para o seu design. Estes itens de informação representam os possíveis elementos do esquema conceitual que está sendo criado. Neste exemplo,

[7] Se estivéssemos considerando todo o processo de desenvolvimento de software, estes itens de informação seriam obtidos dos cenários levantados junto ao usuário final durante o levantamento de requisitos.

consideramos que o projetista propõe inicialmente os itens de informação CD, Gênero, Música e Artista. A representação do *rationale* sobre o design de cada um desses itens será exemplificada nesta seção considerando o modelo formal da UML, descrito a seguir.

4.1.1. O Modelo Formal da UML

Para caracterizar o que significa obter o esquema conceitual de uma aplicação de acordo com o modelo formal da UML, utilizamos parte do metamodelo desta linguagem, ilustrado na Figura 10, para a especificação de um diagrama de classes, segundo a técnica de análise orientada a objetos. O metamodelo completo é descrito com mais detalhes na seção *UML Semantics* da *OMG Unified Modeling Language Specification* (OMG, 2003).

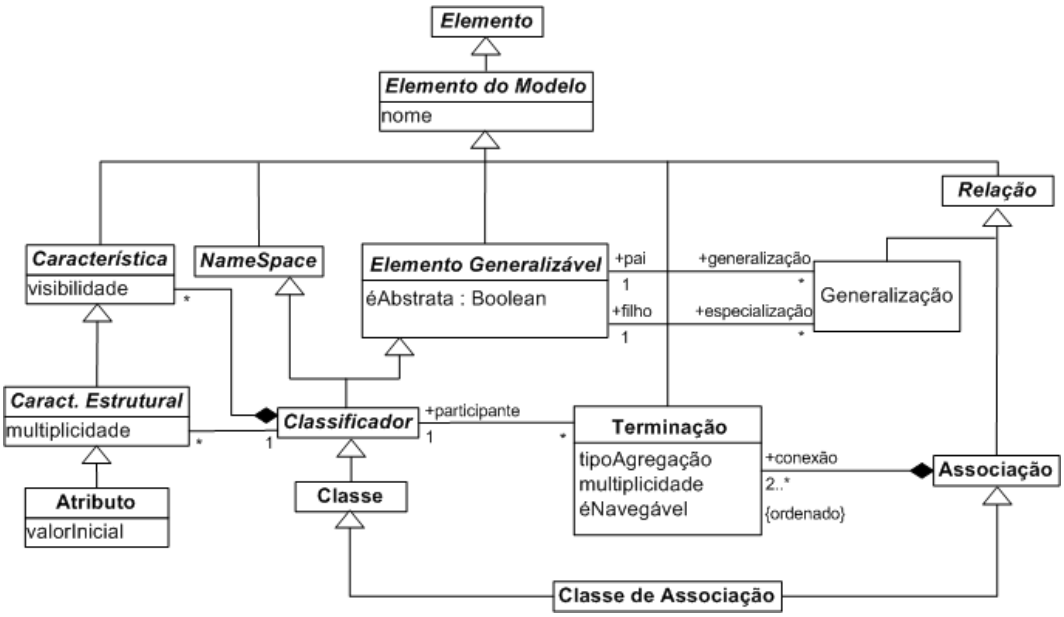

Figura 10 – Modelo formal (parcial) da UML para diagramas de classes

Como podemos observar na figura acima, esta parte do modelo formal da UML descreve os tipos de elementos que podemos ter em um diagrama de classes, os atributos que cada tipo pode ter e as relações que podem existir entre eles. Resumidamente, os tipos de elementos incluem “Classe”, “Atributo”, “Associação”, “Generalização” e “Classe de Associação”. Por herança, um elemento do tipo “Classe” pode ser definido como uma composição de elementos do tipo “Atributo” e um elemento do tipo “Associação” como uma composição de

elementos do tipo “Terminação”, sendo que cada terminação deve especificar o elemento do tipo classe que participa da associação.

Todas as definições de elementos descritas no modelo formal da UML representam opções de design que podem ser usadas para modelar os itens de informação do domínio de conhecimento no qual o projetista está trabalhando. Estas opções de design são usadas na instanciação da ontologia Kuaba para representar o *design rationale* do artefato de software que está sendo produzido, como veremos a seguir.

4.1.2. Rationale sobre o Design dos Elementos CD e Gênero

A representação de *design rationale* geralmente começa com uma questão que estabelece o problema a ser resolvido. Esta questão pode gerar novas questões que representam novos problemas de design relacionados ao problema principal. Para cada questão apresentada, os projetistas podem propor idéias formulando possíveis soluções para o problema expresso na questão.

De acordo com o modelo formal da UML (Figura 10), o primeiro problema a ser resolvido no design de um diagrama de classes é a identificação de seus elementos. A representação deste problema usando o vocabulário da ontologia Kuaba é feita pela instanciação da classe “*Questão*” com o valor “*Quais são os elementos do modelo?*”, como ilustra a Figura 11. Esta figura ilustra uma representação gráfica que nós criamos para facilitar a visualização das instâncias da ontologia Kuaba, mostrando parte do *rationale* usado pelo projetista durante o design do item de informação “Gênero”. Nesta representação, o nó raiz é a questão inicial “*Quais são os elementos do modelo?*” (representada como um retângulo) do tipo “OR”, que é respondida primeiramente pelas idéias de solução “*CD*” e “*Gênero*” (representadas como elipses). Neste exemplo, estas idéias são definidas com base no conhecimento que o projetista tem sobre o domínio, mas poderiam ter sido extraídas do *design rationale* de uma fase anterior, levantamento de requisitos, que não está sendo abordada nesta tese.

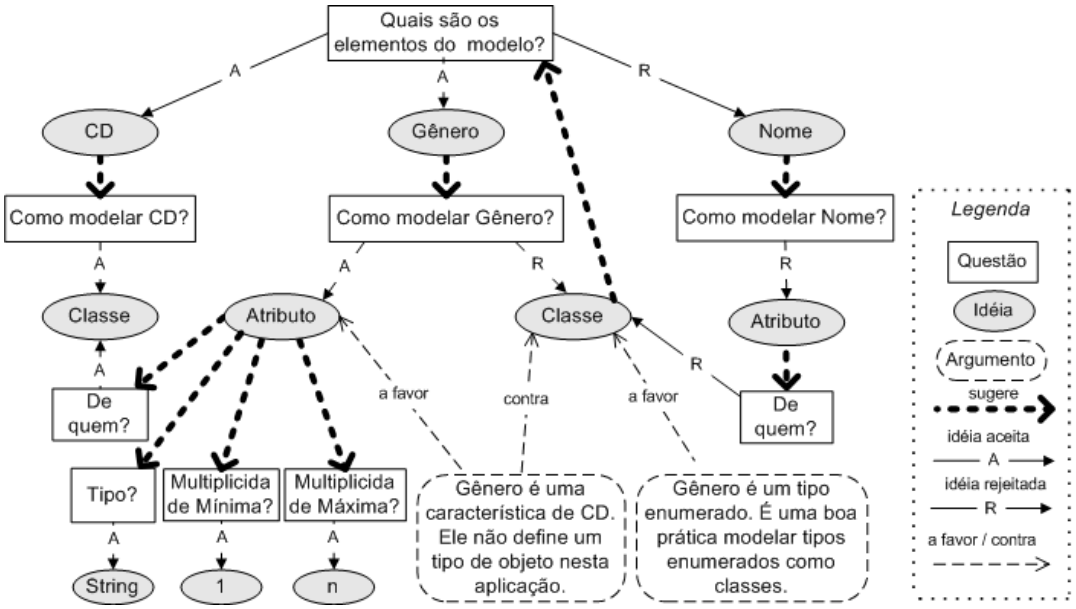

Figura 11 – Exemplo de *design rationale* para o elemento Gênero

Após propor estas primeiras idéias para os elementos do esquema conceitual para o catálogo de CDs, o projetista precisa decidir como cada um desses possíveis elementos será modelado, segundo a linguagem UML, para compor o artefato final, o diagrama de classe. Este novo problema é indicado pela relação "*sugere*" que determina as questões "*Como modelar CD?*" e "*Como modelar Gênero?*", requeridas pelas idéias de solução propostas.

As idéias de solução que podem responder estas novas questões são definidas pelas diferentes opções de design fornecidas pelo modelo formal da UML para diagramas de classes. Analisando as classes concretas deste modelo formal (Figura 10), observamos que os elementos de um diagrama de classe podem ser essencialmente uma classe, um atributo, uma associação ou uma classe de associação. Desta forma, as idéias "C*lasse*" e "A*tributo*", associadas à questão "*Como modelar Gênero?*" são derivadas dessas opções de design. Em uma situação real, o projetista normalmente considera todas as outras opções de design propostas pela UML, mas por questões de simplicidade mostramos apenas estas duas. Os atributos e tipos especificados nas classes do modelo formal da UML também representam opções de design que podem ser utilizadas na instanciação da ontologia Kuaba para representar *design rationale*. Por exemplo, as questões "*Tipo?*", "*Multiplicidade Mínima?*" e "*Multiplicidade Máxima?*" sugeridas pela idéia "*Atributo*" e as idéias de solução associadas a elas são derivadas dos atributos e tipos definidos nas classes do modelo formal da UML.

Com base neste exemplo inicial, é possível perceber como o modelo formal do artefato “guia” a instanciação da ontologia Kuaba fornecendo as opções de design usadas para representar o *design rationale* deste artefato. Este exemplo mostra que, quando o método de design escolhido possui um modelo formal bem definido, é possível automatizar a instanciação de grande parte dos elementos de raciocínio (questões e idéias de solução) que precisam ser registrados durante o design de um artefato. Neste caso, o projetista precisa apenas informar os itens de informação específicos do domínio e os argumentos contra ou a favor das opções de design selecionadas por ele. Observe que, no exemplo da Figura 11, apenas as idéias de solução para a questão inicial e os argumentos (representados como retângulos tracejados) possuem conteúdo informal fornecido pelo projetista.

O quadro abaixo mostra um trecho da instância da ontologia Kuaba, expresso em F-logic, que representa parte do *rationale* usado para o design do elemento ”*Gênero*”.

```
/* Facts */
// Reasoning Elements ----------
genre:idea[hasText->'Genre';
            hasCreationDate->'2004-08-25T09:12:06'; address->>{whatElements};
            hasArgument->>{genreArgument}; isInvolved->domainModelActivity;
            isPresentedBy->carlos; suggests->>{hwModelGenre}].
hwModelGenre:question[hasText->'How to model Genre?'; isDefinedBy->uml;
                            hasCreationDate->'2004-08-25T09:15:02'; hasType->'XOR';
                            isInvolved->domainModelActivity; isPresentedBy->carlos;
                            hasDecision->>{genreAttribDecision;genreClassDecision}].
genreAttribute:idea[hasText->'Attribute';
                        hasCreationDate->'2004-08-25T09:21:02'; address->>{hwModelGenre};
                        hasArgument->>{genreAttribArgument}; isDefinedBy->uml;
                        isPresentedBy->ana; isInvolved->domainModelActivity;
                        suggests->>{whoseAttrGenre;minMultAttribGenre;maxMultAttribGenre}].
genreClass:idea[hasText->'Class';
                    hasCreationDate->'2004-08-25T09:21:12';
                    address->>{hwModelGenre;whoseAttribGenreName}; isDefinedBy->uml;
                    hasArgument->>{genreClassArgument1;genreNameAttribArgument2};
                    isInvolved->domainModelActivity; isPresentedBy->carlos;
                    suggests->>{whatElements}].
genreAttribArgument:argument[hasCreationDate->'2004-08-25T09:21:56';
                                    inFavorOf->>{genreAttribute}; objectsTo->>{genreClass};
                                    isInvolved->domainModelActivity; isPresentedBy->ana;
                                    hasText->'Genre is a feature of CD. It does not define an
                                                object type in this application'].
genreClassArgument:argument[hasCreationDate->'2004-08-25T09:22:00';
                                    inFavorOf->>{genreClass};
                                    isInvolved->domainModelActivity; isPresentedBy->carlos;
                                    hasText->'Genre is an enumerated type. It is a good practice
                                                to model enumerated types as classes'].
```

Quadro 2 - Exemplo de parte do *design rationale* sobre o design do elemento Gênero

Neste trecho, podemos observar que os elementos de raciocínio, cujos valores foram fornecidos pelo modelo formal da UML, possuem a relação "*é indicado por*", que os distingue dos elementos de raciocínio cujos valores são fornecidos pelo projetista.

Como vimos anteriormente, uma classe pode ser vista como uma composição de atributos, de acordo com o modelo formal da UML. Assim, para modelar "*Gênero*" como uma classe, é necessário definir os atributos que compõem esta classe. Isto é representado no *rationale* ilustrado na Figura 11 pela relação "*sugere*" entre a idéia "*Classe*" na sub-árvore de "*Gênero*" e a questão inicial "*Quais são os elementos do modelo?*". Esta relação representa que o projetista precisa identificar outros itens de informação relacionados a gênero, antes de tomar uma decisão sobre o design de "*Gênero*". Caso contrário, o design poderia ser considerado inconsistente.

Analisando o *rationale* ilustrado na Figura 11, vemos que o projetista propôs também o item de informação "*Nome*" e considerou a possibilidade de modelar este item como um atributo. Uma vez que um atributo deve estar associado a uma classe, segundo o modelo formal da UML, a questão "*De quem?*" é sugerida. Esta questão é respondida pela idéia "*Classe*" na sub-árvore de "*Gênero*", representando que o projetista considerou o design de "*Nome*" como um atributo de uma classe "*Gênero*".

Nesta representação de *design rationale* mostramos também as decisões tomadas pelos projetistas, indicando cada idéia de solução para cada questão com um rótulo "A" (para idéias aceitas) ou "R" (para idéias rejeitadas). Então, neste exemplo podemos notar que o projetista decidiu aceitar a idéia "*Atributo*" como uma solução para a questão "*Como modelar Gênero?*", em detrimento da idéia "*Classe*".

O sub-grafo de uma representação de *design rationale* formado de questão e idéias é um grafo AND/OR (Nilsson, 1986), que pode ser visto como uma decomposição do objetivo do nó raiz, que é sempre uma questão. Como vimos no capítulo anterior, o elemento "Questão" da ontologia Kuaba tem um atributo tipo, cujos possíveis valores ("*AND*", "*OR*" e "*XOR*") são usados em regras para sugerir decisões sobre a aceitação ou não das idéias de solução propostas. Por exemplo, um sistema poderia sugerir rejeitar certas idéias baseado na seguinte regra:

Se uma idéia associada a uma questão do tipo "XOR" é aceita pelo projetista, então todas as outras idéias associadas a esta questão podem ser rejeitadas.

Na representação ilustrada na Figura 11, se a decisão de aceitar a idéia de modelar "*Gênero*" como um atributo foi a primeira a ser tomada pelo projetista, um ambiente de design poderia aplicar a regra acima e automaticamente propor que a idéia de modelar "*Gênero*" como uma classe seja rejeitada, dado que existem apenas duas idéias associadas à questão "*Como modelar Gênero?*" e esta questão é do tipo "*XOR*". Neste ponto, o projetista também tem a opção de não aceitar esta sugestão e revisar as possíveis respostas para a questão inicial "*Quais são os elementos do modelo?*", rejeitando "*Gênero*" como um elemento do esquema conceitual. Em qualquer caso, o ambiente de design pode usar as regras definidas pela ontologia Kuaba e também aquelas definidas pelo modelo formal do artefato, para validar as decisões tomadas a fim de garantir a consistência do *design rationale* que está sendo representado, informando ao projetista eventuais contradições. Portanto, a ordem de aceitação ou rejeição das idéias não afeta o *rationale* que está sendo representado.

O quadro abaixo mostra um trecho da instância da ontologia Kuaba, expresso em F-logic, que representa as decisões tomadas sobre as opções de design para modelar "*Gênero*".

```
/* Facts */
// Decisions and Justifications ----------
genreAttribDecision:decision[concludes->genreAttribute; isAccepted->'true';
                                hasDate->'2004-08-25T09:28:12'; isMadeBy->>{carlos};
                                hasJustification->genreAttribJustification].
genreAttribJustification:justification[isDerivedOf->>{genreAttribArgument};
                                    hasText->'In this application we only need to record what is
                                            the genre of a CD. In the conceptual model we
                                            can represent it defining an attribute genre for the
                                            CD class.'].
genreClassDecision:decision[concludes->genreClass; isAccepted->'false';
                                hasDate->'2004-08-25T09:28:12'; isMadeBy->>{carlos};
                                hasJustification->genreClassJustification].
genreClassJustification:justification[isDerivedOf->>{genreClassArgument};
                                     hasText->'Important properties were not found to justify the
                                             representation of genre as a class. Thus, I decided
                                             to model genre as an attribute with multiplicity 1..n
                                             to simplify the conceptual schema.' ].
```

Quadro 3 - Exemplo de representação das decisões sobre o design de Gênero

Este trecho da representação de *rationale* contém a especificação da justificativa final para cada decisão tomada pelo projetista. A representação desta justificativa nos ajuda a compreender melhor como os argumentos apresentados durante o design influenciaram, ou não, a decisão final tomada pela projetista.

Observando novamente a representação gráfica ilustrada na Figura 11 podemos notar que todas as questões de design representadas são definidas pelo modelo formal da UML. Nestes casos, é possível suprimir estas questões para obter uma representação mais simples do *rationale* gerado, visando simplificar a visualização das instâncias da ontologia Kuaba. A Figura 12 ilustra a representação gráfica simplificada.

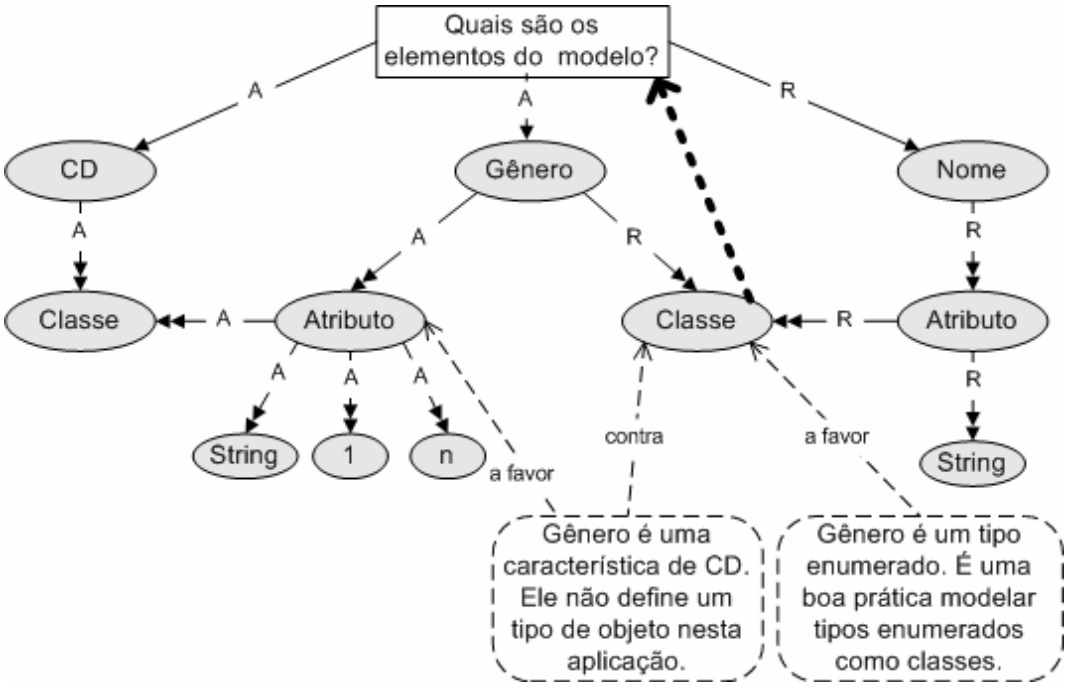

Figura 12 – Representação gráfica simplificada do *design rationale* da Figura 11

Note que esta representação simplificada ilustra apenas a questão principal, as idéias consideradas durante o design com seus respectivos argumentos e as decisões tomadas pelos projetistas. A seta com ponta dupla entre duas idéias indica que uma questão de design definida pelo modelo formal foi omitida na representação gráfica do *design rationale.*

Para completar o exemplo de *rationale* para o design do elemento “CD” podemos considerar que outros itens de informação relacionados a CD, além de “*Gênero*”, foram considerados pelo projetista. Por exemplo, os itens ”*Título*” e “*Preço*”, como ilustra a figura abaixo.

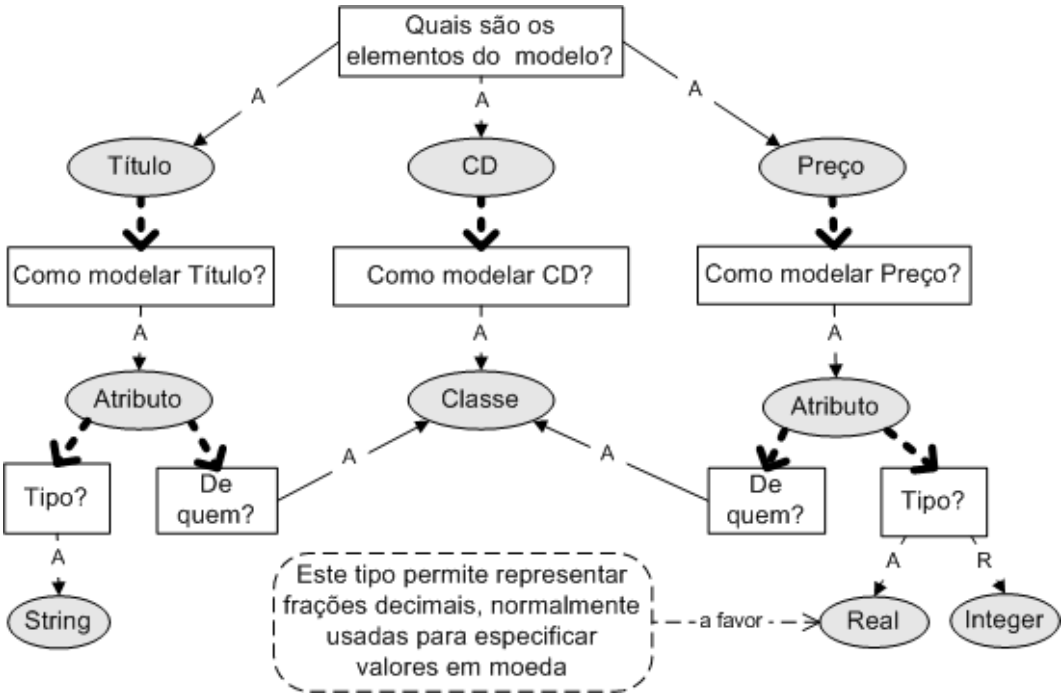

Figura 13 - Exemplo de *design rationale* para o elemento CD

Neste exemplo vemos que o projetista decidiu modelar os itens "*Título*" e "*Preço*" como atributos da classe CD com os tipos "*String*" e "*Real*", respectivamente. A figura abaixo ilustra o artefato resultante do *rationale* representado nas figuras 11 e 13.

Figura 14 – O Artefato CD

Além dos elementos de raciocínio (questões, idéias e argumentos) e das decisões tomadas pelo projetista durante o design, a ontologia Kuaba permite representar também informações sobre os artefatos produzidos. Pelo vocabulário da ontologia Kuaba um artefato pode ser representado como um artefato atômico ou como um artefato composto. A decisão sobre como representar um artefato geralmente depende do tipo do artefato e de sua definição no modelo formal utilizado. De acordo com o modelo formal da UML, uma classe é uma composição de atributos. Portanto, a classe CD ilustrada na Figura 14 pode ser representada como um artefato composto por diversos artefatos atômicos, que representam seus atributos (título, gênero e preço).

O quadro abaixo mostra um trecho da instância da ontologia Kuaba que descreve o artefato CD e sua relação com o elemento de raciocínio que deu origem a ele. Esta relação é descrita como a relação inversa "*é resultado*", definida para a relação "*resulta*" entre os elementos "*Idéia*" e "*Artefato*" no vocabulário da ontologia Kuaba.

```
/* Facts */
// Artifacts ----------
titleArtifact:atomic_artifact[hasName->'Title attribute';
                              hasCreationDate->'2004-08-26T09:30:52';
                              hasDescription->'Attribute of the CD class that contains the name
                                              of the album';
                              isCreatedBy->>{carlos}; isDescribedBy->uml;
                              isResulted->>{title}].
genreArtifact:atomic_artifact[hasName->'Genre attribute';
                              hasCreationDate->'2004-08-26T09:31:12';
                              hasDescription->'Attribute of the CD class that contains the musical
                                              genre of the album';
                              isCreatedBy->>{carlos}; isDescribedBy->uml;
                              isResulted->>{genre}].
priceArtifact:atomic_artifact[hasName->'Price attribute';
                              hasCreationDate->'2004-08-26T09:32:02';
                              hasDescription->'Attribute of the CD class that contains the price
                                              of the album';
                              isCreatedBy->>{carlos}; isDescribedBy->uml;
                              isResulted->>{price}].
cdArtifact:composite_artifact[hasName->'CD class';
                              hasCreationDate->'2004-08-26T09:48:02';
                              isCreatedBy->>{carlos}; isDescribedBy->uml;
                              compositionOf->>{titleArtifact; genreArtifact; priceArtifact};
                              isResulted->>{cd};
                              hasDescription->'Main class of the application. The class CD
                                              contains all attributes that an object CD can
                                              possess in this application'].
```

Quadro 4 - Exemplo de representação do artefato CD como um artefato composto

Note que os itens de informação "*Título*", "*Preço*" e "*Gênero*" estão representados como artefatos atômicos, pois de acordo com o *design rationale* registrado nas figuras 11 e 13, o projetista decidiu modelar estes elementos como atributos de CD.

4.1.3. Rationale sobre o Design do Elemento Música

Além dos itens CD e Gênero, o projetista propôs também o item de informação Música como um elemento para o seu esquema conceitual. A Figura 15 ilustra parte do *rationale* usado por ele para modelar este elemento. De forma

similar ao design do elemento "G*ênero*", o projetista propôs as idéias de solução "*Atributo*" e "*Classe*" para modelar o elemento "*Música*" e precisou identificar outros itens de informação, antes de decidir se este elemento deve ser modelado como uma classe ou não.

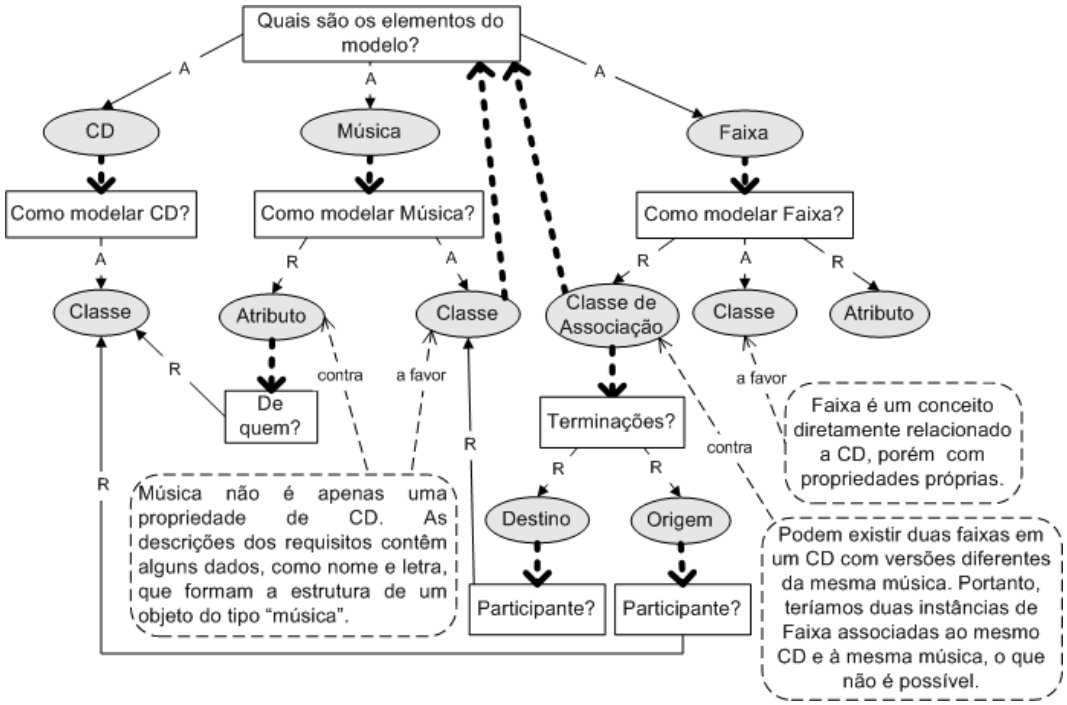

Figura 15 – Exemplo parcial do *design rationale* para o elemento Música

Neste exemplo podemos observar que o projetista apresentou a idéia de solução "*Faixa*" como um possível item de informação relacionado ao elemento Música, e considerou as idéias "*Atributo*", "*Classe*" e "*Classe de Associação*" como possíveis opções para modelar este item em seu diagrama de classes.

Observando novamente o modelo formal da UML, ilustrado na Figura 10, uma classe de associação é uma especialização do elemento classe e do elemento associação. Como uma especialização do elemento associação, uma classe de associação deve ser composta de pelo menos duas terminações. Isto implica que o tipo da questão "*Terminações?*" sugerida pela idéia "*Classe de Associação*" na Figura 15 deve ter o valor "*OR*" na representação de *design rationale* gerada, uma vez que mais de uma idéia de solução deve ser aceita para esta questão. Desta forma, as idéias "*Destino*" e "*Origem*" devem ter associadas a elas decisões iguais, ou seja, ambas devem ser aceitas ou rejeitadas.

Pelo modelo formal da UML, cada terminação proposta para uma associação deve estar relacionada a uma das classes que participam da associação. No

rationale ilustrado na Figura 15, isto é representado pelas questões "*Participante?*" sugeridas para as terminações propostas pelo projetista para uma possível classe de associação "*Faixa*". As idéias que respondem estas questões indicam que o projetista considerou modelar o elemento "*Faixa*" como uma classe de associação entre os elementos "*CD*" e "*Música*", ambos modelados como classes. No entanto, esta idéia de solução foi rejeitada pelo projetista, com base no argumento apresentado contra a idéia "*Classe de Associação*". A Figura 16 ilustra o *rationale* de mais algumas decisões do projetista envolvendo o elemento "*Faixa*".

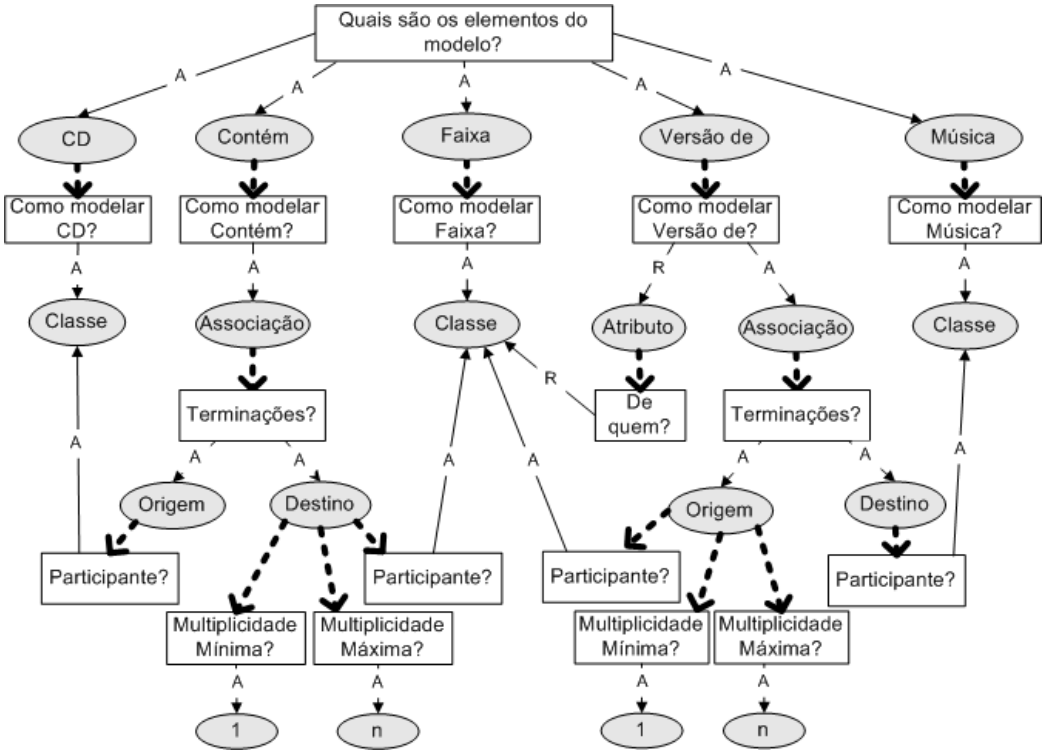

Figura 16 - Exemplo parcial do *design rationale* sobre o elemento Faixa

Como podemos observar, os itens de informação propostos pelo projetista durante o design de um diagrama de classes também podem ser modelados como associações. Neste exemplo, o projetista propõe os itens "*Contém*" e "*Versão de*" como elementos para o diagrama de classes e decide modelá-los como associações entre os elementos "*CD*", "*Faixa*" e "*Música*". As idéias "*1*" e "*n*" que respondem as questões "*Multiplicidade Mínima*" e "*Multiplicidade Máxima*" para cada uma das terminações nas quais o elemento "*Faixa*" participa, indicam que um CD pode ter uma ou mais faixas e que uma música pode ser gravada em uma ou mais faixas do CD. A figura abaixo ilustra o esquema conceitual resultante após as decisões tomadas sobre o design dos elementos "*Música*" e "*Faixa*".

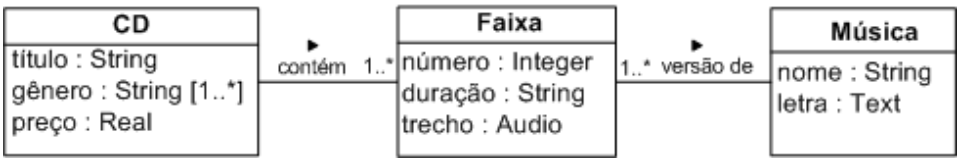

Figura 17 – Esquema conceitual parcial, considerando os elementos Música e Faixa

A representação do *design rationale* que contém os elementos de raciocínio e as decisões sobre a modelagem dos atributos representados nas classes “Faixa” e “Música”, não foi apresentada nesta seção por ser esta bastante semelhante à representação de *rationale* ilustrada na Figura 13 para o design do elemento “CD”.

4.1.4. Rationale sobre o Design do Elemento Artista

Nesta seção, apresentamos alguns exemplos que mostram parte do raciocínio desenvolvido pelo projetista durante o design do item de informação “*Artista*”. Com base na Figura 18, podemos supor que este raciocínio é iniciado a partir do argumento contra a idéia de modelar o elemento “*Cantor*” como uma classe. Em uma análise real do *rationale*, esta suposição poderia ser confirmada através de uma consulta à data e hora de criação de cada elemento de raciocínio incluído na representação. A relação “*sugere*” entre o argumento contra a idéia “*Classe*” na sub-árvore de “*Cantor*” e a questão inicial “*Quais são os elementos do modelo?*”, indica que o projetista precisou propor novos itens de informação para poder modelar os diferentes profissionais relacionados a um CD. Por exemplo, os itens descritos pelas idéias “*Artista*”, “*Intérprete*” e “*Compositor*”.

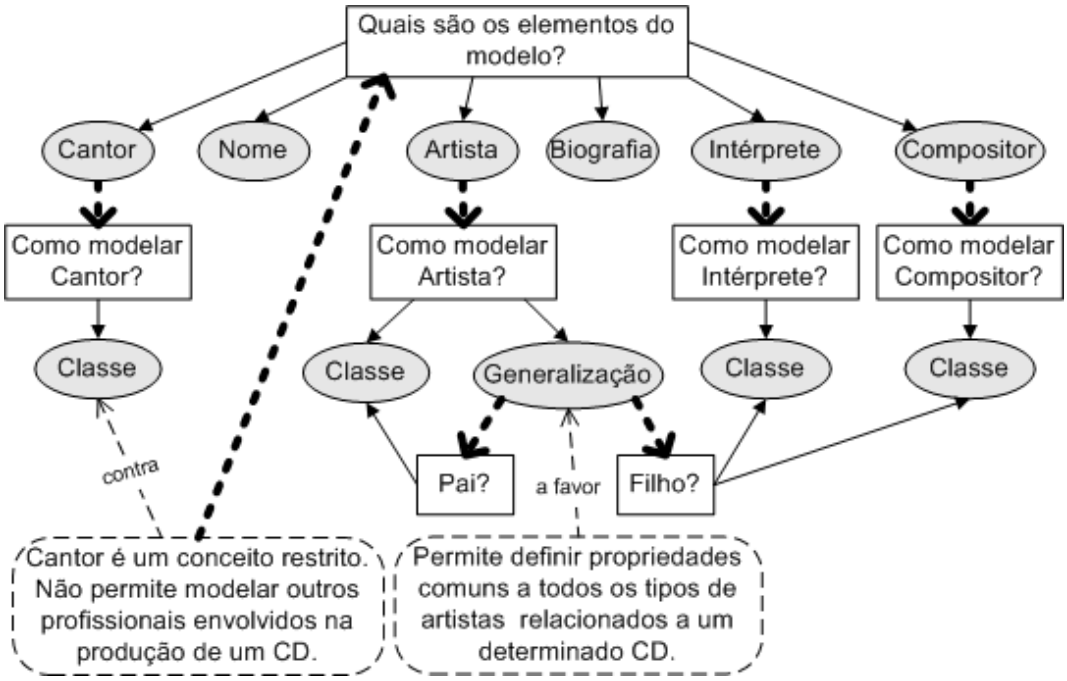

Figura 18 – Exemplo de parte do *rationale* sobre o design do elemento Artista

Neste exemplo, podemos observar que o projetista considerou a idéia "*Generalização*" para modelar o item de informação "*Artista*", que deu origem às questões "*Pai*?" e "*Filho*?". Pelas definições do modelo formal da UML, uma generalização é uma relação entre um elemento "pai" e pelo menos um elemento "filho", que herdará as propriedades definidas no elemento "pai". Geralmente, esses elementos são conhecidos na modelagem orientada a objetos como superclasse e subclasse, respectivamente. No entanto, preferimos usar os termos "pai" e "filho" adotados no modelo formal da UML como valores para as instâncias do elemento "*Questão*" da ontologia Kuaba, uma vez que o conceito de generalização também pode ser aplicado para representar especializações de associações entre classes. Desta forma, no *rationale* representado na Figura 18, o projetista considerou a solução de modelar o elemento "*Artista*" como o "pai" na relação de generalização (uma superclasse) e os elementos "*Intérprete*" e "*Compositor*" como filhos (subclasses) de "*Artista*".

Ao propor as idéias "*Artista*", "*Intérprete*" e "*Compositor*" como elementos do esquema conceitual, o projetista precisa resolver novos problemas de design. Por exemplo, o projetista precisa definir qual é a melhor forma de modelar estes elementos no esquema conceitual e como eles estarão relacionados com os outros elementos já modelados. A Figura 19 ilustra algumas questões e idéias relacionadas à questão sobre como modelar o elemento "*Intérprete*".

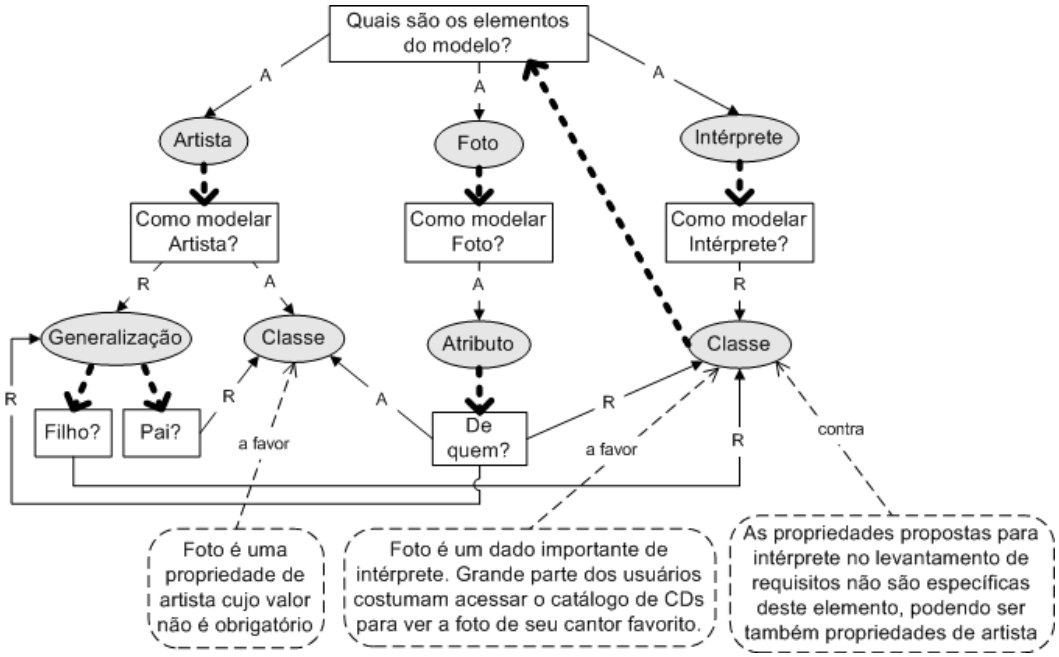

Figura 19 – Exemplo de *design rationale* sobre o elemento Intérprete

Nesta representação, a idéia "*Classe*" proposta para a questão "*Como modelar Intérprete?*", sugere a questão inicial indicando que o projetista precisou definir novos itens de informação relacionados a "*Intérprete*", por exemplo, o item "*Foto*". Este item foi apresentado inicialmente como uma idéia de atributo para o elemento "*Intérprete*", o que poderia justificar a modelagem deste elemento como um filho (subclasse) de "*Artista*". No entanto, analisando o *rationale* podemos perceber que o projetista também considerou modelar "*Foto*" como um atributo do elemento "*Artista*", modelado como uma classe ou como uma generalização. Pelos argumentos e decisões representadas na Figura 19, podemos concluir que o projetista percebeu que não havia atributos específicos para o elemento "*Intérprete*" e decidiu rejeitar as idéias de modelar este elemento como uma classe e como uma especialização de "*Artista*".

Analisando os argumentos apresentados para a idéia "*Classe*" na sub-árvore do elemento "*Intérprete*", podemos notar que não é possível compreende-los sem considerar também as questões "*Como modelar Intérprete?*" e "*De quem?*", que esta idéia responde. Embora isto não esteja representado graficamente na Figura 19, é possível associar um argumento a uma questão usando a relação "*considera*" definida no vocabulário da ontologia Kuaba, como mostra o exemplo no quadro abaixo.

```
/* Facts */
// Reasoning Elements ----------
interpreterClass:idea[hasText->'Class';
                      hasCreationDate->'2004-08-26T19:21:13';
                      address->>{hwModelInterpreter; whoseAttribPhoto};
                      isDefinedBy->uml; suggests->>{whatElements};
                      hasArgument->>{interpreterClassArgument1; interpreterClassArgument2};
                      isInvolved->domainModelActivity; isPresentedBy->carlos].
interpreterClassArgument1:argument[hasCreationDate->'2004-08-26T19:22:56';
                                   inFavorOf->>{interpreterClass};
                                   isInvolved->domainModelActivity; isPresentedBy->ana;
                                   hasText->'Photo is an important data of interpreter.  The
                                            majority of users access the catalogue of CDs to
                                            see the photo of its favourite singer';
                                   considers->whoseAttribPhoto].
interpreterClassArgument2:argument[hasCreationDate->'2004-08-26T19:24:00';
                                   objectsTo->>{interpreterClass};
                                   isInvolved->domainModelActivity; isPresentedBy->carlos;
                                   hasText->'The proposed properties for interpreter in the
                                            survey of requirements are not specific of this
                                            element. They can be also artist's properties.';
                                   considers->hwModelInterpreter].
```

Quadro 5 - Exemplo de representação da relação "*considera*"

Este exemplo mostra a relação "*considera"* representada para os dois argumentos apresentados para a idéia "*Classe*" na sub-árvore do elemento "*Intérprete*". Esta relação permite representar que o argumento é válido apenas para uma das questões respondidas por esta idéia. Desta forma, o primeiro argumento representado no exemplo é válido apenas considerando a idéia "*Classe*" como uma resposta para a questão "*De quem?*", e o segundo argumento é válido apenas considerando esta idéia como uma resposta para a questão "*Como modelar Intérprete?*".

Observando novamente a Figura 19, podemos notar que o projetista decidiu modelar o elemento "*Artista*" como uma classe e o elemento "*Foto*" como um atributo desta classe (rótulos "A" nas setas). Suponha que ao rejeitar as idéias de modelar o elemento "*Intérprete*" como "filho" (subclasse) e o elemento "*Artista*" como "pai" (superclasse), o projetista tenha chegado à conclusão que os perfis profissionais considerados até o momento (Intérprete e Compositor) poderiam ser modelados como papéis exercidos por um objeto "*Artista*" em uma das suas relações com outros objetos do domínio catálogo de CDs. Neste caso, é necessário definir quais são estas relações e como elas podem ser modeladas. A Figura 20

ilustra o *design rationale* desenvolvido pelo projetista para modelar uma possível relação "*interpreta*" no esquema conceitual do catálogo de CDs.

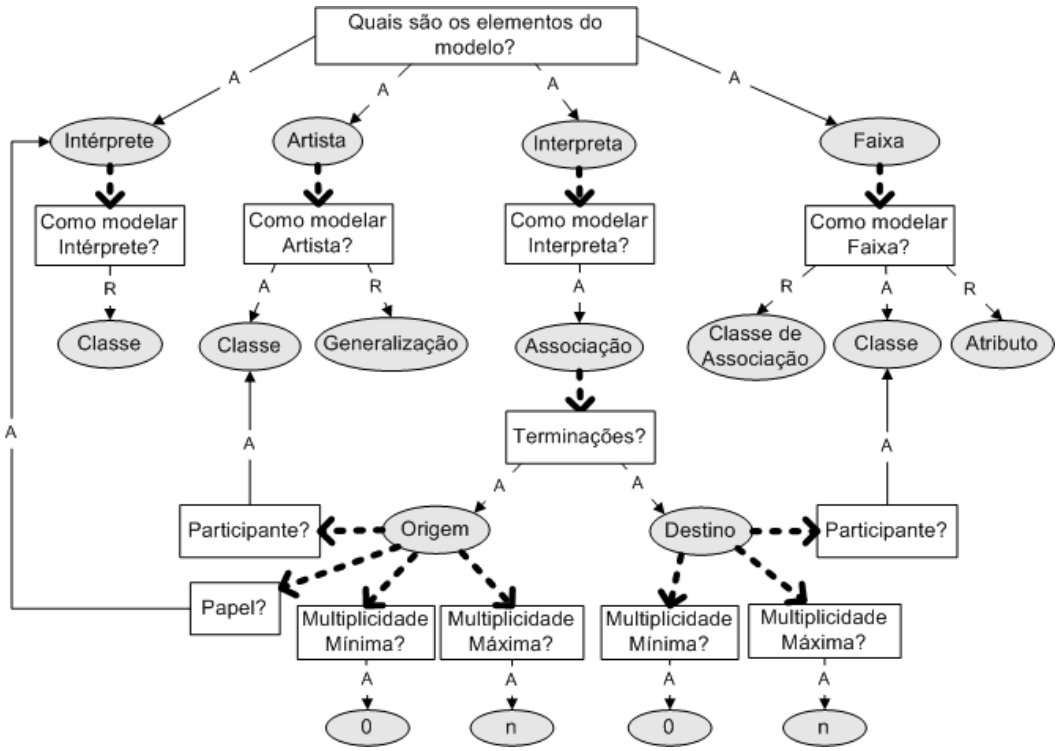

Figura 20 – Exemplo de *design rationale* sobre o elemento Interpreta

Neste exemplo de *design rationale*, podemos observar novamente a utilização do modelo formal da UML na definição das questões sobre como modelar o elemento "*Interpreta*". A idéia de solução inicial para esta questão é modelar o elemento Interpreta como uma associação. Como vimos anteriormente, uma associação é composta de duas ou mais terminações. Para cada uma dessas terminações é necessário definir qual é o elemento participante, o nome do papel que este elemento exerce na associação, as multiplicidades, etc. As terminações consideradas pelo projetista para a possível associação "*Interpreta*" estão representadas pelas idéias "*Origem*" e "*Destino*", que por sua vez sugerem as questões "*Participante?*", "*Papel*?", "*Multiplicidade Mínima?*" e "*Multiplicidade Máxima?*" relacionadas à definição de cada terminação. Note que a idéia "*Intérprete*", proposta inicialmente pelo projetista, também responde a questão "*Papel*?" sugerida pela terminação "*Origem*" na sub-árvore do elemento "*Interpreta*".

Assim, analisando o *design rationale* ilustrado na Figura 20, podemos concluir que o projetista decidiu modelar o elemento "*Interpreta*" como uma

associação binária (duas terminações) entre os elementos "*Artista*" e "*Faixa*", sendo que Artista possui o papel "*Intérprete*" nesta associação. Uma representação semelhante foi gerada para o *design rationale* sobre a modelagem do elemento "*Compõe*" como uma associação entre os elementos "*Artista*" e "*Faixa*". Nesta associação o artista possui o papel "*Compositor*".

Considerando todas as decisões registradas na representação de *design rationale* deste primeiro exemplo, podemos obter o design final do diagrama de classe UML que representa o esquema conceitual para catálogos de CDs, construído ao término do design. A Figura 21 ilustra o artefato final produzido.

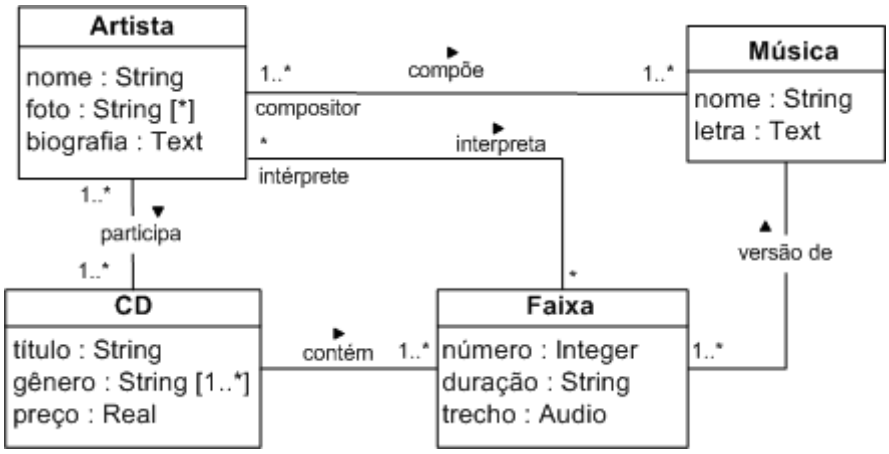

Figura 21 – Esquema Conceitual para o catálogo de CDs

4.2. Design Rationale de um Modelo de Navegação

Nesta seção apresentamos alguns exemplos de representação de *design rationale* de um modelo de navegação de uma aplicação *Web* para o catálogo de CDs, considerando alguns dos conceitos representados no esquema conceitual do exemplo anterior. Como podemos observar no quadro abaixo, neste exemplo o projetista escolheu o método OOHDM (*Object Oriented Hypermedia Design Method*) (Schwabe & Rossi, 1998) para projetar os aspectos de navegação de sua aplicação, definindo assim o modelo navegacional deste método como modelo formal para descrever seus artefatos.

```
/* Facts */
//Design Activicty Information ----------
oohdm:method[hasName->'Object-Oriented Hypermedia Design Method';
                    prescribes->>{navigational_model}].
navigational_model:formal_model[hasName->'Modelo Navegacional';
                                        hasLocalization->'http://www.tecweb.inf.puc-rio.br/oohdm/';
                                        isPrescribedBy->>{oohdm}].
navigationalDesignActivity:activity[hasName->'Navigational Design of the CD Catalogue';
                                        hasStartDate->'2005-11-07T09:30:00';
                                        hasFinishDate->'2005-11-17T10:10:00';
                                        hasExpectedDuration->duration1;
                                        requires->>{role1};
                                        isExecutedBy->>{adriana}].
duration1:expected_duration[hasAmount->10;
                                    hasUnitTime->'day'].
adriana:person[hasName->'Adriana Medeiros'].
role1:role[hasName->'designer'].
```

Quadro 6 - Exemplo de representação do *rationale* sobre uma atividade de design considerando o método OOHDM

O OOHDM é um método integrado para autoria de aplicações hipermídia. Ele provê primitivas de projeto de alto nível e mecanismos de abstração, baseados no paradigma da orientação a objetos, que permitem representar o design de aplicações hipermídia complexas que manipulam grande quantidade de informações estruturadas, tais como aplicações para a *Web*, apresentações multimídia, quiosques, etc.

O método OOHDM propõe as seguintes atividades para o processo de desenvolvimento de aplicações hipermídia:

- Levantamento de Requisitos,
- Projeto Conceitual,
- Projeto de Navegação,
- Projeto da Interface Abstrata e
- Implementação.

A atividade de levantamento de requisitos envolve a especificação de atores, tarefas, cenários, casos de uso e diagramas de interação do usuário (*User Interaction Diagrams*, *UIDs*), que descrevem as principais tarefas que o usuário deseja realizar usando a aplicação hipermídia sendo projetada. Um diagrama de interação do usuário representa graficamente a interação entre o usuário e a aplicação descrita textualmente em um caso de uso.

O projeto conceitual é a atividade responsável pela construção de um modelo conceitual da aplicação, utilizando os princípios da modelagem orientada a objetos, com o acréscimo de algumas primitivas, como perspectivas (múltiplos tipos) de atributo. Este modelo conceitual é definido a partir das descrições dos diagramas de interação do usuário descritos durante a atividade de levantamento de requisitos.

O projeto de navegação é a atividade responsável pela definição da estrutura de navegação de uma aplicação hipermídia. Geralmente, esta estrutura de navegação é definida com base nos artefatos produzidos nas atividades anteriores. No entanto, um projetista experiente pode usar o modelo formal do método OOHDM, que descreve os artefatos que compõem esta estrutura, para modelar a navegação de uma aplicação hipermídia, sem realizar as outras atividades propostas pelo método. Consideramos este cenário de design para os exemplos apresentados nesta seção.

O projeto da interface abstrata especifica como os objetos de navegação serão percebidos e apresentados aos usuários, e a atividade de implementação é responsável pela tradução do projeto da aplicação para um ambiente de implementação.

4.2.1. O Modelo Formal do OOHDM para Navegação

No método OOHDM, uma aplicação é vista como uma visão navegacional sobre o esquema conceitual, possibilitando que diferentes modelos de navegação sejam construídos sobre o mesmo domínio da aplicação, de acordo com o perfil dos usuários e tarefas que eles irão desempenhar (Moura, 1999).

Um modelo de navegação é composto por diferentes artefatos que especificam a estrutura de navegação de uma aplicação hipermídia. Estes artefatos incluem o esquema de contextos de navegação e o esquema de classes navegacionais e são formados pelas seguintes primitivas: classes navegacionais, elos, âncoras, índices, contextos de navegação e classes em contexto. A Figura 22 ilustra parte do modelo formal prescrito pelo método OOHDM para navegação.

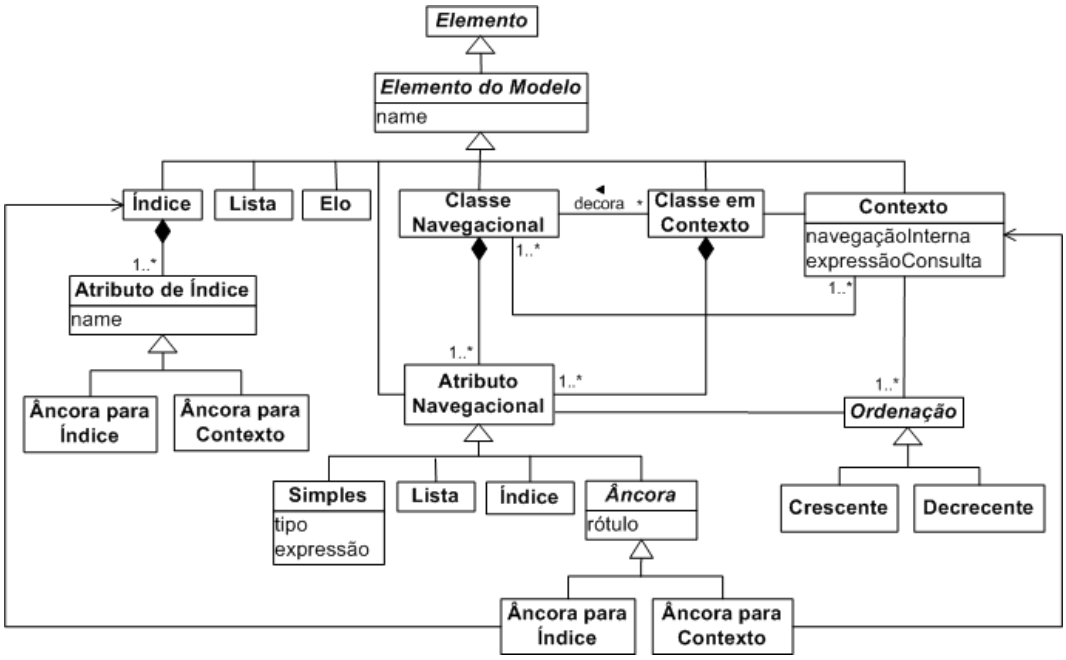

Figura 22 – Modelo Formal (parcial) do OOHDM para navegação

De acordo com este modelo formal, um elemento em um modelo de navegação pode ser basicamente um índice, uma lista, um elo, uma classe navegacional, uma classe em contexto ou um contexto. Contextos definem o conjunto de objetos de navegação que serão explorados pelo usuário em cada momento. Estes objetos são definidos com base nas classes navegacionais relacionadas ao contexto (escopo). Por exemplo, o contexto "*CDs da artista Daniela Mercury*" representa o conjunto de CDs que estarão acessíveis ao usuário, após ele ter selecionado sua artista favorita. Este contexto é formado de objetos da classe navegacional CD, que define as informações do CD (atributos navegacionais) que o usuário poderá consultar.

O acesso aos objetos de um contexto pode ser feito por um índice ou por uma âncora. Um índice é um conjunto de objetos ordenados, onde cada objeto tem pelo menos um atributo do tipo âncora (seletor). Esta âncora aciona um elo para um outro objeto, que pode ser um objeto de um contexto ou de um outro índice. O contexto "*CDs da artista Daniela Mercury*" citado anteriormente pode ser acessado, por exemplo, por um índice de artistas onde um dos objetos é "*Daniela Mercury*". A Figura 23 ilustra o esquema de contextos que representa esta navegação. Uma legenda para a notação é sumarizada no lado direito da figura. Note que o contexto "*CDs por Artista*" inclui um grupo de possíveis contextos,

cujos objetos são definidos pelo nome do artista selecionado pelo usuário no índice "*Artistas*".

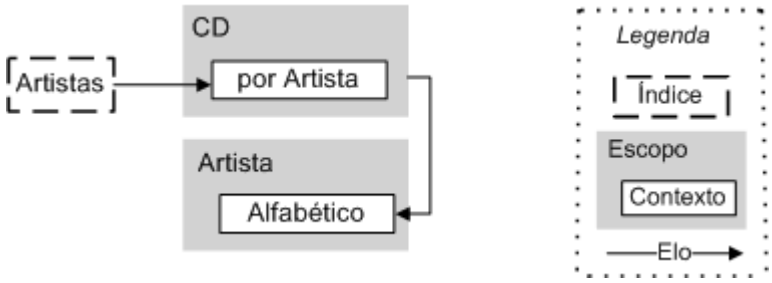

Figura 23 – Exemplo de esquema de contexto

Uma classe em contexto é definida como um "decorador" de uma classe navegacional com informações que só poderão ser acessadas dentro de um contexto específico. Por exemplo, podemos definir uma classe em contexto contendo uma âncora para dar acesso à biografia do artista apenas quando o usuário estiver consultando um CD no contexto "*CDs por Artista*". Assim, neste contexto, além das informações do CD, esta âncora será exibida permitindo que o usuário navegue para as informações do artista no contexto "*Artista Alfabético*", que contém todos os artistas organizados em ordem alfabética.

Algumas dessas definições descritas no modelo formal, ilustrado na Figura 22, representam questões de design que o projetista precisa tratar durante o design de um modelo de navegação usando o método OOHDM. Estas questões de design incluem: a definição dos conjuntos de objetos pelos quais os usuários finais da aplicação *Web* poderão navegar; a escolha das opções de design disponíveis para modelar os conjuntos definidos; a definição dos possíveis caminhos de navegação entre os diferentes conjuntos; e a definição de novos itens de informação para complementar a especificação dos conjuntos modelados, como por exemplo, a especificação da ordenação desejada para os objetos que formam contextos e índices. No caso dos conjuntos modelados como contextos, é necessário ainda definir a navegação interna entre os elementos de um mesmo contexto. A seguir, mostramos como as definições contidas no modelo formal do OOHDM para navegação podem ser usadas para representar as questões e idéias de solução tratadas pelo projetista durante o design de uma aplicação *Web*.

4.2.2. Rationale sobre o Design do Conjunto “CDs de um Artista”

O projeto de navegação de uma aplicação *Web* usando o método OOHDM normalmente envolve o design dos diferentes conjuntos de objetos, aos quais os usuários da aplicação terão acesso durante a navegação. Desta forma, o primeiro problema a ser resolvido pelo projetista é a definição dos conjuntos de objetos que os usuários poderão navegar, como ilustra a Figura 24. Esta figura representa uma parte do *design rationale* usado no design do esquema de contextos ilustrado na Figura 23.

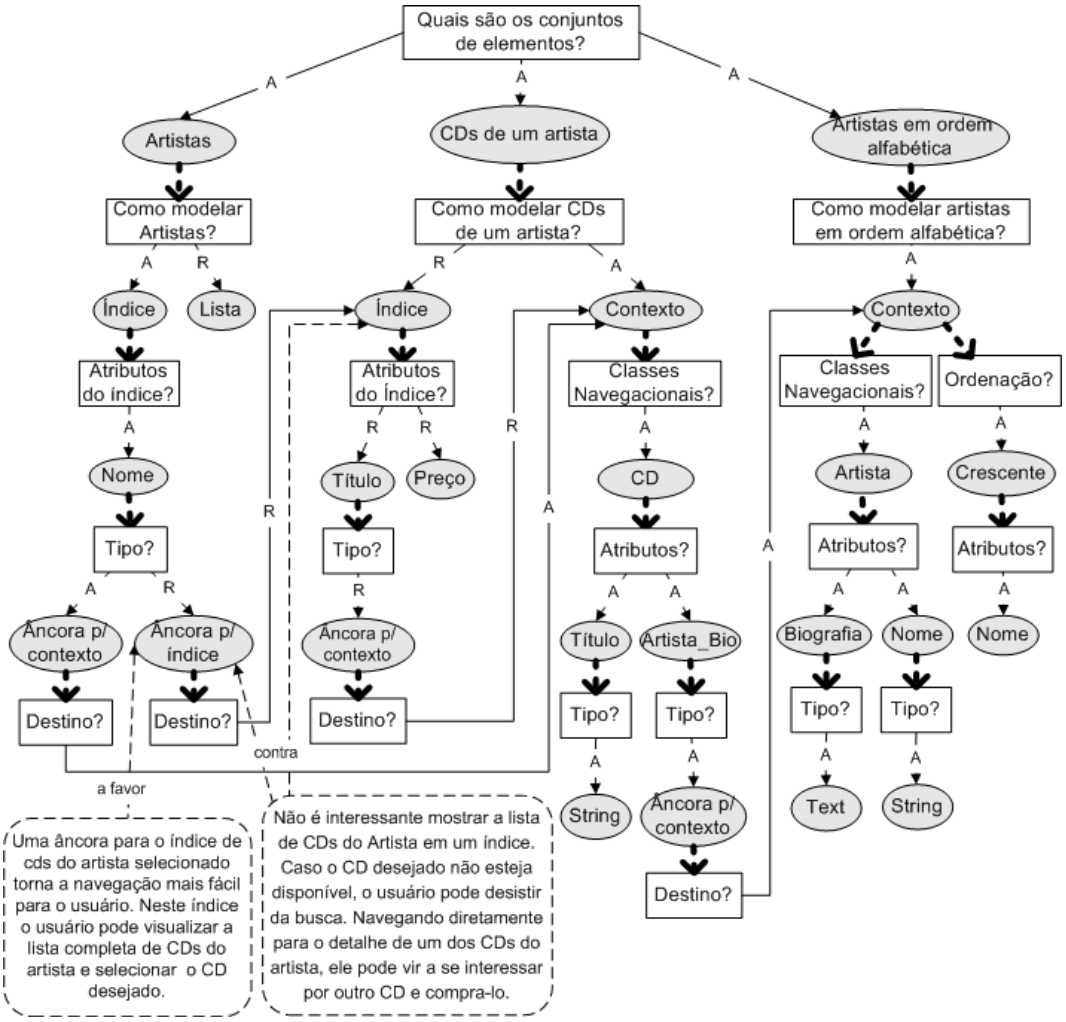

Figura 24 – Parte do *rationale* representando a navegação ilustrada na Figura 23

Nesta representação vemos que o projetista considerou inicialmente os conjuntos “*Artistas*”, “*CDs de um artista*” e “*Artistas em ordem alfabética*” como idéias para a questão inicial “*Quais são os conjuntos de elementos*?”. De acordo com o modelo formal OOHDM para navegação, estes conjuntos de elementos podem ser modelados basicamente como um índice, um contexto ou uma lista.

Desta forma, as idéias "*Índice*" e "*Contexto*" são instanciadas como opções de design para a questão "*Como modelar CDs de um artista?*".

Observe que quando o projetista considera as idéias de modelar os conjuntos "*Artistas*" e "CDs de um artista" como índices, as questões "*Atributos do índice?*" são imediatamente sugeridas. Isto ocorre porque um índice deve estar associado a um ou mais atributos, de acordo com o modelo formal do OOHDM ilustrado na Figura 22. Estas questões são respondidas pelas idéias "*Nome*", "*Título*" e "*Preço*", que correspondem aos respectivos atributos desses índices. Note que estes atributos podem ser do tipo "*Âncora para índice*" ou "*Âncora para contexto*" com destinos diferentes, dependendo da navegação desejada pelo projetista. As idéias propostas para o tipo dos atributos "*Nome*" e "*Título*" dos índices "*Artistas*" *e* "*Cds de um artista*", *respectivamente,* e aquelas que respondem a questão "*Destino?*" na Figura 24, definem duas soluções diferentes para a navegação da aplicação que está sendo projetada.

A primeira solução de navegação parte do índice "*Artistas*", através da âncora "*Nome*", para o contexto "*CDs de um artista*". Esta solução foi aceita pelo projetista (rótulos "A" nas setas entre a questão "*Destino?*" e a idéia "*Contexto*") e corresponde à solução de design final representada pelo esquema de contexto da Figura 23. Acompanhando o *rationale* desta solução, percebemos que o projetista decidiu incluir o atributo "*Artista_Bio"* do tipo "*Âncora para Contexto*" na classe navegacional "*CD*", proposta para o contexto "*Cds de um artista*". Esta âncora permite ao usuário navegar a partir das informações de um CD para a biografia do artista no contexto "*Artista em ordem alfabética"*.

Na segunda solução de navegação, rejeitada pelo projetista (rótulos "R" entre a questão "*Destino?*" e a idéia "*Índice*"), o usuário seleciona o artista desejado no índice "*Artistas*" para acessar a lista de CDs deste artista no índice "*CDs de um artista*". Então, ele pode selecionar o CD desejado usando a âncora "*Título*" neste índice para acessar informações detalhadas do CD no contexto "*CDs de um artista*". Observe que a idéia "*CDs de um artista*" teve duas idéias de solução propostas ("*Índice*" e "*Contexto*") e ambas poderiam ter sido aceitas pelo projetista como soluções para modelar este conjunto. A figura abaixo ilustra a solução de navegação rejeitada pelo projetista, de acordo com a representação de *design rationale* apresentada na Figura 24.

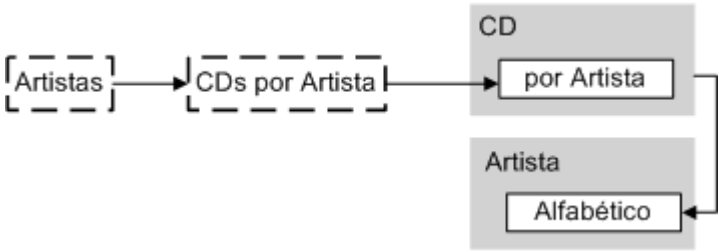

Figura 25 – Esquema de contexto da solução de navegação rejeitada pelo projetista

A partir do *rationale* ilustrado na Figura 24, o projetista pode produzir também parte do artefato que representa o esquema navegacional da aplicação de catálogos de CDs. Este esquema é formado pelas classes navegacionais associadas aos contextos de navegação e pelos elos que permitem a navegação entre eles. Pelo método OOHDM todo contexto está associado a um escopo formado por uma ou mais classes navegacionais. Isto é representado pelas questões "*Classes Navegacionais?*", sugeridas pelas idéias "*Contexto*" para os conjuntos "*CDs de um artista*" e "*Artistas em ordem alfabética*". Cada idéia de solução proposta para estas questões (idéias "*CD*" e "*Artista*") sugere a questão "*Atributos?*", indicando para o projetista que é necessário definir também os atributos que compõem cada classe navegacional proposta. Neste exemplo, podemos observar que o projetista decidiu modelar os atributos "*Nome*" e "*Biografia*" para a classe navegacional "*Artista*" e os atributos "*Título*" e "*Artista_Bio*" para a classe navegacional "*CD*", sendo este último um atributo do tipo âncora para contexto permitindo a navegação entre objetos destas classes. O esquema navegacional resultante dessas idéias é ilustrado na figura abaixo.

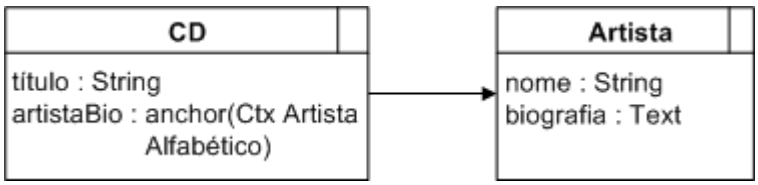

Figura 26 – Exemplo de esquema navegacional

O esquema navegacional acima mostra a notação OOHDM usada para representar classes navegacionais e elos. Nesta notação, uma barra vertical no canto superior direito da classe é usada para diferenciar uma classe navegacional de uma classe conceitual.

No exemplo de *design rationale* apresentado nesta seção podemos notar que a forma de representar o conteúdo fornecido pelos projetistas durante o design é

um pouco diferente daquela usada na representação de *design rationale* do esquema conceitual, apresentado na seção 4.1. Na representação ilustrada na Figura 24, as idéias de domínio fornecidas pelos projetistas respondem questões de design registradas em diferentes partes da representação (diferentes níveis da árvore), enquanto nos exemplos de *design rationale* apresentados na seção 4.1 as idéias de domínio respondem apenas a questão inicial. Por exemplo, na Figura 24 as idéias de domínio "*Nome*", "*Título*" e "*Preço*" respondem questões sugeridas pelas idéias de design "*Índice*". Isto mostra que existe uma certa flexibilidade na construção da representação de *design rationale*, permitindo que os elementos de raciocínio sejam registrados de acordo com as características específicas de cada design.

4.2.3. Rationale sobre o Design do Conjunto "Artistas"

No exemplo anterior, apresentamos o *design rationale* de um esquema de contextos OOHDM para representar os possíveis caminhos de navegação que o usuário pode seguir para acessar os diferentes tipos de informações presentes na aplicação *Web* para o catálogo de CDs. Estes caminhos de navegação podem ser definidos entre contextos diferentes e entre índices e contextos. No entanto, cada contexto modelado em um esquema de contexto possui também uma navegação interna que permite ao usuário navegar entre os objetos de um mesmo contexto. A Figura 27 ilustra parte do *rationale* usado pelo projetista para definir esta navegação interna e outros aspectos relacionados à especificação do contexto "*Artista em ordem alfabética*".

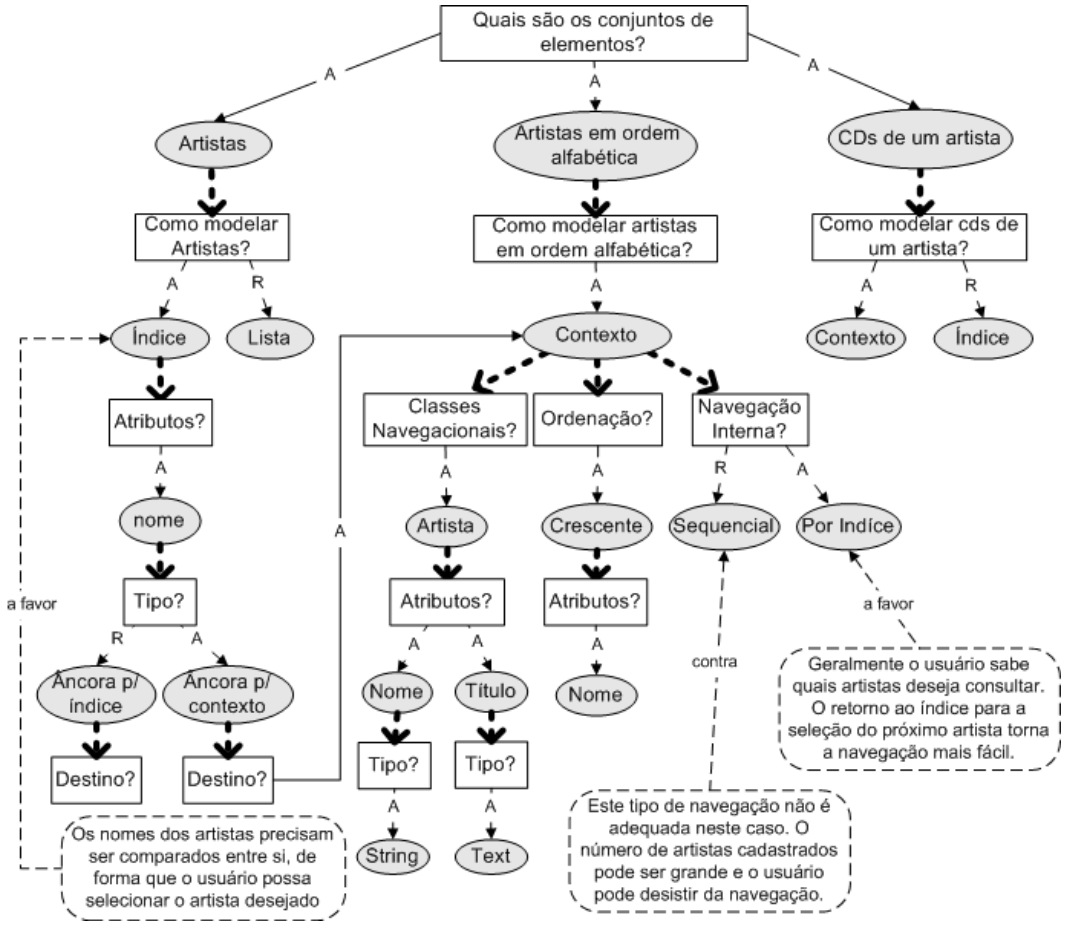

Figura 27 - Parte do *design rationale* representando a especificação de um contexto

Neste exemplo, podemos observar que além da questão "*Classes Navegacionais?*", as questões "*Ordenação?*" e "*Navegação Interna*?" também são sugeridas pela idéia "*Contexto*" proposta para a questão sobre como modelar o conjunto "*Artistas em ordem alfabética*". Estas questões são definidas com base no modelo formal do OOHDM e representam alguns dos problemas de design que o projetista precisa resolver para modelar um conjunto de objetos como um contexto. Um destes problemas é a definição da ordenação usada para organizar os objetos que formam o contexto, representado pela questão *Ordenação?*. Na Figura 27, podemos observar que o projetista decidiu ordenar os objetos do contexto "*Artista em ordem alfabética*" em ordem crescente de nomes, o que corresponde exatamente à intenção do projetista indicada pelo nome do conjunto proposto inicialmente como uma idéia para a questão inicial "*Quais são os conjuntos de elementos?*".

Um outro problema representado neste exemplo é a definição da navegação interna que o usuário poderá seguir quando estiver consultando os objetos deste

contexto. De acordo com o modelo formal do método OOHDM, ilustrado na Figura 22, o elemento “Contexto” possui um atributo “*navegaçãoInterna*” que é uma enumeração contendo os seguintes valores: navegação seqüencial, navegação circular, navegação por índice e navegação livre. Estes valores representam idéias que o projetista pode propor como respostas para a questão “*Navegação Interna*?”. No *design rationale* ilustrado na Figura 27, vemos que o projetista considerou as idéias de solução “*Seqüencial*” e “*Por Índice*”, decidindo aceitar esta última. Na navegação interna “por índice”, a navegação é feita apenas do índice para um elemento do contexto e vice-versa. Não existe a possibilidade de navegação entre os outros elementos do contexto, a não ser através do índice de entrada do contexto. Na navegação seqüencial, após a entrada no contexto o usuário pode navegar por todos os elementos do contexto seguindo uma ordem seqüencial e pré-estabelecida de navegação, incluindo os conceitos de “primeiro”, “último”, “próximo” e “anterior” para os elementos do contexto.

Considerando os argumentos representados na Figura 27 para cada uma dessas opções de navegação interna, a navegação “por índice” é realmente a mais adequada, uma vez que ela pode facilitar a navegação entre os elementos de um contexto composto por um grande número de objetos. Em uma navegação seqüencial, o usuário teria que percorrer vários artistas até encontrar o artista desejado e, em alguns casos, poderia desistir da navegação. No entanto, neste exemplo o projetista poderia ter aceitado também a idéia “*Seqüencial*” definindo assim uma combinação de navegações para os objetos do contexto. Esta combinação permitiria ao usuário navegar pelos elementos do contexto indistintamente por índice ou através de âncoras como “anterior” e “próximo”, flexibilizando um pouco mais a navegação neste contexto.