3
Transições Animadas

Neste capítulo definimos o processo de transição e descrevemos a abordagem elaborada para obtenção de transições animadas, identificando e exemplificando todas as etapas necessárias para a inserção de animações em aplicações *Web*.

O uso de animações tem se concentrado na definição de animações, sobre elementos de interface individualmente, como forma de enfatizar ações que ocorrem durante a interação do usuário. Definimos uma abordagem para definição de animações de forma sistemática e baseada em modelos, que considera toda mudança de interface que ocorre durante o processo de transição, intrínseco às navegações em aplicações hipermídia. As animações são definidas de forma a possibilitar transições suaves, sendo executadas durante a mudança navegacional.

A abordagem estende o uso da metodologia SHDM/OOHDM, dando ênfase à etapa de modelagem de interfaces abstratas, que produz o artefato sobre o qual definimos uma especificação de transições suaves conforme detalhado posteriormente.

3.1. Transições

Como as animações são apresentadas durante o processo de transição, caracterizando uma transição suave ou animada, é importante ressaltar, inicialmente, o que caracteriza tal processo de transição em uma aplicação hipermídia. Durante a navegação em sistemas hipermídia estamos percorrendo os nós navegacionais através dos *hiperlinks* e âncoras que os interconectam, realizando uma transição entre os estados navegacionais e definindo um caminho navegacional. As transições navegacionais ocorrem com freqüência durante o processo de interação. Ilustramos na Figura 6 três nós navegacionais, destacando uma possível transição entre dois deles.

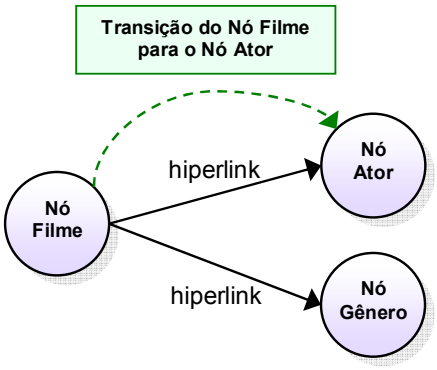

Figura 6 – Transição entre nós navegacionais

As transições geralmente resultam na substituição parcial ou total dos elementos de informação exibidos, apresentando as informações contidas no novo estado navegacional ou nó destino. Quando realizamos tais transições utilizando animações para apresentar as alterações, modificando e exibindo de forma gradual as informações até atingir a interface final do estado destino, definimos uma transição suave. É importante ressaltar que podem ocorrer também transições que são resultados de mudanças de interface, mas que não caracterizam uma mudança navegacional, pois não ocorre uma mudança do nó navegacional. Como uma mudança sobre a forma de visualização dos itens apresentados, ou como por exemplo em uma interface que apresenta um índice com apenas parte dos elementos que o compõem, poderia haver uma transição em que uma nova interface é apresentada com o restante dos elementos do índice. Essas transições também podem ser representadas com animações pois podem apresentar diferentes widgets em cada estado do processo.

Podemos identificar quais as possíveis transições ou caminhos de navegação que resultam em uma transformação da interface através do esquema de contextos navegacionais, definido durante a etapa de modelagem navegacional. Neste esquema podem se representar três primitivas navegacionais: os *landmarks*, as estruturas de acesso e os contextos de navegação. Os *landmarks* são ancoras que podem ser acessadas a partir de qualquer estado navegacional da aplicação, e sendo assim, estarão sempre presentes na interface. Já as estruturas de acesso ou índices, representam um conjunto de âncoras que podem apontar para outros índices ou para nós em um determinado contexto navegacional, exigindo a definição de uma interface específica para sua representação. Por fim, os contextos definem um conjunto de nós, possibilitando acesso às instâncias das

classes navegacionais, exigindo também uma interface específica. Dada as possíveis interfaces, e considerando as possíveis mudanças navegacionais, podemos dizer que as transições suaves podem ser definidas para as seguintes alterações:

- Entre diferentes Contextos;
- Entre Nós de um mesmo Contexto;
- De um índice para outro índice;
- De um índice para um contexto;
- De um contexto para um índice.

Podemos ainda representar alterações sobre interfaces que definem informações adicionais específicas não relacionadas ao modelo conceitual da aplicação (por exemplo, *layouts*) ou mesmo sobre ações que caracterizam mudanças de interface que não caracterizam mudanças navegacionais. Sendo assim, estas representações podem ser definidas sem considerar qualquer relação com elementos com o modelo.

Em nossa proposta, as animações definidas serão apresentadas durante uma tarefa de navegação no sistema, representando apenas transições resultantes de uma mudança de estado navegacional. Sendo assim, embora seja possível a sua realização, não estamos definindo uma forma sistemática para elaboração de animações onde ocorrem alterações de layouts (alterações sobre formas de visualização em uma mesma interface), estilos e por fim, transições que não representam mudança navegacional (por exemplo, itens que não caberiam em uma mesma pagina são quebrados em diversas páginas, sendo assim, ao navegar por essas paginas não ocorrerá mudança do nó navegacional)

As transições podem ser representadas como uma interface intermediária entre um conjunto de estados de interface origem/destino. Este estado intermediário, entretanto, é apenas uma representação das transformações que ocorrerem nos elementos da interface, podendo ser expressa como uma função que interpreta um conjunto de modificações e dispensando assim a definição de uma interface específica para a transição.

Todas as interfaces definidas representam informações, sendo compostas de diversos elementos de interface, ou *widgets* concretos, que são mapeados a partir

da ontologia de *widgets* abstratos através da ontologia de widgets concretos da metodologia SHDM. Durante a transição, geralmente alteramos a interface apresentada e conseqüentemente redefinimos seus elementos. Sendo assim, os *widgets* apresentados na interface origem poderão se manter, alterar ou desaparecer e, da mesma forma, novos *widgets* da interface destino poderão aparecer.

A Figura 7 abaixo ilustra a transição suave com uma interface intermediária lembrando, entretanto que esta interface animada representa apenas um conjunto das modificações dos elementos da interface origem/destino, dispensando a elaboração de uma interface distinta específica.

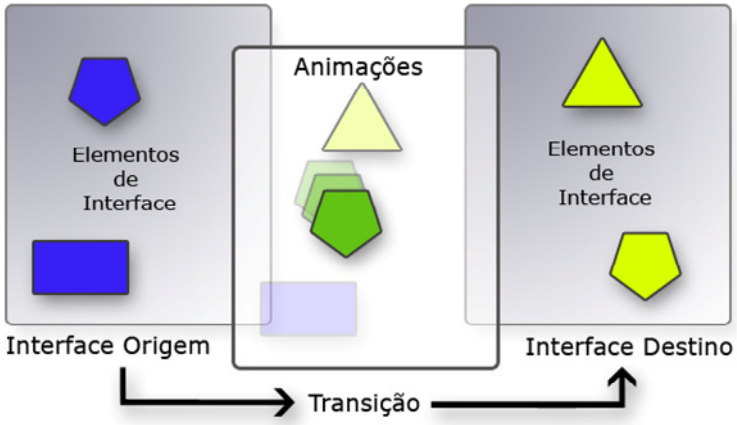

Figura 7 – Transição suave entre interfaces distintas

3.2. Etapas da Abordagem

A abordagem elaborada define o processo de inserção de animações em três etapas ou atividades principais, cada qual produz um artefato para utilização na etapa seguinte, conforme descrito pela Tabela 4 a seguir:

Tabela 4 – Atividades e artefatos da abordagem

Atividade	**Artefato Produzido**
Modelagem das Interfaces	Especificação de Interface Abstrata; Ontologia de Interface Concreta
Modelagem das Transições	Especificação das Animações ; Estrutura Retórica de Transições, Estilo Retórico
Interpretação da Especificação de Transições	Animação da Interface Concreta

Após a definição dessas etapas, possibilitamos a execução das animações para cada transição suave. Descrevemos a seguir cada uma das etapas do processo. Para exemplificação dos modelos resultantes das etapas, representaremos um sistema sobre o domínio de filmes, simplificado no esquema de contexto navegacional, representado pela Figura 8.

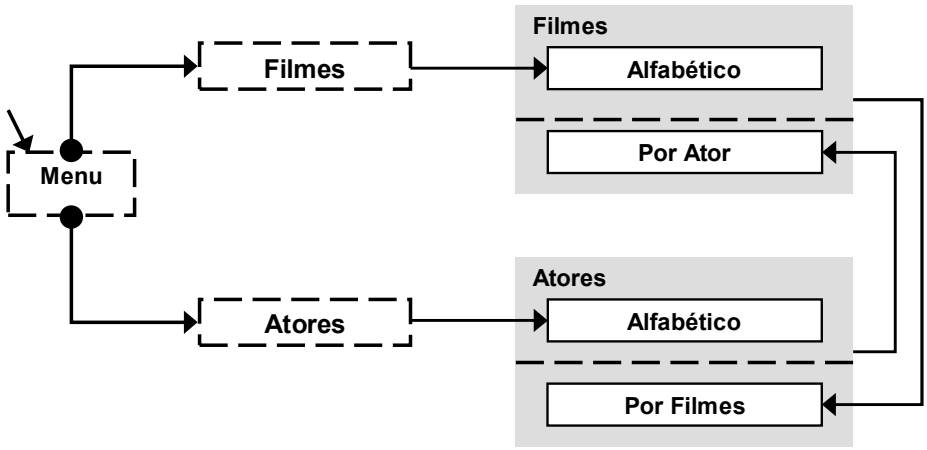

Figura 8 – Um esquema de contextos navegacionais para o domínio de filmes

3.2.1. Modelagem da Interface

Para definição das transições animadas, precisamos inicialmente identificar quais interfaces irão participar da transição e quais *widgets* serão apresentados em cada das interfaces. Podemos assim, definir quais transformações devem ocorrer para obtermos a interface destino a partir do *widget* acionado na interface origem, durante a interação. Com o propósito de especificar essas informações, realizamos a modelagem de interfaces.

Na modelagem da interface definimos então quais *widgets* irão compor as interfaces do sistema, identificando assim, como as informações serão apresentadas e possibilitando a interação do usuário com o sistema. Esta etapa segue a abordagem da metodologia SHDM, definida como modelagem de interface abstrata, onde são utilizadas as ontologias de interface abstrata e concreta para descrição dos elementos de interface. Durante esta modelagem definiremos a partir do modelo navegacional quais objetos navegacionais irão compor a interface na forma de *widgets* abstratos. Conforme descrito no capítulo 2, os *widgets* que compõem a interface abstrata poderão ser de quatro tipos

distintos: ativadores (*SimpleActivator*), exibidores (*ElementExhibitor*), capturadores (*ValueCapturer*) ou compostos (*CompositeInterfaceElement*).

A estrutura exibida pela Figura 9 representa uma interface abstrata do contexto Filmes por ordem alfabética, apresentando todos os *widgets* que a compõem.

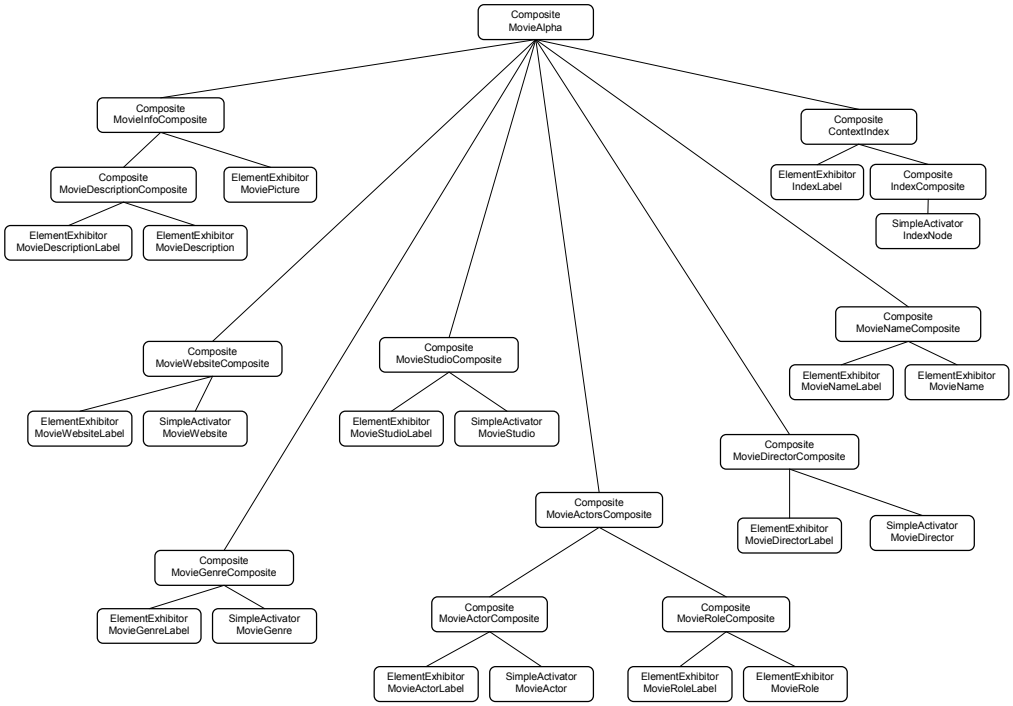

Figura 9 – Widgets que compõem uma interface do contexto filmes por ordem alfabética

Para cada *widget* abstrato apresentado, precisamos definir as propriedades que indicam qual a informação apresentada, sua relação com objetos do modelo navegacional e demais características de sua composição. Na abordagem propomos uma simplificação do conjunto de propriedades definidas na modelagem de interfaces abstratas da metodologia SHDM. As propriedades definem também informações que poderão ser utilizadas durante a execução das transições, possibilitando assim uma comparação entre *widgets* distintos. Descrevemos as propriedades dos *widgets* a seguir:

- *MapsTo*: Indica qual o elemento da ontologia de *widgets* concretos sobre o qual o elemento de interface abstrato será mapeado.

- *fromElement*: Indica qual o elemento do modelo navegacional que o *widget* abstrato representa. O elemento poderá ser um contexto, um índice ou uma âncora.
- *fromAttribute*: Indica qual o atributo da instância do modelo navegacional que o *widget* abstrato apresenta.
- *Value*: Indica o dado que será apresentado pelos *widgets* abstratos do tipo *ElementExhibitor*.
- *HasInterfaceElement*: Propriedade restrita à classe de elementos *CompositeInterfaceElement*, onde definimos quais os widgets abstratos que irão compor o elemento de interface.
- *isRepeated*: Outra propriedade restrita à classe de elementos *CompositeInterfaceElement*, que define se na estrutura do *widget* irá ocorrer a repetição das informações representadas pelos elementos que o compõem.

É importante ressaltar que a especificação de algumas propriedades apresentadas não precisam ser definidas de forma explícita, podendo ser obtidas a partir de outras propriedades e outros *widget*s já definidos. A propriedade *HasInterfaceElement* pode ser definida implicitamente através da descrição da especificação com uma estrutura hierárquica, definida através do aninhamento dos elementos representados na sua composição. A equivalência entre as duas formas de especificações é representada pela Figura 10, a seguir.

```
<Composite Name="MoviesAlpha", isRepeated=false, HasInterfaceElement=[ MovieNameComposite]>

<Composite Name="MovieNameComposite", Repeat=false, Element=MoviesAlpha, Attribute=Name,
 HasInterfaceElement=[ MovieNameLabel, MovieName]>

<Exhibitor Name="MovieNameLabel", Value="Name:">
<Exhibitor Name="MovieName", Element=MoviesAlpha, Attribute=Name >
```

```
<Composite Name="MoviesAlpha", isRepeated=false >

       <Composite Name="MovieNameComposite", Repeat=false, Element=MoviesAlpha, Attribute=Name >

              <Exhibitor Name="MovieNameLabel", Value="Name:">
              <Exhibitor Name="MovieName", Element=MoviesAlpha, Attribute=Name >

       <\Composite>

<\Composite>
```

Figura 10 – Formas de especificação de interface com a propriedade *HasInterfaceElement*

Na abordagem SHDM, existem ainda propriedades de *widgets* abstratos (*blockElement* e *compositionTag*) onde são definidos *tags HTML* e atributos identificadores que permitem marcar elementos como objetos específicos HTML, bem como identificar os elementos para a definição de estilos CSS específicos. Em nossa abordagem, simplificamos o modelo, deixando que tais propriedades sejam definidas através do processo de mapeamento, definido adiante.

Para melhor representação das interfaces abstratas, definimos uma notação gráfica para identificação das diferentes classes de widgets de acordo com a Figura 11. Esta representação foi elaborada de forma a facilitar a comparação entre os elementos que compõem interfaces distintas.

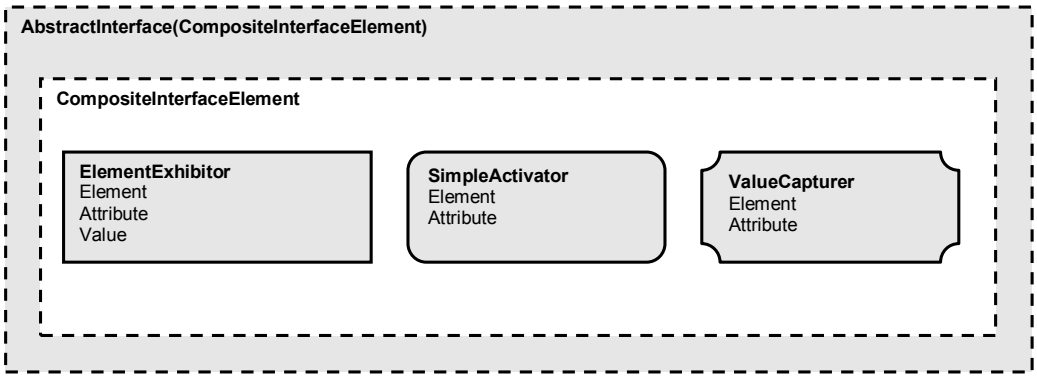

Figura 11 – Notação gráfica para representação de interfaces abstratas

Para exemplificar a representação gráfica de interfaces abstratas, apresentamos na Figura 12 e Figura 14, duas interfaces, a primeira representando um índice de filmes e a segunda representando o contexto filmes por ordem alfabética e na Figura 13 e Figura 15 as representações das interfaces abstratas para cada uma delas, respectivamente.

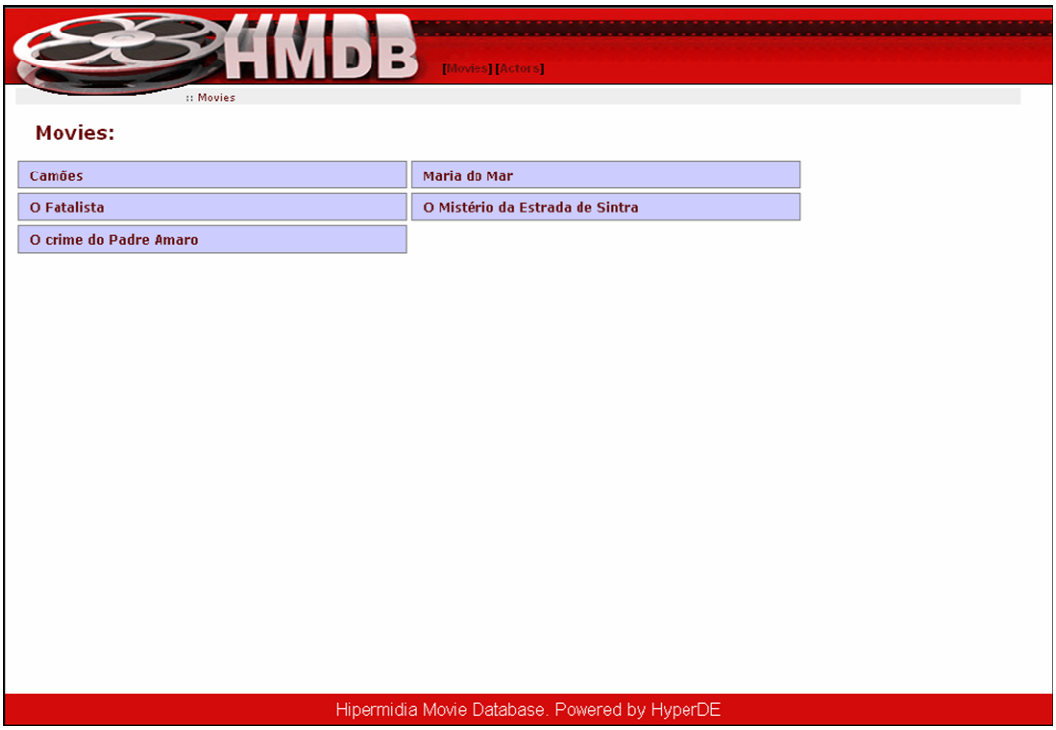

Figura 12 – Interface concreta para um índice de filmes

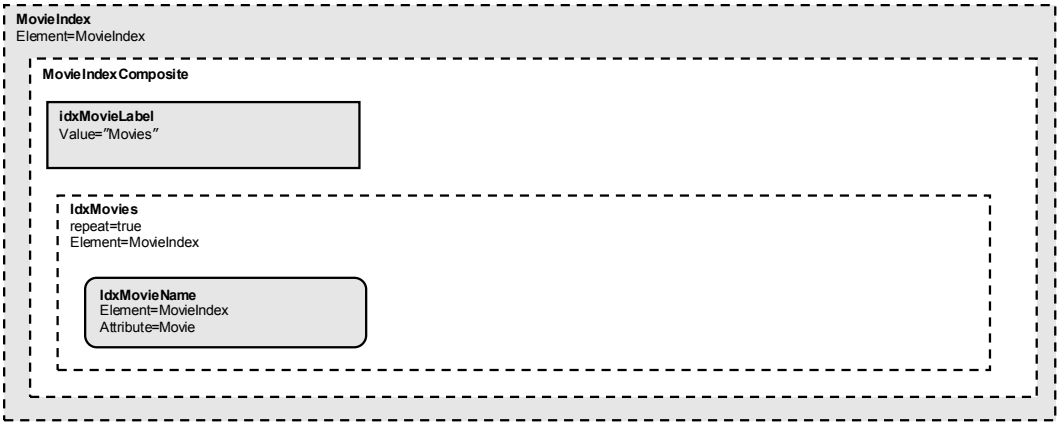

Figura 13 – Interface abstrata para um índice de filmes.

Figura 14 – Interface concreta para um contexto de filmes por ordem alfabética

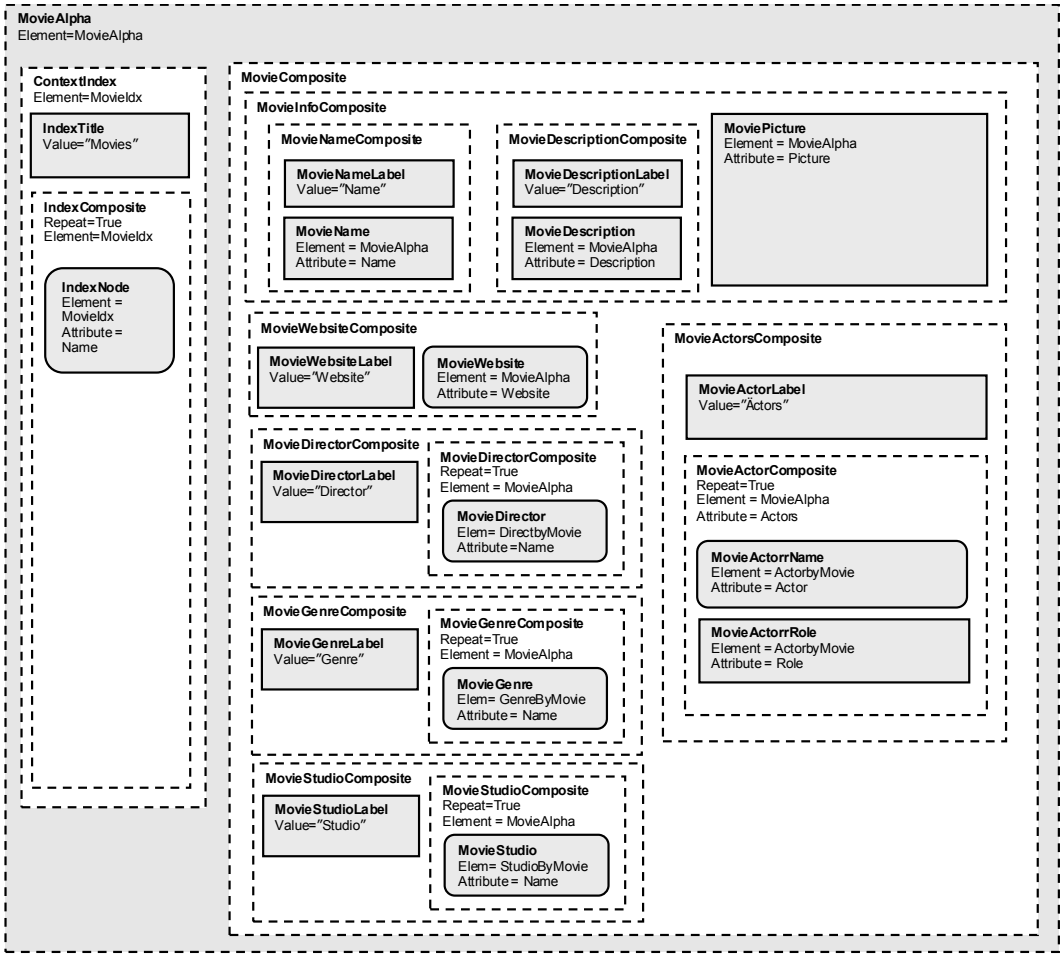

Figura 15 – Interface abstrata para o contexto filmes por ordem alfabética

Definimos ainda na Figura 16 abaixo a especificação da interface abstrata do contexto Filmes por ordem Alfabética, conforme ilustrada pela Figura 15 anterior.

```
<Composite Name="MoviesAlpha", IsRepeated=false, Element=MoviesAlpha>
        <Composite Name="MovieInfoComposite ", IsRepeated=false, Element=MoviesAlpha >
                <Exhibitor Name="MoviePicture", Element=MoviesAlpha, Attribute=Picture>
                <Composite Name="MovieNameComposite ", IsRepeated=false, Element=MoviesAlpha>
                        <Exhibitor Name="MovieNameLabel", Value="Name:">
                        <Exhibitor Name="MovieName", Element=MoviesAlpha, Attribute=Name>
                <\Composite>
                <Composite Name="MovieDescriptionComposite", IsRepeated=false, Element=MoviesAlpha>
                        <Exhibitor Name="MovieDescriptionLabel", Value="Description:">
                        <Exhibitor Name="MovieDescription", Element=MoviesAlpha, Attribute=Description>
                <\Composite>
        <\Composite>
        <Composite Name="MovieWebsiteComposite", IsRepeated=false, Element=MoviesAlpha>
                <Exhibitor Name="MovieWebsiteLabel", Value="Website:">
                <Activator Name="MovieWebsite", Element=MoviesAlpha, Attribute=Website>
        <\Composite>
        <Composite Name="MovieDirectorsComposite", IsRepeated=false Element=MoviesAlpha>
                <Exhibitor Name="MovieDirectorsLabel", Value="Directors:">
                <Composite Name="MovieDirectors", IsRepeated=false Element=MoviesAlpha, Attribute=Directors>
                        <Activator Name="MovieDirector", Element=DirectorsByMovieIdx, Attribute=Name>
                <\Composite>
        <\Composite>
        <Composite Name="MovieGenresComposite", IsRepeated=false, Element=MoviesAlpha>
                <Exhibitor Name="MovieGenresLabel", Value="Genres:">
                <Composite Name="MovieGenres", IsRepeated=true, Element=MoviesAlpha, Attribute=Genres>
                        <Activator Name="MovieGenre", Element=GenresByMovieIdx, Attribute=Name>
                <\Composite>
        <\Composite>
        <Composite Name="MovieStudiosComposite", IsRepeated=false, Element=MoviesAlpha>
                <Exhibitor Name="MovieStudiosLabel", Value="Studios:">
                <Composite Name="MovieStudios", IsRepeated=true, Element=MoviesAlpha Attribute=Studios>
                        <Activator Name="MovieStudio", Element=StudiosByMovieIdx, Attribute=Name>
                <\Composite>
        <\Composite>
        <Composite Name="MovieActorsComposite", IsRepeated=false, Element=MoviesAlpha >
                <Exhibitor Name="MovieActorsLabel", Value="Actors:">
                <Composite Name="MovieActorComposite", IsRepeated=true, Element=ActorbyMovieIdx>
                        <Activator Name="MovieActorName", Element=ActorbyMovieIdx Attribute=Actor>
                        <Exhibitor Name="MovieActorRole", Element=ActorbyMovieIdx Attribute=Role>
                <\Composite>
        <\Composite>
        <Composite Name="ContextIndex", IsRepeated=true, Element=MovieIndex>
                <Exhibitor Name="IndexLabel", Value="Movie Index">
                <Composite Name="IndexComposite", IsRepeated=true, Element=MovieIndex>
                        <Activator Name="IndexNode", Element=MovieIndex Attribute=Name>
```

Figura 16 – Especificação de uma interface do contexto filmes por ordem alfabética

Após a definição das interfaces abstratas, podemos definir a interface concreta que será apresentada ao usuário. Este processo segue a etapa de modelagem de interfaces abstratas, onde realizamos o mapeamento dos *widgets* abstratos para *widgets* concretos, utilizando a ontologia de widgets concretos, onde alguns *widgets* concretos podem estar restritos a determinados tipos de *widgets* abstratos. Essa ontologia permite identificar quais os possíveis elementos concretos que poderão ser apresentados, podendo ser estendido de acordo com a tecnologia ou ambiente utilizado.

Na Figura 12 e Figura 14, ilustradas anteriormente, representamos Interfaces Concretas, mapeadas a partir das interfaces abstratas apresentadas pela Figura 13 e Figura 15, respectivamente. Tais interfaces representam documentos HTML, renderizados por um navegador. Para o mapeamento, utilizamos uma ontologia de widgets concretos onde definimos objetos HTML, em sua maioria, estruturas de

blocos, permitindo a sua definição de estilos através de CSS (*Cascading Style Sheets*).

Conforme descrito anteriormente, assim como na abordagem SHDM, podemos identificar propriedades como *tags* e classes CSS, permitindo que os elementos sejam renderizados de formas específicas no navegador e permitindo também, identificar atributos para referenciar os *widgets* concretos e assim, acessá-los na estrutura do documento. Na nossa abordagem, optamos por não especificar os *tags* através de propriedades do *widget* abstrato, deixando que estes sejam definidos através de objetos na ontologia de widgets concretos. Já para definição de classes ou demais atributos que identificam um elemento no documento, associamos uma propriedade (*WidgetID*) ou um conjunto de propriedades (*fromElement* e *fromAttribute*) ao atributo concreto deste elemento. A forma como a identificação é realizada depende de como o processo de mapeamento será implementado.

3.2.2. Modelagem das Transições

Durante a etapa de modelagem de transições, especificamos as animações que irão ocorrer durante a mudança navegacional e que serão interpretadas durante a execução do sistema, caracterizando assim, transições suaves. A modelagem pode ser subdividida em três passos conforme a Tabela 5, tendo ao final dessas etapas toda definição de transições suaves que irão compor o sistema hipermídia.

Tabela 5 – Atividades para modelagem de transições

Atividade	Descrição
Identificação das Transições	Definição das propriedades que indicam os estados navegacionais que definirão transições suaves
Especificação das Animações de Transição	Definição do conjunto de animações que compõem cada transição suave
Especificação da Estrutura Retórica de Transição	Definição da estrutura retórica para cada transição suave.
Especificação dos Estilos Retóricos de Transição	Definição dos estilos retóricos da transição suave.

Descrevemos melhor a seguir como é o processo de elaboração de cada um dos passos necessários para a modelagem das transições.

3.2.2.1. Identificação das Transições

Como descrito anteriormente, as transições ocorrem devido a uma interação que resulta na alteração entre dois estados navegacionais distintos, mudando assim, os *widgets* presentes em cada estado, ou as informações apresentadas por eles.

Cada um dos possíveis estados navegacionais é representado por uma interface que foi especificada na modelagem de interfaces, sendo assim, para a modelagem das transições, o primeiro passo é identificar através de um par de interfaces origem/destino, quais transições se deseja especificar. Os pares de interfaces definem uma única transição, sendo assim, só podemos ter uma única definição para cada par de interfaces identificados. Após esta definição poderemos indicar então quais animações irão ocorrer e como essas animações serão apresentadas. Sendo assim, para cada transição deveremos definir as duas propriedades a seguir:

- *SourceInterface*: Indica a interface de origem, representando o estado navegacional inicial da transição.
- *TargetInterface*: Indica a interface final, representando o estado navegacional que será apresentado ao fim da transição.

3.2.2.2. Especificação das Animações de Transição

Nesta etapa estabelecemos quais animações serão executadas durante a transição, definindo assim, como as transformações realizadas sobre os *widgets* que compõem as interfaces serão apresentadas. Através desta especificação, definimos então a propriedade *Animations* de cada transição.

Para realização desta especificação temos que identificar inicialmente como cada elemento dos pares de interface origem/destino comportar-se-ão durante a mudança de estado navegacional. Ao analisar duas instâncias de interfaces definidas, podemos compará-las, identificando o que existe de similar, quais as diferenças e estabelecer relações entre as informações apresentadas. Nas

animações queremos apresentar o mesmo princípio de cinematografia, onde indicamos a relação entre as informações importantes durante a mudança, representando assim continuidade na interpretação (May et al. 2003). Esta interpretação é feita utilizando o artefato produzido pela modelagem de interfaces, onde estão descritos todos os *widgets* que compõem cada interface. Os widgets são comparados identificando qual a classe navegacional e o atributo da classe que esta sendo representado por suas informações apresentadas. Através desta propriedade podemos observar *widgets* que apresentam uma mesma informação, identificando similaridades e diferenças entre o conjunto de elementos apresentados. Sendo assim, nos referimos aos elementos das transições como itens de informação, que apresentam dados de instâncias de elementos do modelo navegacional.

Observando as possíveis alterações existentes, foi estabelecido que durante uma transição um elemento, poderá se comportar das seguintes maneiras:

- Um elemento presente na interface origem poderá estar presente também na interface destino, mantendo os mesmos parâmetros de estilo, e conseqüentemente não realizando qualquer alteração. Este comportamento dispensa o uso de qualquer animação.
- O elemento poderá estar presente em ambas as interfaces, porém com parâmetros distintos, exigindo assim, uma transformação sobre suas características e sugerindo o uso de uma animação para manutenção do elemento.
- Um elemento da interface origem poderá não estar presente em uma interface destino, desaparecendo durante a transição. Este comportamento sugere o uso de uma animação para a saída do elemento.
- O elemento poderá estar presente somente na interface destino, inexistindo no estado de origem, surgindo durante a transição. Este comportamento sugere o uso de uma animação para a entrada do elemento.
- Um elemento da interface origem que não está presente na interface destino, mas que possui nela um elemento relacionado, poderá realizar alguma transformação que apresente esta relação, sugerindo uma animação para uma substituição de elementos.

- Um elemento da interface poderá realizar transformações arbitrárias como forma de enfatizar sua presença ou representar alguma ação, independente da sua relação com elementos do outro estado de navegação. Esse elemento pode ainda assumir posteriormente, qualquer outro dos comportamentos definidos. Este comportamento sugere o uso de uma animação para destacar o elemento.

Apresentamos na Interface origem os elementos a seguir:

A – Nome da Atriz (Amália)
B – Nome do Filme (A Arte de Amália)
C – Cartaz do Filme (A Arte de Amália)
D – Foto da Atriz (Amália)
E – Foto do Ator (Joaquim)
F – Nome do Ator (Joaquim)

Na transição é realizado uma animação sobre o item **A**, que indica o item que foi clicado enfatizando-o como forma de fornecer feedback à ação executada.

Os itens **A**, **B** e **C** estarão presente na interface destino, logo, se mantém durante toda transição realizando animações de transformações sobre seu posicionamento e tamanho.

Items como **E** e **F** não estão presentes na interface destino, desaparecendo durante a transição.

Como na interface destino temos um elemento (item **I**) que está relacionado ao item **D** (representa uma foto da mesma atriz), este realizará uma transfomação para apresentar essa relação.

Ilustramos nesta figura a continuação das animações citadas anteriormente sobre os itens **A**, **B** e **C**.

No item **D** apresentamos um efeito do tipo *morph*, onde uma imagem se transforma na outra gradativamente. Esse item passa então a representar o Item **I**.

Como os items **G** e **H** estão presentes somente na interface destino, estes serão inseridos durante a transição.

Por fim temos na interface destino os elementos a seguir:

A – Nome da Atriz (Amália)
B – Nome do Filme (A Arte de Amália)
C – Foto do Filme (A Arte de Amália)
G – Nome do Filme (Fado)
H – Cartaz do Filme (Fado)
I – Foto (distinta da origem) da Atriz (Amália)

Figura 17 – Descrição de uma transição animada

A partir da descrição sobre os possíveis comportamentos dos elementos, definindo como eles se relacionam entre os estados origem/destino e quais transformações podem ser realizadas, determinamos quais animações poderão ocorrer. Temos assim, cinco tipos de animações que poderão compor as transições:

- *InsertElement*: Representa uma animação de entrada, para inserção de um elemento pertencente ao estado destino, durante a transição. Indicando o surgimento de um novo elemento.
- *RemoveElement*: Representa uma animação de saída, para remoção de um elemento pertencente ao estado origem, durante a transição, indicando assim, ausência do elemento no estado destino.
- *MatchElements*: Representa uma animação de manutenção, que realiza transformações para igualar os parâmetros entre um elemento do estado origem e outro do estado destino, indicando a presença do elemento nos dois estados da transição.
- *TradeElements*: Representa uma animação de substituição, que realiza transformações em um elemento do estado origem substituindo-o por outro elemento do estado destino, de forma a indicar uma relação entre eles.
- *EnphasizeElement*: Representa uma animação para destaque de um elemento, realizando transformações sobre ele como forma de enfatizá-lo durante a transição e indicando que uma ação esta sendo realizada.

É importante ressaltar que o designer poderá escolher quais animações ocorrerão ou mesmo não realizar qualquer animação para uma transição. Esta escolha reflete no número de animações que ocorrerão. O designer poderia, por exemplo, querer apresentar apenas animações sobre os widgets considerados como os principais, ou mesmo, limitar a animação sobre um número de elementos quando muitos alterar-se-iam durante uma transição. Assim, pode se evitar um excesso de informações e conseqüentemente uma sensação de confusão pelo usuário.

Com os tipos de animações definidos, podemos especificar através das interfaces abstratas, sobre quais elementos ocorrerão as animações. Para facilitar a

comparação entre os *widgets* que estão presentes na interface, utilizamos a notação gráfica definida na modelagem de interfaces, porém, identificando e descrevendo para cada widget, qual a classe navegacional (se existente) e qual o atributo desta classe que está sendo apresentado. Exemplificamos na Figura 19 um conjunto de animações para a transição definida pelo par de interfaces origem/destino da Figura 18, onde *IdxActorName*, identifica o *widget* clicado.

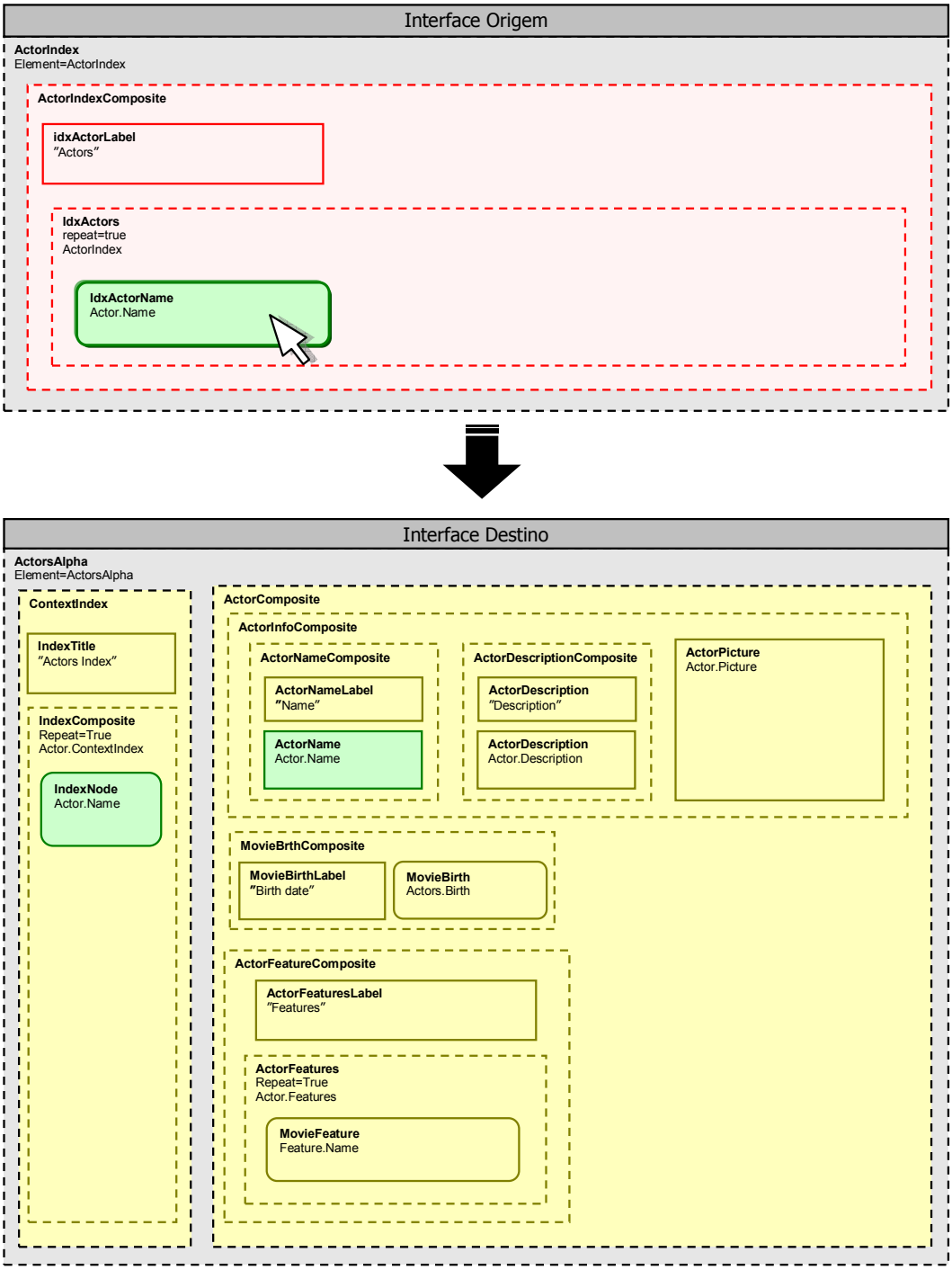

Figura 18 – Mudança de índice de atores para a interface atores por ordem alfabética

```
1  EnphasizeElement Activated_IdxActorName
2  InsertElement ActorComposite
   InsertElement ContextIndex
3  RemoveElement ActorIndexComposite
4  MatchElements Activated_IdxActorName with ActorName
   MatchElements IdxActorName with IndexNode
```

Figura 19 – Conjunto de animações para uma transição suave

Nesta transição são realizadas diversas animações para a alteração das interfaces, nela definimos a princípio que o elemento clicado será enfatizado, definimos também, que as informações da interface destino que não estavam presentes anteriormente serão inseridas. Em seguida, definimos que os elementos que não estarão na interface destino serão removidos e por fim igualamos os parâmetros dos elementos que permanecem durante toda transição.

Dado os tipos de animação que poderão ser especificados para a transição, precisamos identificar quais serão as propriedades de cada animação:

- *Elements*: Indica os *widgets* que irão participar da animação. Quando a animação é do tipo *InsertElement*, *RemoveElement* ou *EnphasizeElement*, definimos apenas um elemento, caso seja do tipo *MatchElements* ou *TradeElements*, definimos o par de elementos origem/destino que irão participar da transformação.
- *RhetoricalStructure*: Indica qual a estrutura retórica de transição que a animação irá pertencer. Através desta propriedade identificaremos a ordem em que as animações ocorrerão na transição, conforme descrito posteriormente.

Pode se observar que existem ainda características das animações como tipos de efeitos e duração, que em nossa abordagem são definidas separadamente através da especificação dos estilos retóricos.

Como dito anteriormente, as animações são especificadas a partir de interfaces abstratas já definidas. Nessas interfaces, entretanto, não sabemos quais dados estarão disponíveis ou mesmo qual a instância, ou nó que estará sendo apresentado. Estas características só poderão ser observadas durante a execução da aplicação, quando todas as informações já estiverem sido definidas e quando as escolhas de navegação serão realizadas. Sendo assim, não podemos prever se os

widgets abstratos especificados representam de fato um mesmo objeto e conseqüentemente, não podemos determinar se as transformações deverão de fato ocorrer durante a transição.

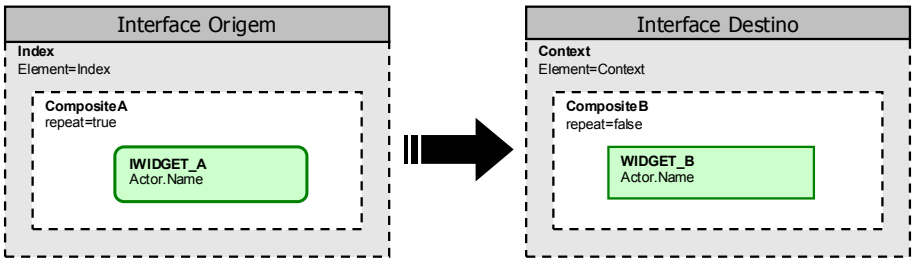

Figura 20 – Transição entre interfaces distintas

Na Figura 20 temos dois widgets A e B que representam um mesmo elemento e atributo do modelo conceitual, sugerindo inicialmente a definição de uma animação do tipo *MatchElement* durante uma transição. Tais elementos, entretanto, poderiam estar referenciando instâncias diferentes, indicando a necessidade de ações diferentes conforme ilustrado pela Figura 21.

Instância representada		**Ação realizada**	
Widget A	**Widget B**	**Widget A**	**Widget B**
João de Andrade	João de Andrade	*MatchElements*	
Maria Cavalcanti	João de Andrade	*RemoveElement*	*InsertElement*
Nenhum (Inexistente)	Maria Cavalcanti	nenhum	*InsertElement*
João de Andrade	Nenhum (Inexistente)	*RemoveElement*	nenhum

Figura 21 – Possíveis ações para as condições de referência distintas

Como existem diferentes formas de representações dos elementos, que dependem dos dados definidos e das navegações realizadas, é aconselhável considerar as possíveis alternativas e tratá-las, obtendo uma transição suave coerente para qualquer condição apresentada. Para possibilitar esse tipo de representação, definimos três condições a seguir, que deverão ser satisfeitas para execução da animação:

- Em animações do tipo *MatchElements* ou *TradeElements*, os dois elementos que a compõe deverão representar uma mesma instância ou nó do sistema (no caso destes elementos serem distintos, eles poderiam ser removidos e/ou inseridos).

- Para qualquer animação ocorrer, todos os elementos que a compõem deverão possuir dados associados, ou seja, devem representar uma instância no sistema.
- Para qualquer animação, todos os elementos que a compõem não poderão participar de qualquer outra animação descrita antes desta na especificação realizada, a não ser que esta seja a animação *EmphasizeElement.*

Dada a transição entre duas instâncias de mesma interface representada pelas interfaces abstratas na Figura 22 (Filmes por ordem alfabética), exemplificamos através da especificação (Figura 23) uma possível definição das animações que compõem esta transição.

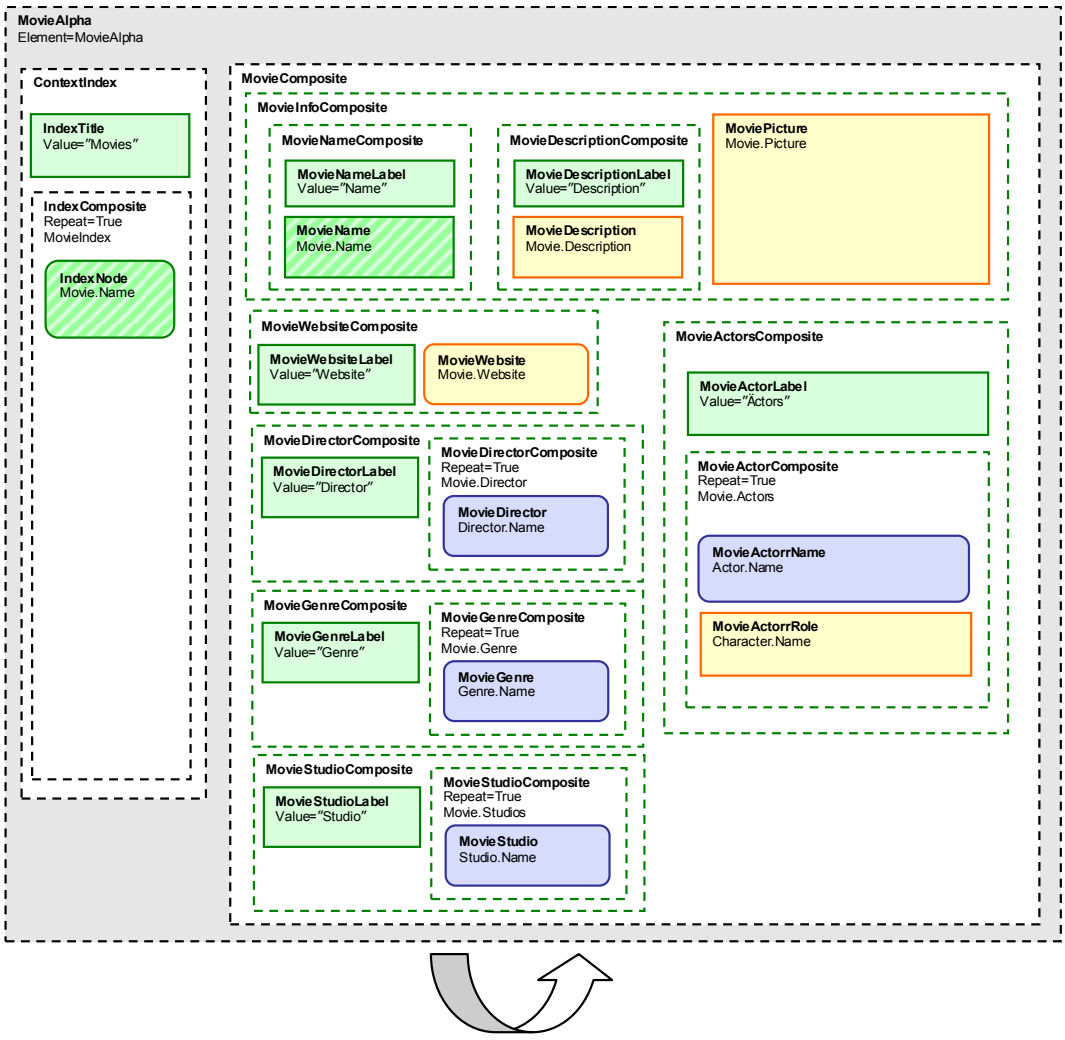

Figura 22 – Transição entre a mesma interface do contexto filmes por ordem alfabética

```
Enphasize Activating_IdxMoviename

Match Elements = (Activating_IndexNode, MovieName), RhetoricalStructure = Main
Match Elements = (MovieName, IndexNode), RhetoricalStructure = Main

Match Elements = (MovieDirector, MovieDirector), RhetoricalStructure = Secondary
Match Elements = (MovieGenre, MovieGenre), RhetoricalStructure = Secondary
Match Elements = (MovieStudio, MovieStudio), RhetoricalStructure = Secondary
Match Elements = (MovieActor, MovieActor), RhetoricalStructure = Secondary

Match Elements = (MovieNameComposite, MovieNameComposite), RhetoricalStructure = Layout
Match Elements = (MovieDescriptionComposite, MovieDescriptionComposite), RhetoricalStructure = Layout
Match Elements = (MovieDirectorComposite, MovieDirectorComposite), RhetoricalStructure = Layout
Match Elements = (MovieGenreComposite, MovieGenreComposite), RhetoricalStructure = Layout
Match Elements = (MovieStudioComposite, MovieStudioComposite), RhetoricalStructure = Layout
Match Elements = (MovieActorsComposite, MovieActorsComposite), RhetoricalStructure = Layout

Remove Elements = (MovieDescription), RhetoricalStructure = Removal
Remove Elements = (MoviePicture), RhetoricalStructure = Removal
Remove Elements = (MovieWebsite), RhetoricalStructure = Removal
Remove Elements = (MovieDirector), RhetoricalStructure = Removal
Remove Elements = (MovieGenre), RhetoricalStructure = Removal
Remove Elements = (MovieStudio), RhetoricalStructure = Removal
Remove Elements = (MovieActor), RhetoricalStructure = Removal

Insert Elements = (MovieDescription), RhetoricalStructure = Insertion
Insert Elements = (MoviePicture), RhetoricalStructure = Insertion
Insert Elements = (MovieWebsite), RhetoricalStructure = Insertion
Insert Elements = (MovieDirector), RhetoricalStructure = Insertion
Insert Elements = (MovieGenre), RhetoricalStructure = Insertion
Insert Elements = (MovieStudio), RhetoricalStructure = Insertion
Insert Elements = (MovieActorComposite), RhetoricalStructure = Insertion

Remove (MovieComposite), RhetoricalStructure = Removal
```

Figura 23 – Especificação de transição entre diferentes instâncias em um mesmo contexto

3.2.2.3. Especificação da Estrutura Retórica de Transição

Utilizamos a retórica nas transições para referir à organização das animações através das relações existentes entre as partes distintas, de forma a apresentar as alterações com coerência para o usuário durante a transição. Assim como realizado por Kennedy & Mercer (2002) para utilização da retórica em animações cinematográficas, podemos comparar a retórica sobre o domínio de transições suaves, com a retórica em textos, definindo três correspondências entre suas características:

- Autor ⇒ Designer;
- Texto ⇒ Animações;
- Frases ou Sentenças ⇒ Estruturas Retóricas de Transição.

Na especificação da estrutura retórica de Transição, estaremos definindo quais são as estruturas que definem esta terceira analogia apresentada. Cada estrutura reunirá animações sobre conjuntos específicos, podendo então definir as características relativas à ordem em que as animações irão ocorrer. Essa

identificação é realizada da mesma forma em que definiríamos um conjunto de palavras como sentenças, e em seguida as organizássemos em uma ordem específica de forma a expressar da melhor maneira o significado do texto.

Para esta especificação precisamos definir então quais serão as estruturas retóricas e, posteriormente, como será organizada a ordem de ocorrência na transição. Note que a definição de quais animações pertencem às estruturas retóricas de transição, será representada através da propriedade *RhetoricalStructure*, na especificação das animações de transição.

Ao definir quais estruturas irão compor a transição, devemos identificar conjuntos de animações que têm algum propósito ou característica em comum, tentando estabelecer coerência entre os itens. Definimos a seguir um possível conjunto de estruturas retóricas, onde descrevemos quais as características dos itens que irão compor cada uma:

1. *Feedback* – Corresponde a animação do item sobre o qual ocorreu a interação, ativando a transição, como forma de enfatizar o elemento, oferecendo feedback;
2. *Layout* – Corresponde a transformações de layout, indicando animações de ajuste da interface durante a transição;
3. *Insert* – Animações sobre itens que serão inseridos durante a transição, identificando todos novos elementos que surgirão na transição;
4. *Remove* – Animações sobre itens que serão removidos durante a transição, identificando os elementos que serão retirados da interface na transição;
5. *Main* – Corresponde a animações sobre os itens que representam as principais transformações, identificando as animações mais importantes da transição;
6. *Secondary* – Corresponde a transformações sobre os itens que não representam as principais mudanças na transição, identificando animações de importância secundária.

Ilustramos na , a estrutura retórica descrita a ordem em que ocorrerão as animações.

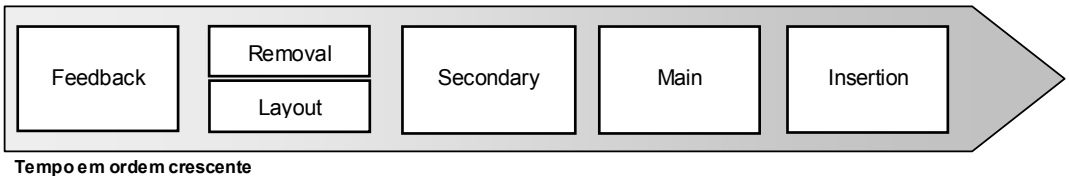

Figura 24 - Exemplo de uma retórica de transição

Na retórica apresentada () realizamos primeiramente as animações de feedback como forma de informar ao usuário que o sistema reconheceu sua ação, preparando-o para as animações que seguirão. Em seguida, retiramos as informações que serão removidas e ajustamos o layout através das animações correspondentes como forma de preparar a interface para as animações seqüentes, direcionando a atenção do usuário. Realizamos então as animações secundárias seguida das principais que correspondem ao núcleo ou núcleos da estrutura, exigindo assim maior atenção do usuário. Por fim, complementamos a interface com as animações de inserção, obtendo assim a interface final da transição.

É importante ressaltar que embora possamos identificar algumas relações sobre as estruturas, definindo uma conexão hierárquica, realizamos este processo de forma subjetiva, sendo necessário um estudo mais detalhado sobre formas sistemáticas para sua definição nas transições, tais como as propostas pela RST (Mann e Thompson, 1988).

Cada estrutura definida deverá ter propriedades que indicarão suas características quanto à ordem de ocorrência. Temos então, para cada estrutura as seguintes propriedades:

- *StructureName*: Indica o nome da estrutura definida;
- *Sequencing*: Indica qual a ordenação dos elementos internos que compõem a estrutura, ou seja, a ordem em que as animações ocorrerão dentro desta estrutura. A ordenação interna poderá ser feita: paralelamente, onde todas as animações se iniciam em um mesmo instante, seqüencialmente, onde as animações serão realizadas uma após a outra e, paralelamente sobre *widgets* semelhantes quanto aos objetos navegacionais que representam e seqüencialmente sobre *widgets* diferentes.
- *Order*: Indica a ordem de ocorrência da estrutura retórica de transição com relação às outras estruturas definidas, indicando assim a ordem em que ocorrerá cada conjunto de animações. Se esta propriedade não for definida,

a estrutura ocorrerá no primeiro instante da transição. Essa propriedade é definida indicando se a estrutura ocorrerá antes ou junto à outra estrutura definida.

3.2.2.4. Especificação dos Estilos Retóricos de Transição

Através da especificação dos estilos retóricos da Transição, definimos quais os efeitos que serão definidos para as animações, bem como qual a sua duração. Optamos por realizar essa definição isoladamente, como forma de possibilitar a separação da especificação das animações e transições, da responsabilidade pela apresentação visual das animações. A separação permite também realizar diferentes especificações que poderiam ser reaproveitados em demais modelos, bem como facilitar a comparação entre diferentes apresentações visuais.

Para a definição dos estilos, precisaremos definir as seguintes propriedades:

- *Effect*: Indica qual efeito (definidos de acordo com o ambiente de execução) será apresentado pela animação. Quando a animação é do tipo *TradeElements*, deverá ser definido um par de efeitos, caso contrário, somente um efeito deverá ser definido.
- *Duration*: Indica o tempo de duração da animação em segundos.

As possíveis opções de efeitos dependerão do tipo de animação, tendo efeitos específicos para cada um. Dependerão ainda da tecnologia e ambiente utilizado e das animações que foram desenvolvidas e disponibilizadas para as transições.

Tais propriedades de estilo retórico permitem representar as animações de formas diversas. Para escolha delas, identificamos qual a melhor forma de representação de cada animação, de acordo com a intenção que se deseja passar ao usuário. Estas intenções são identificadas com o auxílio das estruturas retóricas definidas, que nos permitem analisar qual o papel daquele conjunto de animações na transição, definindo assim, a importância da animação, qual a relação com demais animações apresentadas e como podemos transmitir essas características.

Nesta etapa consideramos as técnicas, práticas e padrões de animações existentes, que auxiliam a determinar qual o melhor efeito e tempo duração a ser empregado. Podemos por exemplo, definir animações importantes com efeitos

chamativos e realizando as animações de forma rápida, direcionando assim a atenção do usuário.

Os estilos retóricos de transição são similares aos estilos definidos em páginas HTML, podendo ser definidos de forma *inline*, na própria especificação da animação, ou através de especificações externas, que denominamos Rhetorical Style Sheets (RHSS). Ao especificarmos os estilos externamente, definimos um RHSS, onde descrevemos as características das animações isoladamente. Neste caso, precisamos referenciar as animações, o que pode ser realizado, estendendo as propriedades das animações com um identificador, do tipo *AnimationID* e utilizando as estruturas retóricas de transição a que pertencem como uma classe de animação que identifica um conjunto de animações específicos. Representamos um RHSS na Figura 25.

```
Enphasize Activating_IdxMoviename, RhetoricalStructure=Feedback, AnimationID = ”Clicked”
Match Elements = (IdxMoviename, IdxMoviename), RhetoricalStructure = Main
Remove Elements = (MovieDescription), RhetoricalStructure = Removal, AnimationID = ”NotImportant”
Insert Elements = (MovieDescription), RhetoricalStructure = Insertion, AnimationID = ”NotImportant”


Rhetorical Style Sheet:

.Feedback #Clicked {
Effect = Grow,
Duration = 1
},

.Match {
Duration = 1
},

#NotImportant{
Effect = Fade
Duration = 1
}
```

Figura 25 – Exemplo de um Rhetorical Style Sheet (RHSS)

Enfatizamos que para determinação da retórica nos baseamos nas práticas, padrões e técnicas elaboradas para a animação. Através da análise destes métodos, podemos observar quais os melhores efeitos a serem aplicados, qual a melhor opção para duração e, qual ordem de animação representará melhor as intenções que se deseja passar durante a transição.

3.2.3. Interpretação da Especificação

Esta etapa define o último passo da abordagem elaborada, quando ocorre a interpretação das especificações da transição suave para a execução das

animações definidas. O processo de interpretação das especificações se dará somente se existir uma transição suave definida durante a mudança navegacional.

Com esta interpretação possibilitamos a geração do produto final da abordagem, que é o sistema hipermídia com interfaces dinâmicas, apresentando transições suaves durante navegações específicas. Para definição da interpretação, deveremos desenvolver uma espécie de máquina virtual, que possibilita a leitura das especificações definidas durante a interação do usuário com o sistema. Esta implementação deverá ser realizada sobre tecnologias específicas, restringindo as funcionalidades e efeitos de animação ao ambiente escolhido. Descrevemos no próximo capítulo uma implementação para interpretação das transições.