8
Referências Bibliográficas

1 SCHWABE, D.; ROSSI, G. The object-oriented hypermedia design model. **Communications of the ACM**, v.38, n.8, p. 45-46, ago. 1995.

2 MOURA, S. S. **Desenvolvimento de Interfaces Governadas por Ontologias para Aplicações na Web Semântica**. Rio de Janeiro, 2004. 150p. Dissertação de Mestrado – Pontifícia Universidade Católica do Rio de Janeiro.

3 BOZZON, A. et al. Conceptual Modeling and code Generation for Rich Internet Applications. In: 6TH INTERNATIONAL CONFERENCE ON WEB ENGINEERING – ICWE '06, 2006, Palo Alto. **Proceedings of the 6th international conference on Web engineering**. New York: ACM Press, 2006. p. 353–360.

4 ALLAIRE, J. **Macromedia Flash MX – A next-generation rich client**. Macromedia, 2002, 14p. Disponível em <http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf>. Acesso em: 24/1/2010.

5 PRECIADO, J. C. et al. Designing Rich Internet Applications with Web Engineering Methodologies. In: 9TH IEEE INTERNATIONAL WORKSHOP ON WEB SITE EVOLUTION – WSE 2007, 2007, Paris. **Proceedings of the 2007 9th IEEE International Workshop on Web Site Evolution**. Washington: IEEE Computer Society, 2007. p. 23-30.

6 YU, J. et al. OpenXUP – an Alternative Approach to Developing Highly Interactive Web Applications**.** In: 6TH INTERNATIONAL CONFERENCE ON WEB ENGINEERING – ICWE '06, 2006, Palo Alto. **Proceedings of the 6th international conference on Web engineering**. New York: ACM Press, 2006. p. 289–296.

7 YU, J. (Ed.). **SUL - Simple User Interface Language 1.0**. OpenXUP.org, 2007, 55p. Disponível em <http://www.openxup.org/TR/sul-20070501.pdf>. Acesso em: 30/1/2010.

8 W3C. **Document Object Model (DOM)**. 2005. Disponível em <http://www.w3.org/DOM/>. Acesso em: 24/1/2010.

9 ASLESON, R.; SCHUTTA, N. T. **Foundations of Ajax**. Apress. 2005. 296 p.

10 GARRETT, J. J. **Ajax: A New Approach to Web Applications**. 2005. Disponível em <http://www.adaptivepath.com/ideas/essays/archives/000385.php>. Acesso em: 24/1/2010.

11 LINAJE, M. et al. On the Implementation of Multiplatform RIA User Interface Components. In: 7TH INTERNATIONAL WORKSHOP ON WEB-ORIENTED SOFTWARE TECHNOLOGIES – IWWOST '08. EIGHTH INTERNATIONAL CONFERENCE ON WEB ENGINEERING (ICWE 2008) WORKSHOPS, 2008, Yorktown Heights, New York. **Proceedings**. Bratislava: Vydavateľstvo STU, 2008. p. 50-55.

12 ADOBE SYSTEMS, Inc. **Adobe Flex 3 Developer Guide**. 2008. 1328p. Disponível em: <http://livedocs.adobe.com/flex/3/devguide_flex3.pdf>. Acesso em: 23/1/2010.

13 ADOBE SYSTEMS, Inc. **Programming Actionscript 3.0**. 2007. 814p. Disponível em: <http://livedocs.adobe.com/flash/9.0/main/flash_as3_programming.pdf>. Acesso em: 23/1/2010.

14 BOJANIC, P. **The Joy of XUL**. 2007. Disponível em <https://developer.mozilla.org/en/The_Joy_of_XUL>. Acesso em: 23/1/2010.

15 LASZLO SYSTEMS, Inc. **OpenLaszlo**: An Open Architecture Framework for Advanced Ajax Applications. 2006. 19p. Disponível em <www.openlaszlo.org/whitepaper/LaszloWhitePaper.pdf>. Acesso em: 20/1/2010.

16 FELDT, K. **Programming Firefox: Building Rich Internet Applications with Xul**. O'Reilly Media, Inc. 2007. 512 p.

17 SONNINO, B.; SONNINO, R. **Introdução ao WPF**. Microsoft Corporation, 2006. Disponível em <http://msdn.microsoft.com/pt-br/library/cc564903.aspx>. Acesso em: 23/1/2010.

18 MORONEY, L. **Silverlight**: Get Started Building A Deeper Experience Across The Web. Microsoft Corporation, 2007. Disponível em <http://msdn.microsoft.com/en-us/magazine/cc163404.aspx>. Acesso em: 23/1/2010.

19 MICROSOFT CORPORATION. **XAML Overview**. 2008. Disponível em <http://msdn.microsoft.com/en-us/library/ms752059.aspx>. Acesso em: 23/1/2010.

20 MORONEY, L. **Introdução ao Silverlight**. Microsoft Corporation, 2007. Disponível em <http://msdn.microsoft.com/pt-br/library/cc580591.aspx>. Acesso em: 23/1/2010.

21 ORACLE CORPORATION. **Develop Expressive Content with the JavaFX Platform**. Disponível em <http://javafx.com/about/overview/>. Acesso em: 24/1/2010.

22 HOMMEL, S. **Learning the JavaFX Script Programming Language**. Oracle Corporation. Disponível em <http://java.sun.com/javafx/1/tutorials/core/>. Acesso em: 24/1/2010.

23 LINAJE, M.; PRECIADO, J.C; SANCHEZ-FIGUEROA, F. A Method for Model Based Design of Rich Inernet Application Interactive User Interfaces. In: 7TH INTERNATIONAL CONFERENCE ON WEB ENGINEERING – ICWE 2007, 2007, Como, Italy. **Proceedings**. Berlin: Springer, 2007. p. 226-241.

24 PRECIADO, J.C. et al. Necessity of methodologies to model Rich Internet Applications. In: 7TH IEEE INTERNATIONAL SYMPOSIUM ON WEB SITE EVOLUTION – WSE 2005, 2005, Budapest. **Proceedings of the Seventh IEEE International Symposium on Web Site Evolution**. Washington: IEEE Computer Society, 2005. p.7-13.

25 MCCARRON, S.; PEMBERTON, S.; RAMAN, T. V. (Eds.). **XML Events**: An Events Syntax for XML. W3C Recommendation 14 October 2003. Disponível em <http://www.w3.org/TR/2003/REC-xml-events-20031014>. Acesso em: 30/1/2010.

26 KOCH, N. et al. Patterns for the Model-Based Development of RIAs. In: 9TH INTERNATIONAL CONFERENCE ON WEB ENGINEERING – ICWE '09, 2009, San Sebastián, Spain. **Proceedings of the 9th International Conference on Web Engineering**. Berlin: Springer, 2009. p. 283-291.

27 OPENAJAX ALLIANCE. **Introducing Ajax and OpenAjax**. Disponível em: <http://www.openajax.org/whitepapers/Introducing%20Ajax%20and%20OpenAjax.php>. Acesso em: 4/07/2009.

28 OPENAJAX ALLIANCE. **OpenAjax Hub 2.0 Specification Introduction**. 2009. Disponível em:

<http://www.openajax.org/member/wiki/OpenAjax_Hub_2.0_Specification_Introduction>. Acesso em: 4/07/2009.

29 OPENAJAX ALLIANCE. **OpenAjax Metadata 1.0 Specification Introduction**. 2009. Disponível em: <http://www.openajax.org/member/wiki/OpenAjax_Metadata_1.0_Specification_Introduction>. Acesso em: 2/07/2009.

30 BOYER, J. M (Ed.). **XForms 1.0 (Third Edition)**: W3C Recommendation 29 October 2007. Disponível em <http://www.w3.org/TR/2007/REC-xforms-20071029/>. Acesso em: 12/08/2009.

31 LUCAS B; AKOLKAR R.; WIECHA C. **collage-overview.pdf**. Collage Programming Model Overview. IBM Corporation, 2008. 80p. Portable Document Format. Arquivo compactado em collage-62.zip, disponível para download em: <http://www.alphaworks.ibm.com/tech/collage/download>. Acesso em: 8/7/2009.

32 HAUBOLD, T.; BEIER, G.; GOLUBSKI, W. Model Driven Development of AJAX-Based User Interfaces. In: ELLEITHY, K (Ed.). **Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering**. Springer Netherlands, 2008. p. 495-499.

33 FONTOURA, M., PREE, W., RUMPE, B. **The UML profile for framework architectures**. Addison-Wesley Professional. 2002. 352 p.

34 LOHMANN, S.; KALTZ, J. W.; ZIEGLER, J. Model-Driven Dynamic Generation of Context-Adaptative Web User interfaces. In: KÜHNE, T. **Models in Software Engineering**: Workshops and Symposia at MoDELS 2006, Genoa, Italy, October 1-6, 2006, Reports and Revised Selected Papers. Berlin: Springer, 2007. p. 116-125.

35 ATTAP TECHNOLOGIES, Inc. **Jitsu Programmers Guide**. 2006. Disponível em: <http://www.jitsu.org/jitsu/guide/index.html>. Acesso em 25/12/2009.

36 COWAN, D.; LUCENA, C. Abstract Data Views: An Interface Specification Concept to Enhance Design for Reuse. **IEEE Transactions on Software Engineering**, vol.21, n.3, p. 229–243, mar. 1995.

37 ROSSI, G. **Um método orientado a objetos para o projeto de aplicações hipermídia**. Rio de Janeiro, 1996. 205p. Tese de Doutorado – Pontifícia Universidade Católica do Rio de Janeiro.

38 SCHWABE, D.; ROSSI, G. **An Object Oriented Approach to Web-Based Application Design**. 1998. 35p. Disponível em: <http://www-di.inf.puc-rio.br/schwabe/papers/TAPOSRevised.pdf>. Acesso em: 26/072008.

39 ANTUNES, D. C. Statecharts. **Revista Bate Byte**, ed. 36, ago./set. 1994. Disponível em: <http://www.pr.gov.br/batebyte/edicoes/1994/bb36/statecharts.htm>. Acesso em: 28/07/2008.

40 COLEMAN, D.; HAYES, F.; BEAR, S. Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design. **IEEE Transactions on Software Engineering**, vol.18, n.1, p. 9–18, jan. 1992.

41 FIALHO, A. **Transições Animadas em Aplicações Web Baseadas em Modelos**. Rio de Janeiro, 2007. 133p. Dissertação de Mestrado – Pontifícia Universidade Católica do Rio de Janeiro.

42 ADIDA, B.; BIRBECK, M. (Eds.). **RDFa Primer**: Bridging the Human and Data Webs. W3C Working Group Note 14 October 2008. Disponível em:

<http://www.w3.org/TR/2008/NOTE-xhtml-rdfa-primer-20081014>. Acesso em: 17/11/2008.

43 ADIDA, B. et al. (Ed.). **RDFa in XHTML: Syntax and Processing**: A collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation 14 October 2008. Disponível em: <http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/>. Acesso em: 1/11/2009.

44 ADDISON WESLEY LONGMAN, Inc. **Programming Ruby**: The Pragmatic Programmer's Guide. 2001. Disponível em <http://www.ruby-doc.org/docs/ProgrammingRuby/> Acesso em 25/12/2009.

45 AKITA, F. **Criando um Chat com Reactor e WebSockets**. 2010. Disponível em <http://www.akitaonrails.com/2010/01/12/criando-um-chat-com-reactor-e-websockets>. Acesso em: 6/2/2010.

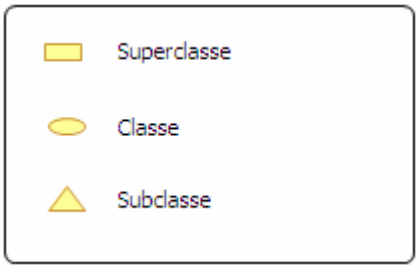
46 HICKSON, I. (Ed.). **The Web Sockets API**. W3C Working Draft 22 December 2009. Disponível em <http://www.w3.org/TR/2009/WD-websockets-20091222/>. Acesso em: 6/2/2010.

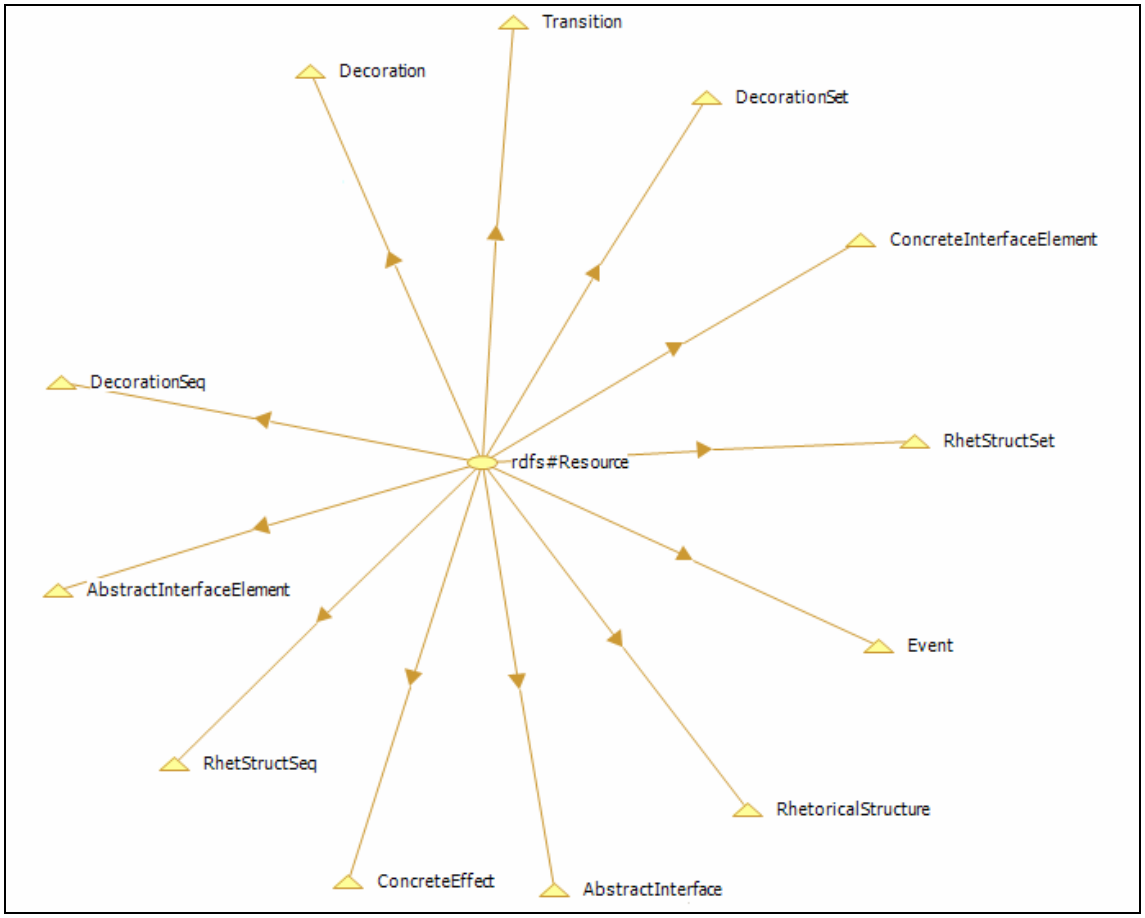
47 GRIGORIK, I. **Ruby & WebSockets: TCP for the Browser**. 2009. Disponível em <http://www.igvita.com/2009/12/22/ruby-websockets-tcp-for-the-browser/>. Acesso em: 6/2/2010.

Apêndice I – Ontologia de Descrição de Interfaces Ricas

Classes da Ontologia

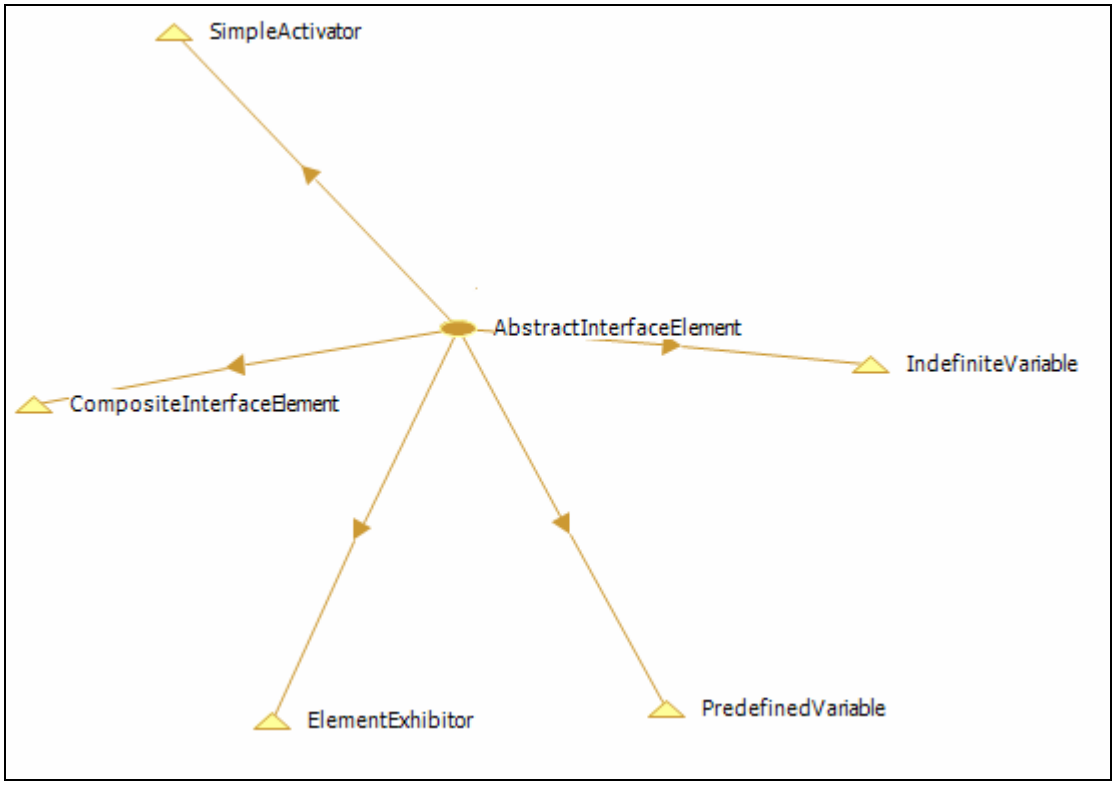
Legenda:





AbstractInterface: Representa a composição final de todos os elementos da interface abstrata.

AbstractInterfaceElement: Representa todos os possíveis elementos que compõem a interface abstrata.



CompositeInterfaceElement: Representa uma composição de elementos abstratos.

ElementExhibitor: Representa elementos que exibem algum tipo de conteúdo, como, por exemplo, um texto ou uma imagem.

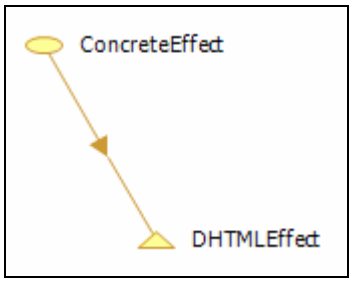
IndefiniteVariable: Representa elementos que permitem a livre entrada de dados, por exemplo, uma caixa de texto em um formulário.

PredefinedVariable: Representa elementos que permitem a seleção de um subconjunto a partir de um conjunto de valores pré-definidos. Exemplos: botões de opção e caixas de seleção.

SimpleActivator: Representa qualquer elemento capaz de reagir a eventos externos, tal como um *link* ou um botão de ação.

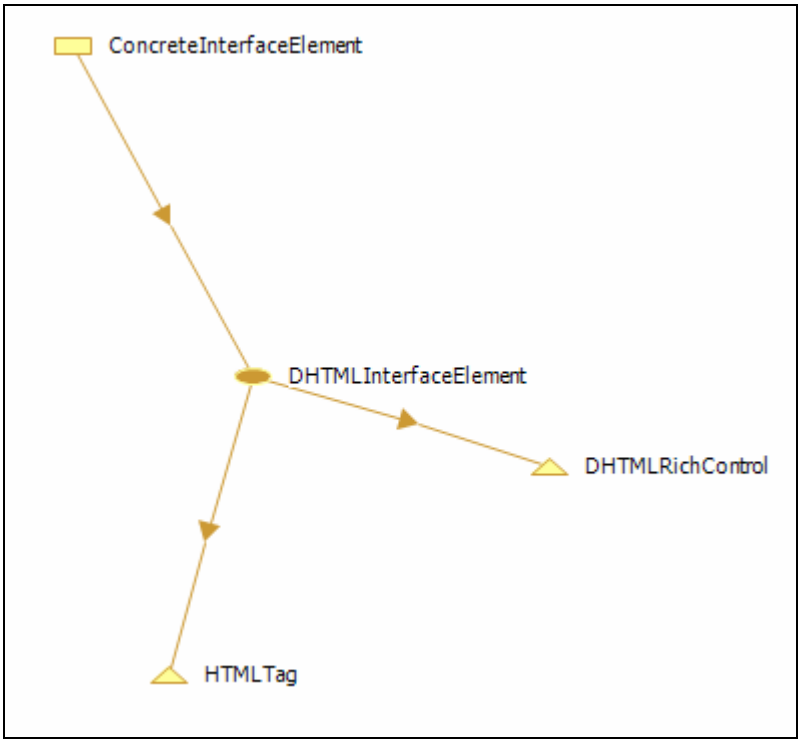
ConcreteEffect: Representa o efeito apresentado por uma decoração.

DHTMLEffect: Representa os efeitos disponíveis no ambiente de implementação DHTML, realizados por programação Javascript.



ConcreteInterfaceElement: Representa os elementos de interface disponíveis nas aplicações Web, em qualquer plataforma.

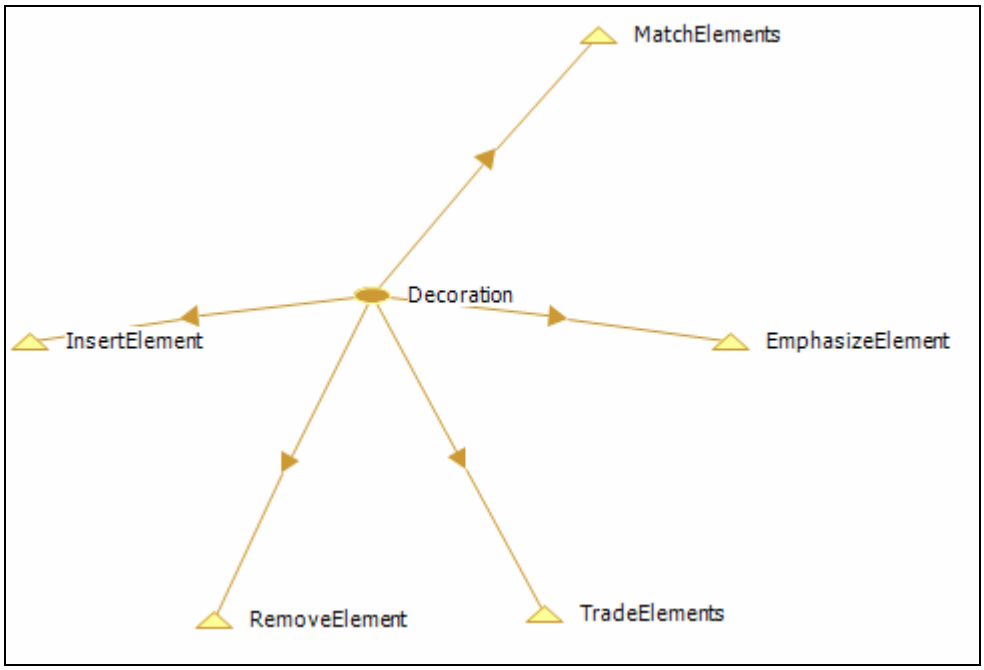
DHTMLInterfaceElement: Representa os elementos de interface disponíveis na plataforma DHTML.



DHTMLRichControl: Representa os elementos de interface que combinam HTML DOM + Javascript + CSS.

HTMLTag: Representa os elementos de interface fornecidos pela linguagem HTML.

Decoration: Alteração sobre os elementos da interface, executada durante uma transição ou em resposta a um evento de interface.



EmphasizeElement: Representa uma decoração para destaque de um elemento, realizando transformações sobre ele como forma de enfatizá-lo durante a transição e indicando que uma ação está sendo realizada.

InsertElement: Representa uma decoração de entrada, para inserção de um elemento pertencente ao estado destino, durante a transição. Indica o surgimento de um novo elemento.

MatchElements: Representa uma decoração 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.

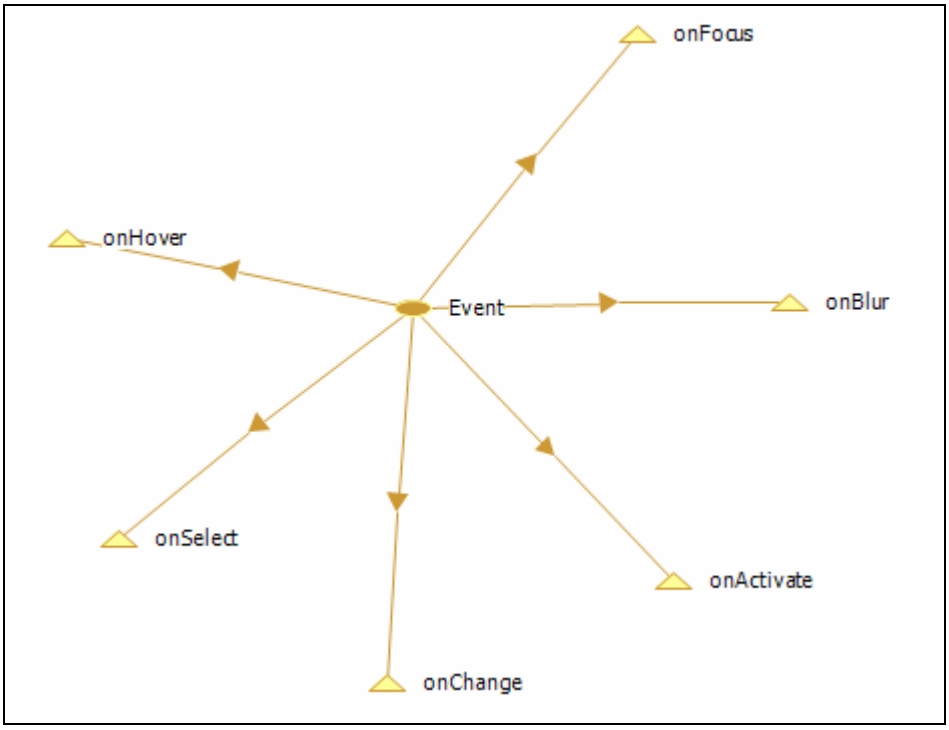
RemoveElement: Representa uma decoração de saída, para remoção de um elemento pertencente ao estado origem, durante a transição, indicando a ausência do elemento no estado destino.

TradeElements: Representa uma decoração 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.

DecorationSeq: Informa a seqüência em que as decorações ocorrem dentro de uma estrutura retórica.

DecorationSet: Informa o conjunto de decorações que ocorrem simultaneamente dentro de uma estrutura retórica.

Event: Ocorrência de uma ação sobre os elementos de interface abstratos.



onActivate: Indica que um *SimpleActivator* foi ativado.

onBlur: Indica que um elemento de entrada de dados (*IndefiniteVariable*/ *PredefinedVariable*) perdeu o foco.

onChange: Indica que um elemento de entrada de dados (*IndefiniteVariable*/ *PredefinedVariable*) perdeu o foco após ter seu conteúdo modificado.

onFocus: Indica que um elemento de entrada de dados (*IndefiniteVariable*/ *PredefinedVariable*) recebeu o foco.

onHover: Indica que o ponteiro do mouse foi movimentado sobre um elemento.

onSelect: Indica que o texto em um elemento de entrada de dados (*IndefiniteVariable*) foi selecionado.

RhetoricalStructure: Representa um conjunto de decorações.

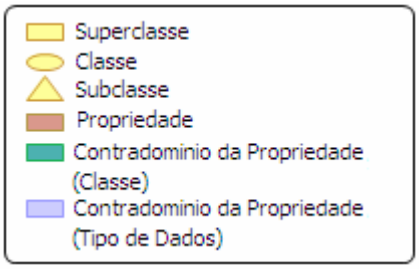
RhetStructSeq: Informa a ordem das estruturas retóricas dentro de uma transição.

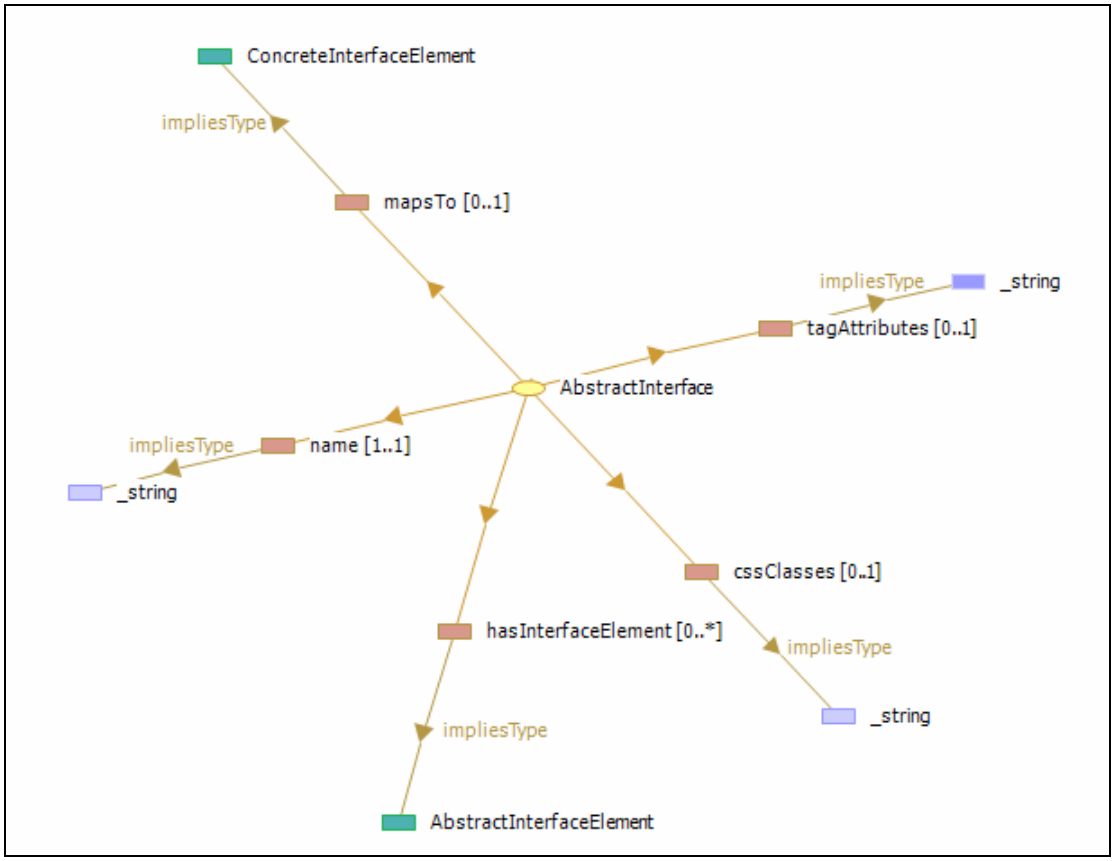
RhetStructSet: Informa o conjunto de estruturas retóricas que ocorrem simultaneamente dentro de uma transição.

Transition: Alteração do estado da interface.

Propriedades da Ontologia

Legenda:





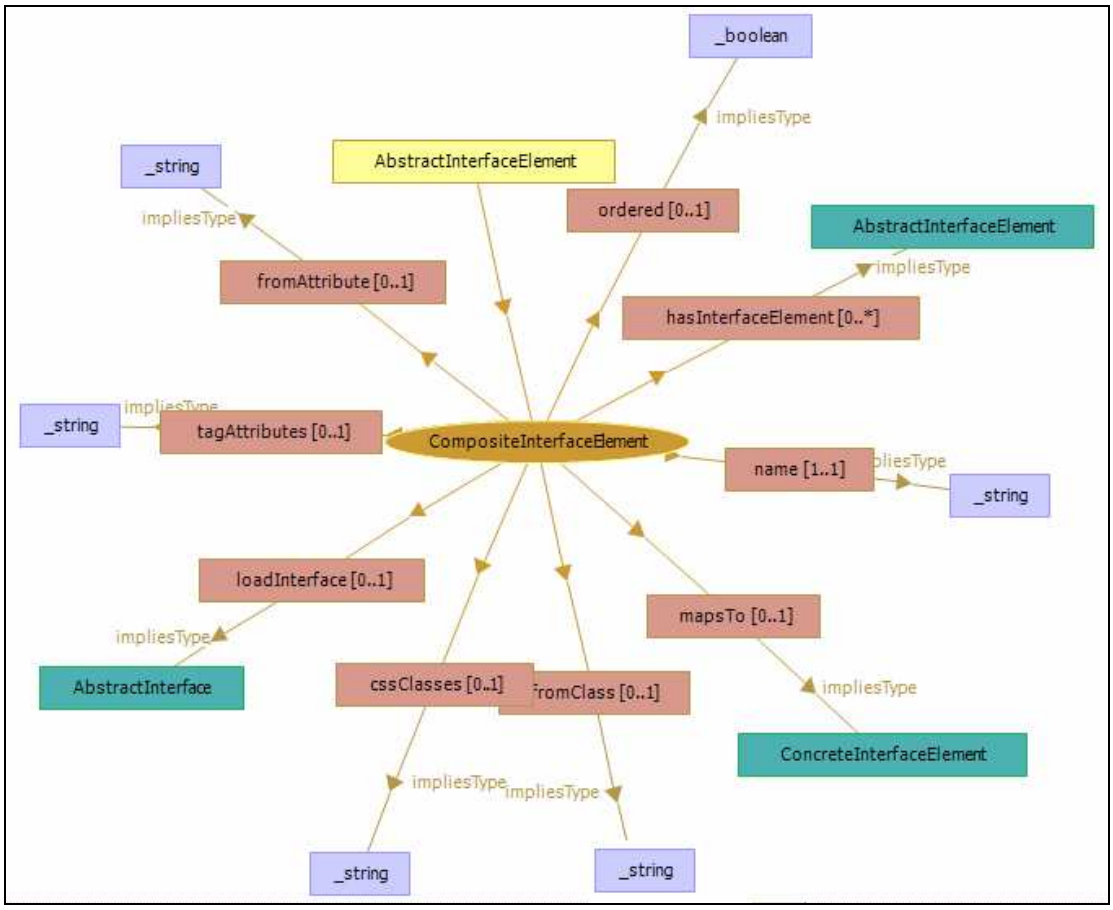
name: Identificador da instância.

hasInterfaceElement: Indica quais instâncias de *AbstractInterfaceElement* compõem o elemento abstrato.

mapsTo: Indica para qual *widget* concreto o elemento abstrato será mapeado.

cssClasses: Indica as classes CSS que serão usadas na tradução para o código HTML concreto.

tagAttributes: Atributos da *tag* HTML que serão incluídos na tradução.



name: Identificador da instância.

hasInterfaceElement: Indica quais instâncias de *AbstractInterfaceElement* compõem o elemento abstrato.

mapsTo: Indica para qual *widget* concreto o elemento abstrato será mapeado.

cssClasses: Indica as classes CSS que serão usadas na tradução para o código HTML concreto.

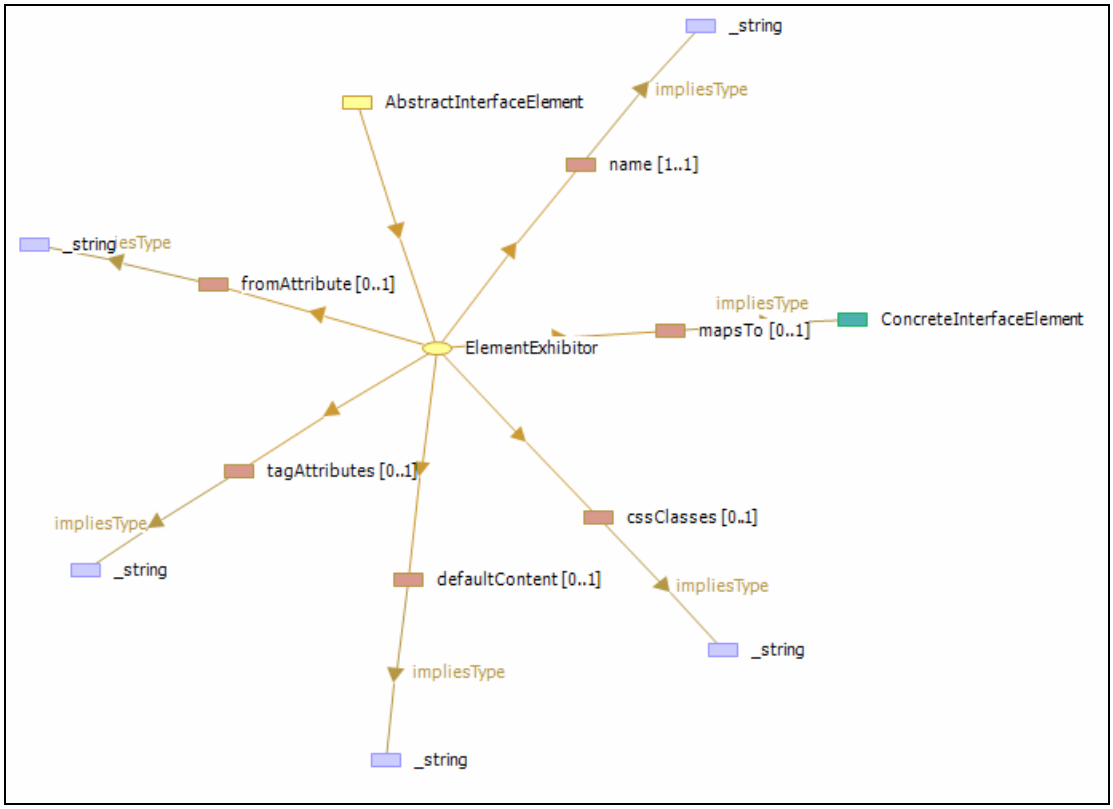
tagAttributes: Atributos da *tag* HTML que serão incluídos na tradução.

fromClass: Indica qual é a classe da ontologia navegacional, do método SHDM, que corresponde à instância da composição.

fromAttribute: Indica qual é o atributo da ontologia navegacional, do método SHDM, que corresponde à instância do composição.

loadInterface: Indica qual instância de *AbstractInterface* será carregada neste composição.

ordered: Indica se os elementos da composição representam uma lista ordenada.



name: Identificador da instância.

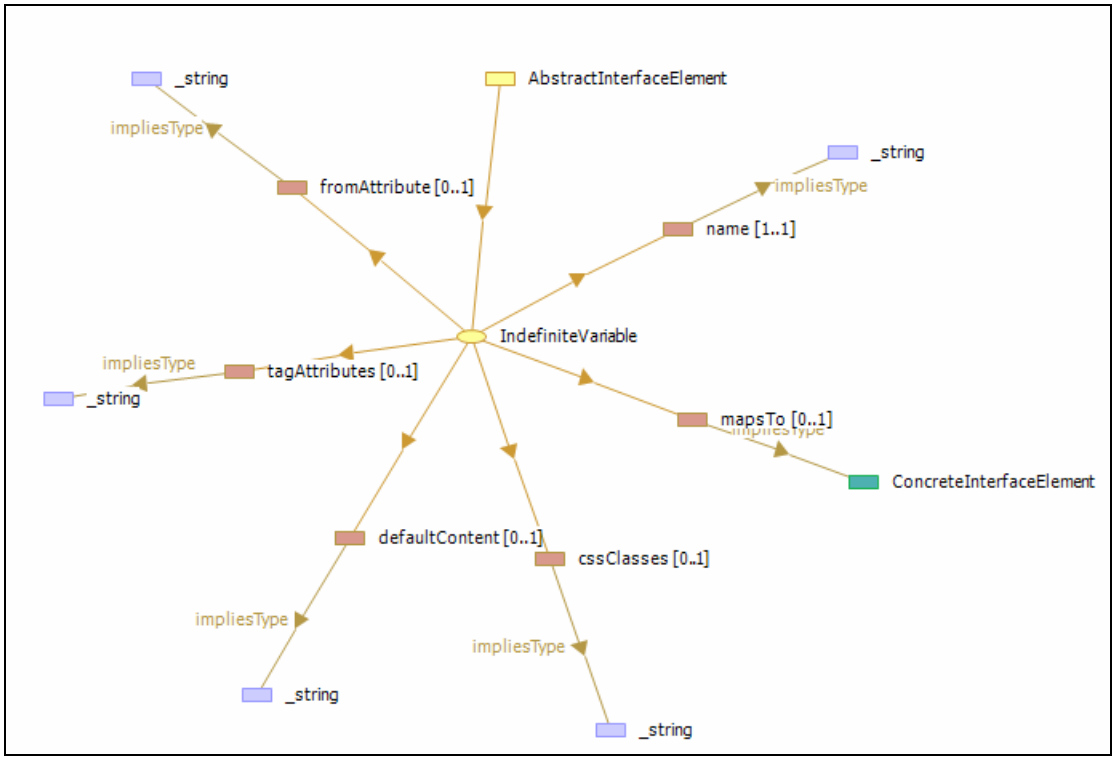
mapsTo: Indica para qual *widget* concreto o elemento abstrato será mapeado.

cssClasses: Indica as classes CSS que serão usadas na tradução para o código HTML concreto.

tagAttributes: Atributos da *tag* HTML que serão incluídos na tradução.

fromAttribute: Indica qual é o atributo da ontologia navegacional, do método SHDM, que corresponde à instância do elemento abstrato.

defaultContent: Representa um valor que será apresentado pelo *widget* concreto.



name: Identificador da instância.

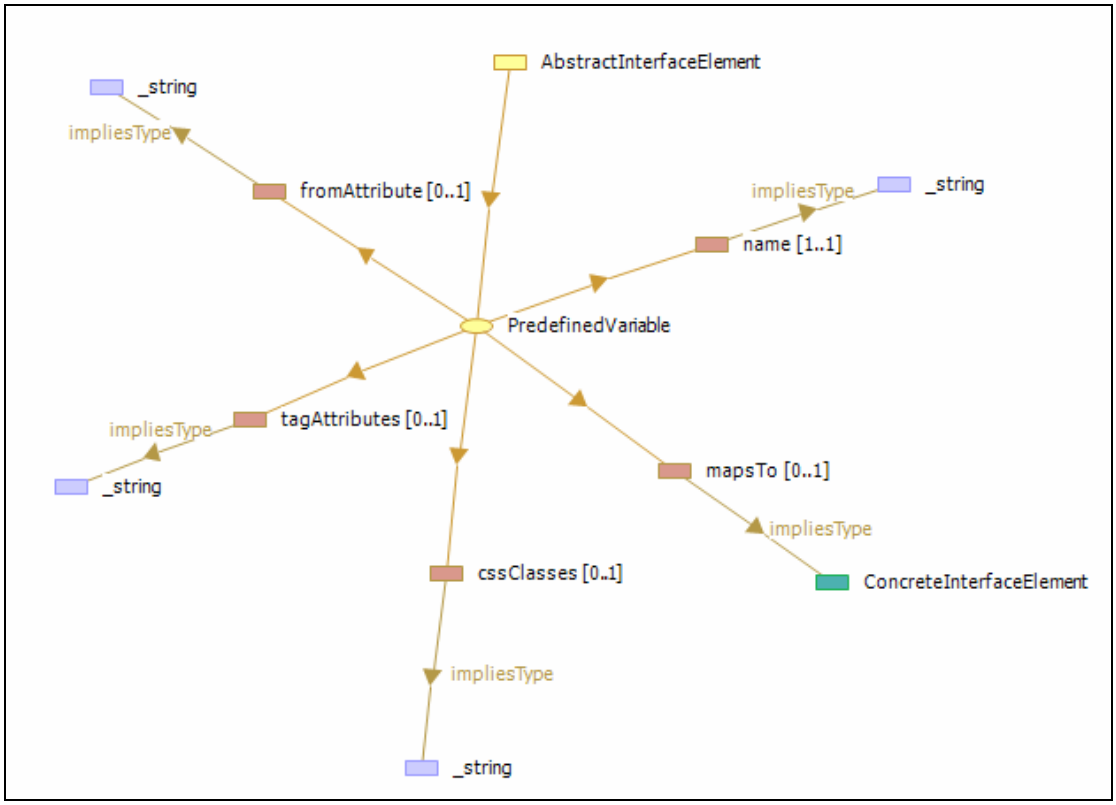
mapsTo: Indica para qual *widget* concreto o elemento abstrato será mapeado.

cssClasses: Indica as classes CSS que serão usadas na tradução para o código HTML concreto.

tagAttributes: Atributos da *tag* HTML que serão incluídos na tradução.

fromAttribute: Indica qual é o atributo da ontologia navegacional, do método SHDM, que corresponde à instância do elemento abstrato.

defaultContent: Representa um valor que será apresentado pelo *widget* concreto.



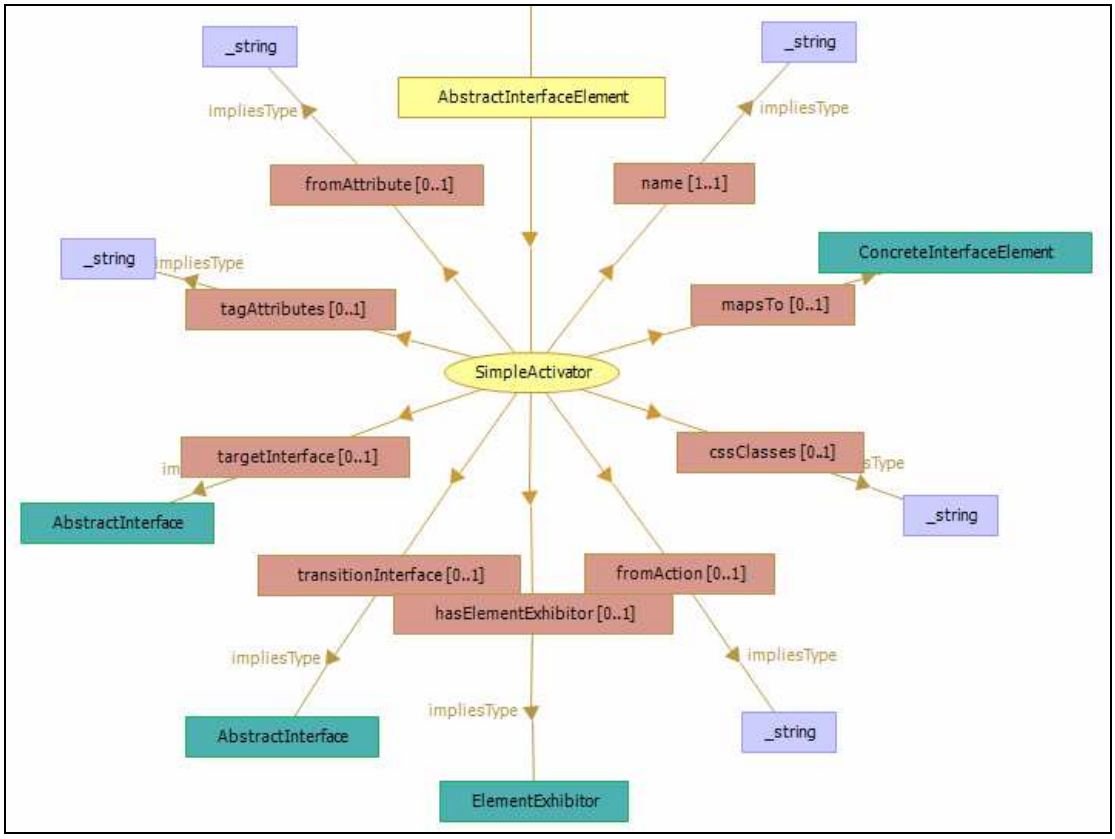
name: Identificador da instância.

mapsTo: Indica para qual *widget* concreto o elemento abstrato será mapeado.

cssClasses: Indica as classes CSS que serão usadas na tradução para o código HTML concreto.

tagAttributes: Atributos da *tag* HTML que serão incluídos na tradução.

fromAttribute: Indica qual é o atributo da ontologia navegacional, do método SHDM, que corresponde à instância do elemento abstrato.



name: Identificador da instância.

mapsTo: Indica para qual *widget* concreto o elemento abstrato será mapeado.

hasElementExhibitor: Indica o elemento da classe *ElementExhibitor* que compõe um *SimpleActivator* mapeado como *Link*..

cssClasses: Indica as classes CSS que serão usadas na tradução para o código HTML concreto.

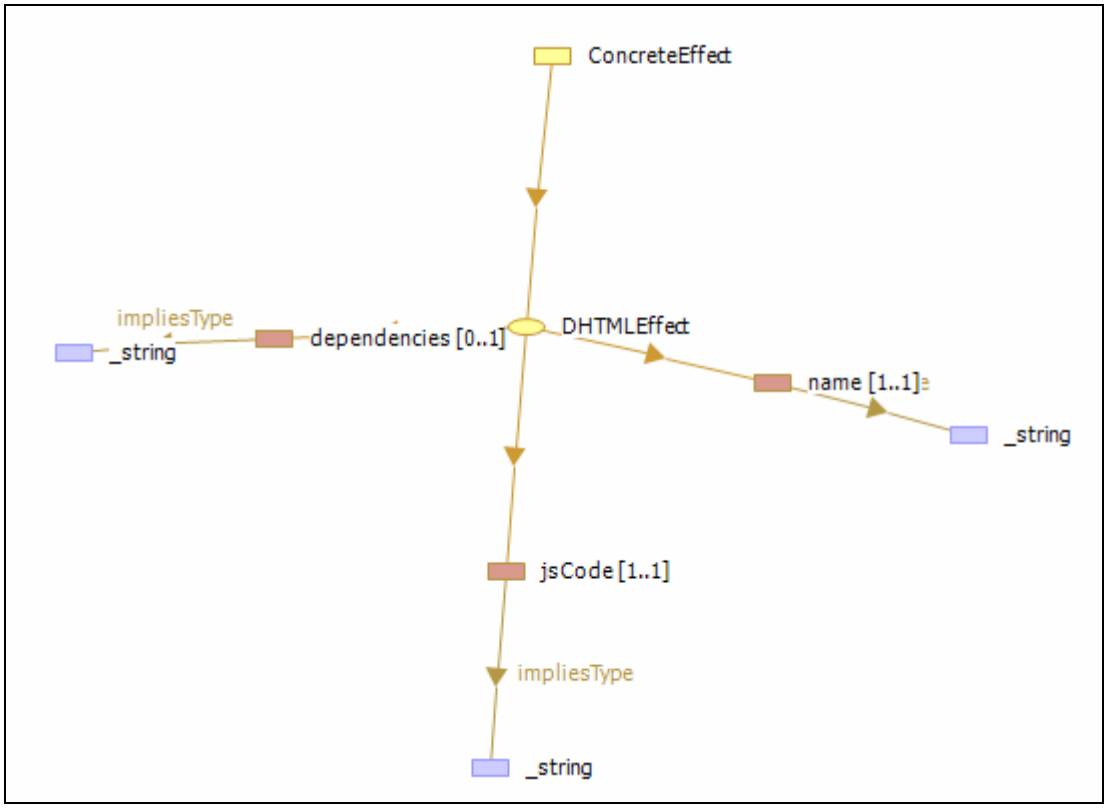
tagAttributes: Atributos da *tag* HTML que serão incluídos na tradução.

fromAttribute: Indica qual é o atributo da ontologia navegacional, do método SHDM, que corresponde à instância do ativador. Quando informado, o ativador dispara uma operação de navegação e o valor do atributo representa a URL de destino da navegação.

fromAction: Indica qual ação no Modelo será executada quando este elemento for ativado.

targetInterface: Indica qual instância da classe *AbstractInterface* será usada para exibir o resultado da ação referenciada pela propriedade *fromAction*.

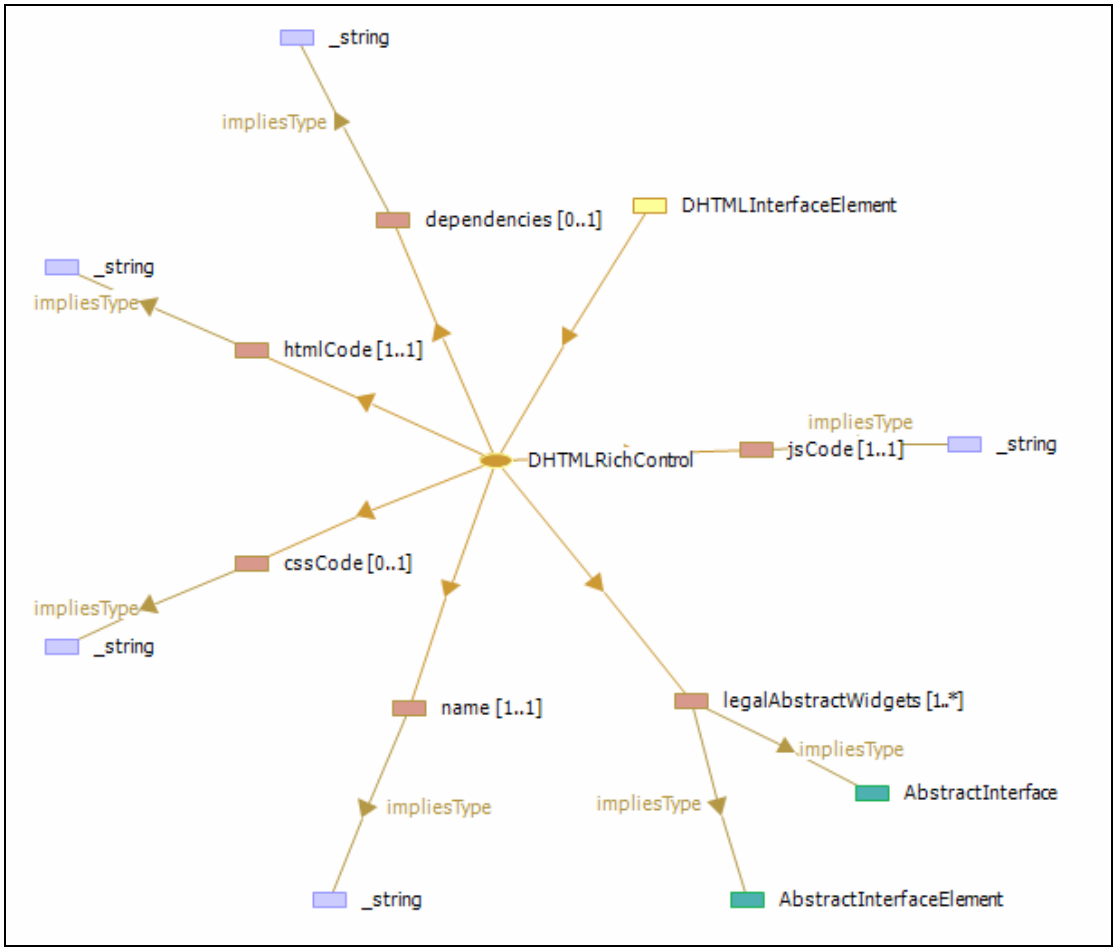
transitionInterface: Indica qual instância da classe *AbstractInterface* será exibida durante a execução da ação referenciada pela propriedade *fromAction*.



name: Identificador da instância.

jsCode: O bloco de código Javascript que contém a lógica necessária para o efeito ser executado.

dependencies: Referencia arquivos externos necessários ao funcionamento do efeito (*scripts* & folhas de estilo).



name: Identificador da instância.

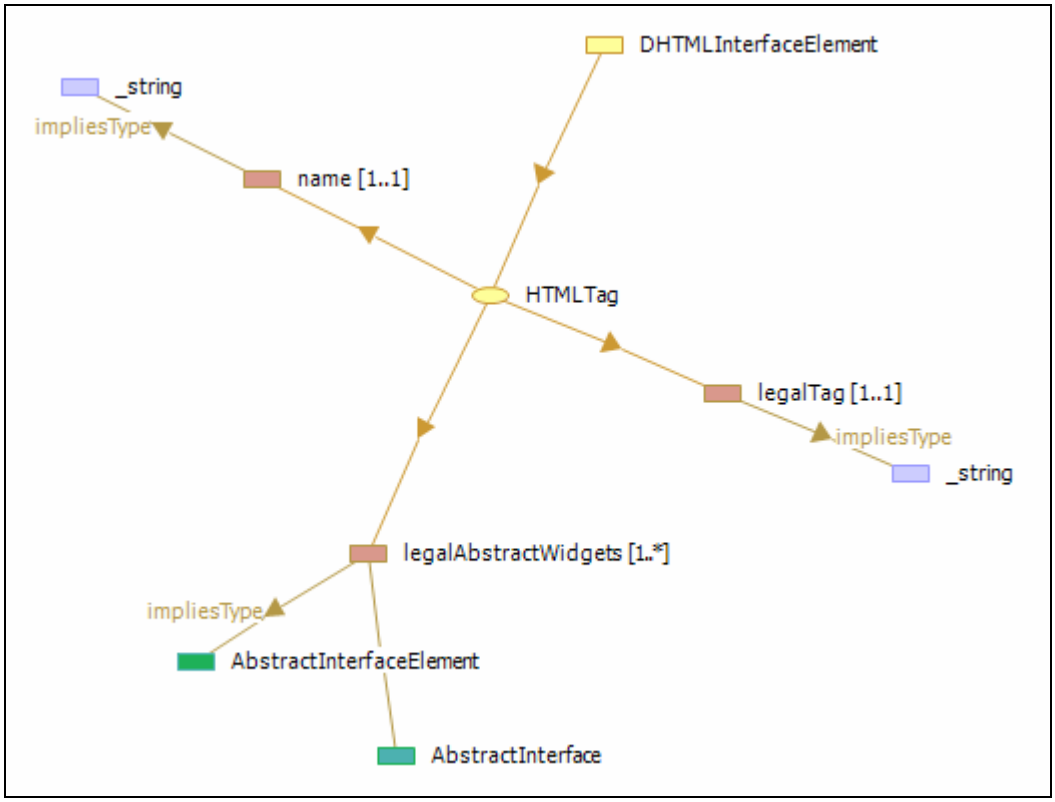
legalAbstractWidgets: Indica quais classes de elementos abstratos podem ser mapeadas para este elemento concreto.

htmlCode: O bloco de código HTML que contém a representação do elemento.

cssCode: Folha de estilo CSS que especifica a aparência do elemento.

jsCode: O bloco de código Javascript que contém a lógica necessária ao funcionamento do elemento.

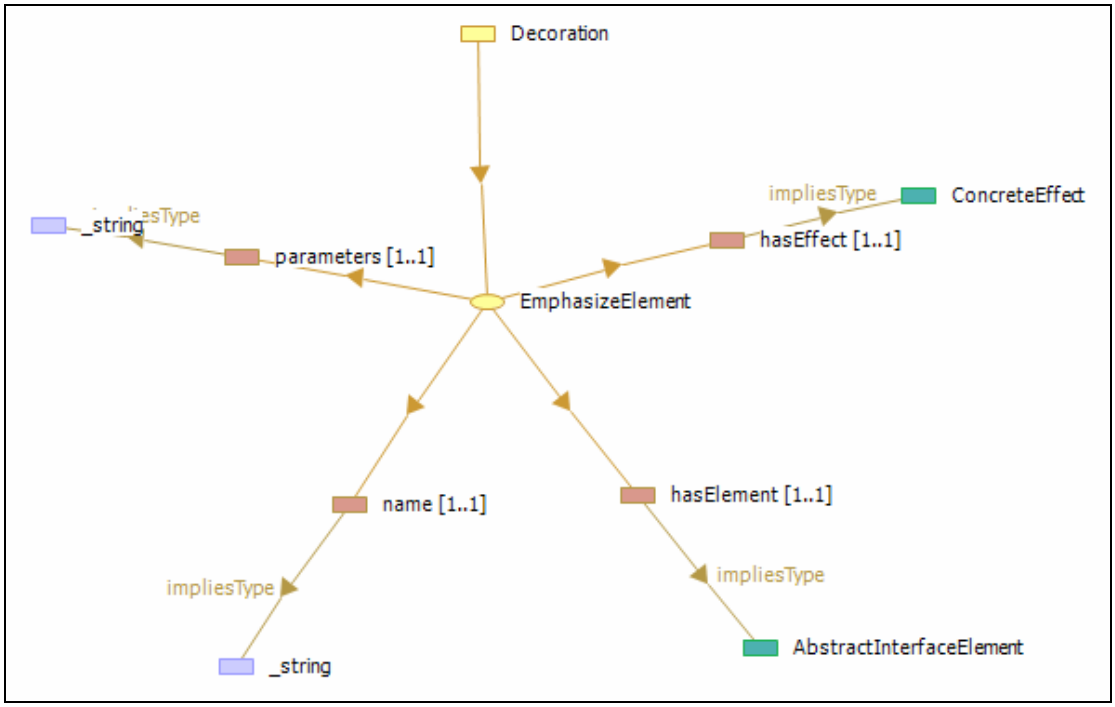
dependencies: Referencia arquivos externos necessários ao funcionamento do efeito (*scripts* & folhas de estilo).



name: Identificador da instância.

legalAbstractWidgets: Indica quais classes de elementos abstratos podem ser mapeadas para este elemento concreto.

legalTag: Indica a tag HTML que deverá ser usada para fazer a tradução do elemento.

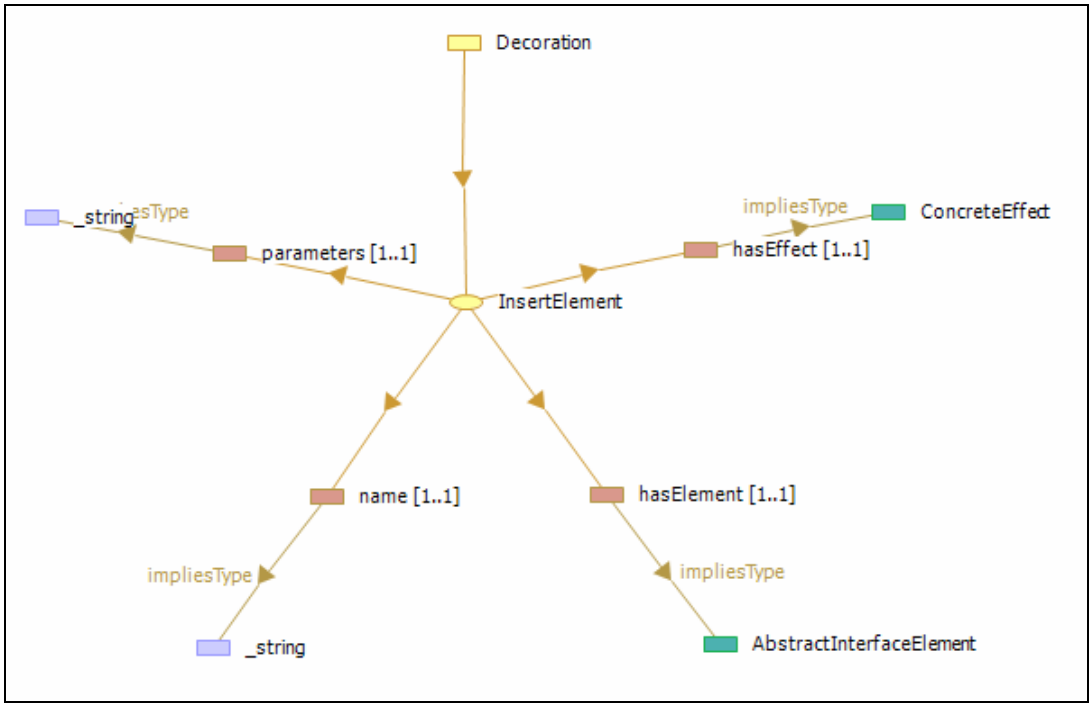


name: Identificador da instância.

hasElement: Indica qual elemento de interface é afetado pela decoração.

hasEffect: Indica o efeito apresentado pela decoração.

parameters: Parâmetros para o efeito aplicado pela decoração.

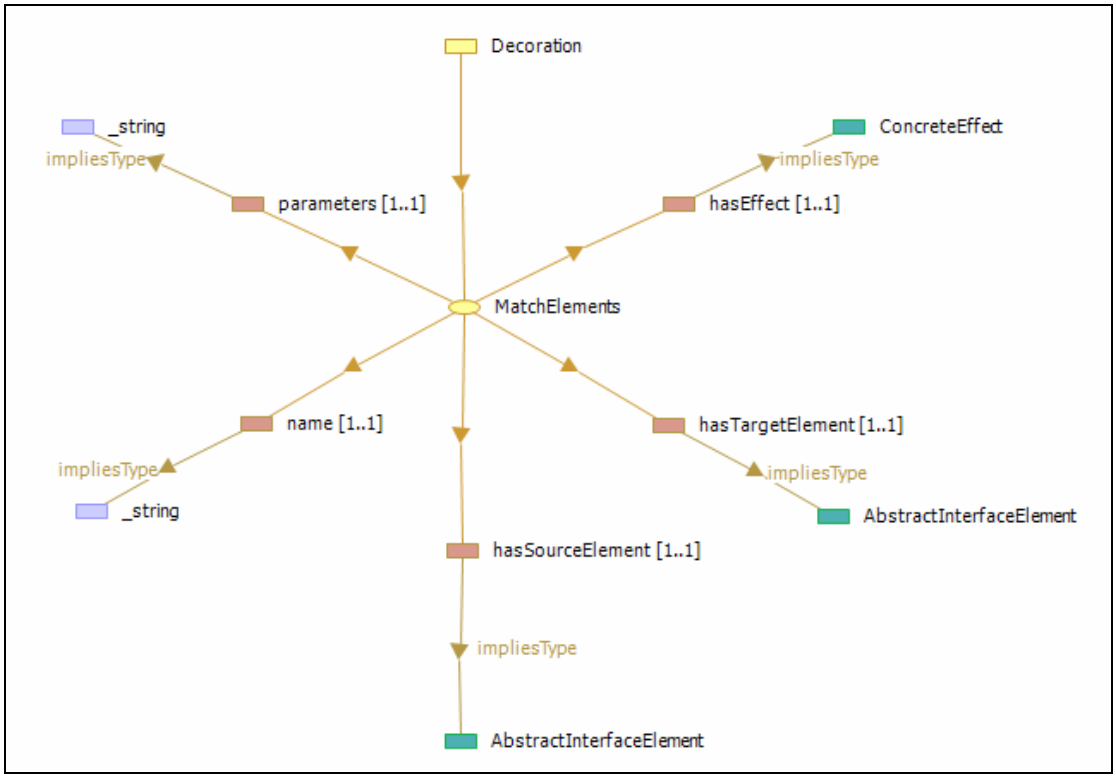


name: Identificador da instância.

hasElement: Indica qual elemento de interface é afetado pela decoração.

hasEffect: Indica o efeito apresentado pela decoração.

parameters: Parâmetros para o efeito aplicado pela decoração.



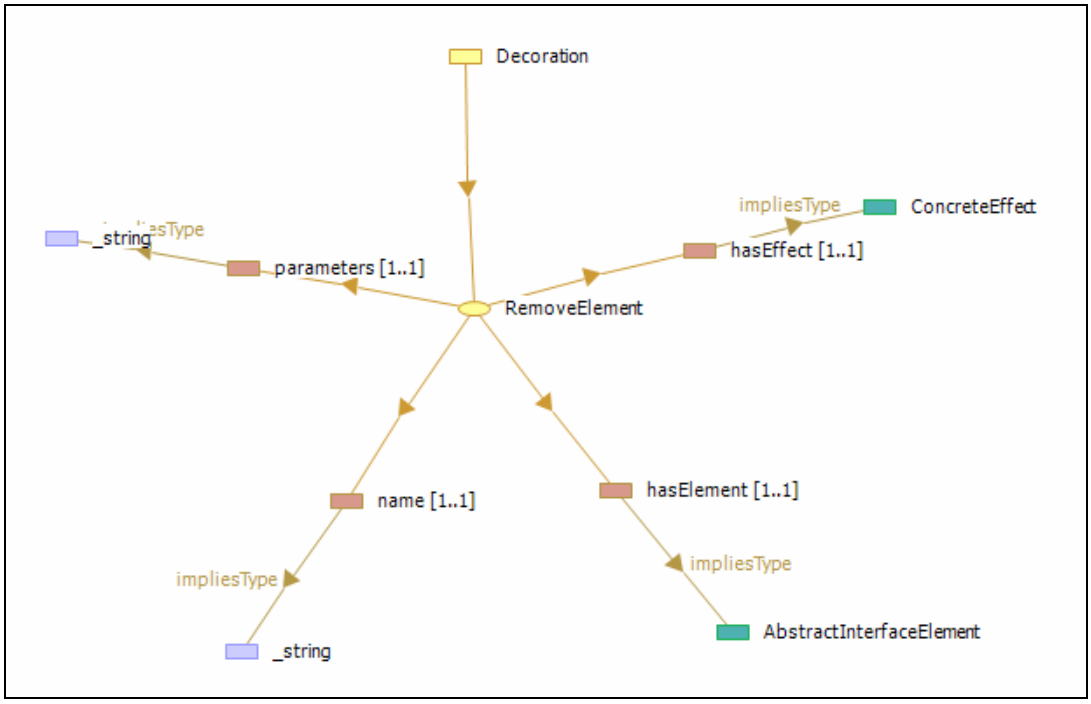
name: Identificador da instância.

hasSourceElement: Indica qual elemento na interface de origem participa da decoração.

hasTargetElement: Indica qual elemento na interface de destino participa da decoração.

hasEffect: Indica o efeito apresentado pela decoração.

parameters: Parâmetros para o efeito aplicado pela decoração.

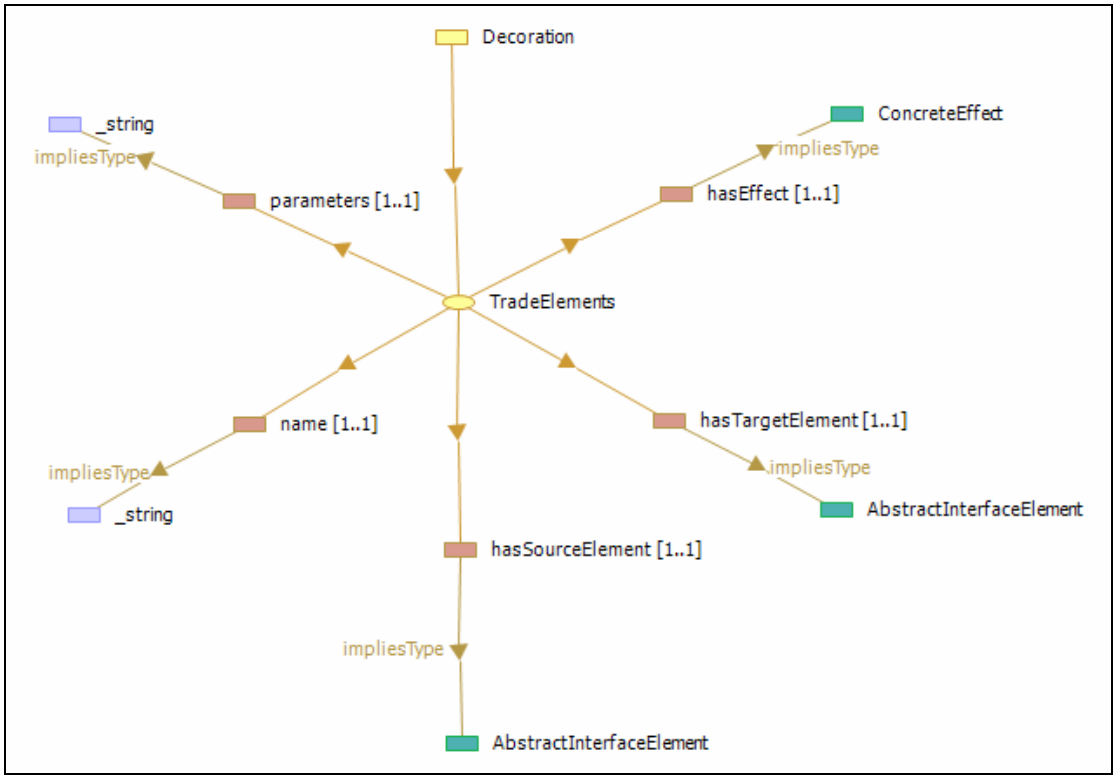


name: Identificador da instância.

hasElement: Indica qual elemento de interface é afetado pela decoração.

hasEffect: Indica o efeito apresentado pela decoração.

parameters: Parâmetros para o efeito aplicado pela decoração.



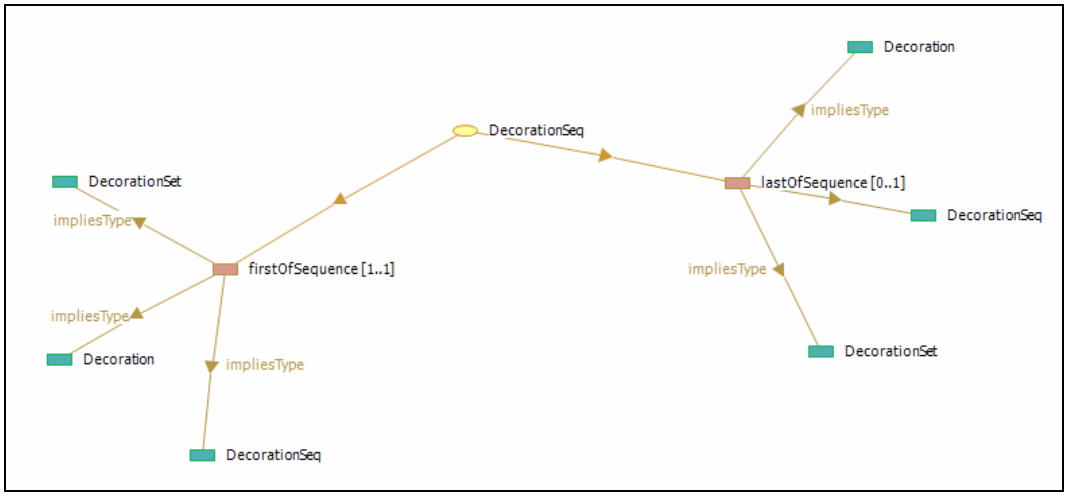
name: Identificador da instância.

hasSourceElement: Indica qual elemento na interface de origem participa da decoração.

hasTargetElement: Indica qual elemento na interface de destino participa da decoração.

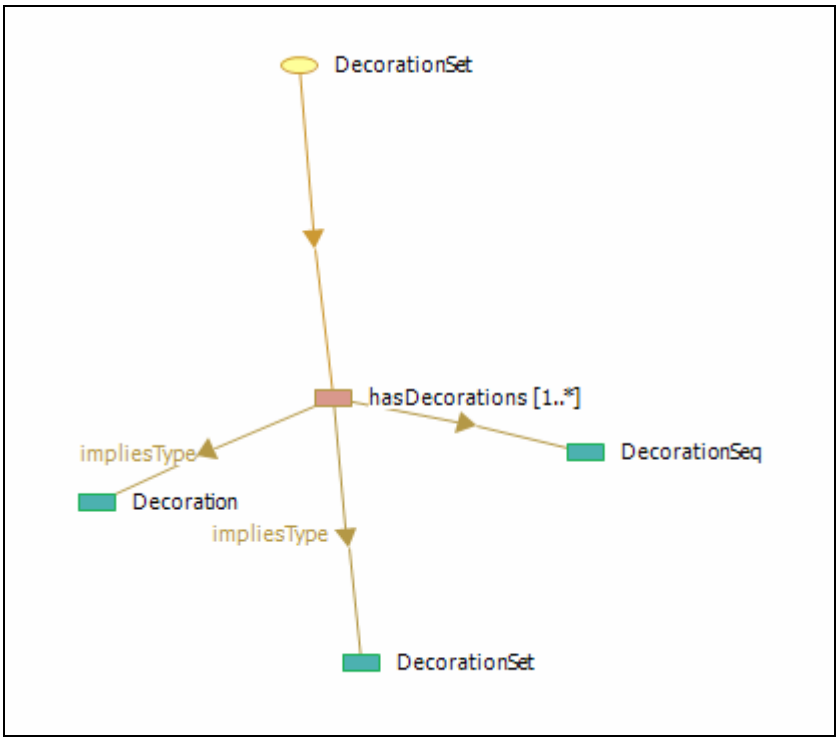
hasEffect: Indica o efeito apresentado pela decoração.

parameters: Parâmetros para o efeito aplicado pela decoração.

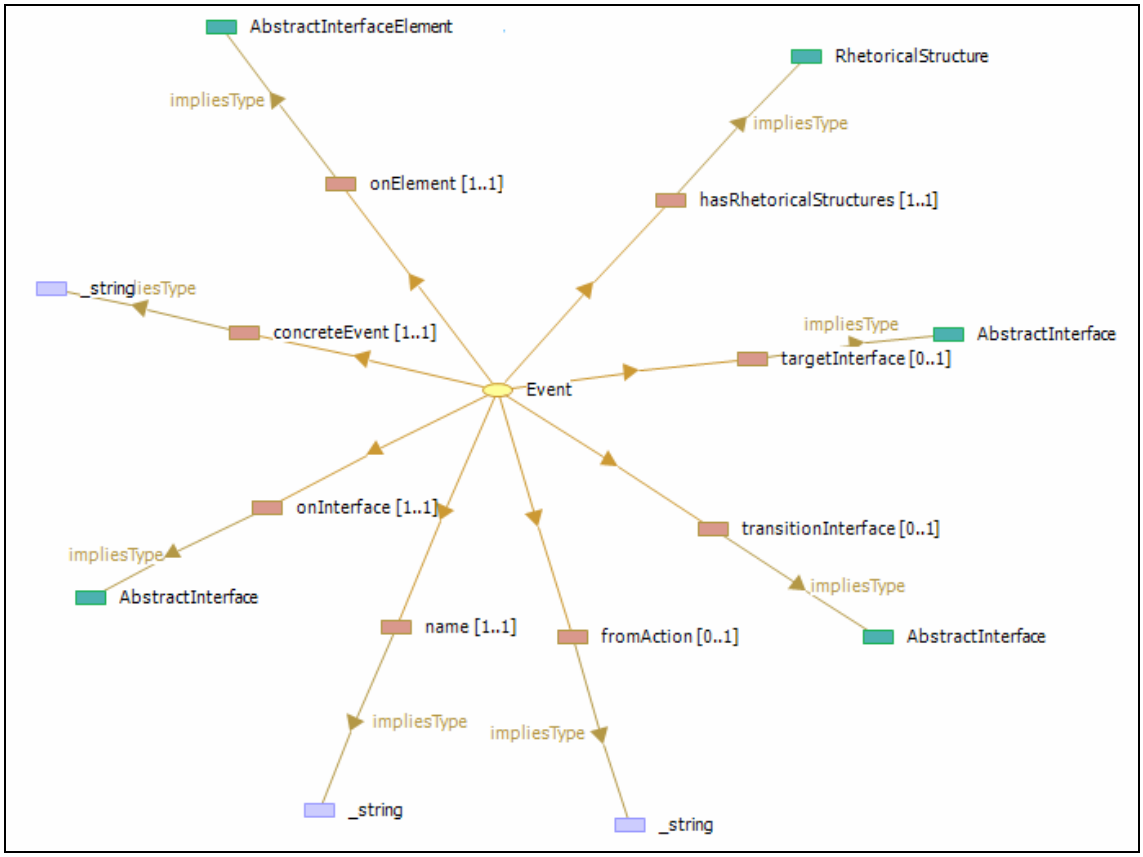


firstOfSequence: Indica o primeiro elemento na seqüência de decorações.

lastOfSequence: Indica o último elemento na seqüência de decorações



hasDecorations: Indica as decorações executadas simultaneamente.



name: Identificador da instância.

onElement: Indica em qual elemento de interface ocorre o evento.

onInterface: Identificador da interface que contém o elemento referenciado em *onElement*.

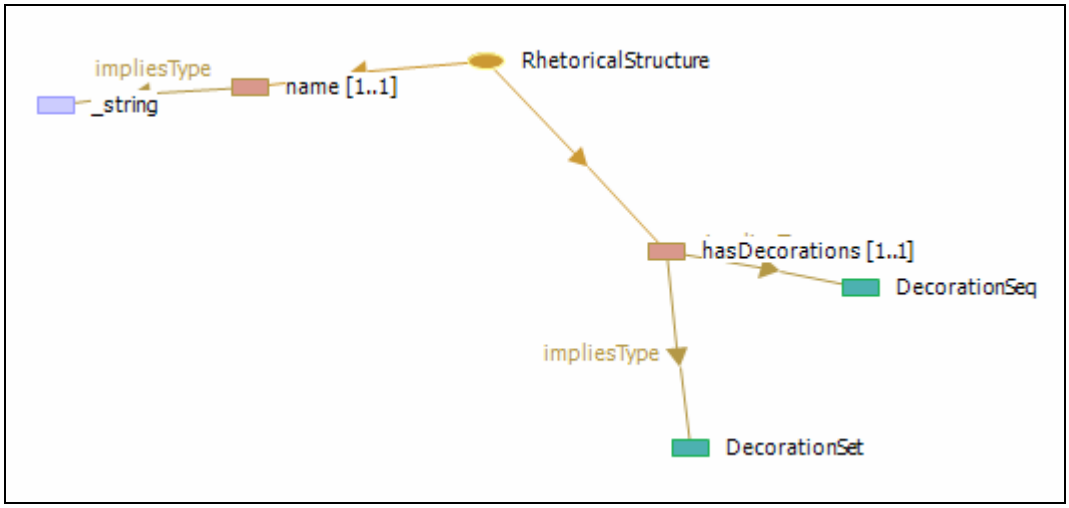
hasRhetoricalStructures: Indica qual estrutura retórica será executada em resposta ao evento.

concreteEvent: Nome do evento no ambiente de implementação.

fromAction: Indica qual ação no Modelo será executada quando ocorrer o evento.

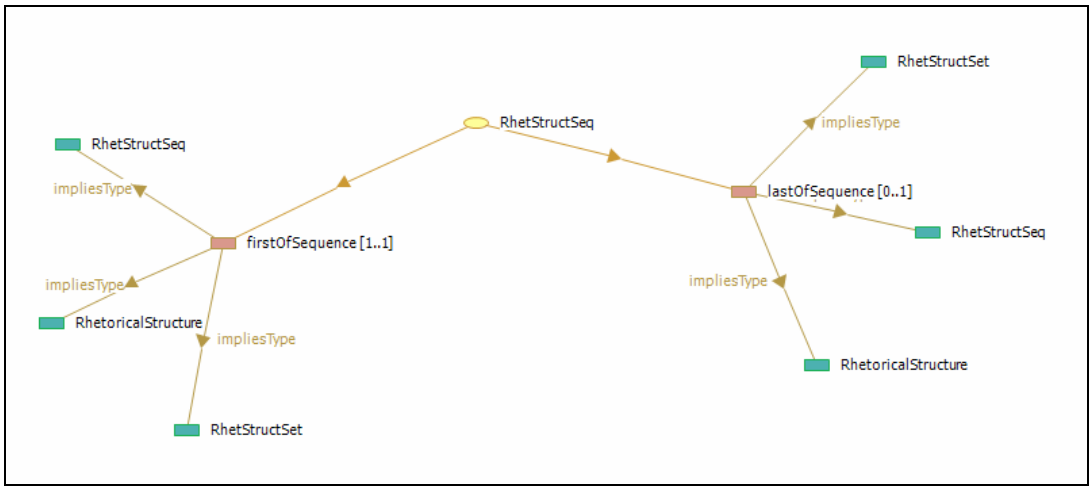
targetInterface: Indica qual instância da classe *AbstractInterface* será usada para exibir o resultado da ação referenciada pela propriedade *fromAction*.

transitionInterface: Indica qual instância da classe *AbstractInterface* será exibida durante a execução da ação referenciada pela propriedade *fromAction*.



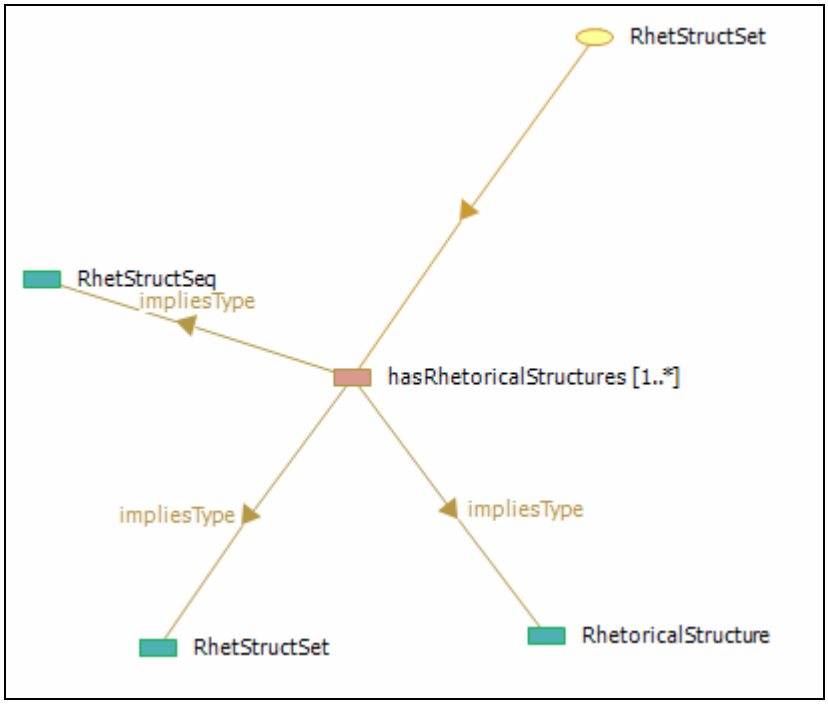
name: Identificador da instância.

hasDecorations: Indica as decorações que compõem a estrutura retórica.

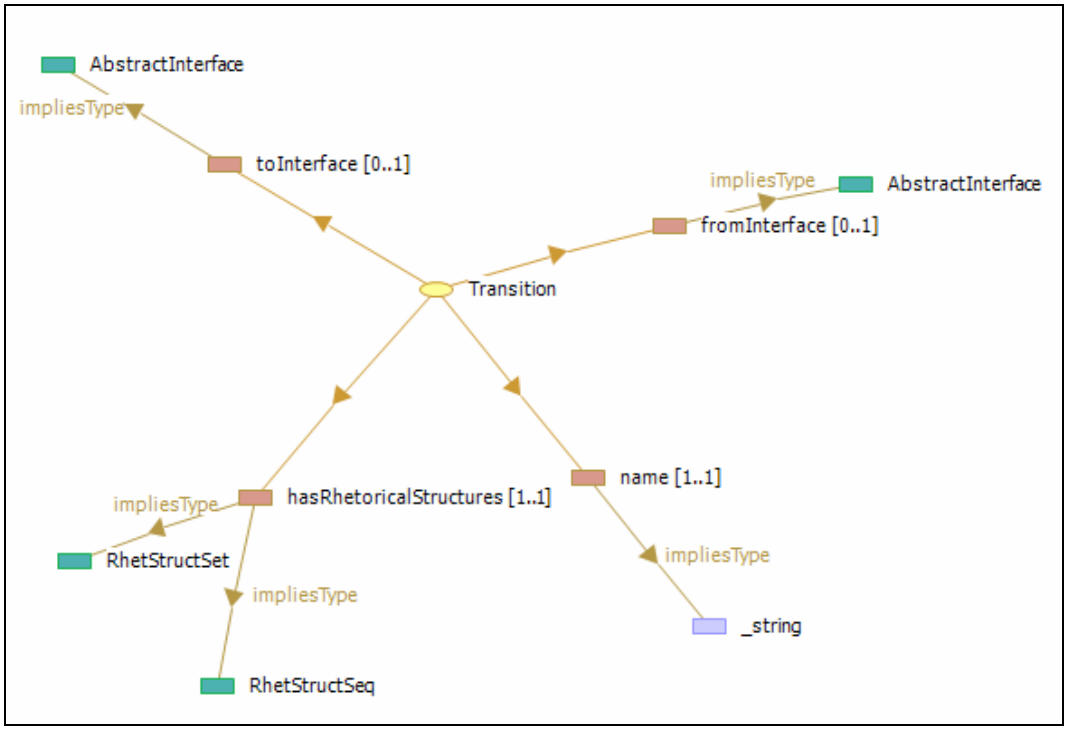


firstOfSequence: Indica o primeiro elemento na seqüência de estruturas retóricas.

lastOfSequence: Indica o último elemento na seqüência de estruturas retóricas.



hasRhetoricalStructures: Indica as estruturas retóricas executadas simultaneamente.



name: Identificador da instância.

hasRhetoricalStructures: Indica qual estrutura retórica compõe a transição.

fromInterface: Indica a interface inicial da transição.

toInterface: Indica a interface final da transição.

Apêndice II – Gramática da Linguagem de Descrição de Widgets: SWUI.dtd[80]

```
<!ELEMENT AbstractInterface ( CompositeInterfaceElement | ElementExhibitor |
IndefiniteVariable | PredefinedVariable | SimpleActivator )* >
<!ATTLIST AbstractInterface tagAttributes CDATA #IMPLIED >
<!ATTLIST AbstractInterface cssClasses CDATA #IMPLIED >
<!ATTLIST AbstractInterface mapsTo CDATA #IMPLIED >
<!ATTLIST AbstractInterface name ID #REQUIRED >

<!ELEMENT CompositeInterfaceElement ( CompositeInterfaceElement | ElementExhibitor
| IndefiniteVariable | PredefinedVariable | SimpleActivator )* >
<!ATTLIST CompositeInterfaceElement tagAttributes CDATA #IMPLIED >
<!ATTLIST CompositeInterfaceElement cssClasses CDATA #IMPLIED >
<!ATTLIST CompositeInterfaceElement loadInterfac[81]e CDATA #IMPLIED >
<!ATTLIST CompositeInterfaceElement fromAttribute CDATA #IMPLIED >
<!ATTLIST CompositeInterfaceElement fromClass CDATA #IMPLIED >
<!ATTLIST CompositeInterfaceElement ordered ( true | false ) #IMPLIED >
<!ATTLIST CompositeInterfaceElement mapsTo CDATA #IMPLIED >
<!ATTLIST CompositeInterfaceElement name ID #REQUIRED >

<!ELEMENT ElementExhibitor EMPTY >
<!ATTLIST ElementExhibitor tagAttributes CDATA #IMPLIED >
<!ATTLIST ElementExhibitor cssClasses CDATA #IMPLIED >
<!ATTLIST ElementExhibitor fromAttribute CDATA #IMPLIED >
<!ATTLIST ElementExhibitor mapsTo CDATA #IMPLIED >
<!ATTLIST ElementExhibitor name ID #REQUIRED >
<!ATTLIST ElementExhibitor defaultContent CDATA #IMPLIED >

<!ELEMENT IndefiniteVariable EMPTY >
<!ATTLIST IndefiniteVariable tagAttributes CDATA #IMPLIED >
<!ATTLIST IndefiniteVariable cssClasses CDATA #IMPLIED >
<!ATTLIST IndefiniteVariable fromAttribute CDATA #IMPLIED >
<!ATTLIST IndefiniteVariable enabled ( true | false ) #IMPLIED >
<!ATTLIST IndefiniteVariable mapsTo CDATA #IMPLIED >
<!ATTLIST IndefiniteVariable name ID #REQUIRED >
<!ATTLIST IndefiniteVariable defaultContent CDATA #IMPLIED >

<!ELEMENT PredefinedVariable EMPTY >
<!ATTLIST PredefinedVariable tagAttributes CDATA #IMPLIED >
```

[80] A opção por utilizar o DTD (*Document Type Definition*) como linguagem de descrição da gramática, em vez do XML Schema, foi motivada pela necessidade de se especificar composições de elementos com cardinalidade 0..N e sem ordem pré-definida. A sintaxe DTD para representar este tipo de regra é bem mais concisa que a equivalente em XML Schema.

```
<!ATTLIST PredefinedVariable cssClasses CDATA #IMPLIED >
<!ATTLIST PredefinedVariable fromAttribute CDATA #IMPLIED >
<!ATTLIST PredefinedVariable mapsTo CDATA #IMPLIED >
<!ATTLIST PredefinedVariable name ID #REQUIRED >

<!ELEMENT SimpleActivator ( ElementExhibitor? ) >
<!ATTLIST SimpleActivator tagAttributes CDATA #IMPLIED >
<!ATTLIST SimpleActivator cssClasses CDATA #IMPLIED >
<!ATTLIST SimpleActivator fromAttribute CDATA #IMPLIED >
<!ATTLIST SimpleActivator fromAction CDATA #IMPLIED >
<!ATTLIST SimpleActivator targetInterface CDATA #IMPLIED >
<!ATTLIST SimpleActivator transitionInterface CDATA  #IMPLIED >
<!ATTLIST SimpleActivator mapsTo CDATA #IMPLIED >
<!ATTLIST SimpleActivator name ID #REQUIRED >
```

Apêndice III – Classes CSS do Módulo de Interfaces Ricas

```
.swui_bold {
  font-weight: bold;
}

.swui_italics {
  font-style: italic;
}

.swui_inline {
  display: inline;
}

.swui_invisible, .swui_ajax_invisible  {
  display: none;
}

.swui_underline {
  text-decoration: underline;
}

.swui_floatLeft {
  float: left;
        margin: inherit;
}

.swui_floatRight {
  float: right;
        margin: inherit;
}

.swui_clearFloat {
  clear: both;
}

.swui_interface {
  margin: 0px;
  padding: 0px;
  width: 80%;
}

.swui_composite {
  padding: 10px 20px;
  margin: 5px;
}
```

```
.swui_landmark {
  border-bottom: 3px solid #e80;
  text-align: center;
}

.swui_navBar {
  padding-top: 0px;
  border-bottom: 1px solid #e80;
  padding-bottom: 35px;
}

.swui_list {
  border-top: 1px solid #e80;
  border-left: 1px solid #e80;
  width: 60%;
}

.swui_node {
  border-left: 1px solid #e80;
  margin: 0 auto;
  width: 70%;
}

.swui_border {
        border: 1px solid #e80;
}
```

Apêndice IV – Interfaces *Default* do Módulo de Interfaces Ricas

DefaultAbstractInterface

```
<AbstractInterface name="DefaultAbstractInterface" mapsTo="Composition"
cssClasses="swui_interface">
    <CompositeInterfaceElement name="Node" mapsTo="Composition" fromClass="NodeClass">
        <%= include DefaultJSONDisplay %>
    </CompositeInterfaceElement>
</AbstractInterface>
```

DefaultContextInterface

```
<AbstractInterface name="DefaultContextInterface" mapsTo="Composition"
cssClasses="swui_interface">

    <CompositeInterfaceElement name="ContextNode" mapsTo="Composition"
fromClass="ContextClass">

        <%= include DefaultLandmarks %>
        <%= include DefaultNavBar %>
        <%= include DefaultContextIndex %>

        <CompositeInterfaceElement name="Node" mapsTo="Composition"
cssClasses="swui_composite swui_node swui_floatRight">
            <%= include DefaultJSONDisplay %>
        </CompositeInterfaceElement>

    </CompositeInterfaceElement>
</AbstractInterface>
```

DefaultIndexInterface

```
<AbstractInterface name="DefaultIndexInterface" mapsTo="Composition" tagAttributes=""
cssClasses="">
```

```
            <%= include DefaultLandmarksForIndexes %>

            <CompositeInterfaceElement name="Index" mapsTo="Composition"
cssClasses="swui_composite">

                <CompositeInterfaceElement name="Entry" mapsTo="Composition" fromClass="IndexClass"
cssClasses="swui_composite">
                    <%= include DefaultJSONDisplay %>
                </CompositeInterfaceElement>

            </CompositeInterfaceElement>

        </AbstractInterface>
```

DefaultLandmarks

```
        <CompositeInterfaceElement name="Landmarks" mapsTo="Composition"
fromAttribute="swui:landmarks" cssClasses="swui_composite swui_landmark" >

            <CompositeInterfaceElement name="Landmark" mapsTo="Paragraph" fromClass="Landmark"
cssClasses="swui_composite swui_inline" >

                <SimpleActivator mapsTo="Link" fromAttribute="swui:landmarks|swui:landmark|swui:target"
name="landmarks-landmark-target" >
                    <ElementExhibitor mapsTo="Text" fromAttribute="swui:landmarks|swui:landmark"
name="landmarks-landmark" />
                </SimpleActivator>

            </CompositeInterfaceElement>

        </CompositeInterfaceElement>
```

DefaultLandmarksForIndexes

```
        <CompositeInterfaceElement name="Landmarks" mapsTo="Composition" cssClasses="swui_composite
swui_landmark" >

            <CompositeInterfaceElement name="Landmark" mapsTo="Paragraph" fromClass="Landmark"
cssClasses="swui_composite swui_inline" >

                <SimpleActivator mapsTo="Link" fromAttribute="swui:landmark|swui:target"
```

```
name="landmarks-landmark-target" >
                        <ElementExhibitor mapsTo="Text" fromAttribute="swui:landmark"
name="landmarks-landmark" />
                    </SimpleActivator>

            </CompositeInterfaceElement>

        </CompositeInterfaceElement>
```

DefaultNavBar

```
        <CompositeInterfaceElement name="NavBar" mapsTo="Composition" cssClasses="swui_composite
swui_navBar" >

            <SimpleActivator mapsTo="Link" fromAttribute="swui:contextPrev|item|swui:target"
name="contextPrev-item-target" cssClasses="swui_floatLeft">
                    <ElementExhibitor mapsTo="Text" fromAttribute="swui:contextPrev|item"
name="contextPrev-item" cssClasses="swui_floatLeft"/>
            </SimpleActivator>

            <SimpleActivator mapsTo="Link" fromAttribute="swui:contextNext|item|swui:target"
cssClasses="swui_floatRight" name="contextNext-item-target" >
                    <ElementExhibitor mapsTo="Text" fromAttribute="swui:contextNext|item"
cssClasses="swui_floatRight" name="contextNext-item" />
            </SimpleActivator>

        </CompositeInterfaceElement>
```

DefaultContextIndex

```
        <CompositeInterfaceElement name="ContextIndex" mapsTo="Composition"
fromAttribute="swui:contextIdx" cssClasses="swui_composite swui_clearFloat swui_floatLeft">

            <CompositeInterfaceElement name="ContextIndexEntry" mapsTo="Composition"
fromClass="ContextIndexEntry">

                    <SimpleActivator mapsTo="Link" fromAttribute="swui:idxEntry|item|swui:target"
name="idxEntry-item-target" >
                        <ElementExhibitor mapsTo="Text" fromAttribute="swui:idxEntry|item"
name="idxEntry-item" />
                    </SimpleActivator>
```

```
        </CompositeInterfaceElement>

    </CompositeInterfaceElement>
```

DefaultJSONDisplay

```
    <%
        ModelUtils.traverseJSON(jsonData,list) do |type,key,value,hash,list|
            if !(key.include? "swui:")
              name = key
              name = name.gsub(/\|/,"-")
              name = name.gsub(/_/,"-")
              if (type=="Item")
                buffer << "<CompositeInterfaceElement mapsTo=\"Composition\" name=\"#{name}Attr\" >"
                  buffer << "<ElementExhibitor name=\"#{name}Lbl\" mapsTo=\"Text\"
defaultContent=\"#{name.capitalize}: \" cssClasses=\"swui_bold\"/>" if !list
              if (hash.has_key? "#{key}|swui:target")
                  buffer << "<SimpleActivator mapsTo=\"Link\" name=\"#{name}Link\"
fromAttribute=\"#{key}|swui:target\">"
              end
                widget = "<ElementExhibitor mapsTo=\"Text\" name=\"#{name}\" fromAttribute=\"#{key}\"/>"
                  widget.gsub!(/mapsTo="Text"/,"mapsTo=\"Image\"") if
(value.match(/.+\.(gif|jpeg|jpg|png|tiff)$/))
                buffer << widget
              if (hash.has_key? "#{key}|swui:target")
                  buffer << "</SimpleActivator>"
              end
                buffer << "</CompositeInterfaceElement>"
              end
              if (type=="OpenList")
            buffer << "<CompositeInterfaceElement mapsTo=\"Composition\" name=\"#{name}List\"
fromAttribute=\"#{key}\" cssClasses=\"swui_composite\" >"
                  buffer << "<ElementExhibitor name=\"#{name}ListLbl\" mapsTo=\"Text\"
defaultContent=\"#{name.capitalize}: \" cssClasses=\"swui_bold\"/>"
                buffer << "<CompositeInterfaceElement mapsTo=\"Composition\" name=\"#{name}\"
fromClass=\"#{name}\" >"
              end
            buffer << "</CompositeInterfaceElement> </CompositeInterfaceElement>" if (type=="CloseList")
              end
        end
    %>
```