Referências Bibliográficas

[1] Gotel, O., Finkelstein, A., *An Analysis of the Requirements Traceability Problem*, in Proc. of the First International Conference on Requirements Engineering, p 94-101, 1994.

[2] Egyed, A., *A Scenario-Driven Approach to Traceability*, in Proc. of the 23rd International Conference on Software Engineering, p 123-134, 2001.

[3] Ramesh, B., Jarke, M., *Toward Models for Requirements Traceability*, IEEE Transactions on Software Engineering, p 58-93, Vol 27, No 1, 2001.

[4] U.S. Department of Defense, *Military Standard 2167A – Defense System Software Development*, Washington, D.C., 1988.

[5] *Capability Maturity Model.* Documentação disponível em http://www.sei.cmu.edu/cmm/cmms/cmms.html

[6] Palmer, J., *Traceability, Software Requirements Engineering*, R.H. Thayer and M. Dorfman, eds., p. 364-374, 1997.

[7] Hamilton, V., Beeby, M., *Issues of Traceability in Integrating Tools*, in Proc. of the IEE Colloquium on Tools and Techniques for Maintaining Traceability during Design, p 4/1-4/3, Dec 1991.

[8] Cybulski, J., Reed, K., *Requirements Classification and Reuse: Crossing Domain Boundaries*, in Proc. of the 6th International Conference on Software Reuse, IEEE, p 190-210, 2000.

[9] Leite, J., C., et al., *Enhancing a Requirements Baseline with Scenarios*, in Proc. of the Third IEEE International Symposium on Requirements Engineering (RE'97) – Annapolis, USA – IEEE Computer Society Press, p 44-53, 1997.

[10] Breitman, K., *Evolução de Cenários*, Tese de Doutorado, PUC/RJ, Maio, 2000.

[11] Murphy, G., Notkin, D., *Lightweight Lexical Source Model Extraction*, ACM Transactions on Software Engineering and Methodology, Vol. 5, No. 3, p 262-292, July 1996.

[12] Gupta, A., *Program Understanding Using Program Slivers -- An Experience Report*, International Conference on Software Maintenance, IEEE, p 66-71, 1997.

[13] Kontogiannis, K., *Evaluation Experiments on the Detection of Programming Patterns Using Software Metrics*, in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 44-55, 1997.

[14] Bojic, D., Velasevic, D., *A Use-Case Driven Method of Architecture Recovery for Program Understanding and Reuse Reengineering*, in Proc. of the 4th European Conference on Software Maintenance and Reengineering, p 23-32, 2000.

[15] Jerding, D., Rugaber, S., *Using Visualization for Architectural Localization and Extraction*, in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 56-65, 1997.

[16] Lange, D., Nakamura, Y., *Object-oriented Program tracing and Visualization*, IEEE Computer, p 63-70, Mai 1997.

[17] Richner, T., Ducasse, S., *Recovering High-Level Views of Object-Oriented Applications from Static and Dynamic Information*, in Proc. of the International Conference on Software Maintenance, IEEE, p 13-22, 1999.

[18] DeBaud, J., *DARE: Domain-Augmented Reengineering*, in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 164-175, 1997.

[19] Girard, J., Koschke, R., Schied, G., *Comparison of Abstract Data Type and Abstract State Encapsulation Detection Techniques for Architectural Understanding*, in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 66-75, 1997,

[20] Wiggerts, T. Bosma, H., Fielt, E., *Scenarios for the Identification of Objects in Legacy Systems* , in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 24-32, 1997.

[21] Yeh, A., Harris, D., Reubenstein, H., *Recovering abstract data types and object instances from a conventional procedural language*, in Proc. of the

Working Conference on Reverse Engineering, IEEE Computer Press, p 227-236, 1995.

[22] Antoniol, G., Fiutem, R., Lutteri, G., Merlo, E., *Program Understanding and Maintenance with the CANTO Environment*, in Proc. of the International Conference on Software Maintenance, p 72-84, 1997.

[23] Liwu, L., *On Managing Classes for Evolving Software*, in Proc. of the 7th International Workshop on Program Comprehension, p 144-150, 1999.

[24] Korel, B., Rilling, J., *Dynamic Program Slicing in Understanding of Program Execution*, in Proc. of the 5th International Workshop on Program Comprehension, 1997.

[25] Zhao, J., *A Slicing-Based Approach to Extracting Reusable Software Architectures*, in Proc. of the 4th European Conference on Software Maintenance and Reengineering, p 215-226, 2000.

[26] Cânfora, G., Cimitile, A., De Lucia, A., Di Lucca, G., *Decomposing Legacy Programs: a First Step Towards Migrating to Client-Server Platforms*, in Proc. of the 6th International Workshop on Program Comprehension, p 136-144, 1998.

[27] Brito, F., Sousa, A., *A Coupling-Guided Cluster Analysis Approach to Reengineer the Modularity of Object-Oriented Systems*, in Proc. of the 4th European Conference on Software Maintenance and Reengineering, p 13-22, 2000.

[28] Mancoridis, S., Mitchell, B., Rorres, C., Chen, Y., Gansner, E., *Using Automatic Clustering to Produce High-Level System Organizations of Source Code*, in Proc. of the 6th International Workshop on Program Comprehension, p 45-53, 1998.

[29] Wiggerts, T., *Using Clustering Algorithms in Legacy Systems Remodularization*, in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 33-43, 1997.

[30] Sartipi, K., Kontogiannis, K., Mavaddat, F., *Architectural Design Recovery using Data Mining Techniques*, in Proc. of the 4th European Conference on Software Maintenance and Reengineering, p 129-140, 2000.

[31] Siff, M., Reps, T., *Identifying Modules Via Concept Analysis*, in Proc. of the International Conference on Software Maintenance, p 170-179, 1997.

[32] Biggerstaff, T., Mitbander, B., Webster, D., *The Concept Assignment Problem in Program Understanding*, in Proc. of the International Conference on Software Engineering, p 482-498, 1993.

[33] Girard, J., Koschke, R., *Finding Components in a Hierarchy of Modules: a Step towards Architectural Understanding*, in Proc. of the International Conference on Software Maintenance, p 58-65, 1997.

[34] Hartman, J., *Understanding Natural Programs Using Proper Decomposition*, in Proc. of the International Conference on Software Engineering, p 62-73, 1991.

[35] Quilici, A., Woods, S., Zhang, Y., *New Experiments with a Constraint-Based Approach to Program Plan Matching*, in Proc. of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 114-123, 1997.

[36] Fiutem, R., Tonella, P., Antoniol, G., Merlo, E., *A Cliche-Based Environment to Support Architectural Reverse Engineering*, in Proc. of the Working Conference on Reverse Engineering, IEEE Computer Press, p 277-286, 1996.

[37] Pinheiro, F., Goguem, J., *An Object Oriented Tool for Tracing Requirements*, IEEE Software, 13(2), p 52-64, 1996.

[38] Murphy, G., Notkin, D., *Reengineering with Reflexion Models: A Case Study*, IEEE Computer, Aug 97, p 29-36, 1997.

[39] Murphy, G., Notkin, D., Sullivan, K., *Software Reflexion Models: Bridging the Gap Between Source and High-Level Models*, in Proceedings of SIGSOFT'95, ACM, 1995.

[40] Haumer, P., et al., *Improving Reviews by Extended Traceability*, proceedings of the 32nd Hawai International Conference on Systems Science, p 3052-3061, 1999.

[41] Lingamarla, S., et al., *System for Automated Validation of Embedded Software in Multiple Operating Configurations*, Automated Software Engineering, 1999.

[42] Liu, K., Alderson, A., Qureshi, Z., *Requirements Recovery from Legacy Systems by Analysing and Modelling Behaviour*, Proceedings of the International Conference on Software Maintenance, IEEE, p 3-12, 1999.

[43] Baxter, I. D., Mehlich, M., *Reverse Engineering is Reverse Forward Engineering*, Proceedings of the 4th Working Conference on Reverse Engineering, IEEE Computer Press, p 104-113, 1997.

[44] Baxter, I., Pidgeon, C., *Software Change Through Design Maintenance*, Proceedings of the International Conference on Software Maintenance, 1997, IEEE Computer Press, p 250-259, 1997.

[45] Baxter, I., *Design Reuse and Scale: Keys to Practical Code Generation and Large Scale Software Maintence*, proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, IEEE Computer Press, p 119-120, 2000.

[46] Neighbors, J., *The Draco Approach to Constructing Software from Reusable Components*, IEEE Transactions on Software Engineering, SE-10, p 564-573, Sep. 1984.

[47] Freeman, P., *A Conceptual Analysis of the Draco Approach to Constructing Software Systems*, IEEE Transactions on Software Engineering, SE-13(7), p 830-844, July 1987.

[48] Wirfs-Brock, R., Wilkerson, B., Wiener, L., *Designing Object-Oriented Software*, Prentice Hall International, Englewood Cliffs, NJ, 1990.

[49] Leite, J.C.S.P, Sant'Anna, M. and Prado, A.F. *Porting Cobol Programs Using a Transformational Approach*, Journal of Software Maintenance: Research and Practice, John Wiley Sons Ltd., Vol. 9, p 3-31, 1997.

[50] Santana, A., Prado, A., Souza, W., Sant'Anna, M., *Automatic Refinement of Distributed Systems Specifications Using Program Transformations*, Proceedings of COMPSAC'98 (1998 Computer Software & Applications Conference), p 154-163, 1998.

[51] Penteado, R., et al. , *Reengineering of Legacy Systems Based on Transformation Using the Oriented Object Paradigm*, in Proc. of the Working Conference on Reverse Engineering, p 144-153, 1998.

[52] Freitas, F.G., *EXL: Uma Linguagem de Extração para Reengenharia de Software*, Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 1997.

[53] Bergmann, U., Leite, J.C., *From Applications Domains to Executable Domains: Achieving Reuse with a Domain Network*, in Proceedings of the 6th International Conference on Software Reuse, p 41-57, 2000.

[54] Bergmann, U., Leite, J.C., *Domain Networks in the Software Development Process*, in Proceedings of the 7th International Conference on Software Reuse, p 194-209, 2002.

[55] Neighbors, J., *Software Construction Using Components*, PhD thesis, University of California at Irvine, 1980.

[56] Freitas, F.G. and Leite, J.C.S.P., *Reusing Domains for the Construction of Reverse Engineering Tools*, Proceedings of the 6th Working Conference on Reverse Engineering, IEEE Computer Press, p 24-35, 1999.

[57] Bergmann, U., *Construção de um Domínio de Desenvolvimento de Software Orientado a Objetos Segundo o Paradigma Draco*, Dissertação de Mestrado, Instituto Militar de Engenharia, 1996.

[58] ISO/TC97/SC21/WG1/FDT/B, *Estelle a Formal Description Technique based on an Extended Transition Model*, ISO, 1986.

[59] Branco, L.H., Prado, A.F., et al., *Automatic Implementation of Distributed Systems Specifications in Model*, Workshop on Object-Oriented Specification Techniques for Distributed Systems and Behaviours, 1999.

[60] Sant´Anna, M., *Circuitos Transformacionais*, tese de doutorado, PUC-Rio, 2001.

[61] Carroll, J.M., *Scenario Based Design: Envisioning Work and Technology in System Development*, John Wiley and Sons, 1995.

[62] Fillipidou, D., *Designing with Scenarios: a critical view on current research and practice*, in Requirements Engineering Journal – edited by Springer Verlag, Vol.3, No.1, p 1-22, 1998.

[63] Breitman, K., Leite, J., *Scenario-Based Software Process*, in Proc. of the 7th International Conference and Workshop on the Engineering of Computer Based Systems, p 375-381, 2000.

[64] Leite, J.C.S.P. et al., *Enhancing a Requirements Baseline with Scenarios*, Requirements Engineering Journal Vol. 2 No. 4, Springer Verlag, December, p 184-198, 1998.

[65] Weidenhaupt, K., Pohl,K., Jarke,M., Haumer,P., *Scenarios in System Development: Current Practice*, IEEE Software, Vol(15), No 2, p 34-45, 1998.

[66] Hughes, T., Martin, Cindy, *Design Traceability for Complex Systems*, proceedings of the 4th Annual Symposiun on Human Interaction with Complex Systems, 1998.

[67] Kautz, H.A., Allen, J.F., *Generalized Plan Recognition*, in Proceedings of the 5th Nat. Conf. AI, p 32-37, 1986.

[68] *Rational Rose Product Family*. Documentação disponível em www.rational.com

[69] *Together ControlCenter*. Documentação disponível em www.togethersoft.com

[70] Wang, J., et al., *An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, p 889-895, Aug 1998.

[71] Quilici, A., Yang, Q., *Applyng Plan Recognition Algorithms to Program Understanding*, in Proceedings of the 11th Knowledge-Based Software Engineering Conference (KBSE), p 96-103, 1996.

[72] Allen, J.F., Kautz, H.A., Pelavin, R.N., Tenenberg, J.D., *Reasoning About Plans*, Morgan Kaufmann Publishers, 1991.

[73] Breitman, K., *Evolução de Cenários*, Tese de Doutorado, PUC/RJ, Maio, 2000.

[74] Marzal, A., Vidal, E., *Computation of Normalized Edit Distance and Applications*, IEEE Transactions on. Pattern Analysis and Machine Intelligence, vol. 15, no. 9, p 926-932, 1993.

[75] *Requirements Capture, Documentation and Validation* – Dagstuhl- Seminar Report 242 – 13.06.99 – 18.06.99 (99241) - Schloss Dagstuhl, 1999.

[76] Lesh, N., Etzioni, O., *A Sound and Fast Goal Recognizer*, in Proc. 14th Int. Joint Conf. AI, p 1704-1710, 1995.

[77] Lin, D., Goebel, R., *A Message Passing Algorithm for Plan Recognition,* in Proc. 12th Int. Joint Conf. AI, volume 1, p 280-285, 1990.

[78] Pinheiro, F., Goguem, J., *An Object Oriented Tool for Tracing Requirements*, IEEE Software, 13(2), p 52-64, 1996.

[79] Antoniol, G., Canfora, G., De Lucia, A., *Maintaining Traceability During Object-Oriented Software Evolution: a Case Study*, in Proceedings of the International Conference on Software Maintenance, p 211-219, 1999.

[80] Baxter, I., *Design Maintenance Systems*, Communications of the ACM, 35(4), p 73-89, 1992.

[81] Holt, R., Pak, J., *Gase: Visualizing Software Evolution-in-the-large*, In Proceedings of the Working Conference on Reverse Engineering, p 163–166, 1996.

Apêndice A
Definição de Tipos de Documentos (DTD) utilizados

A.1.
Léxico Ampliado da Linguagem (LAL)

```
<!ELEMENT symbol (name,alias*,notion*,behavior*) >

<!ELEMENT name (#PCDATA | reference)* >

<!ELEMENT alias (#PCDATA) >

<!ELEMENT notion (#PCDATA | reference)* >
<!ATTLIST notion address CDATA #IMPLIED>
<!ATTLIST notion type (isa|partof|composedby|common) "common">

<!ELEMENT behavior (#PCDATA | reference)* >
<!ATTLIST behavior address CDATA #IMPLIED>

<!ELEMENT reference EMPTY>
<!ATTLIST reference name CDATA #REQUIRED>
<!ATTLIST reference address CDATA #IMPLIED>
<!ATTLIST reference version CDATA #REQUIRED>
<!ATTLIST reference label CDATA #IMPLIED>
```

A.2.
Cenários

```
<!ELEMENT scenario (title , goal* , context* , actor* , resource* , episode*)>

<!ELEMENT title (#PCDATA | reference)*>

<!ELEMENT goal (#PCDATA | reference)*>
<!ATTLIST goal  address CDATA  #IMPLIED >

<!ELEMENT context (#PCDATA | reference)*>
<!ATTLIST context  address CDATA  #IMPLIED >

<!ELEMENT actor (#PCDATA | reference)*>
<!ATTLIST actor  address CDATA  #IMPLIED >

<!ELEMENT resource (#PCDATA | reference)*>
<!ATTLIST resource  address CDATA  #IMPLIED >

<!ELEMENT episode ((definition | execute) , exception* , restriction*)>
<!ATTLIST episode  address CDATA  #IMPLIED >

<!ELEMENT definition (#PCDATA | reference)*>
<!ELEMENT execute (reference)>

<!ELEMENT exception (trigger , (definition | execute))>
<!ATTLIST exception  address CDATA  #IMPLIED >

<!ELEMENT trigger (#PCDATA | reference)*>

<!ELEMENT restriction (#PCDATA | definition | execute)*>
<!ATTLIST restriction  address CDATA  #IMPLIED >

<!ELEMENT reference EMPTY>
<!ATTLIST reference name CDATA #REQUIRED>
<!ATTLIST reference address CDATA #IMPLIED>
<!ATTLIST reference version CDATA #REQUIRED>
<!ATTLIST reference label CDATA #IMPLIED>
```

Apêndice B
Gramáticas dos Domínios Draco-Puc

B.1.
Domínio Draco-Puc para definição de DTDs

```
%%
document          : [elementDecl | attlistDecl | Comment]*  .( , .nl , )
                  ;
elementDecl       : '<!ELEMENT' name elementDefinition  '>'
                  ;
elementDefinition: ('EMPTY' | 'ANY' | mixed | elements)
                  ;
mixed             : '('  '#PCDATA' (  '|' name )* ')*'
                  | '('  '#PCDATA' ')'
                  ;
elements          : (choice | seq) ('?' | '*' | '+')?
                  ;
choice            : '(' cp ('|' cp)+ ')'
                  ;
seq               : '(' cp (',' cp)* ')'
                  ;
cp                : (name | choice | seq) ('?' | '*' | '+')?
                  ;
attlistDecl       : '<!ATTLIST' element_name att_name attDef+ '>'
                  ;
element_name      : name
                  ;
att_name          : name
                  ;
attDef            : attType default
                  ;
attType           : stringType
                  | tokenizedType
                  | enumeratedType
                  ;
stringType        : 'CDATA'
                  ;
tokenizedType     : 'ID'
                  ;
enumeratedType    : notationType
                  | enumeration
                  ;
notationType      : 'NOTATION' '(' not_item ( '|' not_item)* ')'
                  ;
not_item          : name
                  ;
enumeration       : '(' enum_item ( '|' enum_item)* ')'
                  ;
enum_item         : name
                  ;
```

```
default             : '#REQUIRED'
                    | '#IMPLIED'
                    | ('#FIXED'? String)
                    | String
                    | name
                    |
                    ;
name                : Name
                    ;

/***********************************************/
/*              LEXICO                      */
/***********************************************/
String              : \"([^\"])*\"
                    ;
String              : \'([^\'])*\'
                    ;
Name                : ([a-zA-Z]|"-")([a-zA-Z]|[0-9]|"."|":")*
                    ;
Comment             : "<!--"[^"-"]*("-"[^"-"]+)*"-->"
                    ;
IGNORE              : [\t \n]
                    ;
IGNORE              : "//"[^\n]*\n
                    ;
IGNORE              : "/*""/"*([^*/]|[^*]"/"|"*"[^/])*"*""*"*/"
                    ;

%%
```

B.2. Léxico Ampliado da Linguagem (LAL)

```
%{
#define setInsideTAG() BEGIN 0
#define setOutsideTAG() BEGIN REG_OUTSIDETAG
%}

%%

document          : .lm prolog symbol misc*
                  ;
prolog            : xmlDecl? misc* docTypeDecl? misc*
                  ;
xmlDecl           :    XMLDeclStartTag    versionInfo    encodingDecl?    rmDecl?
XMLDeclEndTag
                  ;
versionInfo       : 'version' '=' String
                  ;
misc              : PI
                  ;
docTypeDecl       : .nl StartTag '!DOCTYPE' .sp 'symbol' .sp externalID? EndTag
                  ;
externalID        : 'SYSTEM' .sp String
                  ;
encodingDecl      : 'encoding' '=' String
                  ;
rmDecl            : 'RMD' '=' ( 'NONE' | 'INTERNAL' | 'ALL' )
                  ;
symbol            :  .nl  .slm  .lm(+2)  StartTag  'symbol'   EndTag  (name  alias*  notion*
behavior*)
.slm .lm(-2) .slm '</symbol>'
                  ;
name              : .nl .slm .lm(+2) StartTag 'name'  EndTag (text|reference)*
             .slm .lm(-2) .slm '</name>'
                  ;
alias             : .nl .slm .lm(+2) StartTag 'alias'  EndTag text '</alias>' .lm(-2)
                  ;
notion            :  .nl  .slm  .lm(+2)  StartTag  'notion'   .sp  (  att_address_from_notion  .sp  |
.sp att_type_from_notion .sp )*  EndTag (text|reference)* .slm .lm(-2) .slm '</notion>'
                  ;
att_address_from_notion
                  : 'address' '='  String
                  ;
att_type_from_notion
                  : 'type' '=' enum_att_type_from_notion
                  ;
enum_att_type_from_notion
                  : '"isa"' | '"partof"' | '"composedby"' | '"common"'
                  ;
behavior          : .nl .slm .lm(+2) StartTag 'behavior'  .sp att_address_from_behavior
EndTag (text|reference)* .slm .lm(-2) .slm '</behavior>'
                  ;
att_address_from_behavior
                  : 'address' '='  String   |
                  ;
```

```
reference           : .nl .slm .lm(+2) StartTag 'reference'  .sp ( att_name_from_reference
.sp | .sp att_address_from_reference .sp | .sp att_version_from_reference .sp | .sp
att_label_from_reference .sp )*  EmptyEndTag .lm(-2)
                    ;
att_name_from_reference
                    : 'name' '='  String
                    ;
att_address_from_reference
                    : 'address' '='  String
                    ;
att_version_from_reference
                    : 'version' '='  String
                    ;
att_label_from_reference
                    : 'label' '='  String
                    ;
text                : Text
                    ;
XMLDeclStartTag
                    : "<?xml"       { setInsideTAG(); }
                    ;
XMLDeclEndTag
                    : "?>"       { setOutsideTAG(); }
                    ;
StartTag            : "<"       { setInsideTAG(); }
                    ;
EndTag              : ">"       { setOutsideTAG(); }
                    ;
EmptyEndTag         : "/>"       { setOutsideTAG(); }
                    ;
String              : \"([^\"])*\"
                    ;
String              : \'([^\'])*\'
                    ;
PI                  : "<?"[^"?"]*("?"[^">"]+)*"?>"
                    ;
IGNORE              : [\t \n]
                    ;
IGNORE              : "//"[^\n]*\n
                    ;
IGNORE              : "/*""/"*([^*/]|[^*]"/"|"*"[^/])*"*""*"*/"
                    ;
IGNORE              : "<!--"([^\-]|(\-[^\-]))*"-->"
                    ;
Text                : <REG_OUTSIDETAG>[^\>\<\{\}\[\]\?]*[^ \n\t\>\<\{\}\[\]\?]+[^\>\<\{\}\[\]\?]*
                    ;
Text                : "??"[^\>\<\{\}\[\]\?]*"??"
                    ;
%%
```

B.3. Cenários

```
%{
#define setInsideTAG() BEGIN 0
#define setOutsideTAG() BEGIN REG_OUTSIDETAG
%}
%%
document          : .lm prolog scenario misc*
                  ;
prolog          : xmlDecl? misc* docTypeDecl? misc*
                  ;
xmlDecl                    :  .nl  XMLDeclStartTag  versionInfo  encodingDecl?  rmDecl?
XMLDeclEndTag
                  ;
versionInfo       : 'version' '=' String
                  ;
misc            : PI
                  ;
docTypeDecl       : .nl StartTag '!DOCTYPE' .sp 'scenario' .sp externalID? EndTag
                  ;
externalID        : 'SYSTEM' .sp String
                  ;
encodingDecl      : 'encoding' '=' String
                  ;
rmDecl           : 'RMD' '=' ( 'NONE' | 'INTERNAL' | 'ALL' )
                  ;
scenario           : .nl .slm .lm(+2) StartTag 'scenario'  EndTag (title goal* context* actor*
resource* episode*)
                   .slm .lm(-2) .slm '</scenario>'
                  ;
title           : .nl .slm .lm(+2) StartTag 'title'  EndTag (text|reference)*
                   .slm .lm(-2) .slm '</title>'
                  ;
goal             : .nl .slm .lm(+2) StartTag 'goal'  .sp att_address_from_goal
                   EndTag (text|reference)* .slm .lm(-2) .slm '</goal>'
                  ;
att_address_from_goal
                  : 'address' '='  String   |
                  ;
context           : .nl .slm .lm(+2) StartTag 'context'  .sp att_address_from_context
                   EndTag (text|reference)* .slm .lm(-2) .slm '</context>'
                  ;
att_address_from_context
                  : 'address' '='  String   |
                  ;
actor             : .nl .slm .lm(+2) StartTag 'actor'  .sp att_address_from_actor
                   EndTag (text|reference)* .slm .lm(-2) .slm '</actor>'
                  ;
att_address_from_actor
                  : 'address' '='  String   |
                  ;
resource           : .nl .slm .lm(+2) StartTag 'resource'  .sp att_address_from_resource
                   EndTag (text|reference)* .slm .lm(-2) .slm '</resource>'
                  ;
att_address_from_resource
                  : 'address' '='  String   |
                  ;
```

```
episode          : .nl .slm .lm(+2) StartTag 'episode'  .sp att_address_from_episode
                EndTag ((definition|execute) exception* restriction*) .slm
                .lm(-2) .slm '</episode>'
               ;
att_address_from_episode
               : 'address' '='  String   |
               ;
definition       : .nl .slm .lm(+2) StartTag 'definition'  EndTag (text|reference)*
                .slm .lm(-2) .slm '</definition>'
               ;
execute          : .nl .slm .lm(+2) StartTag 'execute'  EndTag (reference) .slm
                .lm(-2) .slm '</execute>'
               ;
exception        : .nl .slm .lm(+2) StartTag 'exception'  .sp att_address_from_exception
                EndTag (trigger (definition|execute)) .slm .lm(-2) .slm '</exception>'
               ;
att_address_from_exception
               : 'address' '='  String   |
               ;
trigger          : .nl .slm .lm(+2) StartTag 'trigger'  EndTag (text|reference)*
                .slm .lm(-2) .slm '</trigger>'
               ;
restriction      : .nl .slm .lm(+2) StartTag 'restriction'  .sp att_address_from_restriction
                EndTag (text|definition|execute)* .slm .lm(-2) .slm '</restriction>'
               ;
att_address_from_restriction
               : 'address' '='  String   |
               ;
reference        : .nl .slm .lm(+2) StartTag 'reference'  .sp ( att_name_from_reference .sp |
.sp   att_address_from_reference   .sp   |   .sp   att_version_from_reference   .sp   |   .sp
att_label_from_reference .sp )*
                EmptyEndTag .lm(-2)
               ;
att_name_from_reference
               : 'name' '='  String
               ;
att_address_from_reference
               : 'address' '='  String
               ;
att_version_from_reference
               : 'version' '='  String
               ;
att_label_from_reference
               : 'label' '='  String
               ;
text             : Text
               ;
XMLDeclStartTag
               : "<?xml"       { setInsideTAG(); }
               ;
XMLDeclEndTag
               : "?>"       { setOutsideTAG(); }
               ;
StartTag         : "<"       { setInsideTAG(); }
               ;
EndTag           : ">"       { setOutsideTAG(); }
               ;
EmptyEndTag      : "/>"       { setOutsideTAG(); }
               ;
String           : \"([^\"])*\"
```

```
                    ;
String                : \'([^\'])*\'
                    ;
PI                  : "<?"[^"?"]*("?"[^">"]+)*"?>"
                    ;
IGNORE                   : [\t \n]
                    ;
IGNORE                   : "//"[^\n]*\n
                    ;
IGNORE                   : "/*""/"*([^*/]|[^*]"/"|"*"[^/])*"*"*"*/"
                    ;
IGNORE                   : "<!--"([^\-]|(\-[^\-]))*"-->"
                    ;
Text                  : <REG_OUTSIDETAG>[^\>\<\{\}\[\]\?]*[^ \n\t\>\<\{\}\[\]\?]+[^\>\<\{\}\[\]\?]*
                    ;
Text                  : "??"[^\>\<\{\}\[\]\?]*"??"
                    ;
%%
```

B.4.
Domínio de representação de Diferenças (Diff)

```
%%
document            : diffDocument
                    | refineDocument
                    | embeddedStatement
                    ;
diffDocument         : diffElement*
                    ;
refineDocument      : 'Evolution Program' refineStatement* 'end'
                    ;
refineStatement     : 'evolve' String 'to' String 'using' String ';'
                    ;
diffElement         : .nl replaceSpec
                    | .nl insertSpec
                    | .nl removeSpec
                    ;
replaceSpec         : 'replace' .sp typeSpec .sp 'from' .nl .sp .sp any .nl 'to' .nl .sp .sp any .nl
'end'
                    ;
insertSpec          : 'insert' .sp typeSpec .nl .sp .sp any .nl where 'end'
                    ;
removeSpec          : 'remove' .sp typeSpec .nl .sp .sp any .nl where 'end'
                    ;
typeSpec            : 'type' '=' whatChange
                    ;
where               : 'in' .nl  .sp .sp  any .nl
                    ;
embeddedStatement       : locater
                    ;
locater             : 'start' | 'end'    | 'here'
                    ;
any                 : Text*
                    ;
whatChange          : 'text'
                    | 'reference'
                    | 'title'
                    | 'goal'
                    | 'context'
                    | 'actor'
                    | 'resource'
                    | 'episode'
                    | 'definition'
                    ;
String          : \"([^\"])*\"
              ;
String          : \'([^\'])*\'
              ;
Text                : [0-9a-zA-Z]*
                    ;
IGNORE             : [\t \n]
              ;
IGNORE             : "//"[^\n]*\n
              ;
IGNORE             : "/*""/"*([^*/]|[^*]"/"|"*"[^/])*"*""*"*/"
              ;%%
```

B.5. Domínio de Planos (Plan)

```
%%
program            : planLibrary? instanceProgram? planRecognition?
                   ;
/****************************************************************************
                   PLAN LIBRARY
****************************************************************************/
planLibrary        : planLibraryImport
                   | planLibraryDefinition
                   ;
planLibraryImport         : 'loadLibrary' String '.'
                   ;
planLibraryDefinition : .nl 'PlanLibrary'
                    .nl 'begin'
                       .slm  eventDeclaration*
                       .nl .lm(-3) 'end'
                   ;
eventDeclaration : eventIdentification parListdecl .sp eventAttribute declBody
                   ;
eventIdentification       : .slm 'Event' .sp eventName
                   ;
parListdecl        : '('  parDeclaration**','  ')'
                   ;
parDeclaration     : Name
                   | previusVersion
                   | nextVersion
                   | any
                   ;
previusVersion     : Name '-'
                   ;
nextVersion        : Name '+'
                   ;
eventAttribute     : 'is' .sp 'EndEvent'
                   |
                   ;
any                : '*'
                   ;
declBody           : .slm 'begin'
                        specializationRule? decomposition?
                     .slm 'end'
                   | ';'
                   ;
specializationRule        : .nl .slm .lm(+3) .slm 'isa' .sp eventName .slm .lm(-3)
                   ;
decomposition      : .slm .lm(+3) .slm 'composedBy' .slm .lm(+3) decompositionStatement+
.slm .lm(-6)
                   ;
decompositionStatement  : .slm eventName parListdecl decompositionRule ';'
                      ;
decompositionRule         : .sp 'by' .sp ruleName parameters
                   ;
parameters         : .sp 'with' .sp parameterPair++',' .(,,.sp ,)
                   |
                   ;
parameterPair      : Name .sp '=' .sp String
                   ;
```

```
ruleName         : Name
                 ;
eventName        : Name
                 ;

/****************************************************************************
                              INSTANCE PROGRAM
****************************************************************************/
instanceProgram: .nl 'Observations'
                  .nl 'begin'
                     observedEventList
                     .nl .lm(-3) 'end'
                 ;
observedEventList        : observedEvent*
                 ;
observedEvent    : .slm eventName observation
                 ;
observation      : argList value
                 ;
argList          : '('  argValue**','  ')'
                 ;
value            : '[' String ']'
                 |
                 ;
argValue         : String version
                 | '*'
                 ;
version          : ':' Number
                 |
                 ;

/****************************************************************************
                              RECOGNITION PROGRAM
****************************************************************************/
planRecognition  : .nl 'Recognition'
                  .nl 'begin'
                     topInferredEvent*
                  .nl .lm(-3) 'end'
                 ;

topInferredEvent : .nl .slm .lm(+3) .slm inferredEvent '.'
                 ;
inferredEvent    : eventName .sp argList .sp value .sp '{' rationaleStatement+ .slm .lm(-3)
'}'
                 ;
anyEvent         : inferredEvent
                 | observedEvent
                 ;
rationaleStatement       :  rationale
                 ;
rationale        : observedEvent
                 | .nl .slm .lm(+3) .slm 'isa' .sp anyEvent .slm .lm(-3)
                 | .nl .slm .lm(+3) .slm inferredPartOfEvent++',' .(,,.sp ,) .slm .lm(-3)
                 ;
inferredPartOfEvent      : ruleName ':' anyEvent
                 ;

String         : \"([^\"])*\"
             ;
String         : \'([^\'])*\'
```

```
            ;
Name          : [a-zA-Z_][0-9a-zA-Z_]*
            ;
Number          : [0-9]+
                ;
IGNORE          : [\t \n]
            ;
IGNORE          : "//"[^\n]*\n
            ;
IGNORE          : "/*""/"*([^*/]|[^*]"/"|"*"[^/])*"*"*"*/"
            ;
%%
```

Apêndice C
Transformadores Draco-Puc

C.1.
Transformador Dtd2Grm

```
Global-Declaration:
{{dast txt.decls

    #include <include/ulfdebug.h>


#define __IS_TO_ULF_DEBUG__ 1

#ifdef __IS_TO_ULF_DEBUG__
#define ULF_DEBUG_MSG(X) {UlfDebugInfo *aDebugInfo = new UlfDebugInfo(X , 0);
aDebugInfo->save(); delete aDebugInfo; }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->save(); delete aDebugInfo;
}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->setAfter(Z); aDebugInfo-
>save(); delete aDebugInfo; }
#else
#define ULF_DEBUG_MSG(X) { }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) { }
#endif



   char  aux[400];       /* rascunho */
   char  aux1[400];      /* rascunho */
   char  programName[100];
     char  currentElement[100]; /* referencia para o elemento atual em definicao */
     char  currentAttribute[100]; /* referencia para o elemento atual em definicao */
     char  rootElement[100];  /* armazena o primeiro elemento */

    void createDomainDir(char *domainName)
    {
        char aux[300];
        sprintf(aux , "md %s" , domainName);
        system(aux);
        sprintf(aux , "md %s\\src" , domainName);
        system(aux);
        sprintf(aux , "md %s\\bin" , domainName);
        system(aux);
        sprintf(aux , "md %s\\samples" , domainName);
        system(aux);
    }

   void debugMessage(char *msg)
   {
     printf("\nDebugMsg -> %s" , msg);
   }

    void createTokenName( char *tn , char *n) {
        sprintf(tn , "'%s'" , n);
    }

void removeNameSpace( char *tn) {
    for(int i=0;i< strlen(tn);i++) {
        if(tn[i] == ':') {
            tn[i] = '_';
        }
    }
}

void removeNameSpace( char *tn , char *n) {
    strcpy(tn , n);
    removeNameSpace(tn);
}


    void createStartElementTag( char *tn , char *n) {
        sprintf(tn , "StartTag '%s'" , n);
/*      removeNameSpace(tn); */
    }

    void createEndElementTag( char *tn , char *n) {
        sprintf(tn , "'</%s>'" , n);
/*      removeNameSpace(tn); */
    }


    /* regra nao terminal nao pode iniciar com maiuscula */
    void createRuleName(char *n) {
        removeNameSpace(n);
        strlwr(n);
    }

void removeExtension( char *tn , char *n) {
    strcpy(tn , n);
    for(int i=0;i< strlen(tn);i++) {
        if(tn[i] == '.') {
            tn[i] = '\0';
            return;
        }
    }
}



void changeVirgulaPorEspaco(char *n) {
    for(int i=0;i< strlen(n);i++) {
        if(n[i] == ',') {
            n[i] = ' ';
        }
    }
    removeNameSpace(n);
}

void  convertElementDefinition(char *aTarget , char *aSource) {
    strcpy(aTarget , "");
    changeVirgulaPorEspaco(aSource);
    strlwr(aSource);
    strcpy(aTarget , aSource);
}

void createMixedElements(char *t ,char *s) {
    /* Trocar #PCDATA por text */
    int is,it;
    char aux[200];

    for(is=0 , it=0 ;is< strlen(s); is++ , it++) {
        if(s[is] == '#') {
            aux[it] = 't';
            aux[it+1] = 'e';
            aux[it+2] = 'x';
            aux[it+3] = 't';
            aux[it+4] = ' ';
            it += 3;
            is += 6;
        } else {
            aux[it] = s[is];
        }
    }
    aux[it] = '\0';
    convertElementDefinition(t , aux); // pode dar erro ser PCDATA tiver sido substituido
por uma regra que tem maiuscula
}

void createAttributeRuleName(char *arn , char *an , char *en) {
        char anAux[200] , anAux1[200];
        RemoveQuotes(an);
        removeNameSpace(anAux , an);
        removeNameSpace(anAux1 , en);
        sprintf(arn , "att_%s_from_%s" , anAux , anAux1 );
    }

    void createAttributeReference(char *al , char *n) {
        int aCount , aResultCount ;
        char teste[500], anAux[200] , anAux1[200];

        sprintf(anAux, "attribute(\"%s\",*attName)" , n);
        if (KBSolve(anAux)) {
            aResultCount = KBRetrieveLength();

            if(aResultCount == 0) {
                strcpy(al , "");
                return;
            }

            if(aResultCount == 1) {
            strcpy(anAux , KBRetrieve( "*attName" , 1));
                createAttributeRuleName(anAux1 , anAux , n);
                strcpy(teste , " .sp ");
                strcat(teste , anAux1);
                strcpy(al , teste);
                return;
            }

            strcpy(teste , " .sp ( ");
        strcpy(anAux , KBRetrieve( "*attName" , 1));
            createAttributeRuleName(anAux1 , anAux , n);
            strcat(teste , anAux1);

            for(aCount = 2  ; aCount < aResultCount ; aCount++)  {
            strcpy(anAux , KBRetrieve( "*attName" , aCount));
                createAttributeRuleName(anAux1 , anAux , n);
                strcat(teste , " .sp | .sp ");
                strcat(teste , anAux1);
            }

        strcpy(anAux , KBRetrieve( "*attName" , aResultCount));
            createAttributeRuleName(anAux1 , anAux , n);
            strcat(teste , " .sp | .sp ");
            strcat(teste , anAux1);
            strcat(teste , " .sp )* ");
            strcpy(al , teste);
        } else {
             strcpy(al , "");
             return;
        }

    }


    /* Verifica se o atributo e' obrigatorio ou nao, gerando a opcao vazia
    REGRA UTILIZADA:  Nao sera opcional se for #REQUIRED  */
    void createOptionalAttribute(char *n , char *an , char *en) {
        char anAux[200];

        // ver se existe mais de um atributo para o elemento. Se existir nao preciso colocar
a regra vazia pois na regra do elemento ja foi
        // colocado o * . Caso contrario colocar a regra vazia
        sprintf(n , "" );
        sprintf(anAux, "attribute(\"%s\",*attName)" , en);
        if (KBSolve(anAux)) {
            if(KBRetrieveLength() == 1) {
                // colocar a regra vazia
                sprintf(anAux, "defaultAttribute(\"%s\",\"%s\",\"REQUIRED\")" , en , an);
                if (KBSolve(anAux)) {

                } else {
                    sprintf(n , " | ");
                }
            }
        }

        return;


/*
        char anAux[200];
        sprintf(anAux, "defaultAttribute(\"%s\",\"%s\",\"REQUIRED\")" , en , an);
         if (KBSolve(anAux)) {
            sprintf(n , "" );
         } else {
            sprintf(n , " | ");
        }
*/

    }


/*  attributeEnumItem("teste1","href1","true").*/
    void createOptionList(char *n , char *an , char *en) {
        int aCount , aResultCount ;
        char teste[500], anAux[200] , anAux1[200];

        strcpy(teste , "");
        sprintf(anAux, "attributeEnumItem(\"%s\",\"%s\",*option)" , en , an);
         if (KBSolve(anAux)) {
            aResultCount = KBRetrieveLength();
            for(aCount = 1  ; aCount <= aResultCount ; aCount++)  {
            strcpy(anAux , KBRetrieve( "*option" , aCount));
/*              RemoveQuotes(anAux); */
                if(aCount > 1)
                    sprintf(anAux1 , " | '%s'" , anAux);
                else
                    sprintf(anAux1 , "'%s'" , anAux);

                strcat(teste , anAux1);
            }
        }
     strcpy(n , teste);
    }

    /* Verifica se e' FIXED : se for responde 1 e atualiza o valor do parametro
    n para o valor fixo */
    int isFixedAttributeValue(char *n , char *an , char *en) {
        char teste[500], anAux[200] , anAux1[200];

        sprintf(anAux, "defaultAttribute(\"%s\",\"%s\",\"FIXED\")" , en , an);
        if (KBSolve(anAux)) {
            sprintf(anAux, "defaultAttributeValue(\"%s\",\"%s\",*defValue)" , en , an);
            if (KBSolve(anAux)) {
            strcpy(anAux , KBRetrieve( "*defValue" , 1));
                createTokenName( n , anAux);
                return 1;
            }
        }
        return 0;
    }

}}

Global-Initialization: {{dast txt.decls
  strcpy(programName , "erro.txt");
  strcpy(rootElement , "vazio");
  KBClear();

}}


Global-End: {{dast txt.decls
    CHANGE_EXTENSION(programName, ".kb");
    KBWrite(programName);
```

```
}}

/*********************************************************************************
        SET OF TRANSFORMS BuildKBForAttributesInformation
Descricao:
Data:
Autor: Ulf Bergmann
*********************************************************************************/
Set Of Transforms BuildKBForAttributesInformation
  Method
    Search: Top-Down
    Apply: Single Step

  Transform FindAttributeDefinition
      Lhs: {{dast dtd.attlistDecl
            <!ATTLIST [[Name en]] [[Name an]]  [[attDef* AD]] >
      }}
      Post-Match:
      {{dast txt.decls
               COPY_LEAF_VALUE_TO(currentElement, "en");
               COPY_LEAF_VALUE_TO(currentAttribute, "an");
               sprintf(aux, "attribute(\"%s\" , \"%s\")",currentElement , currentAttribute );
               KBAssert(aux);
          APPLY("SearchAttributeFeatures", "AD");
          SKIP_APPLY();
          }}


/*********************************************************************************
        SET OF TRANSFORMS SearchAttributeFeatures
Descricao:
Data:
Autor: Ulf Bergmann
*********************************************************************************/
Set Of Transforms SearchAttributeFeatures
  Trigger: external
  Method
    Apply: Single Step

  Transform CDATAAttType
      Lhs: {{dast dtd.attType
            CDATA
      }}
      Post-Match:
      {{dast txt.decls
               sprintf(aux,   "attributeType(\"%s\"   ,   \"%s\"   ,   \"CDATA\")",currentElement   ,
currentAttribute );
               KBAssert(aux);
          SKIP_APPLY();
          }}

  Transform IDAttType
      Lhs: {{dast dtd.attType
            ID
      }}
      Post-Match:
      {{dast txt.decls
               sprintf(aux,    "attributeType(\"%s\"    ,    \"%s\"    ,    \"ID\")",currentElement    ,
currentAttribute );
               KBAssert(aux);
          SKIP_APPLY();
          }}

  Transform notationAttType
      Lhs: {{dast dtd.attType
            [[notationType NT]]
      }}
      Post-Match:
      {{dast txt.decls
              sprintf(aux,  "attributeType(\"%s\"  ,  \"%s\"  ,  \"NOTATION\")",currentElement  ,
currentAttribute );
              KBAssert(aux);
          SKIP_APPLY();
          }}

  Transform enumerationAttType
      Lhs: {{dast dtd.attType
            [[enumeration ET]]
      }}
      Post-Match:
      {{dast txt.decls
              sprintf(aux,   "attributeType(\"%s\"   ,   \"%s\"   ,   \"ENUM\")",currentElement   ,
currentAttribute );
              KBAssert(aux);
          SKIP_APPLY();
          }}

/*enumeration    : '(' name ( '|' name)* ')'
     ;
*/
  Transform enumerationItemAttType
      Lhs: {{dast dtd.enum_item
            [[name N]]
      }}
      Post-Match:
      {{dast txt.decls
              sprintf(aux,   "attributeEnumItem(\"%s\"   ,   \"%s\"   ,   \"%s\")",currentElement   ,
currentAttribute , expand("[[N]]") );
              KBAssert(aux);
          SKIP_APPLY();
          }}

  Transform notationItemAttType
      Lhs: {{dast dtd.not_item
            [[name N]]
      }}
      Post-Match:
      {{dast txt.decls
              sprintf(aux,  "attributeNotationItem(\"%s\"  ,  \"%s\"  ,  \"%s\")",currentElement  ,
currentAttribute , expand("[[N]]") );
              KBAssert(aux);
          SKIP_APPLY();
          }}


/*
default    : '#REQUIRED'
              | '#IMPLIED'
              | ('#FIXED'? String)
*/
```

```
  Transform fixedStringDefaultAttType
     Lhs: {{dast dtd.default
          #FIXED [[String ST]]
     }}
     Post-Match:
     {{dast txt.decls
            sprintf(aux,  "defaultAttribute(\"%s\"  ,  \"%s\"  ,  \"FIXED\")",currentElement  ,
currentAttribute );
            KBAssert(aux);
            COPY_LEAF_VALUE_TO(aux1, "ST");
            sprintf(aux,  "defaultAttributeValue(\"%s\"  ,  \"%s\"  ,  %s)",currentElement  ,
currentAttribute , aux1 );
            KBAssert(aux);
        SKIP_APPLY();
        }}

  Transform stringDefaultAttType
     Lhs: {{dast dtd.default
          [[String ST]]
     }}
     Post-Match:
     {{dast txt.decls
            COPY_LEAF_VALUE_TO(aux1, "ST");
            sprintf(aux,  "defaultAttributeValue(\"%s\"  ,  \"%s\"  ,  %s)",currentElement  ,
currentAttribute , aux1 );
            KBAssert(aux);
        SKIP_APPLY();
        }}


  Transform requiredDefaultAttType
     Lhs: {{dast dtd.default
          #REQUIRED
     }}
     Post-Match:
     {{dast txt.decls
            sprintf(aux, "defaultAttribute(\"%s\" , \"%s\" , \"REQUIRED\")",currentElement ,
currentAttribute );
            KBAssert(aux);
        SKIP_APPLY();
        }}

  Transform impliedDefaultAttType
     Lhs: {{dast dtd.default
          #IMPLIED
     }}
     Post-Match:
     {{dast txt.decls
            sprintf(aux,  "defaultAttribute(\"%s\"  ,  \"%s\"  ,  \"IMPLIED\")",currentElement  ,
currentAttribute );
            KBAssert(aux);
        SKIP_APPLY();
        }}
```

```
/*******************************************************************************
        SET OF TRANSFORMS CreateGrm
Descricao:
Data:
Autor: Ulf Bergmann
*******************************************************************************/
Set Of Transforms CreateGrm
  Method
    Search: Top-Down
    Apply: Single Step

  Init: {{dast txt.decls
     debugMessage("Inicio da geracao do GRM ...");

     CREATE_WORKSPACE("WSFinalProgram" , "grm");
     CREATE_WORKSPACE("WSRules" , "grm");
     CREATE_WORKSPACE("WSDefaultRulesAtStart" , "grm");
     CREATE_WORKSPACE("WSDefaultRulesAtEnd" , "grm");

     CREATE_WORKSPACE("WSMakefile" , "txt");
     CREATE_WORKSPACE("WSBatchfile" , "txt");
     CREATE_WORKSPACE("WSScriptfile" , "txt");

   strcpy(programName, GET_FIRST_MODULE_NAME());
   CHANGE_EXTENSION(programName, ".grm");
  }}

  End: {{dast txt.decls
      debugMessage("Fim da geracao do GRM ...");
    TEMPLATE("CreateDefaultRulesAtStart");
      PLACE_AT("WSDefaultRulesAtStart");
    END_TEMPLATE;

    TEMPLATE("CreateDefaultRulesAtEnd");
      PLACE_AT("WSDefaultRulesAtEnd");
    END_TEMPLATE;

ULF_DEBUG_MSG_BEFORE("WSDefaultRulesAtStart"          ,          I.GetSystemDAST()-
>GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSDefaultRulesAtStart")));
ULF_DEBUG_MSG_BEFORE("WSDefaultRulesAtEnd"          ,          I.GetSystemDAST()-
>GetLocater(I.GetSystemDAST()->GetCompleteObjectName("WSDefaultRulesAtEnd")));


    TEMPLATE("ComposeAll");
      MOVE1("WSRules" , "rules_place");
      MOVE1("WSDefaultRulesAtStart" , "default_rules_place_start");
      MOVE1("WSDefaultRulesAtEnd" , "default_rules_place_end");
      PLACE_AT("WSFinalProgram");
    END_TEMPLATE;

      removeExtension(aux , programName);

    TEMPLATE("CreateMakefile");
        SET_TEMPL_LEAF_VALUE("domainExtension", aux);
      PLACE_AT("WSMakefile");
    END_TEMPLATE;

    TEMPLATE("CreateBatchfile");
      PLACE_AT("WSBatchfile");
    END_TEMPLATE;

    TEMPLATE("CreateScriptfile");
        SET_TEMPL_LEAF_VALUE("domainExtension", aux);
      PLACE_AT("WSScriptfile");
    END_TEMPLATE;
      createDomainDir(aux);
      sprintf(aux1 , "%s\\src\\%s" , aux , programName);
      RENAME("WSFinalProgram", aux1);

      sprintf(aux1 , "%s\\makefile" , aux);
      RENAME("WSMakefile", aux1);

      sprintf(aux1 , "%s\\tf.bat" , aux);
      RENAME("WSBatchfile", aux1);

      sprintf(aux1 , "%s\\samples\\teste.dsf" , aux);
      RENAME("WSScriptfile", aux1);

    DESTROY_WORKSPACE("WSRules");
    DESTROY_WORKSPACE("WSDefaultRulesAtStart");
    DESTROY_WORKSPACE("WSDefaultRulesAtEnd");
    DESTROY_WORKSPACE(GET_FIRST_MODULE_NAME());
  }}

/*------------------------------------------------------------------------------
        TEMPLATE  ComposeAll
Descricao: Ultimo template a ser executado na geracao de codigo exl
     Integra os diversos WS formando o codigo final.
Data: 23/09/99
Autor: Ulf Bergmann
--------------------------------------------------------------------------------*/
  Template ComposeAll
    Rhs: {{dast grm.description

      %{
        #define setInsideTAG() BEGIN 0
        #define setOutsideTAG() BEGIN REG_OUTSIDETAG
     %}

      %%

    [[rule*  default_rules_place_start]]

    [[rule*  rules_place]]

    [[rule*  default_rules_place_end]]

      %%

    }}

  Template CreateDefaultRulesAtEnd
    Rhs: {{dast grm.rules
text    : Text
;
XMLDeclStartTag    : "<?xml"              { setInsideTAG(); }
;
XMLDeclEndTag      : "?>"                 { setOutsideTAG(); }
```

```
;
StartTag          : "<"        { setInsideTAG(); }
;
EndTag            : ">"        { setOutsideTAG(); }
;
EmptyEndTag       : "/>"             { setOutsideTAG(); }
;
String : \"([^\"])*\"
;
String : \'([^\'])*\'
;
PI          : "<?"[^"?"]*("?"[^">"]+)*"?>"
;
IGNORE            : [\t \n]
;
IGNORE            : "//"[^\n]*\n
;
IGNORE            : "/*""/"*([^*/]|[^*]"/"|"*"[^/])*"*""*"*/"
;
IGNORE : "<!--"([^\-]|(\-[^\-]))*"-->"
;
Text   : <REG_OUTSIDETAG>[^\>\<\{\}\[\]\?]*[^ \n\t\>\<\{\}\[\]\?]+[^\>\<\{\}\[\]\?]*
;
Text              : "??"[^\>\<\{\}\[\]\?]*"??"
                 ;

}}


 Template CreateDefaultRulesAtStart
    Pre-Apply  : {{dast txt.decls
        strcpy( aux1 , rootElement );
        createRuleName(aux1);
        createTokenName( aux , rootElement );
        SET_LEAF_VALUE("root_element_token", aux);
        SET_LEAF_VALUE("root_element", aux1);
    }}
  Rhs: {{dast grm.rules
        document            : .lm prolog [[item_  root_element]] misc*
                            ;
        prolog         : xmlDecl? misc* docTypeDecl? misc*
                            ;
        xmlDecl             :   .nl   XMLDeclStartTag   versionInfo   encodingDecl?   rmDecl?
XMLDeclEndTag
                            ;
        versionInfo   : 'version' '=' String
                            ;
        misc                : PI
                            ;
        docTypeDecl    :  .nl  StartTag  '!DOCTYPE'  .sp  [[item_   root_element_token]]  .sp
externalID? EndTag
                            ;
        externalID          : 'SYSTEM' .sp String
                            ;
        encodingDecl   : 'encoding' '=' String
                            ;
        rmDecl              : 'RMD' '=' ('NONE' | 'INTERNAL' | 'ALL')
                            ;
   }}
  Template CreateMakefile
    Rhs: {{dast txt.decls
!include ../../mkfiles/macros.mk
DOMAIN_NAME    = [[decl domainExtension]]
build        : parser pprinter
!include ../../mkfiles/makedomain.mk
!include ../../mkfiles/suffix.mk
    }}

  Template CreateBatchfile
    Rhs: {{dast txt.decls
cd samples
draco teste.dsf %1
cd ..
    }}

  Template CreateScriptfile
    Rhs: {{dast txt.decls
(begin
  (load-domain [[decl domainExtension]] xml)
  (parse %1.xml l)
  (pp l)
  (exit)
)
    }}


  Transform FindElement
      Lhs: {{dast dtd.elementDecl
              <!ELEMENT [[Name N]] [[elementDefinition ED]]  >
      }}
      Post-Match:
      {{dast txt.decls
              COPY_LEAF_VALUE_TO(currentElement, "N");

              /************************ CUIDADO *******************/
              /* estou considerando que a primeira regra é a root */
              /************************ CUIDADO *******************/
              if(strcmp(rootElement , "vazio") == 0) {
                  strcpy(rootElement , currentElement);
              }
          APPLY("CreateElementRule", "ED");
          SKIP_APPLY();
          }}

  Transform FindAttribute
      Lhs: {{dast dtd.attlistDecl
            <!ATTLIST [[Name en]] [[Name an]]  [[attDef* AD]] >
      }}
      Post-Match:
      {{dast txt.decls
              COPY_LEAF_VALUE_TO(currentElement, "en");
              COPY_LEAF_VALUE_TO(currentAttribute, "an");
              sprintf(aux,       "attributeType(\"%s\",\"%s\",*attType)"       ,currentElement       ,
currentAttribute );
              if (KBSolve(aux)) {
                  strcpy(aux , KBRetrieve( "*attType" , 1));
                  if(strcmp(aux , "\"CDATA\"") == 0)
                      strcpy(aux1 ,"CreateCDataAttributeRule");
                  else if(strcmp(aux , "\"NOTATION\"") == 0 )
```

```
                    strcpy(aux1 ,"CreateNotationAttributeRule");
                else if(strcmp(aux , "\"ENUM\"") == 0 )
                    strcpy(aux1 ,"CreateEnumerationAttributeRule");
                else if(strcmp(aux , "\"ID\"") == 0 )
                    strcpy(aux1 ,"CreateIDAttributeRule");
                else strcpy(aux1 ,"CreateUndefinedAttributeRule");

            TEMPLATE (aux1)
                PLACE_AT("WSRules");
              END_TEMPLATE;
            }
        SKIP_APPLY();
        }}


   Template CreateCDataAttributeRule
     Pre-Apply  : {{dast txt.decls
        createTokenName( aux , currentAttribute);
        SET_LEAF_VALUE("aToken", aux);

        createAttributeRuleName(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("aName", aux);

        if( isFixedAttributeValue(aux , currentAttribute , currentElement) == 0 ) {
            sprintf(aux , " String ");
        }
        SET_LEAF_VALUE("aStringOrDefaultValue", aux);

        createOptionalAttribute(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("anOptional", aux);
     }}
     rhs: {{ dast grm.rule
          [[rule_name aName]] : [[item_ aToken]] '=' [[item_ aStringOrDefaultValue]] [[item_
anOptional]]
          ;
   }}

   Template CreateNotationAttributeRule
     Pre-Apply  : {{dast txt.decls
        createTokenName( aux , currentAttribute);
        SET_LEAF_VALUE("aToken", aux);

        createAttributeRuleName(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("aName", aux);

        createOptionalAttribute(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("anOptional", aux);
     }}
     rhs: {{ dast grm.rule
          [[rule_name aName]] : [[item_ aToken]] '=' Notation [[item_ anOptional]]
          ;
   }}

/*
<!ATTLIST teste1 href1 ( true | false ) "false" >
att_href1 : 'href1' '=' enum_att_href1
                ;
enum_att_href1 : "true"
                 | "false"
                 ;
```

```
*/
Template CreateEnumerationAttributeRule
    Pre-Apply  : {{dast txt.decls
        createTokenName( aux , currentAttribute);
        SET_LEAF_VALUE("aToken", aux);

        createAttributeRuleName(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("aName", aux);

        sprintf( aux1 , "enum_%s" , aux);
        SET_LEAF_VALUE("anEnumListName", aux1);

        if( isFixedAttributeValue(aux , currentAttribute , currentElement) == 0 ) {
            createOptionList(aux , currentAttribute , currentElement);
        }
        SET_LEAF_VALUE("aList", aux);

        createOptionalAttribute(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("anOptional", aux);

    }}
    rhs: {{ dast grm.rules
          [[rule_name  aName]]  :  [[item_  aToken]]  '='  [[item_  anEnumListName]]  [[item_
anOptional]]
          ;
            [[rule_name anEnumListName]] : [[item_ aList]]
            ;
   }}

  Template CreateIDAttributeRule
    Pre-Apply  : {{dast txt.decls
        createTokenName( aux , currentAttribute);
        SET_LEAF_VALUE("aToken", aux);

        createAttributeRuleName(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("aName", aux);

        createOptionalAttribute(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("anOptional", aux);
    }}
    rhs: {{ dast grm.rule
          [[rule_name aName]] : [[item_ aToken]] '=' String [[item_ anOptional]]
         ;
   }}


  Template CreateUndefinedAttributeRule
    Pre-Apply  : {{dast txt.decls
        createTokenName( aux , currentAttribute);
        SET_LEAF_VALUE("aToken", aux);

        createAttributeRuleName(aux , currentAttribute , currentElement);
        SET_LEAF_VALUE("aName", aux);
    }}
    rhs: {{ dast grm.rule
          [[rule_name aName]] : [[item_ aToken]] '=' Undefined
         ;
   }}
```

```
/*********************************************************************************
        SET OF TRANSFORMS CreateElementRule
Descricao:
Data: 16/03/2001
Autor: Ulf Bergmann
*********************************************************************************/
Set Of Transforms CreateElementRule
  Trigger: external
  Method
    Apply: Single Step

  Transform emptyElement
     Lhs: {{dast dtd.elementDefinition
            EMPTY
     }}
     Post-Match:
     {{dast txt.decls
         TEMPLATE ("CreateEmptyElementRule")
           PLACE_AT("WSRules");
         END_TEMPLATE;
        SKIP_APPLY();
        }}

  Transform anyElement
     Lhs: {{dast dtd.elementDefinition
            ANY
     }}
     Post-Match:
     {{dast txt.decls
            sprintf(aux , "*** Element %s is ANY *** NOT IMPLEMENTED",
currentElement);
            debugMessage(aux);
        SKIP_APPLY();
        }}

  Transform pcdataElement
     Lhs: {{dast dtd.elementDefinition
            ( #PCDATA )
     }}
     Post-Match:
     {{dast txt.decls
         TEMPLATE ("CreatePCDATAElementRule")
           PLACE_AT("WSRules");
         END_TEMPLATE;
        SKIP_APPLY();
        }}

  Transform mixedElement
     Lhs: {{dast dtd.elementDefinition
            [[ mixed ME]]
     }}
     Post-Match:
     {{dast txt.decls
            SPRINT_VAR("ME" , aux);
            createMixedElements(aux1 , aux);
         TEMPLATE ("CreateMixedElementRule")
                SET_TEMPL_LEAF_VALUE("mixedString", aux1);
           PLACE_AT("WSRules");
         END_TEMPLATE;
        SKIP_APPLY();
        }}

  Transform elementsElement
     Lhs: {{dast dtd.elementDefinition
            [[ elements E]]
     }}
     Post-Match:
     {{dast txt.decls
            SPRINT_VAR("E" , aux);
            char anAux1[400];
            convertElementDefinition(anAux1 , aux);
         TEMPLATE ("CreateElementsElementRule")
                SET_TEMPL_LEAF_VALUE("elementsString", anAux1);
           PLACE_AT("WSRules");
         END_TEMPLATE;
        SKIP_APPLY();
        }}


/*  TEMPLATES */
   Template CreateEmptyElementRule
     Pre-Apply  : {{dast txt.decls
        strcpy( aux , currentElement);
        createRuleName(aux);
        SET_LEAF_VALUE("aName", aux);

        createStartElementTag( aux , currentElement);
        SET_LEAF_VALUE("aStartTag", aux);

        createAttributeReference(aux , currentElement);
        SET_LEAF_VALUE("anAttList", aux);

     }}
     rhs: {{ dast grm.rule
          [[rule_name  aName]]  :  .nl   .slm  .lm(+2)  [[item_  aStartTag]]  [[item_  anAttList]]
EmptyEndTag .lm(-2)
          ;
   }}

   Template CreatePCDATAElementRule
     Pre-Apply  : {{dast txt.decls
        strcpy( aux , currentElement);
        createRuleName(aux);
        SET_LEAF_VALUE("aName", aux);

        createStartElementTag( aux , currentElement);
        SET_LEAF_VALUE("aStartTag", aux);
        createEndElementTag( aux , currentElement);
        SET_LEAF_VALUE("anEndTag", aux);

        createAttributeReference(aux , currentElement);
        SET_LEAF_VALUE("anAttList", aux);

     }}
     rhs: {{ dast grm.rule
          [[rule_name  aName]]  :  .nl   .slm  .lm(+2)  [[item_  aStartTag]]  [[item_  anAttList]]
EndTag text [[item_ anEndTag]] .lm(-2)
          ;
   }}
  Template CreateElementsElementRule
    Pre-Apply  : {{dast txt.decls
        strcpy( aux , currentElement);
        createRuleName(aux);
        SET_LEAF_VALUE("aName", aux);

        createStartElementTag( aux , currentElement);
        SET_LEAF_VALUE("aStartTag", aux);
        createEndElementTag( aux , currentElement);
        SET_LEAF_VALUE("anEndTag", aux);

        createAttributeReference(aux , currentElement);
        SET_LEAF_VALUE("anAttList", aux);

    }}
    rhs: {{ dast grm.rule
          [[rule_name  aName]]  :  .nl  .slm  .lm(+2)   [[item_  aStartTag]]  [[item_  anAttList]]
EndTag [[item_ elementsString]] .slm .lm(-2) .slm [[item_ anEndTag]]
          ;
   }}

  Template CreateMixedElementRule
    Pre-Apply  : {{dast txt.decls
        strcpy( aux , currentElement);
        createRuleName(aux);
        SET_LEAF_VALUE("aName", aux);

        createStartElementTag( aux , currentElement);
        SET_LEAF_VALUE("aStartTag", aux);
        createEndElementTag( aux , currentElement);
        SET_LEAF_VALUE("anEndTag", aux);

        createAttributeReference(aux , currentElement);
        SET_LEAF_VALUE("anAttList", aux);

    }}
    rhs: {{ dast grm.rule
          [[rule_name  aName]]  :  .nl  .slm  .lm(+2)   [[item_  aStartTag]]  [[item_  anAttList]]
EndTag [[item_ mixedString]] .slm .lm(-2) .slm [[item_ anEndTag]]
          ;
   }}
```

C.2. Transformador HandleDiff

```
Global-Declaration:
{{dast txt.decls
    #include <include/ulfdebug.h>

    int nextSOTIndex = 1;
    Port  *currentTLBPort;

#define __IS_TO_ULF_DEBUG__ 1

#define MAX_STR_SIZE 1000

#ifdef __IS_TO_ULF_DEBUG__
#define ULF_DEBUG_MSG(X) {UlfDebugInfo *aDebugInfo = new UlfDebugInfo(X , 0);
aDebugInfo->save(); delete aDebugInfo; }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->save(); delete aDebugInfo;
}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->setAfter(Z); aDebugInfo-
>save(); delete aDebugInfo; }
#else
#define ULF_DEBUG_MSG(X) { }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) { }
#endif

    void getNextSOTName(char *aName) {
        sprintf(aName , "SOT%d" , nextSOTIndex);
        nextSOTIndex++;
    }

   void reportError(char *msg , RawDASTLocater *aFirst , RawDASTLocater *aSec) {
     printf("\nDebugMsg -> %s" , msg);
        UlfDebugInfo aDebugInfo(msg , 1);
        aDebugInfo.setBefore(aFirst);
        aDebugInfo.setAfter(aSec);
        aDebugInfo.save();
   }

   void reportError(char *msg , RawDASTLocater *aFirst) {
     printf("\nDebugMsg -> %s" , msg);
        UlfDebugInfo aDebugInfo(msg , 1);
        aDebugInfo.setBefore(aFirst);
        aDebugInfo.save();
   }

   void reportError(char *msg ) {
     printf("\nDebugMsg -> %s" , msg);
        UlfDebugInfo aDebugInfo(msg , 1);
        aDebugInfo.save();
   }
}}


Global-Initialization: {{dast txt.decls
```

```
    char aSOTFileName[100] , aux[200];
    strcpy(aux , GET_FIRST_MODULE_NAME() );
    CHANGE_EXTENSION(aux, "_");
    sprintf( aSOTFileName , "SOT_%s.tlb" , aux );
    currentTLBPort = new Port(aSOTFileName , "w");
}}

Global-End: {{dast txt.decls
    currentTLBPort->Flush();
    delete currentTLBPort;
}}


/********************************************************************************
********************************************************************************/
Set Of Transforms MainSet
  Method
    Search: Top-Down
    Apply: Single Step
  Init: {{dast txt.decls

  }}
  End: {{dast txt.decls
  }}

Transform handleDiffDocument
Lhs: {{dast diff.diffDocument
    [[diffElement* program]]
}}
Pre-Apply: {{dast txt.decls
    ULF_DEBUG_MSG_BEFORE("INICIO" , (&l));

    APPLY("HandleText", "program");
    APPLY("HandleDifferences", "program");

    ULF_DEBUG_MSG_BEFORE("FIM" , (&l));
    SKIP_APPLY();
}}



/********************************************************************************
********************************************************************************/
Set Of Transforms HandleText
  Trigger: external
  Method
    Apply: Single Step

/*----------------------------------------------------------------------
 ----------------------------------------------------------------------*/
Transform removeTextMarks
Lhs: {{dast cen.text
    [[Text a]]
}}
Pre-Apply: {{dast txt.decls
    // remover os ??
    char anOld[300] , aNew[300];
    COPY_LEAF_VALUE_TO(anOld, "a");
//  printf("\nConversao texto (LEU ... =>%s<=" , anOld);
    int count = 0;
    for(int i=0; (i < strlen(anOld) - 1) && (i < 300) ; i++) {
        if( (anOld[i] == '?') && (anOld[i+1] == '?') ) {
            i = i + 1;
        } else {
            aNew[count] = anOld[i];
          count++;
        }
    }
    // copiar o ultimo
    if(i < strlen(anOld)) {
        aNew[count] = anOld[i];
        count++;
    }
    aNew[count] = '\0';

    SET_LEAF_VALUE("b", aNew);
//  printf("\nConversao texto DE =>%s<=  EM  =>%s<=" , anOld , aNew);
}}
Rhs: {{dast cen.text
    [[Text b]]
}}


/*********************************************************************************
*********************************************************************************/
Set Of Transforms HandleDifferences
  Trigger: external
  Method
    Apply: Single Step

/*----------------------------------------------------------------------
 ----------------------------------------------------------------------*/
Transform findInsertion
Lhs: {{dast diff.insertSpec
    insert  type = [[whatChange wc]] [[any a]] in [[any b]] end
}}
Pre-Apply: {{dast txt.decls
    ULF_DEBUG_MSG_BEFORE("Achou insert" , (&l));
    printf("\nAchou insert");

    // criar um novo SOT
    char aSOTName[100], aux[100];
    getNextSOTName(aSOTName);
    sprintf(aux , "(SetOfTransforms %s 1 0 1 " , aSOTName);
    currentTLBPort->Write(aux);

    RawDASTLocater aWhatToInsert;
    RawDASTLocater aWhereToInsert;
    RawDASTLocater *aLoc;

  GET_VALUE("a",aWhatToInsert);
  GET_VALUE("b",aWhereToInsert);

    ULF_DEBUG_MSG_BEFORE("a" , (&aWhatToInsert));
    ULF_DEBUG_MSG_BEFORE("b" , (&aWhereToInsert));

    // salvar o header da transformacao
    currentTLBPort->Write("(Transform TFM1 ");
    // criar o LHS
    RawDASTLocater aLHS((aWhereToInsert.GetTmpTree().MakeCopy()));
        // remover a marcacao de lugar
    aLoc = FindPatternOnLocate("FindInsertDiffLocate", &aLHS , TheSetOfTransforms ,
1);
    if(aLoc != NULL) {
        ULF_DEBUG_MSG_BEFORE_AFTER("Construcao do LHS ... antes (lhs / aLoc)" ,
(&aLHS) , aLoc);
//      aLoc->GotoRawCode();
        aLoc->GoFather();
        // ver se tem irmao. se nao tiver removo o pai. se tiver peco para o pai remover o
filho.
        if(aLoc->GetCell()->HasBrother() ||  aLoc->GetCell()->HasLBrother()) { // tem irmao
            TreeCell *aChildToRemove = aLoc->GetCell();
            aLoc->GoFather();
            aLoc->GetCell()->DestroyChild(aChildToRemove);
            ULF_DEBUG_MSG_BEFORE("remover o filho da lista" , aLoc);
        } else {
            aLoc->GoFather();
            aLoc->Remove();
        }

    }   else
        printf("\n aLoc e´ null...");
    ULF_DEBUG_MSG_BEFORE("Construcao do LHS ... depois (rhs)" , (&aLHS));
    // salvar o LHS
    {
        currentTLBPort->Write("(Lhs ");
        aLHS.Write(currentTLBPort);
        currentTLBPort->Write(")");
    }

    // criar o RHS
    RawDASTLocater aRHS((aWhereToInsert.GetTmpTree().MakeCopy()));
        // remover a marcacao de lugar
    aLoc = FindPatternOnLocate("FindInsertDiffLocate", &aRHS , TheSetOfTransforms ,
1);
    if(aLoc != NULL) {
        ULF_DEBUG_MSG_BEFORE_AFTER("Construcao do RHS ... antes (rhs / aLoc)"
, (&aRHS) , aLoc);
        aLoc->GoFather();
        // ver se tem irmao.
        if(aLoc->GetCell()->HasBrother() ||  aLoc->GetCell()->HasLBrother()) { // tem irmao
            UlfDebugInfo *aDebugInfo = new UlfDebugInfo("(Rhs ANTES e DEPOIS do
replace)" , 1);
            aDebugInfo->setBefore((&aRHS));
            ULF_DEBUG_MSG_BEFORE_AFTER("Vai copiar (com irmao)" , (aLoc) ,
(&aWhatToInsert));

           aLoc->AddBrother("TMP");
           aLoc->GoBrother();
            aLoc->AttachDASTCopyFrom(aWhatToInsert);
            aLoc->GoLBrother();
            aLoc->Remove();


//          aLoc.AttachDASTCopyFrom(aWhatToInsert);

            aDebugInfo->setAfter((&aRHS));
            aDebugInfo->save();
```

```
                delete aDebugInfo;
            } else {
/*              aLoc->GoFather();
                ULF_DEBUG_MSG_BEFORE_AFTER("Vai copiar (sem irmao)" , aLoc , (&aWhatToInsert));
                aLoc->AttachDASTCopyFrom(aWhatToInsert );*/
              aLoc->AddBrother("TMP");
              aLoc->GoBrother();
                aLoc->AttachDASTCopyFrom(aWhatToInsert);
                aLoc->GoLBrother();
                aLoc->Remove();

            }
        }   else
            printf("\n aLoc e´ null...");
        ULF_DEBUG_MSG_BEFORE("Construcao do RHS ... depois (rhs)" , (&aRHS));
        // salvar o RHS
        {
            currentTLBPort->Write("(Rhs ");
            aRHS.Write(currentTLBPort);
            currentTLBPort->Write(")");
        }

        // salvar o fim da transformacao
        currentTLBPort->Write(")");

        // finalizar o SOT
        currentTLBPort->Write(")");

}}

/*
    remove type=reference
      {{any cen.reference
            <reference name="malfunction" version="1"/>
        }}
    in
      {{any cen.episode
        <episode address="11"><definition><reference name="facility manager" version="1"/>can correct manually{{reference diffaux.locater start}}<reference name="malfunction" version="1"/>{{reference diffaux.locater end}}undetected by the<reference name="Control system" version="1"/></definition></episode>
        }}
    end
*/
Transform findRemove
Lhs: {{dast diff.removeSpec
    remove  type = [[whatChange wc]] [[any a]] in [[any b]] end
}}
Pre-Apply: {{dast txt.decls
    ULF_DEBUG_MSG_BEFORE("Achou remove" , (&l));
    printf("\nAchou remove");

    // criar um novo SOT
    char aSOTName[100], aux[100];
    getNextSOTName(aSOTName);
    sprintf(aux , "(SetOfTransforms %s 1 0 1 " , aSOTName);
    currentTLBPort->Write(aux);

    RawDASTLocater aWhatToRemove;
    RawDASTLocater aWhereToRemove;

   GET_VALUE("a",aWhatToRemove);
   GET_VALUE("b",aWhereToRemove);

    ULF_DEBUG_MSG_BEFORE("a" , (&aWhatToRemove));
    ULF_DEBUG_MSG_BEFORE("b" , (&aWhereToRemove));

    // salvar o header da transformacao
    currentTLBPort->Write("(Transform TFM1 ");


    // ------------------------------------------------
    //  criar o LHS
    // ------------------------------------------------
    RawDASTLocater aLHS((aWhereToRemove.GetTmpTree().MakeCopy()));

    // encontrar o inicio e o fim
    RawDASTLocater *aStartLoc = FindPatternOnLocate("FindStartDiffLocate", &aLHS ,
TheSetOfTransforms , 1);
    // remover os nos de marcacao de inicio e fim
    if(aStartLoc != NULL) {
        aStartLoc->GoFather();

        RawDASTLocater *aBrother;
        aBrother = new RawDASTLocater(aStartLoc);
        if(!aBrother->GoBrother())   { // nao tem irmao ==> tentar no nivel superior
(problema da lista ou filho unico
            aStartLoc->GoFather();
            delete aBrother;
        }
        aStartLoc->Remove();
    }   else
        printf("\n aStartLoc e´ null...");

    ULF_DEBUG_MSG_BEFORE("Tentando achar endloc em " , (&aLHS));
    RawDASTLocater *anEndLoc =  FindPatternOnLocate("FindEndDiffLocate", &aLHS ,
TheSetOfTransforms , 1);
    ULF_DEBUG_MSG_BEFORE("Resultado de Tentando achar endloc " , (anEndLoc));
    if(anEndLoc != NULL) {
        anEndLoc->GoFather();
        RawDASTLocater *aBrother;
        aBrother = new RawDASTLocater(aStartLoc);
        if(!aBrother->GoBrother())   { // nao tem irmao ==> tentar no nivel superior
(problema da lista ou filho unico
            aStartLoc->GoFather();
            delete aBrother;
        }
        anEndLoc->Remove();
    }   else
        printf("\n anEndLoc eh null...");

    // salvar o LHS
    currentTLBPort->Write("(Lhs ");
    aLHS.Write(currentTLBPort);
    currentTLBPort->Write(")");


    // ------------------------------------------------
    //  criar o RHS
    // -----------------------------------------------
    RawDASTLocater aRHS((aWhereToRemove.GetTmpTree().MakeCopy()));


    // encontrar o inicio
    aStartLoc = FindPatternOnLocate("FindStartDiffLocate", &aRHS ,
TheSetOfTransforms , 1);
        ULF_DEBUG_MSG_BEFORE("Tentando remover (1)..." , aStartLoc);
    // remover os nos de marcacao de inicio e fim
    if(aStartLoc != NULL) {
        RawDASTLocater *aBrother;
        aStartLoc->GoFather();

        aBrother = new RawDASTLocater(aStartLoc);
        if(!aBrother->GoBrother()) { // nao tem irmao ==> tentar no nivel superior
(problema da lista ou filho unico
            aStartLoc->GoFather();
            delete aBrother;
        }

        ULF_DEBUG_MSG_BEFORE("Tentando remover (2)..." , aStartLoc);

        // remover aLoc e todos os irmaos ate encontrar o delimitador de fim
        while(aStartLoc != NULL) {
            ULF_DEBUG_MSG_BEFORE("Tentando remover (3)... " , aStartLoc);
            aBrother = new RawDASTLocater(aStartLoc);
            if(!aBrother->GoBrother()) { // nao tem irmao ==> erro, nao encontrou end no
mesmo nivel
                aBrother = NULL;
                reportError("nao tem irmao ==> erro, nao encontrou end no mesmo nivel" ,
&aRHS);
            } else {
                // ver se o atual ja e´ o end
                RawDASTLocater aTemp(aStartLoc);
                ULF_DEBUG_MSG_BEFORE("Vai pesquisar o end em " , (&aTemp));
                aTemp.GoChild();
                if( FindPatternOnLocate("FindEndDiffLocate", &aTemp ,
TheSetOfTransforms , 0) != NULL) {
                    aBrother = NULL;
                    ULF_DEBUG_MSG_BEFORE("encontrou end" , aStartLoc);
                }
            }
            aStartLoc->Remove();
            aStartLoc = aBrother;
        }
    } else
        printf("\n aStartLoc e´ null...");

    // salvar o RHS
    currentTLBPort->Write("(Rhs ");
    aRHS.Write(currentTLBPort);
    currentTLBPort->Write(")");


    // salvar o fim da transformacao
    currentTLBPort->Write(")");

    // finalizar o SOT
    currentTLBPort->Write(")");
    ULF_DEBUG_MSG_BEFORE_AFTER("LHS e RHS de remove" , (&aLHS) ,
(&aRHS));

}}

Transform findContentChange
Lhs: {{dast diff.replaceSpec
    replace  type = [[whatChange wc]] from [[any a]] to [[any b]] end
}}
Pre-Apply: {{dast txt.decls
    ULF_DEBUG_MSG_BEFORE("Achou replace" , (&l));
    printf("\nAchou replace");

    char aSOTName[100] , aux[100];
    getNextSOTName(aSOTName);
    sprintf(aux , "(SetOfTransforms %s 1 0 1 " , aSOTName);
    currentTLBPort->Write(aux);

    // salvar o header da transformacao
    currentTLBPort->Write("(Transform TFM1 ");

    // salvar o LHS
    {
        currentTLBPort->Write("(Lhs ");
        RawDASTLocater aLoc;
      GET_VALUE("a",aLoc);
        aLoc.Write(currentTLBPort);
        currentTLBPort->Write(")");
    }

    // salvar o RHS
    {
        currentTLBPort->Write("(Rhs ");
        RawDASTLocater aLoc;
      GET_VALUE("b",aLoc);
        aLoc.Write(currentTLBPort);
        currentTLBPort->Write(")");
    }

    // salvar o fim da transformacao
    currentTLBPort->Write(")");

    // finalizar o SOT
    currentTLBPort->Write(")");

}}

Template FindInsertDiffLocate
Lhs: {{dast diffaux.locater
    here
 }}
Template FindStartDiffLocate
Lhs: {{dast diffaux.locater
    start
 }}
Template FindEndDiffLocate
Lhs: {{dast diffaux.locater
    end
 }}
```

C.3.
Transformador Refine

```
Global-Declaration:
{{dast txt.decls
   #include <include/ulfdebug.h>

#define __IS_TO_ULF_DEBUG__ 1

#ifdef __IS_TO_ULF_DEBUG__
#define ULF_DEBUG_MSG(X) {UlfDebugInfo *aDebugInfo = new UlfDebugInfo(X , 0);
aDebugInfo->save(); delete aDebugInfo; }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->save(); delete aDebugInfo;
}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->setAfter(Z); aDebugInfo-
>save(); delete aDebugInfo; }
#else
#define ULF_DEBUG_MSG(X) { }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) { }
#endif



   void removeQuotes( char *source , char *target) {
      int count = 0;
      for(int i=0; i < strlen(source) ; i++) {
         if( (source[i] != '\"') && (source[i] != '\'') ) {
            target[count] = source[i];
           count++;
         }
      }
      target[count] = '\0';
   }

}}

Global-Initialization: {{dast txt.decls


}}

Global-End: {{dast txt.decls


}}

/*******************************************************************************
*******************************************************************************/
Set Of Transforms MainSet
  Method
    Search: Top-Down
    Apply: Single Step
  Init: {{dast txt.decls
     printf("Inicializacao do SOF principal");
  }}
  End: {{dast txt.decls
//   RENAME("versao1.xml", "target.xml");
     printf("Finalizacao do SOF principal");
  }}

Transform FindRefineSTatement
Lhs: {{dast diff.refineStatement
     evolve [[String aSource]] to [[String aTarget]] using [[String aTFM]] ;
}}
Pre-Apply: {{dast txt.decls
     printf("Inicio");

     char aSourceFileName[100];
     char aTargetFileName[100];
     char aTFMFileName[100];
     char aux[100];
     char aSotName[100];

     COPY_LEAF_VALUE_TO(aux, "aSource");
     removeQuotes(aux , aSourceFileName);
     COPY_LEAF_VALUE_TO(aux, "aTarget");
     removeQuotes(aux ,aTargetFileName);
     COPY_LEAF_VALUE_TO(aux, "aTFM");
     removeQuotes(aux , aTFMFileName);

     // leitura do source
     READ_MODULE(aSourceFileName);
     RawDASTLocater *aSource = I.GetSystemDAST()->GetLocater(aSourceFileName);
   aSource->GotoRawCode();

/*   {
         RawDASTLocater *anAuxSource = new RawDASTLocater(aSource);
         TRANSFORM_WS(anAuxSource, "MOSTRATEXTO");
     }
*/
     // carregar a TLB (Transformacao) e aplicar cada aNewSot lido => PARA MANTER A
ORDEM
     Port aTFMPort(aTFMFileName , "r");

     SetOfTransforms* aNewSot = new SetOfTransforms();
     Port aPort("refine.log" , "w");

     while( aNewSot->Read(&aTFMPort) ) {
         TheSetOfTransforms->GetOwner()->RegisterSetOfTransforms(aNewSot);
       strcpy(aSotName , aNewSot->GetName()->value);
         RawDASTLocater *anAuxSource = new RawDASTLocater(aSource);  // necessario
póis perde a posicao atual dentro da tfm ...

         int isToPrintDebugInfo = 1;

         if(isToPrintDebugInfo) {
             aPort.Write("\n ************************************************");
             aPort.Write("\n TENTANDO APLICAR => ");
             aPort.Write(aSotName);
             aPort.Write("\n -----------------------------------------------\n");
             Transform *aTfm = aNewSot->GetTfmByName("TFM1");
             aPort.Write("\n           LHS        \n");
             aPort.Write("\n -----------------------------------------------\n");
             DASTLocater aTemp(aTfm->Lhs);
```

```
            TheShell()->Pprint(&aTemp , &aPort);

            char auxName[50];
            sprintf(auxName , "%s.dast" , aSotName);
            Port auxPort(auxName , "w");
            aTfm->Write(&auxPort);

            aPort.Write("\n ------------------------------------------------\n");
            aPort.Write("\n          RHS        \n");
            aPort.Write("\n ------------------------------------------------\n");
            DASTLocater aTemp1(aTfm->getRhs());
            TheShell()->Pprint(&aTemp1 , &aPort);
            aPort.Write("\n ------------------------------------------------\n");
            aPort.Flush();
        }

        TRANSFORM_WS(anAuxSource, aSotName);

        if(isToPrintDebugInfo) {
            aPort.Write("\n ------------------------------------------------\n");
            aPort.Write("\n RESULTADO DA APLICACAO DE ");
            aPort.Write("\n ------------------------------------------------\n");
            TheShell()->Pprint((DASTLocater*)aSource , &aPort);
            aPort.Write("\n ************************************************");
            aPort.Flush();

            char auxName[50];
            sprintf(auxName , "%s_result.dast" , aSotName);
            Port auxPort(auxName , "w");
            aSource->Write(&auxPort);

        }

    aNewSot = new SetOfTransforms();
  }

/*
    RawDASTLocater *anAuxSource = new RawDASTLocater(aSource);   // necessario
póis perde a posicao atual dentro da tfm ...
    TRANSFORM_WS(anAuxSource, "TESTE1");
*/
    // salvar o arquivo de destino
    Port *aTargetPort = new Port(aTargetFileName , "w");
  aSource->GotoRawCode();
    TheShell()->Pprint((DASTLocater*)aSource , aTargetPort);
    delete aTargetPort;

    SKIP_APPLY();
}}

/*********************************************************************************
*********************************************************************************/
Set Of Transforms TESTE1
  Trigger: external
  Method
    Apply: Single Step

Transform TESTE1Transform
Lhs: {{dast cen.definition
```

```
        <definition>inform     <reference   name="facility   manager"   version="1"/>     of
<reference name="malfunction" version="1"/></definition>
}}
Pre-Apply: {{dast txt.decls
    printf("Achou TESTE1 ....");
}}
Rhs: {{dast cen.definition
    <definition>informs <reference name="facility manager" version="1"/>  of  <reference
name="malfunction" version="1"/></definition>
}}

/*******************************************************************************
*******************************************************************************/
Set Of Transforms MOSTRATEXTO
  Trigger: external
  Method
    Apply: Single Step

Transform findText
Lhs: {{dast cen.text
    [[Text a]]
}}
Pre-Apply: {{dast txt.decls
    char anOld[300] , aNew[300];
    COPY_LEAF_VALUE_TO(anOld, "a");
    printf("\n Texto lido =>%s<=" , anOld);
    SKIP_APPLY();
}}
```

C.4. Transformador de Reconhecimento de Planos

```
Global-Declaration:
{{dast txt.decls
    #include <include/ulfdebug.h>

#define __IS_TO_ULF_DEBUG__ 1

#define MAX_STR_SIZE 1000

#ifdef __IS_TO_ULF_DEBUG__
#define ULF_DEBUG_MSG(X) {UlfDebugInfo *aDebugInfo = new UlfDebugInfo(X , 0);
aDebugInfo->save(); delete aDebugInfo; }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->save(); delete aDebugInfo;
}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) {UlfDebugInfo *aDebugInfo = new
UlfDebugInfo(X , 1); aDebugInfo->setBefore(Y); aDebugInfo->setAfter(Z); aDebugInfo-
>save(); delete aDebugInfo; }
#else
#define ULF_DEBUG_MSG(X) { }
#define ULF_DEBUG_MSG_BEFORE(X,Y) {}
#define ULF_DEBUG_MSG_BEFORE_AFTER(X,Y,Z) { }
#endif

    char aux[MAX_STR_SIZE] , aux1[MAX_STR_SIZE], aux2[MAX_STR_SIZE];
    char currentObservedEventName[MAX_STR_SIZE];
    int newEventInferredThatNeedToBeProcessed;

    RawDASTLocater *definitionsLoc;  /* armazena a definicao dos planos */

    RawDASTLocater *rationaleForCurrentInferredEvent; /*          anyEvent
(observedEvent ou inferredEvent) corrente

            que e´ utilizado como justificativa para o novo evento inferido */

    static RawDASTLocater *WSObservedEvents;          /* armazena os novos eventos
inferidos (gerados) */
    static RawDASTLocater *WSEndEvents;       /* armazena os end eventos inferidos
para a observacao atual */
    static RawDASTLocater *WSFinal;    /* armazena os end eventos inferidos para todas
as observacoes */
    static RawDASTLocater *WSAuxFinal;          /* armazena etmporariamente os planos
reconhecidos e que devem ser duplicados quando da utlizacao de coringas */


    static RawDASTLocater *WSAux;      /* so utilizar para colocar uma arvore que sera
utilizada na aplicacao de um template, ou seja, utilizada
                                        imediatamente depois. Sempre limpar
antes de colocar alguma coisa  */

    static RawDASTLocater *WSCurrentObservedEvents;          /* COntem a copia do
WSObservedEvents anterios com os eventos que estao sendo tratados */

     /* variaveis utilizadas para verificar o resultado da utilizacao dos Sets para verificacao
das declaracoes */
     char roleNameForCurrentObservedEventNameAsPartOf[100];
```

```
  /* variaveis que serao utilizadas para inserir um novo evento inferido */
 char InferredEventNameToBeInserted[MAX_STR_SIZE];

void reportError(char *msg , RawDASTLocater *aFirst , RawDASTLocater *aSec) {
  printf("\nDebugMsg -> %s" , msg);
      UlfDebugInfo aDebugInfo(msg , 1);
      aDebugInfo.setBefore(aFirst);
      aDebugInfo.setAfter(aSec);
      aDebugInfo.save();
}

void reportError(char *msg , RawDASTLocater *aFirst) {
  printf("\nDebugMsg -> %s" , msg);
      UlfDebugInfo aDebugInfo(msg , 1);
      aDebugInfo.setBefore(aFirst);
      aDebugInfo.save();
}

void reportError(char *msg ) {
  printf("\nDebugMsg -> %s" , msg);
      UlfDebugInfo aDebugInfo(msg , 1);
      aDebugInfo.save();
}

 void removeSpace( char *source , char *target) {
      int count = 0;
      for(int i=0; i < strlen(source) ; i++) {
          if( (source[i] != ' ') && (source[i] != '\n') ) {
              target[count] = source[i];
            count++;
          }
      }
      target[count] = '\0';
 }

 void removeQuotes( char *source , char *target) {
      int count = 0;
      for(int i=0; i < strlen(source) ; i++) {
          if( (source[i] != '\"') && (source[i] != '\'') ) {
              target[count] = source[i];
            count++;
          }
      }
      target[count] = '\0';
 }

 float removeQuotesAndGetFloatValue(char *source) {
      int count = 0;
      char target[MAX_STR_SIZE];
      for(int i=0; i < strlen(source) ; i++) {
          if( (source[i] != '\"') && (source[i] != '\'') ) {
              target[count] = source[i];
            count++;
          }
      }
      target[count] = '\0';
      return atof(target);
 }
```

```
    int removeQuotesAndGetIntValue(char *source) {
        int count = 0;
        char target[MAX_STR_SIZE];
        for(int i=0; i < strlen(source) ; i++) {
            if( (source[i] != '\"') && (source[i] != '\'') ) {
                target[count] = source[i];
              count++;
            }
        }
        target[count] = '\0';
        return atoi(target);
    }

    // responde a ordem de aChar em aString (indexOf...)
    int findCharOrderIn(char aChar , char *aStr) {
        for(int i=0; i < strlen(aStr) ; i++) {
            if( aStr[i] == aChar ) {
                return i;
            }
        }
        return -1;
    }
    void dumpRationaleToString(char *aStr) {
        char aTemp[MAX_STR_SIZE];
        Port *anOut = new Port("lixo" , "w");
        TheShell()->Pprint((DASTLocater*)rationaleForCurrentInferredEvent , anOut);
        anOut->DumpToString(aTemp);
        removeSpace(aTemp , aStr);
        delete anOut;
    }

int preProcessNewEventToBeLateProcessed(char *anEventName , char *aRationaleStr)
{
    char aStr[MAX_STR_SIZE];

    sprintf(aStr, "inferredEvent(%s,%s)", anEventName , aRationaleStr);
    if(KBSolve(aStr)) {  /* ja existe o evento na KB */
        return(0);
    } else {
        __KBAssert(aStr);
        /* ver se eh um endEvent */
        sprintf(aStr, "endEvent(\"%s\")", anEventName);
        if(KBSolve(aStr)) {
            sprintf(aStr, "recognizedPlan(%s,%s)", anEventName , aRationaleStr);
            __KBAssert(aStr);
            return(1);  /* end event */
        }
        newEventInferredThatNeedToBeProcessed = 1;
        return (2);
    }
}

int isSameCurrentRationale(RawDASTLocater *aLoc , RawDASTLocater *anotherLoc) {
    /* Ver se e´ a mesma regra */
    char anAux[MAX_STR_SIZE];

    RawDASTLocater a(aLoc) , b(anotherLoc);
REVISAO_CODIGO_SUSPEITO(1,(&a));
REVISAO_CODIGO_SUSPEITO(2,(&b));
    /* o primeiro filho corresponde ao tipo da regra:
```

```
    rationale1 => observacao
    rationale2 => heranca
    rationale3 => parte de
    */
    char ruleType;
    if( a.GoChild() && b.GoChild()) {
        if( strcmp( a.GetCurrent()->value , b.GetCurrent()->value) != 0)
            return 0;  /* regras diferentes */
        if( strcmp( "rationale1" , a.GetCurrent()->value ) == 0) {      /* tratar observacao */
            return 0; /* observacao e´ sempre diferente */
        }
        if( strcmp( "rationale2" , a.GetCurrent()->value ) == 0) {      /* tratar heranca */
            /* localizar o eventName de cada um e ver se sao o mesmo */
            a.GoChild(); b.GoChild();
            a.GoBrother();
            b.GoBrother();
            while(a.GoChild() && b.GoChild()) {
                if(strcmp( "eventName1" , a.GetCurrent()->value ) == 0) {
                    a.GoChild();
                    b.GoChild();
                    if( strcmp( a.GetCurrent()->value , b.GetCurrent()->value) == 0)
                        return (1);
                    else
                        return (0);
                }
            }
            reportError("Erro   na   comparacao:   situacao   nao   prevista   (nao   achou
eventName)");
            return 0;
        }

        if( strcmp( "rationale3" , a.GetCurrent()->value ) == 0) {      /* tratar parte de */
            /* localizar o ruleName */
            while(a.GoChild() && b.GoChild()) {
                if(strcmp( "ruleName1" , a.GetCurrent()->value ) == 0) {
                    /* ver se o filho de ruleName (o nome da regra) eh o mesmo */
                    RawDASTLocater a1(a) , b1(b);
                    a1.GoChild();
                    b1.GoChild();
                    if( strcmp( a1.GetCurrent()->value , b1.GetCurrent()->value) != 0)
                        return (0);
                    /* ver se o irmao que e´ o evento e´ o mesmo */
                    a.GoBrother();
                    b.GoBrother();
                    while(a.GoChild() && b.GoChild()) {
                        if(strcmp( "eventName1" , a.GetCurrent()->value ) == 0) {
                            a.GoChild();
                            b.GoChild();
                            if( strcmp( a.GetCurrent()->value , b.GetCurrent()->value) == 0)
                                return (1);
                            else
                                return (0);
                        }
                    }
                    return 0;
                }
            }
            return 0;
        }
    } else {
        reportError("ERRO: Situacao nao prevista na comparaco");
    }
    return(1);
}

float getTotalDecompositionRuleValueForEvent(char *anEventName) {
    char anAux[MAX_STR_SIZE];
    int aResultCount , aCount;
    float aValue = 0;

//                                                    rule(              "nomeEventoTodo"
,"nomeEventoParte","regra","valorDoPeso","nrTotalDeParametros").
  sprintf(anAux , "rule(\"%s\",*epn,*rn,*aValue,*nrParam)", anEventName);
    if (KBSolve(anAux)) {
        aResultCount = KBRetrieveLength();
        for(aCount = 1  ; aCount <= aResultCount ; aCount++)  {
        strcpy(anAux , KBRetrieve( "*aValue" , aCount));
            aValue += removeQuotesAndGetFloatValue(anAux);
        }
    }
    return aValue;
}

float getValueForRationaleList(char *anEventName , RawDASTLocater *anOriginalRoot )
{
    char anAux[MAX_STR_SIZE];
    RawDASTLocater aRoot(anOriginalRoot);
    float aValue = 0;

    if( strcmp( "observedEvent1" , aRoot.GetCurrent()->value ) == 0) {
        char aSpecializationEvent[100];
        aRoot.GoChild();
        aRoot.GoChild();
        strcpy(aSpecializationEvent , aRoot.GetCurrent()->value );
        aRoot.GoFather();
        aRoot.GoBrother();// estou em observation
        RawDASTLocater aNewRat(aRoot);
        // obter o valor para o evento que e´ a especializacao
        aValue = getValueForRationaleList( aSpecializationEvent , &aNewRat );
        return aValue;
    }

    if( strcmp( "inferredEvent1" , aRoot.GetCurrent()->value ) == 0) {
        char aSpecializationEvent[100];
        aRoot.GoChild();
        aRoot.GoChild();
        strcpy(aSpecializationEvent , aRoot.GetCurrent()->value );
        aRoot.GoFather();
        aRoot.GoBrother();
        aRoot.GoBrother();
        aRoot.GoBrother();
        RawDASTLocater aNewRat(aRoot);
        // obter o valor para o evento que e´ a especializacao
        aValue = getValueForRationaleList( aSpecializationEvent , &aNewRat );
        return aValue;
    }

    if( strcmp( "observation1" , aRoot.GetCurrent()->value ) == 0) {
        aRoot.GoChild();
        aRoot.GoBrother();
        if( strcmp( "value2" , aRoot.GetCurrent()->value ) == 0)     //     nao     tem     valor
declarado na observacao.
            aValue = 1; // default
        else
            if( strcmp( "value1" , aRoot.GetCurrent()->value ) == 0) {
                aRoot.GoChild();
                strcpy(anAux , aRoot.GetCurrent()->value );
                aValue = removeQuotesAndGetFloatValue(anAux);
            } else {
                reportError("ERRO : SITUACAO NAO PREVISTA (calculo da observacao)" ,
anOriginalRoot);
                aValue = 0;
            }
        return aValue;
    }

    if( strcmp( "rationale2" , aRoot.GetCurrent()->value ) == 0) {   // tratar heranca
        char aSpecializationEvent[100];
        aRoot.GoChild();
        aRoot.GoBrother();
      // ver se eh observedEvent1 ou
        aRoot.GoChild();
        aRoot.GoChild();
        strcpy(aSpecializationEvent , aRoot.GetCurrent()->value );
        aRoot.GoFather();
        aRoot.GoFather();  // estou em observedEvent1 ou inferredEvent1
        RawDASTLocater aNewRat(aRoot);
        // obter o valor para o evento que e´ a especializacao
        aValue = getValueForRationaleList( aSpecializationEvent , &aNewRat );
        return aValue;
    }

    if( strcmp( "rationale3" , aRoot.GetCurrent()->value ) == 0) {   // tratar parte de
        float aWeight;
        float valorTotalPesos;
        char anEventPartName[MAX_STR_SIZE];
        char aRuleName[100]; // nome da regra

        aRoot.GoChild();
        aRoot.GoChild();
        aRoot.GoChild();
        aRoot.GoChild();
        strcpy(aRuleName , aRoot.GetCurrent()->value );
        aRoot.GoFather();
        aRoot.GoBrother();  // inferredEvent1 ou observedEvent1
        aRoot.GoChild();
        aRoot.GoChild();
        strcpy(anEventPartName , aRoot.GetCurrent()->value );
        aRoot.GoFather();
        aRoot.GoFather();  // retornei para inferredEvent ou observedEvent1
        RawDASTLocater aNewRat(aRoot);

        // obter o valor do peso na KB
//                                          rule(             "nomeEventoTodo"
,"nomeEventoParte","regra","valorDoPeso","nrTotalDeParametros").
        char aQuery[200];
       sprintf(aQuery    ,    "rule(\"%s\",\"%s\",\"%s\",*aValue,*nrParam)",    anEventName
,anEventPartName , aRuleName );
        if (KBSolve(aQuery)) {
        strcpy(anAux , KBRetrieve( "*aValue" , 1));
```

```
            aWeight = removeQuotesAndGetFloatValue(anAux);
         } else {
            reportError("ERRO: Nao encontrou o valor da regra na KB" );
            reportError(aQuery);
            aWeight = 1;
         }

        // obter o valor para o evento que e´ parte de
        float aTempValue = getValueForRationaleList( anEventPartName , &aNewRat );
        aValue = aWeight * aTempValue;

        // obter o valor total dos pesos
        valorTotalPesos = getTotalDecompositionRuleValueForEvent(anEventName);
        aValue = aValue / valorTotalPesos;
        return (aValue);
    }

    float aTempValue;
    if(strcmp( "_lst1_rationaleStatement" , aRoot.GetCurrent()->value ) == 0) {

        float aMaxSpecializationValue = 0;
        if(aRoot.GoChild() == 0 ) {
            reportError("ERRO: nao achou rat Stat" );
            return 0;
        }
        do { // estou em rationaleStatement1
            if( aRoot.GoChild() == 0 ) {
                reportError("ERRO: nao achou rat Stat" );
                return 0;
            }

            // float aMaxSpecializationValue = 0
            aTempValue = getValueForRationaleList( anEventName , &aRoot );

            // ver se o valor corresponde a uma especializacao. Se for soh deve ser
computada a de maior valor.
            if(strcmp( "rationale2" , aRoot.GetCurrent()->value ) == 0) { // especializacao
                if(aTempValue > aMaxSpecializationValue)
                    aMaxSpecializationValue = aTempValue;
                aTempValue = 0;
            } else {
                if(strcmp( "rationale3" , aRoot.GetCurrent()->value ) == 0) {
                        // nothing ...
                } else {
                    reportError("ERRO: situacao nao prevista no computo da lista de
rationales" , (&aRoot) );
                }
            }
            aValue = aValue + aTempValue;

            aRoot.GoFather();
        } while(aRoot.GoBrother());

        // somar o valor da maior especializacao
        aValue = aMaxSpecializationValue + aValue;

        return aValue;
    }

    reportError("NAO ENTROU EM NADA");
    return 0;
}

// responde o indice de aPatternList que contiver aPattern / -1 se nao tiver
int getIndexOf(char *aPattern , char **aPatternList , int aSize) {
    int i;
    for(i=0;i<aSize;i++) {
        if( strcmp( aPatternList[i] , aPattern) == 0)
            return i;
    }
    return -1;
}

//  Verifica se nao vai existir parametros duplicados quando do uso do coringa
// => nao aceitar que ocorra (A,B,A) p. ex.
int canMergeCoringa(RawDASTLocater *anOld , RawDASTLocater *aNew) {
    char anUsedParNames[10][15];
    int anIndex , aSize;
    char anAux;

    // percorrer a lista antiga e atualizar anUsedParNames
    RawDASTLocater anOldParList(*anOld);
    anOldParList.GoChild();
    anOldParList.GoChild();
    aSize = 0;

  do {
        if(strcmp(anOldParList.GetCurrent()->value , "argValue1") == 0) {
            anOldParList.GoChild();
            strcpy(anUsedParNames[aSize] , anOldParList.GetCurrent()->value);
            anOldParList.GoBrother();
            anOldParList.GoChild();
            strcat(anUsedParNames[aSize] , anOldParList.GetCurrent()->value);
            anOldParList.GoFather();
            anOldParList.GoFather();
         } else
                strcpy(anUsedParNames[aSize] , "*");
        aSize++;

    } while( anOldParList.GoBrother() );

    // percorrer a nova lista. Para cada valor a ser inserido, ver se ele esta na mesma
posicao de anUsedParNames.
    // se nao estiver, retornar 0;
    RawDASTLocater aNewParList(*aNew);
    aNewParList.GoChild();
    aNewParList.GoChild();
    anIndex = 0;
  do {
        if(strcmp(aNewParList.GetCurrent()->value , "argValue1") == 0) {
            char aNewAux[20];
            aNewParList.GoChild();
            strcpy(aNewAux , aNewParList.GetCurrent()->value);
            aNewParList.GoBrother();
            aNewParList.GoChild();
            strcat(aNewAux , aNewParList.GetCurrent()->value);
            aNewParList.GoFather();
            aNewParList.GoFather();
            // ver se anUsedParNames tem aNewParList. Se tiver em posicao diferente de
anIndex => retornar 0
```

```
            int i;
            for(i=0;i<aSize;i++) {
                if( strcmp( anUsedParNames[i] , aNewAux) == 0) {
                    if(i != anIndex)
                        return 0;
                    break;
                }
            }
        }
        anIndex++;
    } while( aNewParList.GoBrother() );
    return 1;
}

// verifica as duas listas de parametros. Retorna:
// 0 => totalmente diferentes, ou seja nao casam nem com o coringa.
// 1 => casam com o coringa
// 2 => casam sem precisar do coringa
int compareParList(RawDASTLocater *anOld , RawDASTLocater *aNew) {
    RawDASTLocater anOldParList(*anOld) , aNewParList(*aNew);

REVISAO_CODIGO_SUSPEITO(4,(&anOldParList));
REVISAO_CODIGO_SUSPEITO(5,(&aNewParList));
    anOldParList.GoChild(); aNewParList.GoChild();
    anOldParList.GoChild(); aNewParList.GoChild();
    int aResult = 2;
    char anAux[100];
   do {
        if(strcmp(anOldParList.GetCurrent()->value , "argValue1") == 0) {
            // OLD eh um valor
            if(strcmp(aNewParList.GetCurrent()->value , "argValue1") == 0) {
                // OLD eh um valor e NEW eh um valor
                // ver se sao iguais. 2 passos => comparar o nome e comparar a versao
                anOldParList.GoChild(); aNewParList.GoChild();
                if(strcmp(anOldParList.GetCurrent()->value    ,    aNewParList.GetCurrent()-
>value) == 0) { // comparacao do nome
                    // nomes iguais, ver a versao
                    anOldParList.GoBrother(); aNewParList.GoBrother();
                    if(  (strcmp(anOldParList.GetCurrent()->value  ,  "version1")  ==  0)  &&
(strcmp(aNewParList.GetCurrent()->value , "version1") == 0) ) {
                         // os dois tem um numero de versao: comparar se sao iguais
                        anOldParList.GoChild(); aNewParList.GoChild();
                        if(strcmp(anOldParList.GetCurrent()->value                           ,
aNewParList.GetCurrent()->value) == 0) { // comparacao da versao
                            // mesma versao, continua como esta
                        } else {
                            // versoes diferentes: sao diferentes
                            return 0; // valores diferentes
                        }
                        anOldParList.GoFather(); aNewParList.GoFather();
                    } else {
                        if(  (strcmp(anOldParList.GetCurrent()->value  ,  "version2")  ==  0)  &&
(strcmp(aNewParList.GetCurrent()->value , "version2") == 0) ) {
                            // os dois nao tem numero de versao: sao iguais , continua como
esta

                        } else { // um tem o nr de versao e o outro nao: sao diferentes
                            return 0; // valores diferentes
                        }
                    }
                    anOldParList.GoFather(); aNewParList.GoFather();
                    // valores exatamente iguais, continua como esta
                } else {
                    return 0; // valores diferentes
                }
            } else {
                // OLD eh um valor e NEW eh um coringa
                aResult = 1;
            }
        } else {
            // OLD eh um coringa
            if(strcmp(aNewParList.GetCurrent()->value , "argValue1") == 0) {
                // OLD eh um coringa e NEW eh um valor
                aResult = 1;
            } else {
                // OLD eh um coringa e NEW eh um coringa
            }

        }

    } while( anOldParList.GoBrother() && aNewParList.GoBrother() );

    if( (aResult == 1) && (canMergeCoringa(anOld , aNew) == 0) ) {
        return 0; // valores diferentes
    }
    return aResult;
}


void mergeParList(RawDASTLocater *anOld , RawDASTLocater *aNew) {
    RawDASTLocater anOldParList(*anOld) , aNewParList(*aNew);

REVISAO_CODIGO_SUSPEITO(6,(anOld));
REVISAO_CODIGO_SUSPEITO(7,(aNew));

    anOldParList.GoChild(); aNewParList.GoChild();
    anOldParList.GoChild(); aNewParList.GoChild();
   do {
        if(strcmp(aNewParList.GetCurrent()->value , "argValue1") == 0) {
            // NEW eh um valor, ou seja, e´necessario realizar o merge em OLD
            if(strcmp(anOldParList.GetCurrent()->value , "argValue1") == 0) {
                // OLD tambem eh um valor, verificar se eh o mesmo. Se nao for => ERRO
                anOldParList.GoChild(); aNewParList.GoChild();
                if(strcmp(anOldParList.GetCurrent()->value    ,    aNewParList.GetCurrent()-
>value) != 0) {
                    // ERRO : valores diferentes
                    reportError("ERRO: tentando realizar o MERGE de parametros que sao
incompativeis");
                    return;
                }
                anOldParList.GoFather(); aNewParList.GoFather();
            } else {
                // OLD eh um coringa, trocar pelo valor de new
                anOldParList.AttachDASTCopyFrom(aNewParList);

            }
        } else {
            // NEW eh um coringa, nao eh necessario fazer nada
        }
    } while( anOldParList.GoBrother() && aNewParList.GoBrother() );
```

```
}

int getParameter(RawDASTLocater *aLoc , char *aParName , char *aParValue) {
    char anAux[MAX_STR_SIZE];

REVISAO_CODIGO_SUSPEITO(8,(aLoc));
    if(strcmp(aLoc->GetCurrent()->value , "parameters2") == 0) {  // nao existe declaracao
de parametros
        return 0;
     }

    if(strcmp(aLoc->GetCurrent()->value , "parameters1") != 0) {
        reportError("ERRO : SITUACAO NAO PREVISTA (parameters1)" , aLoc);
        return 0;
     }

    aLoc->GoChild();
    aLoc->GoBrother();
    aLoc->GoChild();
    // estou no primeiro paremeterPair
    do {
        // ver se e´o parametro desejado
        RawDASTLocater aTemp(aLoc);
        aTemp.GoChild();
        if(strcmp(aTemp.GetCurrent()->value , aParName) == 0) {
            aTemp.GoBrother();
            char anAux[MAX_STR_SIZE];
            strcpy(anAux , aTemp.GetCurrent()->value);
            removeQuotes( anAux , aParValue);
            return 1;
        }
    } while(aLoc->GoBrother());
    return 0;
}

}}


Global-Initialization: {{dast txt.decls


}}
Global-End: {{dast txt.decls
        KBWrite("teste.kb");

    char anAux[100];
    strcpy(anAux , GET_FIRST_MODULE_NAME() );
    CHANGE_EXTENSION(anAux, "_rec.plan");
    RENAME("WSFinal", anAux);

    Port aPort(anAux , "w");
    PRINT_WS("WSFinal" , (&aPort));


    DESTROY_MODULE(GET_FIRST_MODULE_NAME());
    DESTROY_MODULE("WSObservedEvents");
    DESTROY_MODULE("WSEndEvents");
    DESTROY_MODULE("WSAux");
    DESTROY_MODULE("WSAuxFinal");
    DESTROY_MODULE("WSCurrentObservedEvents");
    DESTROY_MODULE("rationaleForCurrentInferredEvent");
    if(definitionsLoc != NULL)
        delete (definitionsLoc);

}}
/*******************************************************************************
*******************************************************************************/
Set Of Transforms MainSet
  Method
    Search: Top-Down
    Apply: Single Step
  Init: {{dast txt.decls
       KBClear(); /* tem que ser aqui pois armazena tb os endEvents */
  }}
  End: {{dast txt.decls
  }}

/*--------------------------------------------------------------------
 ---------------------------------------------------------------------*/
  Transform MainTransform
      Lhs: {{dast plan.program
             loadLibrary [[String aLibName]] .
           [[instanceProgram ip]]
      }}
      Pre-Apply:
      {{dast txt.decls
             /* Problema no Draco, solucionado com quebra-galho:
                 o READ_MODULE nao pode vir depois do CREATE_MODULE. */

         char aLibFileName[100];
         COPY_LEAF_VALUE_TO(aux, "aLibName");
         removeQuotes(aux , aLibFileName);
         READ_MODULE(aLibFileName);

         CREATE_MODULE("WSObservedEvents");
         CREATE_OBJECT_IN_MODULE("plan",                    "WSObservedEvents",
"WSObservedEvents", 1);
         WSObservedEvents     =     I.GetSystemDAST()->GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSObservedEvents"));
         WSObservedEvents->GoChild();
         WSObservedEvents->GoBrother();
         WSObservedEvents->GotoRawCode();

         CREATE_MODULE("WSEndEvents");
         CREATE_OBJECT_IN_MODULE("plan", "WSEndEvents", "WSEndEvents", 1);
         WSEndEvents         =       I.GetSystemDAST()->GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSEndEvents"));
         WSEndEvents->GoChild();
         WSEndEvents->GoBrother();
         WSEndEvents->GotoRawCode();

         CREATE_MODULE("WSCurrentObservedEvents");
         CREATE_OBJECT_IN_MODULE("plan",               "WSCurrentObservedEvents",
"WSCurrentObservedEvents", 1);
         WSCurrentObservedEvents                =                   I.GetSystemDAST()-
>GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSCurrentObservedEvents"));
         WSCurrentObservedEvents->GoChild();
         WSCurrentObservedEvents->GoBrother();
         WSCurrentObservedEvents->GotoRawCode();
         CREATE_MODULE("rationaleForCurrentInferredEvent");
         CREATE_OBJECT_IN_MODULE("plan",          "rationaleForCurrentInferredEvent",
"rationaleForCurrentInferredEvent", 1);
         rationaleForCurrentInferredEvent                   =                  I.GetSystemDAST()-
>GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("rationaleForCurrentInferredEvent"));
         rationaleForCurrentInferredEvent->GoChild();
         rationaleForCurrentInferredEvent->GoBrother();
         rationaleForCurrentInferredEvent->GotoRawCode();

         CREATE_MODULE("WSAux");
         CREATE_OBJECT_IN_MODULE("plan", "WSAux", "WSAux", 1);
         WSAux             =             I.GetSystemDAST()->GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSAux"));
         WSAux->GoChild();
         WSAux->GoBrother();
         WSAux->GotoRawCode();

         CREATE_MODULE("WSFinal");
         CREATE_OBJECT_IN_MODULE("plan", "WSFinal", "WSFinal", 1);
         WSFinal            =            I.GetSystemDAST()->GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSFinal"));
         WSFinal->GoChild();
         WSFinal->GoBrother();
         WSFinal->GotoRawCode();
      TEMPLATE("copyEmptyObservedEvent");
           PLACE_AT("WSFinal");
         END_TEMPLATE;

         CREATE_MODULE("WSAuxFinal");
         CREATE_OBJECT_IN_MODULE("plan", "WSAuxFinal", "WSAuxFinal", 1);
         WSAuxFinal          =          I.GetSystemDAST()->GetLocater(I.GetSystemDAST()-
>GetCompleteObjectName("WSAuxFinal"));
         WSAuxFinal->GoChild();
         WSAuxFinal->GoBrother();
         WSAuxFinal->GotoRawCode();
      TEMPLATE("copyEmptyObservedEvent");
           PLACE_AT("WSAuxFinal");
         END_TEMPLATE;

         RawDASTLocater *aPLanLib = I.GetSystemDAST()->GetLocater(aLibFileName);
       aPLanLib->GotoRawCode();

         TRANSFORM_WS(aPLanLib, "HandlePlanLibrary");
         APPLY("HandleObservedFactsSet", "ip");
         TRANSFORM_WS(WSFinal, "CalculateEndEventValue");
      }}

/*******************************************************************************
*******************************************************************************/
Set Of Transforms HandlePlanLibrary
  Trigger: external
  Method
    Apply: Single Step
  End: {{dast txt.decls
       KBWrite("teste.kb");
  }}
/*-----------------------------------------------------------------------
 -----------------------------------------------------------------------*/
 Transform LocateEvent
   Lhs: {{dast plan.eventDeclaration
       Event [[Name en]] [[parListdecl al]]  [[eventAttribute ea]] [[declBody db]]
   }}
    Post-Match:
    {{dast txt.decls
           COPY_LEAF_VALUE_TO(aux, "en");

           // ver se eh endEvent
           if(FindPatternOnLVar("endEventPattern" , "ea" , env, TheSetOfTransforms)) {
              sprintf(aux1, "endEvent(\"%s\")" , aux );
              KBAssert(aux1);
           }

           // achar o nr de parametros
           int aParNumber = 0;
           RawDASTLocater aLoc;
           GET_VALUE("al", aLoc );
REVISAO_CODIGO_SUSPEITO(9,(&aLoc));
           aLoc.GoChild();
           aLoc.GoChild();
           while( strncmp(aLoc.GetCurrent()->value , "parDeclaration" , 14) == 0 ) {
              aParNumber++;
              if(!aLoc.GoBrother())
                 break;
           }
           sprintf(aux1, "event(\"%s\",\"%d\")" , aux , aParNumber );
           KBAssert(aux1);
           SKIP_APPLY();
       }}

 Template endEventPattern
   Lhs: {{dast plan.eventAttribute
       is EndEvent
   }}

/*----------------------------------------------------------------------
 ----------------------------------------------------------------------*/
 Transform LocateDefinitions
    Lhs: {{dast plan.program
         [[planLibrary pl]]
    }}
    Post-Match:
    {{dast txt.decls
           definitionsLoc = new RawDASTLocater;
         GET_VALUE("pl",(*definitionsLoc));
           SKIP_APPLY();
       }}

Transform putDecompositionValuesInKB
   Lhs: {{dast plan.decomposition
       composedBy [[decompositionStatement* ds]]
   }}
   Pre-Apply:   {{ dast txt.decls
       RawDASTLocater *aLoc;
       aLoc = new RawDASTLocater();
       GET_VALUE("ds", (*aLoc) );

       // encontrar o nome do evento para a regra atual
       RawDASTLocater aTemp(aLoc);
```

```
        strcpy(aux , "");
        // subir ate EventDeclaration1
        while(aTemp.GoFather()) {
            if(strcmp(aTemp.GetCurrent()->value , "eventDeclaration1") == 0) {
                // descer ate o nome do evento
                aTemp.GoChild();
                aTemp.GoChild();
                aTemp.GoBrother();
                if(strcmp(aTemp.GetCurrent()->value , "eventName1") != 0) {
                    reportError("ERRO : SITUACAO NAO PREVISTA");
                    delete (aLoc);
                    SKIP_APPLY();
                }
                aTemp.GoChild();
                strcpy(aux , aTemp.GetCurrent()->value);
                /* criar a lista de parametros */
                aTemp.GoFather();
                aTemp.GoFather();
                aTemp.GoBrother();
                if(strncmp(aTemp.GetCurrent()->value , "parListdecl" , 11) != 0) {
                    reportError("ERRO : SITUACAO NAO PREVISTA");
                    delete (aLoc);
                    SKIP_APPLY();
                }
                aTemp.GoChild();
                aTemp.GoChild();
                int i=0;

                do {
                    aTemp.GoChild();
                    aux1[i] = aTemp.GetCurrent()->value[0];
                    i++;
                    aTemp.GoFather();
                } while ( ( aTemp.GoBrother() ) && (strncmp(aTemp.GetCurrent()->value ,
"parDeclaration",14) == 0));

                aux1[i] = '\0';  // aux1 contem o string com os parametros na ordem da
declaracao
                break;
            }
        }

        if(strcmp(aux , "") == 0) {
            reportError("ERRO : SITUACAO NAO PREVISTA");
            delete (aLoc);
            SKIP_APPLY();
        }

        /* criar a lista de parametros */

        do {
            aLoc    =    FindPatternOnLocate("FindNextPartOfDeclaration",    aLoc    ,
TheSetOfTransforms , 1);
            if(aLoc == NULL) {
                    break;
            }
        } while(aLoc->GoBrother());
        delete (aLoc);
        /* COlocar a totalizacao*/
    SKIP_APPLY();
```

```
  }}

Template FindNextPartOfDeclaration
    Lhs: {{dast plan.decompositionStatement
        [[Name en]] [[parListdecl ald]] by [[Name rn]] [[parameters p]]  ;
    }}
  Post-Match: {{ dast txt.decls
        // WARNING => estou usando o valor global aux1
    if (is_bound("rn")) {
            char eventPartName[MAX_STR_SIZE] , ruleName[MAX_STR_SIZE];
            COPY_LEAF_VALUE_TO(eventPartName, "en");
            COPY_LEAF_VALUE_TO(ruleName, "rn");
            RawDASTLocater aLoc;
            GET_VALUE("p", aLoc );
            char aValue[30];
            if(getParameter(&aLoc , "weight" , aValue) == 0)
                strcpy(aValue , "1"); // valor default para regra que nao tem peso

            /* colocar a ordem do parametros que estao em adl em relacao a regra TODO*/
            GET_VALUE("ald", aLoc );
            aLoc.GoChild();
            aLoc.GoChild();
            int aPartOrder = 1;
            do {   /* para cada parametro */
              // aux1 contem o string com os parametros na ordem da declaracao

                RawDASTLocater aNewTempLoc((&aLoc));
                int anOrder = -1;
                if( strcmp( aNewTempLoc.GetCurrent()->value , "parDeclaration1" ) == 0) {
    /* tratar Name */
                    aNewTempLoc.GoChild();
                    if((anOrder  =  findCharOrderIn(aNewTempLoc.GetCurrent()->value[0]  ,
aux1)) != -1) {
                        anOrder = anOrder + 1;
                        //                      ruleParOrder(         "nomeEventoTodo"
,"nomeEventoParte","regra","nrOrdemNoEventoparte","nrOrdemNoEventoTodo","difVers
ao").
                      sprintf(aux2    ,    "ruleParOrder(\"%s\",\"%s\",\"%s\",\"%d\",\"%d\",\"0\")",
aux ,eventPartName , ruleName , aPartOrder , anOrder);
                        KBAssert(aux2);
                    } else {
                    reportError("Nao  achou  algum  parametro  do  evento  parte  na  lista  de
parametros do evento todo");
                    }
              } else {
                    if( strcmp( aNewTempLoc.GetCurrent()->value , "parDeclaration2" ) == 0)
{   /* tratar previusVersion */
                        aNewTempLoc.GoChild();
                        aNewTempLoc.GoChild();
                        if((anOrder = findCharOrderIn(aNewTempLoc.GetCurrent()->value[0] ,
aux1)) != -1) {
                            anOrder = anOrder + 1;
                            //                      ruleParOrder(         "nomeEventoTodo"
,"nomeEventoParte","regra","nrOrdemNoEventoparte","nrOrdemNoEventoTodo","difVers
ao").
                          sprintf(aux2                                                    ,
"ruleParOrder(\"%s\",\"%s\",\"%s\",\"%d\",\"%d\",\"+1\")", aux ,eventPartName , ruleName
, aPartOrder , anOrder);
                            KBAssert(aux2);
                        } else {
```

```
                        reportError("Nao achou algum parametro do evento parte na lista de
parametros do evento todo");
                        }
                    } else {
                        if( strcmp( aNewTempLoc.GetCurrent()->value , "parDeclaration3" )
== 0) {    /* tratar nextVersion */
                            aNewTempLoc.GoChild();
                            aNewTempLoc.GoChild();
                            if((anOrder      =       findCharOrderIn(aNewTempLoc.GetCurrent()-
>value[0] , aux1)) != -1) {
                                anOrder = anOrder + 1;
                                //                    ruleParOrder(       "nomeEventoTodo"
,"nomeEventoParte","regra","nrOrdemNoEventoparte","nrOrdemNoEventoTodo","difVers
ao").
                               sprintf(aux2  ,  "ruleParOrder(\"%s\",\"%s\",\"%s\",\"%d\",\"%d\",\"-
1\")", aux ,eventPartName , ruleName , aPartOrder , anOrder);
                                KBAssert(aux2);
                            } else {
                            reportError("Nao achou algum parametro do evento parte na lista
de parametros do evento todo");
                            }
                      } else {
                            if( strcmp( aNewTempLoc.GetCurrent()->value , "parDeclaration4"
) == 0) {  /* tratar any */
                               sprintf(aux2                                                  ,
"ruleParOrder(\"%s\",\"%s\",\"%s\",\"%d\",\"*\",\"0\")",  aux  ,eventPartName  ,  ruleName  ,
aPartOrder);
                                KBAssert(aux2);
                          }
                        }
                    }
                }

                aPartOrder++;
            }  while  (  (  aLoc.GoBrother()  )  &&  (strncmp(aLoc.GetCurrent()->value  ,
"parDeclaration" , 14) == 0));

            //                                          rule(              "nomeEventoTodo"
,"nomeEventoParte","regra","valorDoPeso","nrTotalDeParametros").
          sprintf(aux2  ,  "rule(\"%s\",\"%s\",\"%s\",\"%s\",\"%d\")",  aux  ,eventPartName  ,
ruleName , aValue , aPartOrder - 1);
            KBAssert(aux2);
      }
    }}
/*********************************************************************************
*********************************************************************************/
Set Of Transforms HandleObservedFactsSet
  Trigger: external
  Method
    Apply: Single Step
/*-------------------------------------------------------------------
 ---------------------------------------------------------------------*/
  Transform LocateObservedEvent
     Lhs: {{dast plan.observedEvent
         [[Name anEvent]] [[argList pl]] [[value v]]
     }}
     Post-Match:
     {{dast txt.decls
            char aCurrentEventName[MAX_STR_SIZE];
    /* INICIALIZACAO */
          COPY_LEAF_VALUE_TO(currentObservedEventName, "anEvent");
          /* INICIO DA GERACAO DA ARVORE PARA O EVENTO OBSERVADO */
          strcpy(aCurrentEventName ,  currentObservedEventName);

       CLEAR_OBJECT(WSEndEvents);
    TEMPLATE("copyEmptyObservedEvent");
        PLACE_AT("WSEndEvents");
       END_TEMPLATE;

          CLEAR_OBJECT(rationaleForCurrentInferredEvent);
        TEMPLATE("createRationale");
             TRANSPORT_VALUE("anEvent");
             TRANSPORT_VALUE("pl");
             TRANSPORT_VALUE("v");
           PLACE_AT("rationaleForCurrentInferredEvent");
          END_TEMPLATE;

          /* inicializacao */
          CLEAR_OBJECT(WSObservedEvents);
      TEMPLATE("copyEmptyObservedEvent");
            PLACE_AT("WSObservedEvents");
          END_TEMPLATE;

          /* Tratar o evento observado */
          newEventInferredThatNeedToBeProcessed = 0;
          TRANSFORM_WS((definitionsLoc), "FindPlanSet"); /* gerou e colocou em
WSObservedEvents */
          int anIntCount = 1;
          while( newEventInferredThatNeedToBeProcessed != 0) {
             anIntCount++;

             /* passando os eventos observados para o novo WS */
             CLEAR_OBJECT(WSCurrentObservedEvents);
           TEMPLATE("copyToTarget");
              MOVE1("WSObservedEvents" , "pr");
               PLACE_AT("WSCurrentObservedEvents");

             END_TEMPLATE;
             /* inicializacao */
             CLEAR_OBJECT(WSObservedEvents);
             PLACE("copyEmptyObservedEvent", "WSObservedEvents")

             /* tratar os eventos observados */
             newEventInferredThatNeedToBeProcessed = 0;
             TRANSFORM_WS(WSCurrentObservedEvents,
"HandleGeneratedFactsSet");
        }
/* FINALIZACAO */
    /* MERGE COM OS EVENTOS INFERIDOS DAS OBSERVACOES ANTERIORES */
    /* Pegar os eventos em WSEndEvents e realizar o merge com os ja existentes em
WSFinal */

          TRANSFORM_WS(WSEndEvents, "MergeEndEvents");
          SKIP_APPLY();
       }}
       Template copyEmptyObservedEvent
       Rhs: {{dast plan.planRecognition
         Recognition
          begin
          end
     }}

         Template copyToTarget
         Rhs: {{dast plan.program
                   [[planRecognition pr]]
       }}

         // rationale para eventos observados
         Template createRationale
         Rhs: {{dast plan.observedEvent
               [[Name anEvent]] [[argList pl]] [[value v]]
       }}



/*******************************************************************************
*******************************************************************************/
Set Of Transforms HandleGeneratedFactsSet
  Trigger: external
  Method
    Search: Top-Down
    Apply: Single Step
      /* Exhaustive     Single Step     Search: Bottom-Up */
/*-----------------------------------------------------------------------
 ------------------------------------------------------------------------*/
 /* Observacao: no LHS estou procurando apenas por regras que tenham 1 rationale,
apesar de na gramatica poderem ser varios. Isto nao tem reflexo
              aqui por enquanto pois so vao existir mais de um rationale apos ser dado o
merge.
 */
  Transform LocateGeneratedEvent
     Lhs: {{dast plan.topInferredEvent
             [[Name anEvent]] [[argList pl]] [[value aValue]] { [[rationale rat]] } .
   }}
   Pre-Apply: {{dast txt.decls
             char aLocalEventName[MAX_STR_SIZE];
             COPY_LEAF_VALUE_TO(currentObservedEventName, "anEvent");
             strcpy(aLocalEventName ,  currentObservedEventName);

             CLEAR_OBJECT(rationaleForCurrentInferredEvent);
           TEMPLATE("createRationale");
                 TRANSPORT_VALUE("anEvent");
                 TRANSPORT_VALUE("rat");
                 TRANSPORT_VALUE("pl");
               PLACE_AT("rationaleForCurrentInferredEvent");
             END_TEMPLATE;

             TRANSFORM_WS((definitionsLoc), "FindPlanSet");

             DESTROY_CURRENT_TREE();
        }}

     Template createRationale
     Rhs: {{dast plan.inferredEvent
         [[Name anEvent]] [[argList pl]] { [[rationale rat]] }
   }}
```

```
/*********************************************************************************
*********************************************************************************/
Set Of Transforms FindPlanSet
  Trigger: external
  Method
    Apply: Single Step
/*---------------------------------------------------------------------
 ---------------------------------------------------------------------*/
Transform LocateCurrentEventDeclaration
     Lhs: {{dast plan.eventDeclaration
          Event [[Name en]] [[parListdecl ad]] [[eventAttribute ea]] [[declBody db]]
     }}
     Pre-Apply:    {{ dast txt.decls
          COPY_LEAF_VALUE_TO(aux, "en");

          if(strcmp(currentObservedEventName , aux) == 0) { /* eh a decl do proprio evento
*/
               APPLY("HandleSpecializationRule", "db");
          } else {
               /* verificar se o evento e´ parte do atual */
               strcpy(InferredEventNameToBeInserted , aux);
               APPLY("HandleDecompositionRules", "db");
          }
      SKIP_APPLY();
     }}

/*********************************************************************************
*********************************************************************************/
Set Of Transforms HandleDecompositionRules
  Trigger: external
  Method
    Apply: Single Step
  Init: {{dast txt.decls
          strcpy(roleNameForCurrentObservedEventNameAsPartOf , "");
  }}
  End: {{dast txt.decls
  }}
/*---------------------------------------------------------------------
 ---------------------------------------------------------------------*/
Transform CurrentEventDeclaredAsPartOf
     Lhs: {{dast plan.decompositionStatement
          [[Name en]] [[parListdecl al]] by [[Name rn]] [[parameters p]]  ;
     }}
     Pre-Apply:    {{ dast txt.decls
      SPRINT_VAR("en", aux2);


          if(strcmp(currentObservedEventName , aux2) == 0) {  // achou a declaracao que o
evento currentObservedEventName e´ parte de InferredEventNameToBeInserted
               COPY_LEAF_VALUE_TO(roleNameForCurrentObservedEventNameAsPartOf,
"rn");
               TRANSFORM_WS(WSObservedEvents,
"AddNewDecompositionInferredEvent");
          }
     SKIP_APPLY();
   }}

/*********************************************************************************
*********************************************************************************/
Set Of Transforms HandleSpecializationRule
```

```
 Trigger: external
 Method
   Apply: Single Step
/*----------------------------------------------------------------------
 ----------------------------------------------------------------------*/
Transform GeneralizationForCurrentEvent
    Lhs: {{dast plan.specializationRule
        isa [[Name n]]
    }}
    Pre-Apply:    {{ dast txt.decls
        COPY_LEAF_VALUE_TO(InferredEventNameToBeInserted, "n");
        TRANSFORM_WS(WSObservedEvents, "AddNewGeneralizationInferredEvent");
    SKIP_APPLY();
  }}

/*********************************************************************************
*********************************************************************************/
Set Of Transforms AddNewGeneralizationInferredEvent
 Trigger: external
 Method
   Apply: Single Step

/*----------------------------------------------------------------------
 ----------------------------------------------------------------------*/
Transform AddNewGeneralizationInferredEventTransformation
    Lhs: {{dast plan.planRecognition
      Recognition
        begin
         [[topInferredEvent* oe]]
     end
  }}
    Pre-Apply: {{dast txt.decls
        SET_LEAF_VALUE("en", InferredEventNameToBeInserted);
        SET_VALUE("rat", rationaleForCurrentInferredEvent);
        /* achar o valor dos parametros que estao em rationaleForCurrentInferredEvent e
copiar para o novo evento inferido */
        RawDASTLocater *aLoc;
        aLoc = new RawDASTLocater();

        aLoc = FindPatternOnLocate("FindParList", rationaleForCurrentInferredEvent ,
TheSetOfTransforms , 1);
        if(aLoc == NULL) {
                ULF_DEBUG_MSG("valor null na procura por parList");
        } else {
                SET_VALUE("pl", aLoc );
        }

        /* ver se o evento / objeto ja nao foi inserido anteriormente */
        dumpRationaleToString(aux1);
        sprintf(aux, "\"%s\"", aux1);
        int                          aResult                          =
preProcessNewEventToBeLateProcessed(InferredEventNameToBeInserted , aux);
        if(aResult == 0)
            return(0);
        if(aResult == 1) {    /* endEvent */
            CLEAR_OBJECT(WSAux);
            TEMPLATE("copyEndInferredEvend");
                SET_TEMPL_LEAF_VALUE("en", InferredEventNameToBeInserted);
                SET_TEMPL_VALUE("pl", aLoc );
                SET_TEMPL_VALUE("rat", rationaleForCurrentInferredEvent);
```

```
              INSTANTIATE_TEMPLATE_AT(WSAux);
            END_TEMPLATE;
            TRANSFORM_WS(WSEndEvents, "AddTopInferredEventToWS");
            return(0); /* se e´ um endEvent nao precisa ser verificado novamente, ou seja,
nao devemos inserir em WSObservedEvents */
        }
    }}
    Rhs: {{dast plan.planRecognition
      Recognition
        begin
            [[topInferredEvent* oe]]
            [[Name en]] [[argList pl]] { isa [[anyEvent rat]] } .
        end
  }}

    Template copyEndInferredEvend
    Rhs: {{dast plan.topInferredEvent
        [[Name en]] [[argList pl]] { isa [[anyEvent rat]] } .
  }}

    Template FindParList
    Lhs: {{dast plan.argList
        ( [[argValue* pv]] )
    }}

/*******************************************************************************
*******************************************************************************/
Set Of Transforms AddNewDecompositionInferredEvent
  Trigger: external
  Method
    Apply: Single Step
/*---------------------------------------------------------------------
 ---------------------------------------------------------------------*/
Transform AddNewDecompositionInferredEventTransformation
    Lhs: {{dast plan.planRecognition
      Recognition
        begin
         [[topInferredEvent* oe]]
     end
  }}
    Pre-Apply: {{dast txt.decls
        SET_LEAF_VALUE("en", InferredEventNameToBeInserted);
        SET_LEAF_VALUE("ruleName",
roleNameForCurrentObservedEventNameAsPartOf);
        SET_VALUE("rat", rationaleForCurrentInferredEvent);

// CRIAR A LISTA DE PARAMETROS PARA O NOVO EVENTO QUE VAI SER
INSERIDO
        // achar o valor dos parametros que estao em rationaleForCurrentInferredEvent
        RawDASTLocater *anOldParList;
        anOldParList = new RawDASTLocater();
        anOldParList                    =                    FindPatternOnLocate("FindParList",
rationaleForCurrentInferredEvent , TheSetOfTransforms , 1);
        if(anOldParList == NULL) {
                ULF_DEBUG_MSG("valor null na procura por parList");
                SKIP_APPLY();
        }

        // criar a nova lista de parametros com todos os elementos preenchidos com o
coringa (*)
```

```
        // obter o nr de parametros KB => event( "nomeEventoTodo"
,"nrTotalDeParametros")

        char anAux[200];
        int aParNumber;
        sprintf(anAux , "event(\"%s\",*nrParam)", InferredEventNameToBeInserted );
        if (KBSolve(anAux)) {
        strcpy(anAux , KBRetrieve( "*nrParam" , 1));
            aParNumber = removeQuotesAndGetIntValue(anAux);
        } else {
            ULF_DEBUG_MSG("nao achou o nr total de parametros da regra");
            ULF_DEBUG_MSG(anAux);
            SKIP_APPLY();
        }
        RawDASTLocater *aNewParList;
        aNewParList = new RawDASTLocater(WSAux->GetTmpTree().MakeCopy());
        CLEAR_OBJECT(aNewParList);
        TEMPLATE("createEmptyParList");
          INSTANTIATE_TEMPLATE_AT(aNewParList);
        END_TEMPLATE;

        for(int i=0;i<aParNumber;i++) {
            TRANSFORM_LOCATER( aNewParList , "AddNewEmptyParSOT");
        }

        // percorrer a lista de parametros anteriores e setar a nova lista
        anOldParList->GoChild();
        anOldParList->GoChild();
        int aParOrderInPart = 1;
        int aParOrderInWhole;
        do {
            // para cada parametro em anOldParList ler a KB para:
            //    => ver a ordem dele na regra todo
            //    => ver se o elemento e´ a versao anterior ou posterior na regra todo.
          sprintf(anAux , "ruleParOrder(\"%s\",\"%s\",\"%s\",\"%d\",*anOrder,*aDifVersion)",
InferredEventNameToBeInserted , currentObservedEventName ,
roleNameForCurrentObservedEventNameAsPartOf , aParOrderInPart);
            if (!KBSolve(anAux)) {
                ULF_DEBUG_MSG("nao achou o nr total de parametros da regra (1)");
                ULF_DEBUG_MSG(anAux);
                SKIP_APPLY();
            }
        strcpy(anAux , KBRetrieve( "*anOrder" , 1));
            if(strcmp(anAux , "\"*\"") != 0) {
                aParOrderInWhole = removeQuotesAndGetIntValue(anAux);
                // setar em aNewParList o parametro correspondente a aParOrderInWhole
                // achar a posicao em aNewParList
                RawDASTLocater *aTempLoc;
                aTempLoc = new RawDASTLocater(aNewParList);
                aTempLoc->GotoRawCode();
REVISAO_CODIGO_SUSPEITO(12,(aTempLoc));
                aTempLoc->GoChild();
                aTempLoc->GoChild();
                for(int i=1;i<aParOrderInWhole;i++) {
                    if(!aTempLoc->GoBrother())
                        reportError("ERRO na procura de parametros" );
                }
UlfDebugInfo aDebugInfo1("TESTE PARA VER SE rationaleForCurrentInferredEvent
Mudou" , 1);
aDebugInfo1.setBefore(rationaleForCurrentInferredEvent);
```

```
            // copiar o valor anterior (anOldParList) para a posicao atual (aTempLoc) da
nova lista (aNewParList)

UlfDebugInfo aDebugInfo2("Teste de copia de parametro" , 1);
aDebugInfo2.setBefore(aTempLoc);

/*
            RawDASTLocater aNewAux((anOldParList->GetTmpTree().MakeCopy())); //
FUNCIONA MAS NAO DUPLICA
            aTempLoc->Replace(aNewAux);
*/
            if(strcmp(anOldParList->GetCurrent()->value , "argValue1") == 0) {
               char aStr[30];
               char aCurrentVersionStr[10] , aNewVersion[10];
               {
                  RawDASTLocater anAux(anOldParList);
                  anAux.GoChild();
                  strcpy(aStr , anAux.GetCurrent()->value);
                  anAux.GoBrother();
                  anAux.GoChild();
                  strcpy(aCurrentVersionStr , anAux.GetCurrent()->value);
               }

               char aDifVersionStr[20];
            strcpy(aDifVersionStr , KBRetrieve( "*aDifVersion" , 1));
               int aDifVersion = removeQuotesAndGetIntValue(aDifVersionStr);
               if(aDifVersion != 0)        {
                  int aCurrentVersion = atoi(aCurrentVersionStr);
                  if(aCurrentVersionStr == 0)
                     reportError("Erro na leitura da versao do evento observado na
geracao da versao para o evento inferido");
                  sprintf(aNewVersion , "%d" , aCurrentVersion + aDifVersion);
               } else
                  strcpy(aNewVersion , aCurrentVersionStr);

               RawDASTLocater    *aNewAux    =    new    RawDASTLocater(WSAux-
>GetTmpTree().MakeCopy());
               CLEAR_OBJECT(aNewAux);
               TEMPLATE("copyParValue");
                  SET_TEMPL_LEAF_VALUE("aStr", aStr);
                  SET_TEMPL_LEAF_VALUE("aVersion", aNewVersion);
                 INSTANTIATE_TEMPLATE_AT(aNewAux);
               END_TEMPLATE;
               aTempLoc->Replace(*aNewAux);
            } else {
            printf("\nERRRRRRRRRRROOOOOOOOOO");
               reportError("Falta especificacao de como copiar ..." , anOldParList);
               RawDASTLocater                             aNewAux((anOldParList-
>GetTmpTree().MakeCopy())); // FUNCIONA MAS NAO DUPLICA
               aTempLoc->Replace(aNewAux);
            }
         }
         aParOrderInPart++;
      } while(anOldParList->GoBrother() );
      SET_VALUE("pl", aNewParList );

      /* ver se o evento / objeto ja nao foi inserido anteriormente */
      dumpRationaleToString(aux1);
```

```
        sprintf(aux,      "\"%s:%s\")",      roleNameForCurrentObservedEventNameAsPartOf,
aux1);
        int                                      aResult                                      =
preProcessNewEventToBeLateProcessed(InferredEventNameToBeInserted , aux);
        if(aResult == 0)
            return(0);
        if(aResult == 1) {    /* endEvent */
            CLEAR_OBJECT(WSAux);
            TEMPLATE("copyEndInferredEvend");
                SET_TEMPL_LEAF_VALUE("en", InferredEventNameToBeInserted);
                SET_TEMPL_LEAF_VALUE("ruleName",
roleNameForCurrentObservedEventNameAsPartOf);
                SET_TEMPL_VALUE("rat", rationaleForCurrentInferredEvent);
                SET_TEMPL_VALUE("pl", aNewParList);
              INSTANTIATE_TEMPLATE_AT(WSAux);
            END_TEMPLATE;

            TRANSFORM_WS(WSEndEvents, "AddTopInferredEventToWS");

            SKIP_APPLY(); /* se e´ um endEvent nao precisa ser verificado novamente, ou
seja, nao devemos inserir em WSObservedEvents */
        }
    }}
    Rhs: {{dast plan.planRecognition
      Recognition
        begin
            [[topInferredEvent* oe]]
            [[Name en]] [[argList pl]] { [[Name ruleName]] : [[anyEvent rat]] } .
        end
  }}

    Template copyEndInferredEvend
    Rhs: {{dast plan.topInferredEvent
        [[Name en]] [[argList pl]] { [[Name ruleName]] : [[anyEvent rat]] } .
  }}
    Template FindParList
    Lhs: {{dast plan.argList
        ( [[argValue* pv]] )
    }}
    Template createEmptyParList
    Rhs: {{dast plan.argList
      ( )
  }}
    Template copyParValue
    Rhs: {{dast plan.argValue
        [[String aStr]] [[version aVersion]]
  }}


/*********************************************************************************
*********************************************************************************/
Set Of Transforms AddTopInferredEventToWS
  Trigger: external
  Method
    Apply: Single Step

Transform AddTopInferredEventToWSTransformation
    Lhs: {{dast plan.planRecognition
      Recognition
```

```
        begin
         [[topInferredEvent* oe]]
     end
  }}
    Pre-Apply: {{dast txt.decls
        SET_VALUE("aNewEvent", WSAux);
    }}

    Rhs: {{dast plan.planRecognition
      Recognition
        begin
            [[topInferredEvent* oe]]
            [[topInferredEvent aNewEvent]]
        end
  }}

Set Of Transforms AddNewEmptyParSOT
  Trigger: external
  Method
    Apply: Single Step

Transform AddNewEmptyPar
    Lhs: {{dast plan.argList
        ( [[argValue* pvl]] )
  }}

    Rhs: {{dast plan.argList
        ( [[argValue* pvl]] , * )
  }}
/*******************************************************************************
*******************************************************************************/
Set Of Transforms MergeEndEvents
  Trigger: external
  Method
    Search: Top-Down
    Apply: Single Step

/* acha o proximo endEvent em WSEndEvents a ser mergeado em WSFinal */
Transform LocateNewEndEvent
    Lhs: {{dast plan.topInferredEvent
        [[Name aName]] [[argList pl]] [[value aValue]] { [[rationaleStatement* rat]] } .
  }}
    Pre-Apply: {{dast txt.decls
        char aCurrentEndEventName[100];
        int aCurrentEventHasBeenProcessed = 0; //  marca  se  o  evento  atual  ja  foi
processado de alguma maneira.
        int needToCopySavedEvents = 0;

        COPY_LEAF_VALUE_TO(aCurrentEndEventName, "aName");

        /* REGRAS:
        1) Se existir o mesmo endEvent entao juntar os rationales
        2) Se nao existir incluir um novo
        */

        /* procurar o mesmo endEvent em WSFinal*/
        RawDASTLocater    *aFinalEvent    ,    *aFinalEventRat    ,    *aNewEndEvent    ,
*aNewEndEventRat ;
      aNewEndEvent = new RawDASTLocater(l);
```

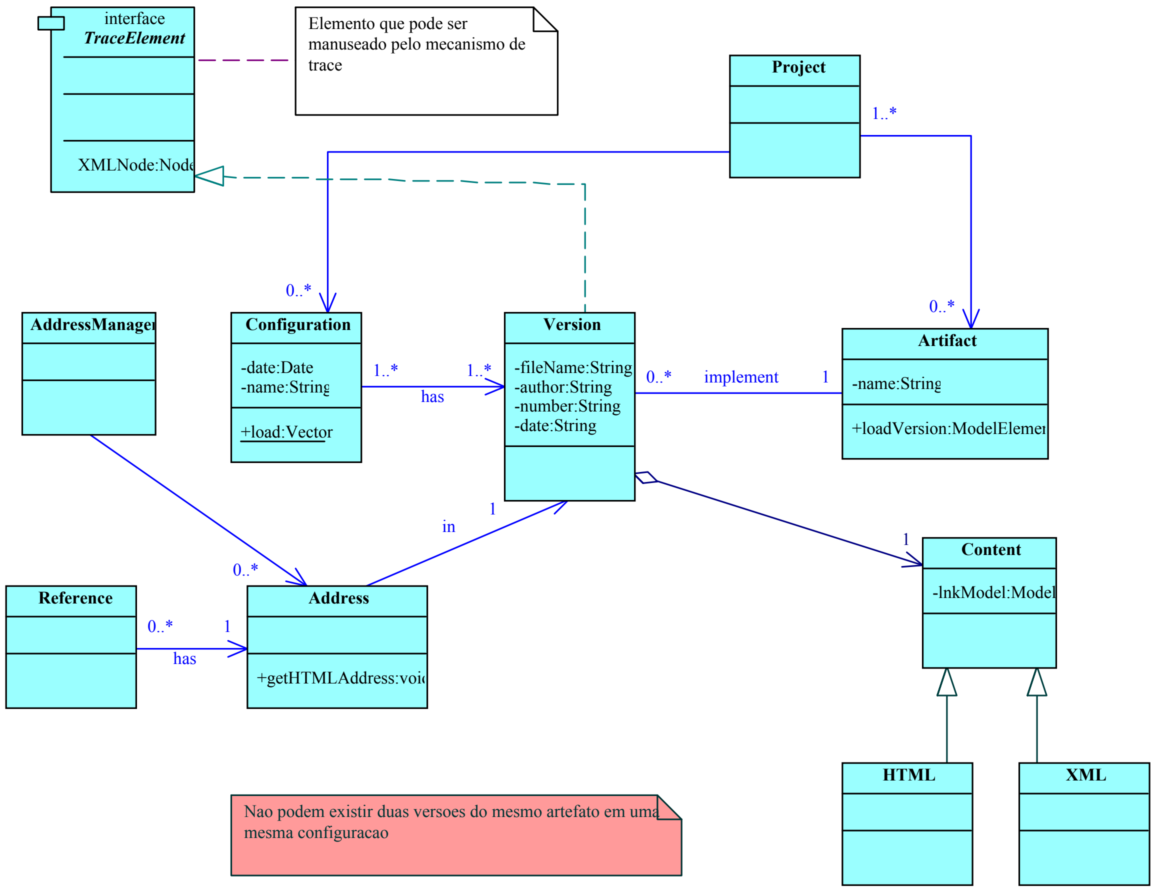
```
        aFinalEvent = FindPatternOnLocate("nextTopInferredEvend", WSFinal ,
TheSetOfTransforms , 1);
        int aLastFinalEventIndexProcessed = 1;
        while(aFinalEvent != NULL) {
            RawDASTLocater *anAuxFinalEvent;
          anAuxFinalEvent = new RawDASTLocater(aFinalEvent);
REVISAO_CODIGO_SUSPEITO(13,(anAuxFinalEvent));
            anAuxFinalEvent->GoChild();
            anAuxFinalEvent->GoChild();
            anAuxFinalEvent->GoChild();

            // ver se deve ser realizado o Merge
            // regras:
            // 1) mesmo endEvent name
            // 2) mesmos valores nos parametros
            // 3) se for coringa (*) => duplicar a estrutura que tem o coringa e substitui-lo
pelo valor do parametro. manter a estrutura
            //    original do coringa

            if( strcmp(anAuxFinalEvent->GetCurrent()->value , aCurrentEndEventName) ==
0) {/* ver se e´ o mesmo endEvent */
                // eh o mesmo endEvent: verificar se os parametros casam
                // procurando os parametros dos eventos
                RawDASTLocater *aFinalEventParList , *aNewEndEventParList;
                aNewEndEventParList = new RawDASTLocater();
              GET_VALUE("pl",(*aNewEndEventParList));

                aFinalEventParList = new RawDASTLocater();
                aFinalEventParList = FindPatternOnLocate("FindParList", aFinalEvent ,
TheSetOfTransforms , 1);
                if(aFinalEventParList == NULL) {
                    return 0;
                }

                int aParListComparationResult = compareParList(aFinalEventParList ,
aNewEndEventParList);

                if(aParListComparationResult == 1) {
                    // casam com o coringa. Salvar o atual e colocar wm WSAuxFinal
                    CLEAR_OBJECT(WSAux);
                    WSAux->AttachDASTCopyFrom(*aFinalEvent);
                    TRANSFORM_WS(WSAuxFinal, "AddTopInferredEventToWS");

                    // inserir o novo em WSAuxFinal pois o coringa pode ser utilizado em
outra instanciacao
                    needToCopySavedEvents = 1;
                    CLEAR_OBJECT(WSAux);
                    WSAux->AttachDASTCopyFrom(l);  // tentando desta maneira...
                    TRANSFORM_WS(WSAuxFinal, "AddTopInferredEventToWS");
                // atualizar os parametros do atual
                    mergeParList(aFinalEventParList , aNewEndEventParList);
                }

                if( (aParListComparationResult == 1) || (aParListComparationResult == 2) ) {
                // realizar o merge do atual
                    aFinalEventRat = FindPatternOnLocate("nextInferredEvend", aFinalEvent
, TheSetOfTransforms , 1);
                    aNewEndEventRat = FindPatternOnLocate("nextInferredEvend",
aNewEndEvent , TheSetOfTransforms , 1);
                    int aDone = 0;
                    while((aDone == 0) && (aFinalEventRat != NULL) && (aNewEndEventRat
!= NULL)) {
                        int   anAuxResult   =   isSameCurrentRationale(aFinalEventRat   ,
aNewEndEventRat);
                        if(!anAuxResult) { // ver se eh o mesmo rationale atual
                                // se nao for aplicar o merge e retornar
                            aFinalEventRat->GoFather();
                            ADD_COPY_TO_SON(aFinalEventRat , (*aNewEndEventRat));
                            aDone = 1;
                        }
                    // eh o mesmo rationale atual: caminhar para o proximo e realizar nova
interacao se for o caso
                        if(aFinalEventRat->GoChild() && aNewEndEventRat->GoChild()) {
                            aFinalEventRat    =    FindPatternOnLocate("nextInferredEvend",
aFinalEventRat , TheSetOfTransforms , 1);
                            aNewEndEventRat  =   FindPatternOnLocate("nextInferredEvend",
aNewEndEventRat  , TheSetOfTransforms , 1);
                        } else {
                            aFinalEventRat = NULL;
                            aNewEndEventRat =  NULL;
                        }
                    }  // voltando para nova interacao
                    if(aDone == 0) {
                        sprintf(aux  ,  "***  Termino  do  Merge  para  o  evento  %s  =>  ERRO  :
SITUACAO NAO PREVISTA " , aCurrentEndEventName);
                        reportError(aux);
                        delete (anAuxFinalEvent);
                        return (1);
                    }
                }

                if( (aParListComparationResult == 1) || (aParListComparationResult == 2) ) {
                    aCurrentEventHasBeenProcessed = 1;
                }
            }
            // nao foi realizado o merge. Pegando o proximo endEvent em WSFinal
            aFinalEvent   =   FindPatternOnLocate("nextTopInferredEvend",   WSFinal   ,
TheSetOfTransforms , 1);  // pegar novamente em WSFinal o primeiro

            for(int j=0 ; j < aLastFinalEventIndexProcessed ; j++) {
                if(!aFinalEvent->GoBrother()) {
                    aFinalEvent = NULL;
                    break;
                }
                aFinalEvent  =  FindPatternOnLocate("nextTopInferredEvend",  aFinalEvent  ,
TheSetOfTransforms , 1);
            }
            aLastFinalEventIndexProcessed ++;
        }

        if(aCurrentEventHasBeenProcessed == 0) {          /*  o  evento  nao  foi  processado
=> inserir o novo em WSFinal */
            // -----------------------------------------------
            //   INSERCAO DE UM NOVO END EVENT
            // -----------------------------------------------
            CLEAR_OBJECT(WSAux);
            TEMPLATE("copyEndInferredEvend");
                TRANSPORT_VALUE("aName");
                TRANSPORT_VALUE("aValue");
                TRANSPORT_VALUE("rat");
                TRANSPORT_VALUE("pl");
              INSTANTIATE_TEMPLATE_AT(WSAux);
            END_TEMPLATE;
            TRANSFORM_WS(WSFinal, "AddTopInferredEventToWS");
        }

        if(needToCopySavedEvents == 1) {
            // copiar do WSAuxFinal em WSFinal
            // pegar cada endEvent em WSAuxFinal, colocar em WSAux a aplicar a
transformacao de insercao
            RawDASTLocater auxLoc(WSAuxFinal);
            auxLoc    =    FindPatternOnLocate("nextTopInferredEvend",    &auxLoc    ,
TheSetOfTransforms , 1);
            do {
                CLEAR_OBJECT(WSAux);
                WSAux->AttachDASTCopyFrom(auxLoc);
                TRANSFORM_WS(WSFinal, "AddTopInferredEventToWS");

                //   auxLoc   =   FindPatternOnLocate("nextTopInferredEvend",   auxLoc   ,
TheSetOfTransforms , 1);
            } while(auxLoc.GoBrother());

            CLEAR_OBJECT(WSAuxFinal);
         TEMPLATE("copyEmptyObservedEvent");
              PLACE_AT("WSAuxFinal");
            END_TEMPLATE;
        }
        SKIP_APPLY();
    }}

    Template nextInferredEvend
    Lhs: {{dast plan.rationaleStatement
        [[rationale otherRat]]
   }}

    Template nextTopInferredEvend
    Lhs: {{dast plan.topInferredEvent
        [[Name  aNewName]]  [[argList  pl  ]]  [[value  aNewValue]]  {  [[rationaleStatement*
otherRat]] } .
   }}

    Template copyEndInferredEvend
    Rhs: {{dast plan.topInferredEvent
        [[Name aName]] [[argList pl]] [[value aValue]] { [[rationaleStatement* rat]] } .
   }}

    Template FindParList
    Lhs: {{dast plan.argList
        ( [[argValue* pv]] )
    }}
    Template copyEmptyObservedEvent
    Rhs: {{dast plan.planRecognition
      Recognition
        begin
     end
   }}

/*********************************************************************************
*********************************************************************************/
Set Of Transforms CalculateEndEventValue
```

```
 Trigger: external
 Method
   Search: Top-Down
   Apply: Single Step

 Init: {{dast txt.decls
        Port aPort("ResultLog.log" , "w");
        aPort.Flush();
 }}

Transform LocateNewEndEvent
    Lhs: {{dast plan.topInferredEvent
        [[Name aName]] [[argList pl]] [[value aValue]] { [[rationaleStatement* rat]] } .
  }}

    Pre-Apply: {{dast txt.decls
        COPY_LEAF_VALUE_TO(aux, "aName");
        RawDASTLocater aRat;
        GET_VALUE("rat" , aRat);
        aRat.GoFather();
        float aValue = getValueForRationaleList(aux , &aRat );
        sprintf(aux1 , "[\'%5.2f\']" , aValue);
        SET_LEAF_VALUE("aNewValue", aux1);

        if(1) {  // TESE - TESTES => Gera um relatorio final
            char anAux[MAX_STR_SIZE];
            Port aPort("ResultLog.log" , "a");

            sprintf(anAux , "\n%s\t" , aux);
            aPort.Write(anAux);
            aPort.Flush();

            RawDASTLocater aParList;
            GET_VALUE("pl" , aParList);
            TheShell()->Pprint((DASTLocater*)(&aParList) , &aPort);

            sprintf(anAux , "\t%f" , aValue);
            aPort.Write(anAux);

            aPort.Flush();
        }
    }}
    Rhs: {{dast plan.topInferredEvent
        [[Name aName]] [[argList pl]] [[value aNewValue]] { [[rationaleStatement* rat]] } .
  }}
    Template nextInferredEvend
    Lhs: {{dast plan.rationaleStatement
        [[rationale otherRat]]
  }}
    Template nextTopInferredEvend
    Lhs: {{dast plan.topInferredEvent
        [[Name  aNewName]]  [[argList  pl]]   [[value  aNewValue]]  {  [[rationaleStatement*
otherRat]] } .
  }}
```

Apêndice D
Modelos do protótipo da Ferramenta DiffTraceTool

D.1.
Diagrama de Classes – Controle de Configuração

O diagrama mostrado a seguir apresenta as classes necessárias ao controle das configurações do sistema. Estas classes estão agrupadas no pacote *managerpkg*.

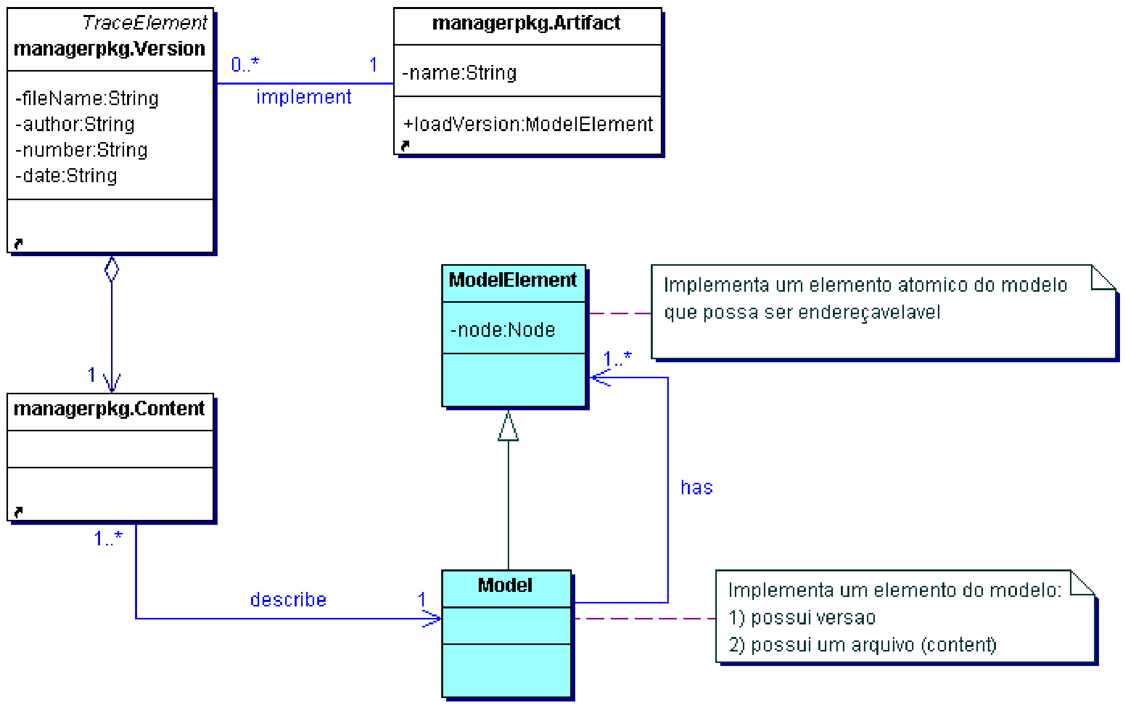


- Interface *TraceElement*: é o parâmetro utilizado pela ferramenta no acionamento do algoritmo TreeDiff. Esta interface deve ser implementada por todas as classes que representam os artefatos que podem ser submetidos ao algoritmo e deve saber responder o nó DOM raiz do artefato em XML;

- Classe *Project*: gerencia as varias configurações e artefatos do sistema em desenvolvimento, permitindo que a ferramenta seja utilizada no desenvolvimento de vários projetos.
- Classe *Artifact*: representa um artefato do processo de desenvolvimento. Um artefato pode ter várias versões ao longo do processo.
- Classe *Configuration*: gerencia as versões dos artefatos existentes em uma configuração do sistema.
- Classe *Version*: representa uma versão de algum artefato e corresponde a um arquivo em disco. A classe implementa a interface *TraceElement* e possui as funcionalidades necessárias a realizar a leitura de arquivos XML e HTML, o seu armazenamento como instâncias da classe *Content*.
- Classe *Content*: armazena o conteúdo de uma versão.

D.2. Diagrama de Classes - Modelo

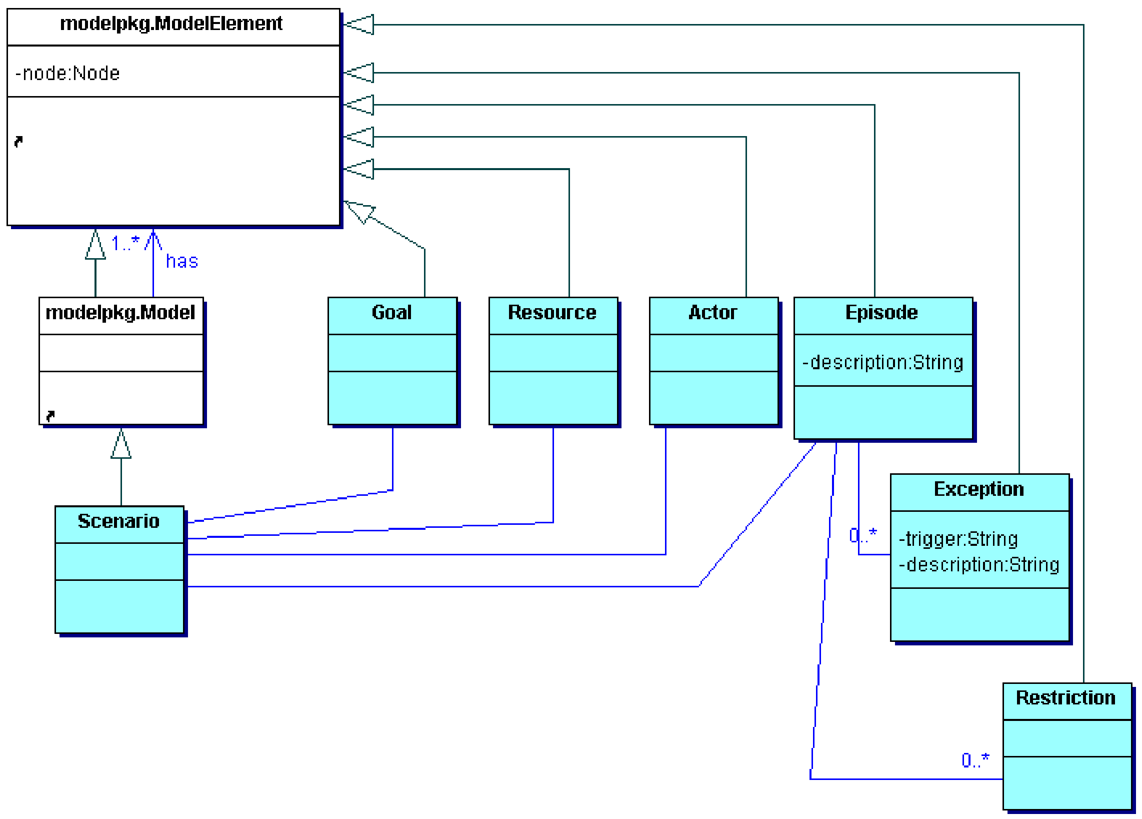
A manipulação do conteúdo dos artefatos é realizada através das classes do pacote *modelpkg* apresentadas no diagrama a seguir.



- Classe *Model*: superclasse abstrata de todas as classes representativas de um artefato do processo de desenvolvimento. A classe *Content* do pacote *managerpkg* faz referência a uma instância de alguma subclasse de *Model*.
- Classe *ModelElement*: superclasse abstrata que representa qualquer entidade que pode ser endereçável, ou seja, podem existir referências a esta entidade através dos elementos *reference* em XML (ver seção 4.1). Cada instância desta classe possui um nó DOM com o conteúdo do elemento.

D.3. Diagrama de Classes – Modelo de Cenários

As classes do pacote *modelpkg* são abstratas e devem ser criadas as subclasses especificas para o modelo do artefato utilizado. No caso do exemplo apresentado nesta tese, foram criadas as classes necessárias a manipulação de cenários que possuem a estrutura definida na seção 3.1. As classes específicas para os cenários são apresentadas no diagrama a seguir.



Apêndice E
Bibliotecas de Planos

E.1.
Reconhecimento de Relacionamentos

```
PlanLibrary
begin
Event subSetContextSimilarity(A,B) ;
Event high_contextSimilarity(A,B)
begin
  isa subSetContextSimilarity

end
Event low_contextSimilarity(A,B) ;
Event contextInsideContext(A,B)
begin
  isa subSetContextSimilarity

end
Event scenarioInsideContext(A,B) ;
Event high_goalSimilarity(A,B) ;
Event low_goalSimilarity(A,B) ;
Event high_actorSimilarity(A,B)
begin
  isa oneOrMoreActorSimilarity

end
Event oneOrMoreActorSimilarity(A,B) ;
Event high_resourceSimilarity(A,B) ;
Event low_resourceSimilarity(A,B) ;
Event high_episodeSimilarity(A,B) ;
Event low_episodeSimilarity(A,B) ;
Event scenarioIsExceptionOfScenario(A,B) ;
Event ScenarioIsEpisodeOfScenario(A,B) ;
Event complement(A,B) is EndEvent
begin
  composedBy
    high_contextSimilarity(A,B) by context with weight = '1.5';
    high_goalSimilarity(A,B) by goal with weight = '2.0';
    high_actorSimilarity(A,B) by actor with weight = '1.5';
    high_resourceSimilarity(A,B) by resource with weight = '1.0';
    low_episodeSimilarity(A,B) by episode with weight = '4.0';

end
Event equivalence(A,B) is EndEvent
begin
  composedBy
    high_contextSimilarity(A,B) by context with weight = '1.5';
    high_goalSimilarity(A,B) by goal with weight = '2.0';
    high_actorSimilarity(A,B) by actor with weight = '1.5';
    low_resourceSimilarity(A,B) by resource with weight = '1.0';
    high_episodeSimilarity(A,B) by episode with weight = '4.0';

end
Event subSet(A,B) is EndEvent
begin
  composedBy
    subSetContextSimilarity(A,B) by context with weight = '3.0';
    low_goalSimilarity(A,B) by goal with weight = '1.5';
    high_actorSimilarity(A,B) by actor with weight = '2.5';
    low_episodeSimilarity(A,B) by episode with weight = '3.0';

end
Event preCondition(A,B) is EndEvent
begin
  composedBy
    scenarioInsideContext(A,B) by context with weight = '4';
    high_actorSimilarity(A,B) by actor with weight = '3';
    low_episodeSimilarity(A,B) by episode with weight = '3';

end
Event possiblePrecedence(A,B) is EndEvent
begin
  composedBy
    high_contextSimilarity(A,B) by context with weight = '4';
    low_goalSimilarity(A,B) by goal with weight = '3';
    oneOrMoreActorSimilarity(A,B) by actor with weight = '3';

end
Event detour(A,B) is EndEvent
begin
  composedBy
    scenarioInsideContext(A,B) by context with weight = '2';
    high_goalSimilarity(A,B) by goal with weight = '2';
    high_actorSimilarity(A,B) by actor with weight = '2';
    high_resourceSimilarity(A,B) by resource with weight = '2';
    scenarioIsExceptionOfScenario(A,B) by episode with weight = '1';
    ScenarioIsEpisodeOfScenario(A,B) by episode with weight = '1';

end
Event exception(A,B) is EndEvent
begin
  composedBy
    scenarioIsExceptionOfScenario(A,B) by episode with weight = '10';

end
Event inclusion(A,B) is EndEvent
begin
  composedBy
    ScenarioIsEpisodeOfScenario(A,B) by episode with weight = '10';

end
end
```

E.2. Reconhecimento de Operações

```
PlanLibrary
begin
/*******************  Observed events  *******************/
/* Relationships */
   Event complement(A,B);
   Event equivalence(A,B);
   Event subSet(A,B);
   Event inclusion(A,B);
/* Configuration Manager */
   Event scenarioRemoved(A);
   Event scenarioAdded(A);
/* Episode changes */
   Event episodeRemoved(A,B)  /* A: scenario  B: episode */
      begin
        isa episodeModification
      end
   Event episodeAdded(A,B)  /* A: scenario  B: episode */
      begin
        isa episodeModification
      end
   Event episodeContentChanged(A)
      begin
        isa episodeModification
      end
/*******************  Intermediate events  *******************/
   Event episodeModification(A);
   Event episodeMoved(A,B,C) /* A: firstScenario  B: secondScenario   C: episode */
      begin
        composedBy
          episodeRemoved(A,C) by source with weight = '1.0';
          episodeAdded(B,C) by target with weight = '1.0';
      end
/*******************  Final events  *******************/
   Event fusionOperation(A,B) is EndEvent;  /* B foi unido em A */
   Event fusionOperationInOldScen(A,B)   /* B foi unido em A */
      begin
         isa fusionOperation
        composedBy
          scenarioRemoved(B) by r1 with weight = '1.0';
          complement(A-,B) by r2 with weight = '1.0';
          episodeMoved(B,A,*) by r3 with weight = '1.0';
      end
   Event fusionOperationInNewScen(A,B,C) /* B foi unido em A no novo cenario C */
      begin
         isa fusionOperation
        composedBy
          scenarioRemoved(A) by r1 with weight = '1.0';
          scenarioRemoved(B) by r2 with weight = '1.0';
          scenarioAdded(C) by r3 with weight = '1.0';
          complement(A,B) by r4 with weight = '1.0';
          episodeMoved(B,C,*) by r5 with weight = '1.0';
          episodeMoved(A,C,*) by r6 with weight = '1.0';
      end
   Event encapsulationOperation(A,B) is EndEvent  /* B foi encapsulado em A */
      begin
```

```
            composedBy
               scenarioRemoved(B) by r1 with weight = '1.0';
               equivalence(A-,B) by r2 with weight = '1.0';
               episodeMoved(B,A,*) by r3 with weight = '1.0';
         end
    Event splitOperation(A,B,C) is EndEvent  /* A foi dividido em B e C */
         begin
            composedBy
               scenarioRemoved(A) by r1 with weight = '1.0';
               scenarioAdded(B) by r2 with weight = '1.0';
               scenarioAdded(C) by r3 with weight = '1.0';
               complement(B,C) by r4 with weight = '1.0';
               episodeMoved(A,B,*) by r5 with weight = '1.0';
               episodeMoved(A,C,*) by r6 with weight = '1.0';
         end
    Event multSplitOperation(A,B,C) is EndEvent  /* A isola comp comum a B e C */
         begin
            composedBy
               scenarioAdded(A) by r1 with weight = '1.0';
               inclusion(A,B+) by r2 with weight = '1.0';
               inclusion(A,C+) by r3 with weight = '1.0';
               episodeMoved(B,A,*) by r4 with weight = '1.0';
               episodeMoved(C,A,*) by r5 with weight = '1.0';
         end
    Event exclusionOperation(A) is EndEvent  /*  */
         begin
            composedBy
               scenarioRemoved(A) by r1 with weight = '1.0';
         end
    Event inclusionOperation(A) is EndEvent  /*  */
         begin
            composedBy
               scenarioAdded(A) by r1 with weight = '1.0';
         end
    Event modificationOperation(A) is EndEvent  /*  */
         begin
            composedBy
               episodeModification(A) by r1 with weight = '1.0';
         end
    Event specializationOperation(A,B,C) is EndEvent  /* A foi dividido em B e C */
         begin
            composedBy
               scenarioAdded(B) by r1 with weight = '1.0';
               scenarioAdded(C) by r2 with weight = '1.0';
               equivalence(B,C) by r3 with weight = '1.0';
               subSet(A,B) by r4 with weight = '1.0';
               subSet(A,C) by r5 with weight = '1.0';
               episodeMoved(A-,B,*) by r6 with weight = '1.0';
               episodeMoved(A-,C,*) by r7 with weight = '1.0';
         end
    Event extensionOperation(A,B) is EndEvent  /* B foi criado e eh uma esp de A */
         begin
            composedBy
               scenarioAdded(B) by r1 with weight = '1.0';
               equivalence(A,B) by r2 with weight = '1.0';
         end
end
```