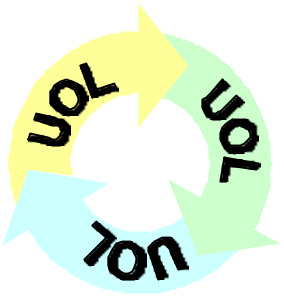

Universal Object Language 1.2

Appendices

Authors

Recerca Informàtica
Daimler-Benz Research and Technology

OMG Document: ad/98-07-08
Version 1.2 /T-UOL-19980708
July 8th, 1998

The most recent updates of the Universal Object Language can be found, via worldwide web, at: http://www.recercai.com

Primary Contacts for the UOL submission:

Recerca Informàtica	Joan M. Moral	uol@arrakis.es
Daimler-Benz Research and Technology	Mario Jeckle	mario.jeckle@dbag.ulm.DaimlerBenz.COM

Table of Contents

The table of Contents contains entries for both the Specification and the Appendices.

A UOL Grammar without UML constructs

In this section, we present a tested BNF grammar. A parser implemented in Java of this grammar can be obtained at http://www.recercai.com.

We give the grammar sorted by the use of the productions.

identifier: [a-zA-Z_0-9]+
Integer_constant: ([1-9][0-9]*|0)
Character_constant: "([^\t\n] | (\\[^\t\n])) "
Range: [1-9][0-9]* {mdot} ([1-9][0-9]*| *)
Float_constant: ([0-9]+.[0-9]*([eE][+|-]?[0-9]+)?
Comment: -- [^(--)\n]
String: ' ([^'\\] | \\[ntbrf\\""]|[0-7][0-7]?|[0-3][0-7][0-7] | [\n\r])* '
TextMultiline: text " [^"] "
CommentMultiline: comment " [^"] "
Anonymous ?[0-9]*

adaptation	**by**	**else**	**implies**	**like**	**package**	**result**	**to**
addonly	**call**	**end**	**import**	**link**	**postcondition**	**select**	**true**
all	**class**	**exception**	**in**	**model**	**precondition**	**static**	**undefine**
and	**constrained**	**false**	**infix**	**none**	**prefix**	**stereotype**	**unique**
any	**constraint**	**feature**	**inherit**	**not**	**raise**	**stereotyped**	**values**
as	**current**	**from**	**inout**	**instance**	**redefine**	**subsystem**	**viewed**
of	**attached**	**deferred**	**frozen**	**interface**	**or**	**relation**	**tag**
with	**BIT**	**diagrams**	**implements**	**is**	**out**	**rename**	**then**
xor							

Bag	**Collection**	**else**	**endif**	**enum**	**if**	**Set**	**Sequence**
then							

(**Non association**
\+ - **Left association**
*
< /= = Xor or And implies >= <=
< > . .. ^ / **Non association**
UPlus UMinus **Right association**
not **Left association**

Start_production -> Model_declaration;
Start_production -> Package_declaration;
Model_declaration -> **model** Model_name Package_decl_list_opt View_element_decl_list_opt **end**;
Model_name -> *identifier*;
View_element_decl_list_opt -> **diagrams** View_element_decl_list **end**;
View_element_decl_list_opt -> ;
View_element_decl_list -> View_element_declaration;

View_element_decl_list -> View_element_decl_list **';'** View_element_declaration;
View_element_declaration -> View_element_name_list **':'** View_element_kind;
View_element_name -> *identifier*;
View_element_kind -> *identifier* Extension_use Invariant_opt;
Package_decl_list_opt -> Package_decl_list;
Package_decl_list_opt -> ;
Package_decl_list -> Package_decl_list Package_or_subsystem_declaration;
Package_decl_list -> Package_or_subsystem_declaration;
Package_or_subsystem_declaration -> Subsystem_declaration;
Package_or_subsystem_declaration -> Package_element_decl;
Package_or_subsystem_declaration -> Use_of_tagged_value;
Package_or_subsystem_declaration -> Use_of_constraint;
Package_or_subsystem_declaration -> Use_of_stereotype;
Package_declaration -> **package** Package_name Viewed_with_opt Extension_use Package_inheritance_opt Package_import_opt Package_element_decl_list_opt Invariant_opt **end**;
Package_name -> *identifier*;
Viewed_with_opt -> **viewed with** View_element_name_list;
Viewed_with_opt -> ;
View_element_name_list -> View_element_name_list **','** View_element_name Position;
View_element_name_list -> View_element_name Position;
Position -> **'('** Dec **','** Dec Third_dimension **')'**;
Position -> ;
Third_dimension -> **','** Dec;
Third_dimension -> ;
Package_inheritance_opt -> **inherit** Package_name_list;
Package_inheritance_opt -> ;
Package_name_list -> Package_name_list **','** Use_of_constraint_opt Package_name Discriminator_opt;
Package_name_list -> Use_of_constraint_opt Package_name Discriminator_opt;
Package_import_opt -> **import** Package_import_list ;
Package_import_opt -> ;
Package_import_list -> Package_import_list **','** Package_import_elem;
Package_import_list -> Package_import_elem;
Package_import_elem -> Visibility_opt Element_path_opt As_alias **from** Package_name;
Visibility_opt -> ;
Visibility_opt -> **'{'** Visibility **'}'**;
As_alias -> **as** Alias;
As_alias -> ;
Element_path_opt -> ;
Element_path_opt -> Element_path;
Element_path -> Element_name;
Element_path -> Element_path Path Element_name;
Alias -> Element_name;
Package_element_decl_list_opt -> **is** Package_element_decl_list;
Package_element_decl_list_opt -> ;
Package_element_decl_list -> Visibility_opt Package_element_decl;
Package_element_decl_list -> Package_element_decl_list Visibility_opt Package_element_decl;
Package_element_decl -> Package_declaration;
Package_element_decl -> Interface_declaration;
Package_element_decl -> Class_declaration;

Package_element_decl -> Relation_declaration;
Package_element_decl -> Extension_declaration;
Package_element_decl -> Usecase_abstraction;
Package_element_decl -> Activity_model;
Package_element_decl -> Comment_definition;
Package_element_decl -> Actor_or_exception Light_body;
Package_element_decl -> Component_or_node Ultra_light_body;
Package_element_decl -> Object_declaration
Object_declaration -> Object_name Formal_generics_opt instance of Invariant_opt Element_path Extension_use Viewed_with_opt Object_body_opt **end**;
Object_name -> *identifier*;
Object_body_opt -> ;
Object_body_opt -> **is** Attribute_value_list;
Attribute_value_list -> *identifier* **is** Expression;
Attribute_value_list -> Attribute_value_list ';' identifier **is** Expression;
Comment_definition -> CommentMultiline Attach_or_view;
Attach_or_view -> **attached to** Name;
Attach_or_view -> viewed with View_element_name_list;
Actor_or_exception -> **actor**;
Actor_or_exception -> **exception**;
Light_body -> Name Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Features Invariant_opt **end**;
Component_or_node -> **component**;
Component_or_node -> **node**;
Ultra_light_body -> Name Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Identifier_list Invariant_opt **end**;
Package_element_decl -> Collaboration_declaration;
Subsystem_declaration -> Subsystem_header Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Package_import_opt Interface_operation_declaration_opt Package_element_decl_list_opt Invariant_opt **end**;
Subsystem_header -> Deferred_opt subsystem Subsystem_name;
Subsystem_name -> *identifier*;
Extension_declaration_opt -> ;
Extension_declaration_opt -> Extension_declaration_opt Extension_declaration;
Extension_declaration -> Constraint;
Extension_declaration -> Tagged_values;
Extension_declaration -> Stereotype;
Extension_use -> Use_of_stereotype_opt List_use_consr_tag;
Use_of_stereotype_opt -> Use_of_stereotype;
Use_of_stereotype_opt -> ;
List_use_consr_tag -> List_use_consr_tag Use_of_tagged_value;
List_use_consr_tag -> ;
Constraint -> **constraint** Constraint_def_list **end**;
Constraint_def_list -> Constraint_name **is** **'{'** Constraint_expression **'}'**;
Constraint_def_list -> Constraint_def_list Constraint_name **is** **'{'** Constraint_expression **'}'**;
Constraint_expression -> OCL_expression;
Constraint_expression -> TextMultiline;
Constraint_name -> *identifier*;
OCL_expression -> OCLexpression;
Use_of_constraint -> **constrained by** **'{'** Constraint_expression **'}'**;
Use_of_constraint_opt -> ;
Use_of_constraint_opt -> Use_of_constraint;
Tagged_values -> **tag values** Tagged_values_def_list **end**;
Tagged_values_def_list -> Tagged_values_def;

Tagged_values_def_list -> Tagged_values_def_list Tagged_values_def;
Tagged_values_def -> Extension_name;
Tagged_values_def -> Extension_name **is** Initial_value;
Extension_name -> *identifier*;
Use_of_tagged_value -> **with tag values** '**(**' Property_list '**)**';
Property_list -> Property;
Property_list -> Property_list '**,**' Property;
Property -> LT Extension_name GT;
Property -> LT Extension_name '**,**' Expression GT;
Stereotype -> **stereotype** Extension_name **of** Base_class Icon_opt Stereotype_parent_opt Stereotype_extension_dec **end**;
Stereotype_extension_dec -> Stereotype_extension_dec Constraint;
Stereotype_extension_dec -> Stereotype_extension_dec Tagged_values;
Stereotype_extension_dec -> ;
Base_class -> Classifier_name;
Icon_opt -> ;
Icon_opt -> **viewed as** String;
Stereotype_parent_opt -> **inherit** Stereotype_parent_list;
Stereotype_parent_opt -> ;
Stereotype_parent_list -> Use_of_constraint_opt Extension_name Discriminator_opt;
Stereotype_parent_list -> Stereotype_parent_list '**,**' Use_of_constraint_opt Extension_name Discriminator_opt;
Use_of_stereotype -> **stereotyped with** Extension_name;
String -> *'STRINGstart' 'STRINGliteral' 'STRINGend'*;
identifier -> *OCLtypeOrName*;
identifier -> *Anonymous*;
Name_comma_list -> Name;
Name_comma_list -> Name_comma_list '**,**' Name;
String_list -> String;
String_list -> String_list '**,**' String;
Classifier_name -> Element_path;
Formal_generics_opt -> ;
Formal_generics_opt -> Formal_generics;
Formal_generics -> '**[**' Formal_generic_list '**]**';
Formal_generic_list -> Formal_generic;
Formal_generic_list -> Formal_generic_list '**,**' Formal_generic;
Formal_generic -> Element_name Invariant_opt;
Element_name -> *identifier*;
Expression -> Call;
Expression -> Operator_expression;
Expression -> Equality;
Expression -> Manifest_constant;
Expression -> Manifest_array;
Call -> Parenthesized_qualifier_opt Call_chain;
Call_chain -> Unqualified_call;
Call_chain -> Call_chain '**.**' Unqualified_call;
Parenthesized_qualifier_opt -> ;
Parenthesized_qualifier_opt -> Parenthesized_qualifier;
Parenthesized_qualifier -> Parenthesized '**.**';
Parenthesized -> '**(**' Expression '**)**';
Unqualified_call -> Entity Actuals_opt;
Entity -> *identifier*;
Entity -> **result**;

(UOL 1.2)

Entity -> **current**;
Actuals_opt -> ;
Actuals_opt -> Actuals;
Actuals -> '**(**' Actual_list '**)**';
Actual_list -> Actual;
Actual_list -> Actual_list '**,**' Actual;
Actual -> Expression;
Operator_expression -> Parenthesized;
Operator_expression -> Unary_expression;
Operator_expression -> Binary_expression;
Unary_expression -> Unary Expression;
Unary -> **not**;
Unary -> '**+**';
Unary -> '**-**';
Binary_expression -> Expression Binary Expression;
Binary -> '**+**';
Binary -> '**-**';
Binary -> '*****';
Binary -> '/';
Binary -> '<';
Binary -> '>';
Binary -> *LTE*;
Binary -> *GTE*;
Binary -> *DD*;
Binary -> *DS*;
Binary -> '^';
Binary -> **and**;
Binary -> **or**;
Binary -> **xor**;
Binary -> **implies**;
Manifest_constant -> Boolean_constant;
Manifest_constant -> Character_constant;
Manifest_constant -> Integer_constant;
Manifest_constant -> Float_constant;
Manifest_constant -> String;
Boolean_constant -> **true**;
Boolean_constant -> **false**;
Float_constant -> *'FLOATINGconstant0'*;
Manifest_array -> *ANGLEBL* Expression_list *ANGLERR*;
Expression_list -> Expression;
Expression_list -> Expression_list '**,**' Expression;
Comparision -> '**=**';
Comparision -> *NOTEQUAL*;
Equality -> Expression Comparision Expression;
Inheritance -> **inherit** Parent_list;
Parent -> Use_of_constraint_opt Class_type Discriminator_opt;
Parent_list -> Parent;
Parent_list -> Parent_list '**;**' Parent;
Discriminator_opt -> ;
Discriminator_opt -> '**(**' Name '**)**';
Class_type -> Class_name Actual_generics_opt;

Actual_generics_opt -> ;
Actual_generics_opt -> Actual_generics;
Actual_generics -> **'['** Type_list **']'**;
Type_list -> Type;
Type_list -> Type_list **','** Type;
Type -> Class_type;
Type -> Class_type_expanded;
Type -> Anchored;
Type -> Bit_type;
Class_type_expanded -> **expanded** Class_type;
Anchored -> **like** Anchor;
Anchor -> *identifier*;
Anchor -> **current**;
Bit_type -> **BIT** Constant;
Constant -> Manifest_constant;
Constant -> Entity;
Rename_opt -> ;
Rename_opt -> Rename;
Rename -> **rename** Rename_list;
Rename_list -> Rename_pair;
Rename_list -> Rename_list **','** Rename_pair;
Rename_pair -> Feature_name **as** Feature_name;
Feature_name -> *identifier*;
Feature_name -> Prefix;
Feature_name -> Infix;
Infix -> **infix** **'('** Infix_operator **')'**;
Infix_operator -> Binary;
Infix_operator -> *identifier*;
Prefix -> **prefix** **'('** Prefix_operator **')'**;
Prefix_operator -> Unary *identifier*;
New_exports_opt -> ;
New_exports_opt -> New_exports;
New_export_item -> Clients Feature_set;
New_export_list -> New_export_item;
New_export_list -> New_export_list **';'** New_export_item;
New_exports -> **export** New_export_list;
Class_list -> Class_name;
Class_list -> Class_list **','** Class_name;
Clients -> **'{'** Class_list **'}'**;
Feature_set -> Identifier_list;
Feature_set -> **all**;
Undefine_opt -> ;
Undefine_opt -> Undefine;
Undefine -> **undefine** Feature_list;
Select_opt -> ;
Select_opt -> Select;
Select -> **select** Feature_list;
Relation_declaration -> **relation** Relation_name Extension_declaration_opt Extension_use Relation_inheritance_opt Link_list_opt Features_attrib_or_Oper Invariant_opt **end**;
Relation_inheritance_opt -> ;
Relation_inheritance_opt -> Relation_inheritance;
Link_list_opt -> ;

(UOL 1.2)

Link_list_opt -> Link_list;
Relation_name -> *identifier*;
Relation_inheritance -> **inherit** Parent_relation_list;
Parent_relation_list -> Parent_relation;
Parent_relation_list -> Parent_relation_list **';'** Parent_relation;
Parent_relation -> Use_of_constraint_opt Relation_type Relation_feature_adaptation_opt;
Relation_feature_adaptation_opt -> ;
Relation_feature_adaptation_opt -> **adaptation** Relation_feature_adaptation;
Relation_type -> Relation_path;
Relation_path -> Element_path;
Relation_feature_adaptation -> Rename_opt New_exports_opt Undefine_opt Relation_redefine_opt Select_opt **end**;
Relation_redefine_opt -> ;
Relation_redefine_opt -> Relation_redefine;
Relation_redefine -> **redefine** Feature_or_redef;
Feature_or_redef -> Feature_list;
Feature_or_redef -> Redefine_with_list;
Redefine_with_list -> Redefine_pair ;
Redefine_with_list -> Redefine_with_list **','** Redefine_pair;
Redefine_pair -> Feature_name with Feature_name;
Link_list -> **link** Type_or_dependency ;
Assoc_entity_opt -> ;
Assoc_entity_opt -> **with** Classifier_name;
Type_or_dependency -> Type_link_two_list Assoc_entity_opt;
Type_or_dependency -> Dependency_list Dependency_description_opt;
Type_link_two_list -> Type_link Cardinality_opt **','** Type_link_list;
Cardinality_opt -> ;
Cardinality_opt -> Cardinality2;
Cardinality2 -> **'['** Cardinality **']'**;
Cardinality -> Range_list Range_last;
Range_last -> *Range*;
Range_last -> Int_or_star;
Range_mid -> *Range* **','** ;
Range_mid -> Integer_constant **','**;
Range_list -> ;
Range_list -> Range_list Range_mid;
Int_or_star -> *Integer_constant*;
Int_or_star -> *Star*;
Type_link_list -> Type_link Cardinality_opt;
Type_link_list -> Type_link_list **','** Type_link Cardinality_opt;
Type_link -> Classifier_name;
Dependency_list -> Dependency;
Dependency_list -> Dependency_list **','** Dependency;
Dependency -> Element_path **to** Element_path;
Dependency_description_opt -> **is** TextMultiline;
Dependency_description_opt -> ;

B UOL Grammar with UML constructs

In this section, we present a tested BNF grammar. A parser implemented in Java of this grammar can be obtained at http://www.recercai.com.

We give the grammar sorted by the use of the productions. We also include the parts of the UML diagrams that the UOL refers to. Including the UML diagrams shows clearly the direct mapping that exists between UML and UOL.

As we have said, introducing UML concepts makes trasference more efficient, but its use is optional. The parser is implemented using this grammar and can parse UOL with or without the UML constructs.

identifier:	[a-zA-Z_0-9]+
Integer_constant:	([1-9][0-9]*|0)
Character_constant:	"([^\t\n] | (\\[^\t\n])) "
Range:	[1-9][0-9]* {mdot} ([1-9][0-9]*| *)
Float_constant:	([0-9]+.[0-9]*([eE][+|-]?[0-9]+)?
Comment:	-- [^(--)\n]
String:	' ([^'\\] | \\[ntbrf\\'"']|[0-7][0-7]?|[0-3][0-7][0-7] | [\n\r])* '
TextMultiline:	text " [^"] "
CommentMultiline:	comment " [^"] "
Anonymous	?[0-9]*

adaptation	**after**	**action**	**actions**	**activity**	**actor**	**addonly**	**all**
alternative	**and**	**any**	**as**	**attached**	**BIT**	**branch**	**by**
call	**class**	**collaboration**	**component**	**composite**	**concurrent**	**constrained**	**constraint**
course	**current**	**creates**	**deep**	**deferred**	**diagrams**	**else**	**end**
entry	**event**	**exit**	**expanded**	**export**	**extend**	**extension**	**exception**
false	**feature**	**final**	**flow**	**fork**	**from**	**frozen**	**history**
implements	**implies**	**import**	**in**	**infix**	**inherit**	**initial**	**inout**
interface	**is**	**join**	**like**	**link**	**machine**	**model**	**node**
none	**not**	**of**	**old**	**or**	**out**	**package**	**partitioned**
prefix	**raise**	**redefine**	**request**	**relation**	**rename**	**result**	**select**
shallow	**signal**	**simple**	**state**	**static**	**stereotype**	**stereotyped**	**subactivity**
submachine	**subsystem**	**synchronous**	**tag**	**then**	**to**	**transition**	**trigger**
true	**undefine**	**unique**	**use**	**usecase**	**values**	**viewed**	**when**
with	**xor**						

Bag	**Collection**	**else**	**endif**	**enum**	**if**	**Set**	**Sequence**
then							

(**Non association**

\+ - **Left association**

*

< /= = Xor or And implies >= <=

< > . .. ^ / **Non association**

UPlus UMinus **Right association**

not **Left association**

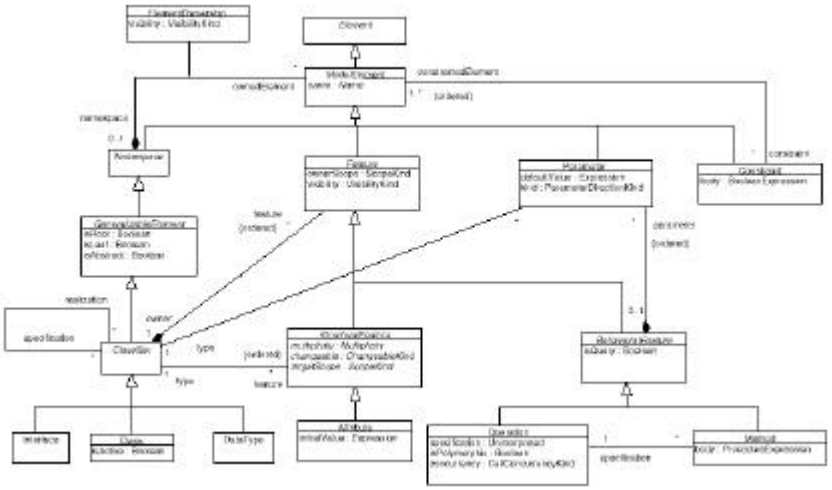

Fig. 1 Core Package - Backbone

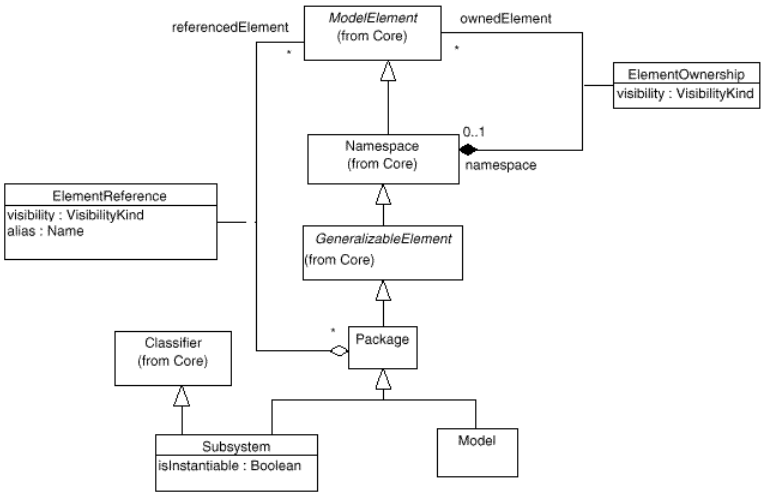

Fig. 2 Model Management

Start_production	->	Model_declaration;
Start_production	->	Package_declaration;
Model_declaration	->	**model** Model_name Package_decl_list_opt View_element_decl_list_opt **end**;
Model_name	->	*identifier*;
View_element_decl_list_opt	->	**diagrams** View_element_decl_list **end**;
View_element_decl_list_opt	->	;
View_element_decl_list	->	View_element_declaration;
View_element_decl_list	->	View_element_decl_list '**;**' View_element_declaration;
View_element_declaration	->	View_element_name_list '**:**' View_element_kind;
View_element_name	->	*identifier*;
View_element_kind	->	*identifier* Extension_use Invariant_opt;
Package_decl_list_opt	->	Package_decl_list;
Package_decl_list_opt	->	;

(UOL 1.2)

Package_decl_list	->	Package_decl_list Package_or_subsystem_declaration;
Package_decl_list	->	Package_or_subsystem_declaration;
Package_or_subsystem_declaration	->	Subsystem_declaration;
Package_or_subsystem_declaration	->	Package_element_decl;
Package_or_subsystem_declaration	->	Use_of_tagged_value;
Package_or_subsystem_declaration	->	Use_of_constraint;
Package_or_subsystem_declaration	->	Use_of_stereotype;

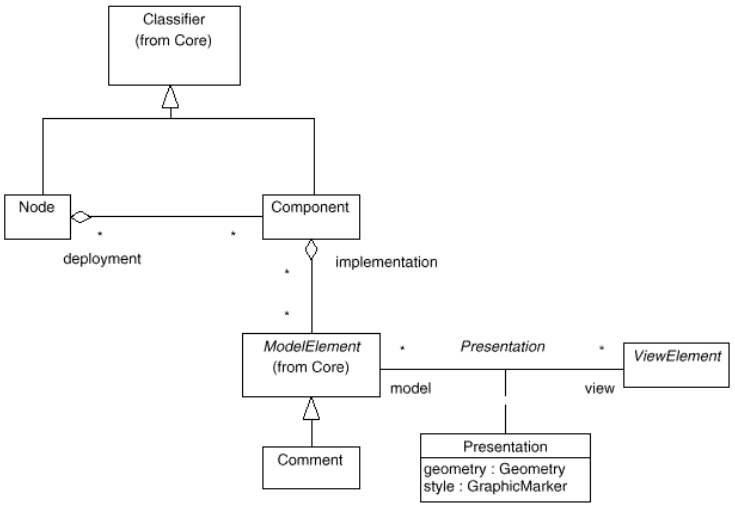

Fig. 3 Auxiliary Elements - Physical Structures and View Elements

Package_declaration	->	**package** Package_name Viewed_with_opt Extension_use Package_inheritance_opt Package_import_opt Package_element_decl_list_opt Invariant_opt **end**;
Package_name	->	*identifier*;
Viewed_with_opt	->	**viewed with** View_element_name_list;
Viewed_with_opt	->	;
View_element_name_list	->	View_element_name_list **,'** View_element_name Position;
View_element_name_list	->	View_element_name Position;
Position	->	**('** Dec **,'** Dec Third_dimension **)'**;
Position	->	;
Third_dimension	->	**,'** Dec;
Third_dimension	->	;
Package_inheritance_opt	->	**inherit** Package_name_list;
Package_inheritance_opt	->	;
Package_name_list	->	Package_name_list **,'** Use_of_constraint_opt Package_name Discriminator_opt;
Package_name_list	->	Use_of_constraint_opt Package_name Discriminator_opt;
Package_import_opt	->	**import** Package_import_list ;
Package_import_opt	->	;
Package_import_list	->	Package_import_list **,'** Package_import_elem;
Package_import_list	->	Package_import_elem;
Package_import_elem	->	Visibility_opt Element_path_opt As_alias **from** Package_name;
Visibility_opt	->	;
Visibility_opt	->	**{'** Visibility **}'**;
As_alias	->	**as** Alias;
As_alias	->	;
Element_path_opt	->	;
Element_path_opt	->	Element_path;
Element_path	->	Element_name;
Element_path	->	Element_path Path Element_name;
Alias	->	Element_name;
Package_element_decl_list_opt	->	**is** Package_element_decl_list;
Package_element_decl_list_opt	->	;
Package_element_decl_list	->	Visibility_opt Package_element_decl;

(UOL 1.2)

Package_element_decl_list -> Package_element_decl_list Visibility_opt Package_element_decl;
Package_element_decl -> Package_declaration;
Package_element_decl -> Interface_declaration;
Package_element_decl -> Class_declaration;
Package_element_decl -> Relation_declaration;
Package_element_decl -> Extension_declaration;
Package_element_decl -> Usecase_abstraction;
Package_element_decl -> Activity_model;
Package_element_decl -> Comment_definition;
Package_element_decl -> Actor_or_exception Light_body;
Package_element_decl -> Component_or_node Ultra_light_body;
Package_element_decl -> Object_declaration
Object_declaration -> Object_name Formal_generics_opt instance of Invariant_opt Element_path Extension_use Viewed_with_opt Object_body_opt **end**;
Object_name -> *identifier*;
Object_body_opt -> ;
Object_body_opt -> **is** Attribute_value_list;
Attribute_value_list -> *identifier* **is** Expression;
Attribute_value_list -> Attribute_value_list ';' identifier **is** Expression;
Comment_definition -> CommentMultiline Attach_or_view;
Attach_or_view -> attached to Name;
Attach_or_view -> viewed with View_element_name_list;
Actor_or_exception -> **actor**;
Actor_or_exception -> **exception**;
Light_body -> Name Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Features Invariant_opt **end**;
Component_or_node -> **component**;
Component_or_node -> **node**;
Ultra_light_body -> Name Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Identifier_list Invariant_opt **end**;
Package_element_decl -> Collaboration_declaration;

Subsystem_declaration -> Subsystem_header Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Package_import_opt Interface_operation_declaration_opt Package_element_decl_list_opt Invariant_opt **end**;
Subsystem_header -> Deferred_opt subsystem Subsystem_name;
Subsystem_name -> *identifier*;

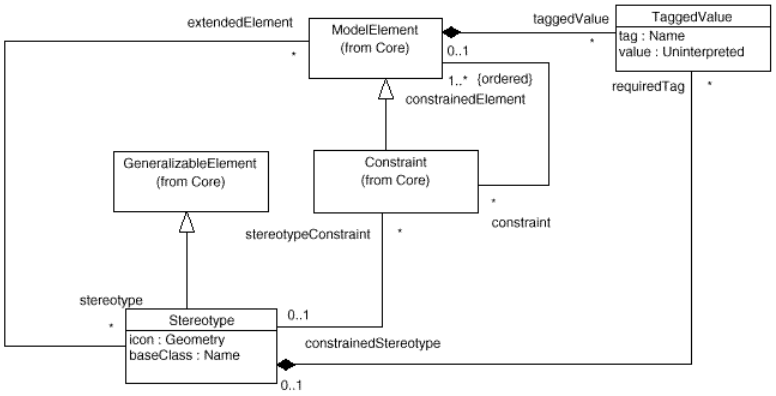

Fig. 4 Extension Mechanisms

Extension_declaration_opt	->	;
Extension_declaration_opt	->	Extension_declaration_opt Extension_declaration;
Extension_declaration	->	Constraint;
Extension_declaration	->	Tagged_values;
Extension_declaration	->	Stereotype;
Extension_use	->	Use_of_stereotype_opt List_use_consr_tag;
Use_of_stereotype_opt	->	Use_of_stereotype;
Use_of_stereotype_opt	->	;
List_use_consr_tag	->	List_use_consr_tag Use_of_tagged_value;
List_use_consr_tag	->	;
Constraint	->	**constraint** Constraint_def_list **end**;
Constraint_def_list	->	Constraint_name **is** '**{**' Constraint_expression '**}**';
Constraint_def_list	->	Constraint_def_list Constraint_name **is** '**{**' Constraint_expression '**}**';
Constraint_expression	->	OCL_expression;
Constraint_expression	->	TextMultiline;
Constraint_name	->	*identifier*;
OCL_expression	->	OCLexpression;
Use_of_constraint	->	**constrained by** '**{**' Constraint_expression '**}**';
Use_of_constraint_opt	->	;
Use_of_constraint_opt	->	Use_of_constraint;
Tagged_values	->	**tag values** Tagged_values_def_list **end**;
Tagged_values_def_list	->	Tagged_values_def;
Tagged_values_def_list	->	Tagged_values_def_list Tagged_values_def;
Tagged_values_def	->	Extension_name;
Tagged_values_def	->	Extension_name **is** Initial_value;
Extension_name	->	*identifier*;
Use_of_tagged_value	->	**with tag values** '**(**' Property_list '**)**';
Property_list	->	Property;
Property_list	->	Property_list '**,**' Property;
Property	->	LT Extension_name GT;
Property	->	LT Extension_name '**,**' Expression GT;
Stereotype	->	**stereotype** Extension_name **of** Base_class Icon_opt Stereotype_parent_opt Stereotype_extension_dec **end**;
Stereotype_extension_dec	->	Stereotype_extension_dec Constraint;
Stereotype_extension_dec	->	Stereotype_extension_dec Tagged_values;
Stereotype_extension_dec	->	;

(UOL 1.2)

Base_class	->	Classifier_name;
Icon_opt	->	;
Icon_opt	->	**viewed as** String;
Stereotype_parent_opt	->	**inherit** Stereotype_parent_list;
Stereotype_parent_opt	->	;
Stereotype_parent_list	->	Use_of_constraint_opt Extension_name Discriminator_opt;
Stereotype_parent_list	->	Stereotype_parent_list **','** Use_of_constraint_opt Extension_name Discriminator_opt;
Use_of_stereotype	->	**stereotyped with** Extension_name;
String	->	*'STRINGstart' 'STRINGliteral' 'STRINGend'*;
identifier	->	*OCLtypeOrName*;
identifier	->	*Anonymous*;

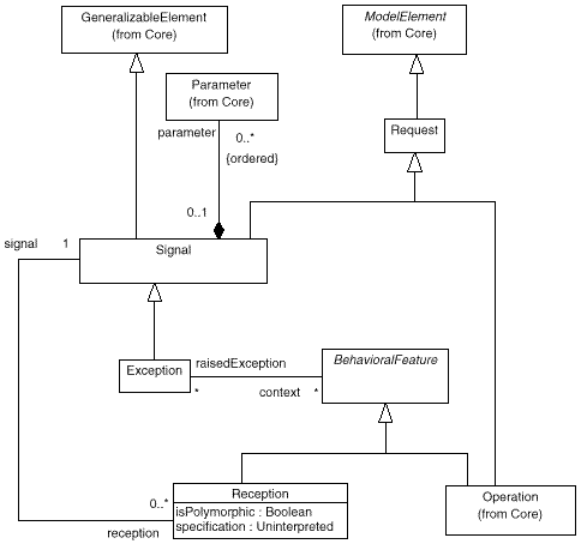

Fig. 5 Common Behavior - Requests

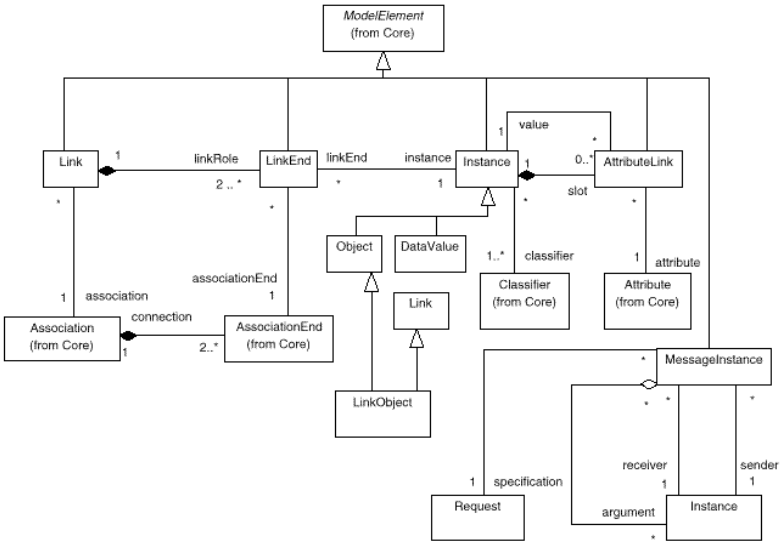

Fig. 6 Common Behavior - Instances and Links

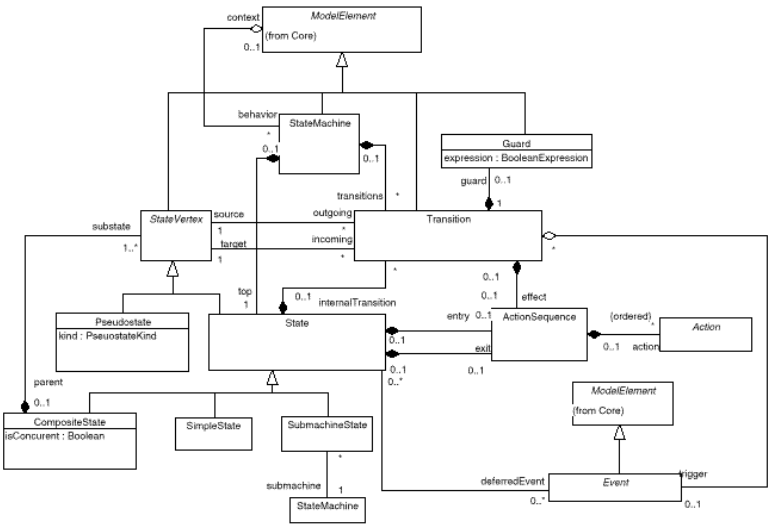

Fig. 7 State Machines - Main

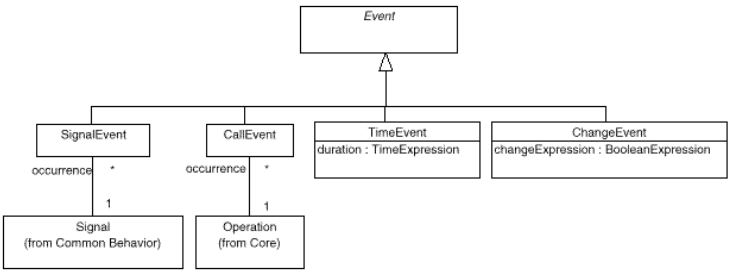

Fig. 8 State Machines - Events

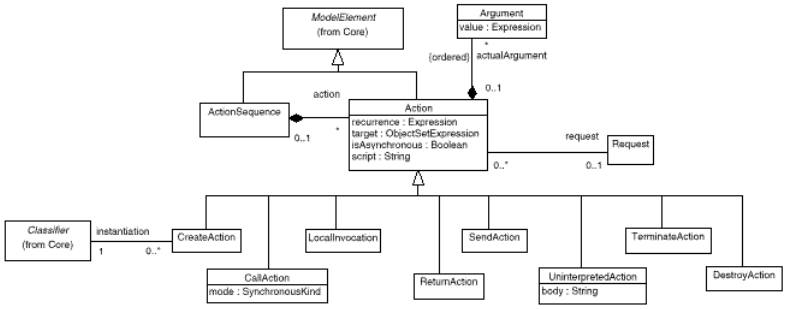

Fig. 9 Common Behavior - Actions

State_machine	->	**state machine** Name Viewed_with_opt Constraint_use_def_opt Machine_body **end**;
Name	->	*identifier*;
Path_name	->	Name;
Path_name	->	Path_name *Path* Name;
Constraint_use_def_opt	->	;
Constraint_use_def_opt	->	Constraint Invariant_opt;
Constraint_use_def_opt	->	Use_of_constraint;
Machine_body	->	Composite_state Transition_list Action_def_list;
State_definition	->	**state** Name Viewed_with_opt Constraint_use_def_opt Action_opt Internal_transition_list Deferred_events_opt;
Action_opt	->	;
Action_opt	->	**actions** EntryExit_action_opt;
EntryExit_action_opt	->	**entry** Action_list;
EntryExit_action_opt	->	**exit** Action_list;
EntryExit_action_opt	->	**entry** Action_list exit Action_list;
Composite_state	->	**composite** State_definition Concurrent_state_list;
Composite_state	->	**composite** State_definition State_list;
Concurrent_state_list	->	**concurrent** State_list ;
Concurrent_state_list	->	Concurrent_state_list concurrent State_list ;
State_list	->	State_kind;
State_list	->	State_list State_kind;
State_kind	->	Simple_state **end**;

State_kind -> Pseudostate **end**;
State_kind -> Submachine **end**;
State_kind -> Machine_body **end**;
Simple_state -> **simple** State_definition;
Pseudostate -> Pseudostate_kind Name Constraint_use_def_opt Action_opt;
Pseudostate_kind -> **deep history**;
Pseudostate_kind -> **shallow history**;
Pseudostate_kind -> **initial**;
Pseudostate_kind -> **final**;
Pseudostate_kind -> **join**;
Pseudostate_kind -> **fork**;
Pseudostate_kind -> **branch**;
Submachine -> **submachine** Name Viewed_with_opt Constraint_use_def_opt Machine_body;
Internal_transition_list -> ;
Internal_transition_list -> Internal_transition_list **transition** When_or_after Trigger_opt Action_sequence_opt;
When_or_after -> **when** Guard_expression ;
When_or_after -> **after** Time_expression ;
Time_expression -> *Integer_constant* String;
Deferred_events_opt -> ;
Deferred_events_opt -> **deferred** Deferred_event_list;
Deferred_event_list -> **event** Name;
Deferred_event_list -> Deferred_event_list **event** Name;
Guard_expression -> Expression;
Guard_expression -> *TextMultiline*;
Trigger_opt -> ;
Trigger_opt -> Trigger_expression;
Trigger_expression -> **call** Operation_use;
Trigger_expression -> **trigger** Signal_or_time_or_change;
Signal_or_time_or_change -> Signal_definition;
Signal_or_time_or_change -> **after** Time_expression;
Signal_or_time_or_change -> **when** Boolean_expression;
Boolean_expression -> TextMultiline;
Transition_definition -> **transition** Name **from** Path_name_or_initial **to** Path_name_or_final Guard_expression_opt Trigger_opt Action_sequence_opt;
Path_name_or_initial -> Path_name;
Path_name_or_initial -> **initial**;
Path_name_or_final -> Path_name;
Path_name_or_final -> **final**;
Transition_list -> ;
Transition_list -> Transition_list Transition_definition;
Guard_expression_opt -> ;
Guard_expression_opt -> When_or_after;
Action_sequence_opt -> ;
Action_sequence_opt -> **actions** Action_list;
Action_list -> *identifier*;
Action_list -> Action_list **,** *identifier*;
Action_def_list -> ;
Action_def_list -> Action_def_list Action_definition;
Action_definition -> Synchronous_opt **action** Name Recurrence_opt Script_opt Object_set_expression_opt Request_opt Action_kind;
Script_opt -> ;
Script_opt -> String;

(UOL 1.2)

Synchronous_opt -> ;
Synchronous_opt -> **synchronous**;
Recurrence_opt -> ;
Recurrence_opt -> **'('** Expression **')'**;
Object_set_expression_opt -> ;
Object_set_expression_opt -> **to** Object_set_expression;
Object_set_expression -> Name;
Object_set_expression -> Object_set_expression **','** Name;
Request_opt -> ;
Request_opt -> **request** Operation_or_signal;
Operation_or_signal -> Operation_use;
Operation_or_signal -> Signal_definition;
Signal_definition -> **signal** Name Reception_opt Exception_opt;
Reception_opt -> ;
Reception_opt -> **to** Name_comma_list;
Name_comma_list -> Name;
Name_comma_list -> Name_comma_list **','** Name;
Exception_opt -> ;
Exception_opt -> Exception_list;
Exception_list -> **raise** Exception_use;
Exception_list -> Exception_list **raise** Exception_use;
Exception_use -> Name **from** Name_comma_list;
Action_kind -> ;
Action_kind -> **call** Operation_use;
Action_kind -> **creates** *identifier*;
Action_kind -> *TextMultiline*;
Operation_use -> Name **'.'** Name **'('** Expression_list **')'**;

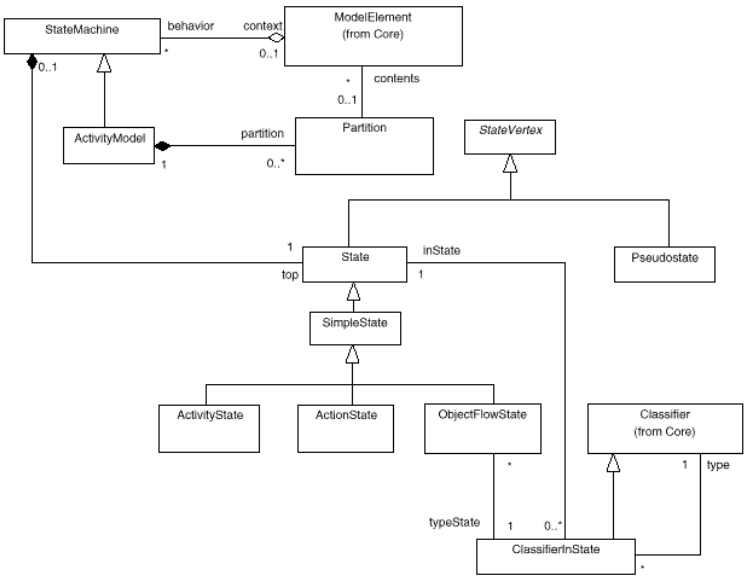

Fig. 10 Activity Models

Activity_model	->	**activity** Name Viewed_with_opt Constraint_use_def_opt Activity_body **end**;
Activity_body	->	Activity_state Transition_list Action_def_list;
Activity_state	->	**composite** State_definition Partition_opt Act_concurrent_state_list;
Activity_state	->	**composite** State_definition Partition_opt Act_state_list;
Act_concurrent_state_list	->	**concurrent** Act_state_list ;
Act_concurrent_state_list	->	Act_concurrent_state_list concurrent Act_state_list;
Act_state_list	->	Act_state_kind Partition_opt **end**;
Act_state_list	->	Act_state_list Act_state_kind Partition_opt **end**;
Partition_opt	->	;
Partition_opt	->	**partitioned in** Name;
Act_state_kind	->	Activity_body;
Act_state_kind	->	Act_simple_state ;
Act_state_kind	->	Pseudostate ;
Act_state_kind	->	Subactivity ;
Act_simple_state	->	Action_state;
Act_simple_state	->	Object_flow_state;
Action_state	->	**state** Name Viewed_with_opt Constraint_use_def_opt Action_opt;
Object_flow_state	->	State_definition flow Name **'['** Name **']'** Invariant_opt;
Subactivity	->	**subactivity** Name Viewed_with_opt Constraint_use_def_opt Activity_body;

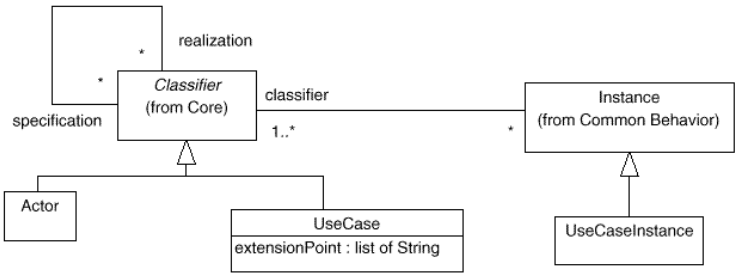

Fig. 11 Use Cases

Usecase_abstraction	->	Usecase_definition;
Usecase_abstraction	->	Usecase_instance;
Usecase_abstraction	->	Usecase_extension;
Usecase_definition	->	**usecase** Name Formal_generics_opt Usecase_inherit_opt Usecase_use_opt Usecase_actor_opt Features_attrib_or_Oper TextMultiline_opt Usecase_alt_course_opt Usecase_extension_point_opt **end**;
Usecase_instance	->	**usecase** Name Formal_arguments_opt Type_mark_opt **is** Usecase_method_list **end**;
Usecase_extension	->	**extend** Usecase_path with Usecase_path_list **in** Extension_point;
Usecase_method_list	->	Usecase_method ;
Usecase_method_list	->	Usecase_method_list Usecase_method ;
Usecase_method	->	*identifier* **is** Expression;
Usecase_inherit_opt	->	;
Usecase_inherit_opt	->	**inherit** Name_inherit_list;
Usecase_use_opt	->	;
Usecase_use_opt	->	**use** Identifier_list;
Usecase_actor_opt	->	;
Usecase_actor_opt	->	**actor** Identifier_list;
TextMultiline_opt	->	;
TextMultiline_opt	->	TextMultiline;
Usecase_extension_point_opt	->	;
Usecase_extension_point_opt	->	**extension in** Extension_point_list;
Extension_point	->	String;
Extension_point_list	->	String_list;
Name_inherit_list	->	Use_of_constraint_opt Name Discriminator_opt;
Name_inherit_list	->	Name_inherit_list **,'** Use_of_constraint_opt Name Discriminator_opt;
Usecase_alt_course_opt	->	;
Usecase_alt_course_opt	->	**alternative course** TextMultiline;
Usecase_path	->	Name;
Usecase_path	->	Usecase_path Path Name;
Usecase_path_list	->	Usecase_path;
Usecase_path_list	->	Usecase_path_list **,'** Usecase_path;
String_list	->	String;
String_list	->	String_list **,'** String;

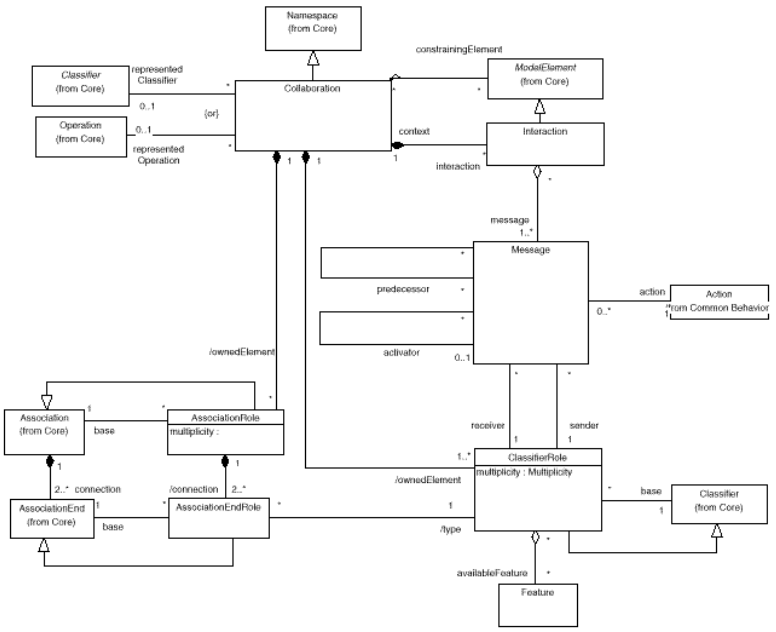

Fig. 12 Collaborations

Collaboration_declaration	->	**collaboration** Collaboration_name Formal_generics_opt Viewed_with_opt Implementation_of_opt Class_or_intf_or_rel_decl_list Action_def_list Message_list_opt **end**;
Collaboration_name	->	*identifier*;
Implementation_of_opt	->	;
Implementation_of_opt	->	**implements** Classifier_or_operation;
Classifier_or_operation	->	Element_name;
Class_or_intf_or_rel_decl_list	->	;
Class_or_intf_or_rel_decl_list	->	Class_or_intf_or_rel_decl_list Class_declaration;
Class_or_intf_or_rel_decl_list	->	Class_or_intf_or_rel_decl_list Element_name;
Class_or_intf_or_rel_decl_list	->	Class_or_intf_or_rel_decl_list Interface_declaration;
Class_or_intf_or_rel_decl_list	->	Class_or_intf_or_rel_decl_list Relation_declaration;
Message_list_opt	->	Message_list;
Message_list_opt	->	;
Message_list	->	Message_list Message;
Message_list	->	Message;
Message	->	**actions** Action_list **to** Classifier_name **from** Classifier_name;
Classifier_name	->	Element_path;
Formal_generics_opt	->	;
Formal_generics_opt	->	Formal_generics;
Formal_generics	->	'**[**' Formal_generic_list '**]**';
Formal_generic_list	->	Formal_generic;
Formal_generic_list	->	Formal_generic_list '**,**' Formal_generic;
Formal_generic	->	Element_name Invariant_opt;
Element_name	->	*identifier*;
Expression	->	Call;
Expression	->	Operator_expression;

(UOL 1.2)

Expression -> Equality;
Expression -> Manifest_constant;
Expression -> Manifest_array;
Call -> Parenthesized_qualifier_opt Call_chain;
Call_chain -> Unqualified_call;
Call_chain -> Call_chain '**.**' Unqualified_call;
Parenthesized_qualifier_opt -> ;
Parenthesized_qualifier_opt -> Parenthesized_qualifier;
Parenthesized_qualifier -> Parenthesized '**.**';
Parenthesized -> '**(**' Expression '**)**';
Unqualified_call -> Entity Actuals_opt;
Entity -> *identifier*;
Entity -> **result**;
Entity -> **current**;
Actuals_opt -> ;
Actuals_opt -> Actuals;
Actuals -> '**(**' Actual_list '**)**';
Actual_list -> Actual;
Actual_list -> Actual_list '**,**' Actual;
Actual -> Expression;
Operator_expression -> Parenthesized;
Operator_expression -> Unary_expression;
Operator_expression -> Binary_expression;
Unary_expression -> Unary Expression;
Unary -> **not**;
Unary -> '+';
Unary -> '-';
Binary_expression -> Expression Binary Expression;
Binary -> '+';
Binary -> '-';
Binary -> '*****';
Binary -> '/';
Binary -> '<';
Binary -> '>';
Binary -> *LTE*;
Binary -> *GTE*;
Binary -> *DD*;
Binary -> *DS*;
Binary -> '^';
Binary -> **and**;
Binary -> **or**;
Binary -> **xor**;
Binary -> **implies**;
Manifest_constant -> Boolean_constant;
Manifest_constant -> Character_constant;
Manifest_constant -> Integer_constant;
Manifest_constant -> Float_constant;
Manifest_constant -> String;
Boolean_constant -> **true**;
Boolean_constant -> **false**;
Float_constant -> *'FLOATINGconstant0'*;

Manifest_array -> *ANGLEBL* Expression_list *ANGLERR*;
Expression_list -> Expression;
Expression_list -> Expression_list **','** Expression;
Comparision -> **'='**;
Comparision -> *NOTEQUAL*;
Equality -> Expression Comparision Expression;
Inheritance -> **inherit** Parent_list;
Parent -> Use_of_constraint_opt Class_type Discriminator_opt;
Parent_list -> Parent;
Parent_list -> Parent_list **';'** Parent;
Discriminator_opt -> ;
Discriminator_opt -> **'('** Name **')'**;
Class_type -> Class_name Actual_generics_opt;
Actual_generics_opt -> ;
Actual_generics_opt -> Actual_generics;
Actual_generics -> **'['** Type_list **']'**;
Type_list -> Type;
Type_list -> Type_list **','** Type;
Type -> Class_type;
Type -> Class_type_expanded;
Type -> Anchored;
Type -> Bit_type;
Class_type_expanded -> **expanded** Class_type;
Anchored -> **like** Anchor;
Anchor -> *identifier*;
Anchor -> **current**;
Bit_type -> **BIT** Constant;
Constant -> Manifest_constant;
Constant -> Entity;
Rename_opt -> ;
Rename_opt -> Rename;
Rename -> **rename** Rename_list;
Rename_list -> Rename_pair;
Rename_list -> Rename_list **','** Rename_pair;
Rename_pair -> Feature_name **as** Feature_name;
Feature_name -> *identifier*;
Feature_name -> Prefix;
Feature_name -> Infix;
Infix -> **infix '('** Infix_operator **')'**;
Infix_operator -> Binary;
Infix_operator -> *identifier*;
Prefix -> **prefix '('** Prefix_operator **')'**;
Prefix_operator -> Unary *identifier*;
New_exports_opt -> ;
New_exports_opt -> New_exports;
New_export_item -> Clients Feature_set;
New_export_list -> New_export_item;
New_export_list -> New_export_list **';'** New_export_item;
New_exports -> **export** New_export_list;
Class_list -> Class_name;
Class_list -> Class_list **','** Class_name;

(UOL 1.2)

Clients	->	**'{'** Class_list **'}'**;
Feature_set	->	Identifier_list;
Feature_set	->	**all**;
Undefine_opt	->	;
Undefine_opt	->	Undefine;
Undefine	->	**undefine** Feature_list;
Select_opt	->	;
Select_opt	->	Select;
Select	->	**select** Feature_list;

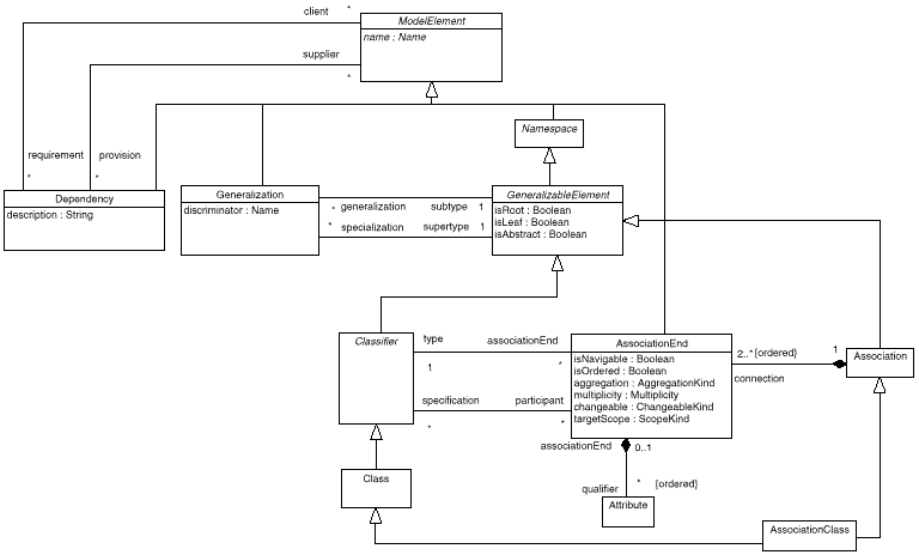

Fig. 13 Core Package - Relationships

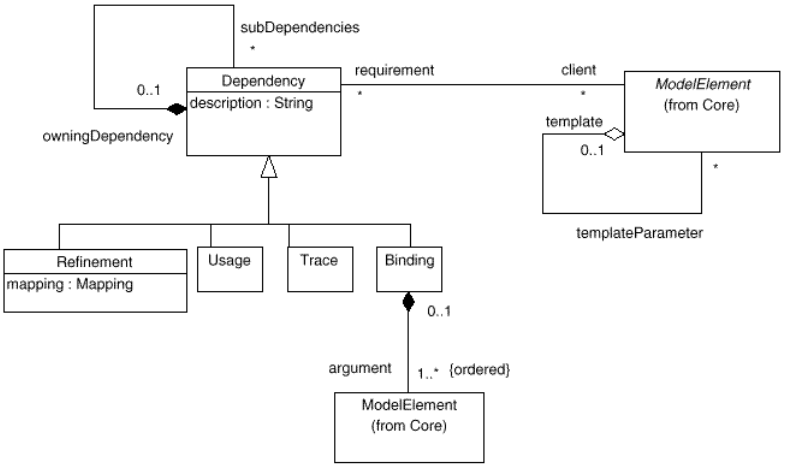

Fig. 14 Auxiliary Elements - Dependencies and Templates

Relation_declaration	->	**relation** Relation_name Extension_declaration_opt Extension_use Relation_inheritance_opt Link_list_opt Features_attrib_or_Oper Invariant_opt **end**;
Relation_inheritance_opt	->	;
Relation_inheritance_opt	->	Relation_inheritance;
Link_list_opt	->	;
Link_list_opt	->	Link_list;
Relation_name	->	*identifier*;
Relation_inheritance	->	**inherit** Parent_relation_list;
Parent_relation_list	->	Parent_relation;
Parent_relation_list	->	Parent_relation_list **';'** Parent_relation;
Parent_relation	->	Use_of_constraint_opt Relation_type Relation_feature_adaptation_opt;
Relation_feature_adaptation_opt	->	;
Relation_feature_adaptation_opt	->	**adaptation** Relation_feature_adaptation;
Relation_type	->	Relation_path;
Relation_path	->	Element_path;
Relation_feature_adaptation	->	Rename_opt New_exports_opt Undefine_opt Relation_redefine_opt Select_opt **end**;
Relation_redefine_opt	->	;
Relation_redefine_opt	->	Relation_redefine;
Relation_redefine	->	**redefine** Feature_or_redef;
Feature_or_redef	->	Feature_list;
Feature_or_redef	->	Redefine_with_list;
Redefine_with_list	->	Redefine_pair ;
Redefine_with_list	->	Redefine_with_list **','** Redefine_pair;
Redefine_pair	->	Feature_name with Feature_name;
Link_list	->	**link** Type_or_dependency ;
Assoc_entity_opt	->	;
Assoc_entity_opt	->	**with** Classifier_name;
Type_or_dependency	->	Type_link_two_list Assoc_entity_opt;
Type_or_dependency	->	Dependency_list Dependency_description_opt;
Type_link_two_list	->	Type_link Cardinality_opt **','** Type_link_list;
Cardinality_opt	->	;
Cardinality_opt	->	Cardinality2;
Cardinality2	->	**'['** Cardinality **']'**;

(UOL 1.2)

```
Cardinality                 ->  Range_list Range_last;
Range_last                  ->  Range;
Range_last                  ->  Int_or_star;
Range_mid                   ->  Range ',' ;
Range_mid                   ->  Integer_constant ',';
Range_list                  ->  ;
Range_list                  ->  Range_list Range_mid;
Int_or_star                 ->  Integer_constant;
Int_or_star                 ->  Star;
Type_link_list              ->  Type_link Cardinality_opt;
Type_link_list              ->  Type_link_list ',' Type_link Cardinality_opt;
Type_link                   ->  Classifier_name;
Dependency_list             ->  Dependency;
Dependency_list             ->  Dependency_list ',' Dependency;
Dependency                  ->  Element_path to Element_path;
Dependency_description_opt  ->  is TextMultiline;
Dependency_description_opt  ->  ;
```

C UOL code for the MOF meta-meta-model

The following model is the MOF specification writed in UOL

```
model MOF

   --
   -- Package Reflective
   --
   package Reflective
   is {any}

   --
   -- Exceptions
   --
   exception AlreadyCreated
   end -- exception AlreadyCreated

   exception BadPosition
   end -- exception BadPosition

   exception ConstraintError
      feature {any}
        constraint_designator : RefObject;
        offending_values : ErroneousValues;
        explanation_text : stringt
      end -- feature
   end -- exception ConstraintError

   exception InvalidDesignator
      feature {any}
        designator : DesignatorType;
        element_kind : string
      end -- feature
   end -- exception InvalidDesignator

   exception InvalidLink
   end -- exception InvalidLink

   exception InvalidObject
      feature {any}
        designator : DesignatorType;
        obj : RefObject;
        type_expected : TypeCode
      end -- feature
   end -- exception InvalidObject
```

(UOL 1.2)

```
  exception InvalidValue
    feature {any}
      designator : DesignatorType;
      element_kind : string;
      value : ValueType;
      type_expected : TypeCode
    end -- feature
  end -- exception InvalidValue

  exception MissingParameter
    feature {any}
      designator : DesignatorType
    end -- feature
  end -- exception MissingParameter

  exception NotFound
  end -- exception NotFound

  exception NotSet
  end -- exception NotSet

  exception PragmaNotPresent
  end -- exception PragmaNotPresent

  exception PragmaNotSupported
  end -- exception PragmaNotSupported

  exception OtherException
    feature {any}
      exception_designator : DesignatorType;
      exception_values : ValuesType
    end -- feature
  end -- exception OtherException

  exception SemanticError
    feature {any}
      error_name : string;
      offending_values : ErroneousValues;
      explanation_text : string
    end -- feature
  end -- exception SemanticError

  exception StructuralError
    feature {any}
      violations : StructuralViolationSet
    end -- feature
  end -- exception StructuralError

  --
  -- Interfaces
  --
  interface RefBaseObject
    feature {any}
      deferred metaObject():DesignatorType
        is text "A reflective object reveals its nature through the
          meta-object returned by the metaObject operation. The CORBA
```

```
          object type of the returned meta-object depends on the usage
          conventions which govern this use of the Reflective
          interfaces.
          The operation may return an invalid (nil) object reference if
          no descriptive meta-objects for the object are available.";
      deferred itself(in otherObject : RefBaseObject):boolean
        is text "The itself operation tests whether this object and
          another object provided an argument are identical. The
          precise semantics of this operation depends on the usage
          conventions.";
      deferred repository_container(): RefBaseObject
        is text "The repository_container operation returns the
          repository container for an object. The precise semantics of
          this operation depends on the usage conventions."
    end -- feature
  end -- interface RefBaseObject

  interface RefObject
    inherit RefBaseObject
    feature {any}
      deferred isInstanceOf(in someType : DesignerType; in
        considerSubtypes : boolean):boolean
        is text "This operation tests whether the RefObject is an
          instance of the type described by the someType meta-object.
          If the considerSubtypes argument is true, an object whose
          type is a subtype of the type described by someType will be
          considered as an instance of the type. When the type system
          does not have a notion of subtyping, the considerSubtypes
          argument is ignored.";
      deferred createInstance(in args[0..*] : ValueType):RefObject
        is text "This createInstance operation creates a new instance of
          the RefObject's most derived type. The args list gives the
          initialization parameters for the new object as required. If
          it is inappropriate to create an object (e.g. the type is
          abstract) or the arguments are incorrect, exceptions will be
          raised.
          If a meta-model supports multiple factory operations for a
          given type, the usage conventions should define how to decide
          which factory operation to use based args provided. (The
          invokeOperation operation is also available as a possible
          alternative.)";
      deferred allObjects(in includeSubtypes : boolean):RefObject
        is text "The allObjects operation returns the collection of all
          known instances of this objects model type that are present
          in the context. If the model has a well defined notion of
          subtyping and includeSubtypes is true, the resulting
          collection will also include the object instances that are
          subtypes. The specific definition of what constitutes the
          object's context should be defined in the usage
          conventions.";
      deferred setValue(in features : DesignatorType; in value:
        ValueType)
        is text "The setValue operation assigns a new value to a feature
          (e.g. attribute or object link) of the object as allowed by
          the object's type and its meta-model. The precise meaning of
          assignment, the kinds of features it applies to, and the set
```

```
        of exceptions that can be raised are dependent on the usage
        conventions.
        Standard exceptions are raised if the feature is not a known
        changeable feature of the object, if the type of value is
        incorrect, or if the update would violate structural or
        semantic integrity constraints.
        For meta-models that support collection valued features, the
        setValue operation replaces the current collection value with
        a new one. More specialized operations are defined below for
        updating the individual values of a feature collection.";
    deferred value(in features : DesignatorType): ValueType
      is text "The value operation fetches the current value of some
        feature. The precise method of obtaining the value returned
        and the set of exceptions that can be raised are dependent on
        the usage conventions.
        Standard exceptions are raised if the feature is not known,
        is not appropriate for fetching, or if some semantic error
        was detected during the access attempt.";
    deferred addValue(in features : DesignatorType)
      is text "The addValue operation allows the user to add a new
        component value into a feature that is a collection. The
        precise semantics of the add operation depends on the usage
        conventions.
        Standard exceptions are raised if the feature is not a known
        changeable collection, if the value has the wrong type, or if
        the add operation would violate structural or semantic
        integrity rules.";
    deferred addValueBefore(in features : DesignatorType;
      in value: ValueType;
      in existingValue : ValueType)
      is text "The addValueBefore operation is similar to the addValue
        operation, except that the user can control the point at
        which the new value is inserted into the collection. The
        operation is intended to be used to update ordered
        collections with unique elements. The precise semantics of
        the insertion (including the position) depend on the usage
        conventions.
        NotFound should be raised if existingValue is not a member of
        the collection. Standard exceptions are raised for other
        errors as described above.";
    deferred addValueAt(in features : DesignatorType;
      in value: ValueType;
      in position : long)
      is text "The addValueAt operation is similar to the
        addValueBefore operation, except that the user specifies the
        insertion point using an index. The operation is intended to
        be used to update ordered collections with non-unique
        elements. The precise semantics of the insertion (including
        the position) depend on the usage conventions.
        If the position value does not denote a valid location in the
        collection, InvalidPosition should be raised. Standard
        exceptions are raised for other errors as described above.";
    deferred modifyValue(in features : DesignatorType;
      in existingValue: ValueType;
      in newValue: ValueType)
```

```
    is text "The modifyValue operation replaces one value in a
      feature collection with another one. The operation is
      intended to be used to update collections with unique
      elements. The precise semantics of the update depend on the
      usage conventions.
      Standard exceptions are raised if the feature is not a known
      changeable collection, if the newValue has the wrong type or
      if the update would violate structural or semantic integrity
      constraints. Similarly, the NotFound exception should be
      raised if the existingValue is not in the collection.";
  deferred modifyValueAt(in features : DesignatorType;
    in newValue: ValueType;
    in position: long)
    is text "The modifyValueAt operation is similar to the
      modifyValue operation,except that the existing value to be
      modified is specified by positionrather than by value. The
      operation is intended to be used to updatecollections with
      non-unique elements. The precise semantics of themodification
      (including the position) depend on the usage conventions.
      Standard exceptions are raised if the feature is not a known
      changeable collection, if the newValue has the wrong type or
      if the update would violate structural or semantic integrity
      constraints. If position does not denote a valid location in
      the collection, InvalidPosition will be raised.";
  deferred removeValue(in features : DesignatorType;
    in existingValue: ValueType)
    is text "The removeValue operation is used to remove a value
      from a feature collection. The operation is intended to be
      used with collections with unique elements. The precise
      semantics of the deletion depend on the usage conventions.
      Standard exceptions are raised if the feature is not a known
      changeable collection, if the existingValue has the wrong
      type, or if the removal would violate structural or semantic
      integrity constraints. If the existingValue is not present in
      the collection, NotFound will be raised.";
  deferred removeValueAt(in features : DesignatorType;
    in position : long)
    is text "The removeValueAt operation is similar to the
      removeValue operation except that the value to be removed is
      specified by an index. The operation allows removal from
      collections with non-unique elements. The precise semantics
      of the deletion depend on the usage conventions.
      Standard exceptions are raised if the feature is not a known
      changeable collection, or if the removal would violate
      structural or semantic integrity constraints. If the position
      is out of range for the collection, InvalidPosition will be
      raised.";
  deferred invokeOperation(inout args[0..*]: ValueType):ValueType
    is text "The invokeOperation operation causes a model-specific
      operation to be invoked with the arguments supplied. The
      conventions for encoding and passing arguments, results and
      exception values should be defined by the usage conventions.
      Exceptions are raised if requestedOperation does not
      designate an operation for the object, or the number or types
      of the args are not as required. If the invoked operation
```

```
            raises an exception, this is mapped onto OtherException, with
            exception parameters as defined by the usage conventions."
      end -- feature
    end -- interface RefObject

    interface RefAssociation
      inherit RefBaseObject
      feature {any}
        deferred exists(in someLink : Link):boolean
          is text "The exists operation tests whether the supplied link is
            a member of this association. The precise meaning of link
            existence depends on the usage conventions The ends of as
            Link must be given in the order defined by the association's
            defining model. The InvalidLink exception will be raised if
            the CORBA object types of the two ends of someLink do not
            match the defining model.";
        deferred query(in queryEnd : DesignatorType; in queryObject :
          RefObject):RefObject
          is text "Given an object value for one end of a link, the query
            operation returns the set of objects linked to it in the
            association. The precise semantics of linked to are
            determined by the usage conventions.
            The operation should raise InvalidDesignator if the queryEnd
            designator is not a known association end, or InvalidObject
            if the queryObject is invalid, or does not have an
            appropriate type.";
        deferred addLink(in newLink : Link)
          is text "The addLink operation adds newLink into the set of
           links that comprise the association. If the association has
           ordering semantics, the link should be inserted at the last
           position for that ordering. As with other operations, the
           precise semantics of link insertion depends on the usage
           conventions.
           The addLink operation should raise InvalidLink if the newLink
           object references are invalid or have the wrong type, and other
           standard exceptions if adding the link violates structural or
           semantic integrity constraints.";
        deferred addLinkBefore(in newLink : Link;
          in positionEnd : DesignatorType;
          in positionValue : RefObject )
          is text "The addLinkBefore operation adds newLink at a
           particular position in the association. The precise semantics
           of link insertion (including the position at which the link is
           inserted) depends on the usage conventions. The operation
           should raise standard exceptions if the newLink object
           references are invalid or have the wrong type, if either of
           positionEnd or positionValue is inappropriate, if positionValue
           is not found in an existing link, or if adding the link
           violates structural or semantic constraints.";
        deferred modifyLink(in existingLink : Link;
          in positionEnd : DesignatorType;
          in positionValue : RefObject )
          is text "The modifyLink operations replaces one or other of the
           end values for an existing link with a new value. The precise
           semantics of the update depend on the usage conventions.
           However, it is recommended that the update be modeled as a
```

```
          single logical operation rather separate deleteLink and an
          addLink operations. Standard exceptions should be raised if the
          arguments are incorrect or the modification violates structural
          or semantic constraints.";
      deferred removeLink(in existingLink : Link)
        is text "The removeLink operation removes the existingLink from
          the association. The precise semantics of link removal depend
          on the usage conventions. Standard exceptions should be raised
          if existingLink is invalid, does not exist, or if removing it
          would violate structural or semantic constraints."
    end -- feature
  end -- interface RefAssociation

  interface RefPackage
    inherit RefBaseObject
    feature {any}
      deferred getClassRef(in type : DesignatorType): RefObject
        is text "The getClassRef operation returns the object reference
          for a class proxy object for a given type in the package's
          model. The precise type and purpose of the proxy object depends
          on the usage conventions. If the meta-model does not require
          the use of a class proxy object for the type, then an invalid
          object reference should be returned. The InvalidDesignator
          exception should be raised if type does not denote a type in
          the package's meta-object description.";
      deferred getAssociation(in association: DesignatorType):
        RefAssociation
        is text "The getAssociation operation returns a object reference
          for an association object that holds links between the types in
          the context of the package instance. The precise type and
          semantics of the association object returned depend on the
          usage conventions. The InvalidDesignator exception should be
          raised if association does not denote a valid association in
          the package's meta-object description.";
      deferred getNestedPackage(in nestedPackage: DesignatorType):
        RefPackage
        is text "The getSubpackage operation returns a package instance
          object reference for a package nested with the current one. The
          precise type and semantics of the package object returned
          depend on the usage conventions. The InvalidDesignator
          exception should be raised if nestedPackage does not denote a
          nested package in the package's meta-object description."
    end -- feature
  end -- interface RefPackage

  end -- package Reflective

  --
  -- Package ModelPackage
  --
  package ModelPackage
    import from Reflective
    is {any}

      --
      -- Tag Values
```

(UOL 1.2)

```
--
tag values
  AssociationEndName
  IsNavigable
  Aggregation
end

--
-- Exceptions
--
exception BadKindString
end -- exception BadKindString

exception FormatNotSupported
end -- exception FormatNotSupported

exception IllformedExternalizedObject
  feature {any}
    explanation : string
  end -- feature
end -- exception IllformedExternalizedObject

exception NameNotFound
  feature {any}
    name : NameType
  end -- feature
end -- exception NameNotFound

exception NameNotResolved
  feature {any}
    missing_name : NameType;
    resolved_part : NameTypeList
  end -- feature
end -- exception NamenotResolved

exception ObjectNotExternalizable
  feature {any}
    explanation : string
  end -- feature
end -- exception ObjectNotExternalizable

--
-- Classes
--
deferred class ModelElement
  inherit Reflective::RefObject
  feature {none}
    name : NameType;
    annotation : AnnotationType;
    qualifiedName constrained by
    {not self.oclIsTypeOf(Package) implies self.container->size = 1
     and self. isTypeOf(Package) implies self.container->size < 2 }
    : NameTypeSeq
  end -- feature
  feature {any}
```

```
deferred verify(in depth:DepthType; out violations[0..*]:
 ConstraintViolation):VerificationResultKind
 is text " Each ModelElement is capable of checking its own
   correctness, as defined by the inherent properties of
   metamodels described in this specification, and constraints
   that hold over the ModelElement. The client of the operation
   specifies whether the operation should propagate to any
   ModelElements that this ModelElement might contain (if it is
   capable of containing elements), or whether it should return
   after only checking itself. The verify operation checks
   inherent constraints on the object and its attributes plus
   any constraints contained by the object.. The operation
   returns valid if all verification checks passed; otherwise it
   returns invalid. A parameter returns representations of any
   constraint violations detected. If the operation returns
   invalid, this parameter must not be empty. When the depth
   argument is deep, and this element (and, by definition, all
   its dependent elements) are published, the operation returns
   published.";
deferred isFrozen():boolean
 is text " Reports the freeze status of a ModelElement. A
   ModelElement, at any particular time, is either frozen or not
   frozen. All ModelElements of a published model are
   permanently frozen.";
deferred setName(in newName:NameType)
   constrained by { text "constraints are defined on the
    Namespace-Contains-ModelElement association, which may be
    violated as a consequence of this operation"}
 is text "Provides the only means of changing the name of an
   existing ModelElement.";
deferred findRequiredElements(in kinds[1..*]:DependencyKind; in
   recursive:boolean):ModelElement
 is text "Supports selecting a subset of the dependents, based
   on the kind of dependency. In addition, this operation can
   return the transitive closure of all required elements, when
   the argument recursive is set to true. Setting the kinds
   parameter to all, the operation can take advantage of the
   recursive find, without restricting to specific dependency
   kinds. This operation is more powerful than the
   requiredElements reference.";
deferred isVisible(in otherElement:ModelElement): boolean
 is text "Determines whether the supplied ModelElement is
   visible to this ModelElement. If necessary, the operation
   searches through the transitive closure of the ModelElements
   that are visible to this ModelElement. For a complete
   description on visibility rules, and their implication on
   building models, see Section 6.5 of MOF specification
   (ad970804) Rules of Importing and Visibility.";
deferred isRequiredBecause(out reason:DependencyKind):boolean
 is text "Determines whether the supplied ModelElement is
   required by this ModelElement, according to the DependsOn
   derived association. If the supplied ModelElement is required
   by this ModelElement, this operation returns the kind of
   dependency. If the supplied ModelElement is indirectly
   required by this ModelElement through transitivity, the
   dependency is categorized as indirect. If the supplied
```

```
          ModelElement is not required by this ModelElement, the reason
          output argument is an empty string.";
        deferred removeElement()
         is text "Removes this ModelElement from existence (from the
          perspective of the modeler). In doing so, this operation
          removes the Link between the ModelElement and its container.
          The operation also removes the Link between this ModelElement
          and its class-level representation (which keeps track of all
          of its instances in the MofRepository). For each attribute of
          this element which contains one or more ModelElements, this
          removeElement operation is invoked. For each reference of
          this element, the corresponding Link is removed, and the
          referenced element may have the removeElement operation
          invoked, depending on the defined aggregation of the
          reference (See Section 6.2.3 of MOF specification (ad970804)
          Aggregations). If this ModelElement is linked to an element
          in another ModelRepository, the removeElement operation could
          result in a dangling Link. See Section 6.7.2 of MOF
          specification (ad970804) Defining Models across Repositories
          for a discussion of such issues";
        deferred copyElement(in container:Namespace):Modelelement
         is text"Supports the copying of a model. This operation is not
          intended for users of the MOF."
    end -- feature
    constrained by { text "The attribute values of a ModelElement
     which is frozen cannot be have the values of its attributes
     changed. This constraint applies to all subtypes of ModelElement,
     and constrains all their attributes as well. A Link cannot be
     changed or removed if both a ModelElement which is frozen is
     connected and that ModelElement has a Reference corresponding to
     that Link; also a new Link cannot be created which connects a
     ModelElement which is frozen if the ModelElement has a Reference
     corresponding to the candidate new Link."}
end -- class ModelElement

deferred class Namespace
    inherit ModelElement
    feature {any}
      deferred lookupElement(in name: NameType):ModelElement
       is text "Searches for a contained element with the designated
         name. If the operation finds a ModelElement, referenced by
         container, which has a name equal to the supplied name, the
         operation returns that element. Otherwise, no element is
         returned.";
      deferred resolveQualifiedName(in qualifiedName[1..*]:NameType)
         raise NameNotResolved
       is text "Searches for a contained element with the designated
         qualified name. Depth of the target ModelElement within the
         containment hierarchy, with respect to this Namespace, must
         match the number of names in the NameType list. return type:
         ModelElement (exactly one). If no element is found, an
         exception is raised.";
      deferred nameIsValid(in proposedName: NameType):boolean
       is text "Determines whether the specified name can be used
         within the namespace. The operation verifies that the name is
         valid according to the constraint on names and that the name
```

```
        is not already used within the namespace. The operation
        returns true only if both conditions are met. The nameIsValid
        operation supports renaming ModelElements";
      deferred findElementsByType(in ofType:Class; in
        includeSubtypes:boolean):ModelElement
      is text "Returns all the ModelElements identified by the
       contents reference defined for this Namespace that are of the
       type supplied. The returned list of ModelElements is a subset
       of the ModelElements contained by this Namespace. This
       operation can either return only those ModelElements that
       exactly match the specified type or those ModelElements that
       are instances of the specified type and one or more of its
       subtypes. Because ModelElement is an abstract type, invoking
       this operation with the ofType argument specified as
       ModelElement and the includeSubtypes argument set to false
       returns an empty list. Because ModelElement is the base type
       for all instances which can be contained by a Namespace,
       Invoking the operation with the ofType argument specified as
       ModelElement and includeSubtypes set to true returns all the
       contained elements of the Namespace."
  end -- feature
end -- class Namespace

class GeneralizableElement
  inherit Namespace
  feature {none}
    visibility : VisibilityType;
    isAbstract : boolean;
    isRoot : TristateType;
    isLeaf : TristateType;
    allSupertypes[0..*] : GeneralizableElement
  end -- feature
  feature {any}
    deferred lookupElementExtended(in name : NameType; in
     includeSubtypes : boolean):ModelElement
     is text "Returns the element whose name matches the provided
      name. Like findElement defined for Namespace, this operation
      searches the elements contained by the GeneralizableElement.
      However, the lookupElementExtended also searches the elements
      contained by all GeneralizableElement supertypes (direct and
      indirect) of the GeneralizableElement. Subtypes can include a
      larger overall are for the lookup. Package, a subtype of
      GeneralizableElement, also considers the elements brought
      into this Namespace through the use of the Import type.";
    deferred findElementsByTypeExtended(in ofType :
     Type):ModelElement
     is text "Provides an extension of the findElementsByType
      defined for Namespace so that contained elements of all
      supertypes (direct and indirect) of the GeneralizableElement
      are included in the search. The order of the returned
      elements is determined by the order of the elements contained
      in the  GeneralizableElements and a depth-first traversal of
      the supertypes. Subtypes can include a larger overall are for
      the lookup. Package, a subtype of GeneralizableElement, also
      considers the elements brought into this Namespace through
      the use of Import"
```

(UOL 1.2)

```
    end -- feature
  end -- class GeneralizableElement

  class TypedElement
    inherit ModelElement
  end -- class TypedElement

  deferred class Classifier
    inherit GeneralizableElement
  end -- class Classifier

  class Class
    inherit Classifier
    feature {none}
      isSingleton :boolean
    end -- feature
    constrained by {((self.contents - Set { Class, DataType,
     MofAttribute, Reference, Operation,MofException, Constraint })-
     >isEmpty) and (self.isSingleton implies self.AllInstances < 2)
     and (self.isAbstract implies not self.isSingleton )}
  end -- class Class

  class DataType
    inherit Classifier
    feature {none}
      typeCode
           constrained by {text "
             allTypes : Set{TCKind}
             allTypes = Set{ self.kind }->union(
             if Set{tk_struct, tk_union, tk_except}
                  ->includes(self.kind)
             then self.member_types->collect(mt | mt.allTypes)->
             union(
               if self.kind = tk_union then
                 self.discriminator_type.allTypes
               else { }
               endif
             )
             else if Set{tk_sequence, tk_array, tk_alias}
                       ->includes(self.kind) then
                    self.content_type.allTypes
                  else{ }
             endif "}
           : TypeDescriptor
    end -- feature
    constrained by {text "((self.contents - Set{ TypeAlias, Tag })
      ->isEmpty) and (self.typeCode.allTypes->intersection(Set{
      tk_void, tk_principal, tk_except })->isEmpty) and
      (self.typeCode.allTypes->forAll(tc| {set of MOF data type names)
      ->includes(tc.name) or self.contents->Select(c |
      c.isOfType(TypeAlias))-> collect(ta | ta.name)
      ->includes(tc.name) or true))"}
  end -- class DataType

  deferred class Feature
    inherit ModelElement
```

```
  feature {none}
    visibility: VisibilityType
  end -- feature
end -- class Feature

class StructuralFeature
  inherit Feature; TypedElement
  feature {none}
    multiplicity:MultiplicityType;
    ischangeable:boolean
  end -- feature
end -- class StructuralFeature

class MofAttribute
  inherit StructuralFeature
  feature {none}
    isDerived:boolean
  end -- feature
end -- class MofAttribute

class Reference
  inherit StructuralFeature
    constrained by {(self.muliplicity =
      self.referencedEnd.multiplicity
      and (self.scope = instance_level)
      and (self.isChangeable = self.referencedEnd.isChangeable)
      and (self.type = self.referencedEnd.type))}
end -- class Reference

deferred class BehavioralFeature
  inherit Namespace; Feature
end -- class BehavioralFeature

class Operation
  inherit BehavioralFeature
  feature {none}
    isQuery:boolean
  end -- feature
  constrained by {((self.contents - Set { Parameter, Constraint })
    ->isEmpty
    and (self.contents->Select(c | c.oclIsTypeOf(Parameter))
    ->Select(p : Parameter | p.direction = return)->size < 2))}
end -- class Operation

class MofException
  inherit BehavioralFeature
    constrained by {((self.contents - Set { Parameter, Constraint
      })->isEmpty)
      and (self. contents->Select(c | c.oclIsTypeOf(Parameter))
      ->forAll(p : Parameter | p.direction = Out))}
end -- class MofException

class Association
  constraint
    consOneNotNone is {(self.contents->Select(c |
      c.oclIsTypeOf(AssociationEnd))->
```

(UOL 1.2)

```
          Select(r | r.aggregation <> None)->size < 2)}
    end -- Constraint
    inherit Classifier
    feature {none}
      isDerived : boolean
    end -- feature
    constrained by {((self.contents - Set{ AssociationEnd, Constraint
      })->isEmpty)
      and (self.supertypes->isEmpty )
      and (self.isRoot = dont_care and self.isLeaf = dont_care )
      and (not self.isAbstract )
      and (self.visibility = public )
      and (self.contents->Select(c | c.oclIsTypeOf(AssociationEnd))-
        >size = 2 )}
end -- class Association

class AssociationEnd
  inherit TypedElement
  feature {none}
    multiplicity : MultiplicityType;
    aggregation:AggregationType;
    isNavigable:boolean;
    isChangeable:boolean;
    otherEnd:AssociationEnd
  end -- feature
  constrained by {(consoneNotNone)
    and (not self.referent->isEmpty implies self.isNavigable)
    and (otherEnd = self.container.contents->
      Select(c | c.oclIsKindOf(AssociationEnd) and c <> self))
    and (self.multiplicity.isOrdered implies not
      self.otherEnd.multiplicity.isOrdered)}
end -- class AssociationEnd

class Package
  inherit GeneralizableElement
  feature {any}
  deferred copyModel(in target: Namespace):Modelelement
   is text "This operation is only available to top-level Packages
    (Packages not contained by another Package). The copyModel
    operation creates a copy of the model in the target Namespace.
    For copying a MOF-compliant metamodel, this Namespace must be an
    instance of MofRepository. The boundaries of the copy are
    determined by the transitive closure of the model's
    requiredElements (elements it depends on). The operation could
    result in copying not only this model, but imported Packages
    from this model's repository or other repositories. If however,
    an imported Package is already in the target repository, it is
    not copied.";
  deferred externalize(in format:string;inout stream:Any)
    raise ObjectNotExternalizable,FormatNotSupported
   is text "Externalize the Package and all of its ModelElements
    (transitive closure on the containment hierarchy) in a format
    specified by the format parameter, into a stream of type any.";
  deferred internalize(in format:string;in stream:Any):Package
    raise FormatNotSupported,IllformedExternalizedObject
```

```
    is text "Translate a model in some external format specified by
      the format parameter, encoded in the stream, into a package"
  end -- feature
  constrained by {((self.contents - Set { Package, Type, DataType,
    Association, MofException, Constraint, Import })->isEmpty)
    and (not self.isAbstract)}
end -- class Package

class Import
  inherit ModelElement
  feature {none}
    visibility:boolean
  end -- feature
  constrained by {(self.container.isVisible(e|
    e=self.importedNamespace))
    and (self.container.extendedNamespace->
    forAll(e | e.qualifiedName =
    self.importedNamespace.qualifiedName implies e = self))}
end -- class Import

class Parameter
  inherit TypedElement
  feature {none}
    direction:DirectionType;
    multiplicity:MultiplicityType
  end -- feature
end -- class Parameter

class Constraint
  inherit ModelElement
  feature {none}
    expression:Any;
    language:string;
    evaluationPolicy:EvaluationType
  end -- feature
  constrained by {(self.constrainedElements->forAll(c | not
    c.oclIsTypeOf(Constraint))) and (self.constrainedElements->
    forAll(c | self.container.extendedContents->includes(c|)))}
end -- class Constraint

class Constant
  inherit TypedElement
  feature {none}
    value:Any
  end -- feature
  constrained by {self.value.oclIsTypeOf(e|e=self.type)}
end -- class Constant

class Tag
  inherit ModelElement
  feature {none}
    tagId:string;
    value:Any
  end -- feature
end -- class Tag
```

(UOL 1.2)

```
class TypeAlias
  inherit TypedElement
  feature {none}
    multiplicity:MultiplicityType
  end -- feature
end -- class TypeAlias

--
-- Relationships
--
relation DependsOn
  link ModelElement[0..*],ModelElement[0..*]
  feature {ModelElement}
    with tag values (<AssociationEndName,provider>)
  end -- feature
  feature {ModelElement}
    with tag values
      (<AssociationEndName,dependent>,<IsNavigable,false>)
  end -- feature
end -- relation DependsOn

relation Aliases
  link Import[0..*],Namespace[1]
  feature {Import}
    with tag values (<AssociationEndName,importer>,
      <IsNavigable,false>)
  end -- feature
  feature {Namespace}
    with tag values (<AssociationEndName,imported>)
  end -- feature
end -- relation Aliases

relation Contains
  link ModelElement[0..*],Namespace[0..1]
  feature {ModelElement}
    with tag values (<AssociationEndName,containedElement>);
    constrained by {ordered}
  end -- feature
  feature {Namespace}
    with tag values (<AssociationEndName,container>,
        <IsNavigable,false>,
        <Aggregation,Composite>)
  end -- feature
end -- relation Contains

relation Constrains
  link ModelElement[1..*],Constraint[0..*]
  feature {ModelElement}
    with tag values (<AssociationEndName,constrainedElement>)
  end -- feature
end -- relation Constrains

relation AttachesTo
  link Tag[0..*],ModelElement[1..*]
  feature {Tag}
    with tag values (<IsNavigable,false>)
```

```
  end -- feature
end -- relation AttachesTo

relation Generalizes
  link GeneralizableElement[0..*],GeneralizableElement[0..*]
  feature {GeneralizableElement}
    with tag values (<AssociationEndName,supertype>);
    constrained by {ordered}
  end -- feature
  feature {GeneralizableElement}
    with tag values
      (<AssociationEndName,subtype>,<IsNavigable,false>)
  end -- feature
end -- relation Generalizes

relation IsOfType
  link Classifier[1],TypedElement[0..*]
  feature {Classifier}
    with tag values (<AssociationEndName,type>)
  end -- feature
  feature {TypedElement}
    with tag values (<IsNavigable,false>)
  end -- feature
end -- relation IsOfType

relation CanRaise
  link Operation[0..*],MofException[0..*]
  feature {Operation}
    with tag values (<IsNavigable,false>)
  end -- feature
  feature {MofException}
    constrained by {ordered}
  end -- feature
end -- relation CanRaise

relation RefersTo
  link Reference[0..*],AssociationEnd[1]
  feature {Reference}
    with tag values
      (<AssociationEndName,referent>,<IsNavigable,false>)
  end -- feature
  feature {AssociationEnd}
    with tag values (<AssociationEndName,referencedEnd>)
  end -- feature
end -- relation RefersTo

relation Exposes
  link Reference[0..*],AssociationEnd[1]
  feature {Reference}
    with tag values
      (<AssociationEndName,referrer>,<IsNavigable,false>)
    end -- feature
  feature {AssociationEnd}
    with tag values (<AssociationEndName,exposedEnd>)
  end -- feature
end -- relation Exposes Exposes
```

(UOL 1.2)

```
   end -- package ModelPackage
end -- model MOF
```

D UOL grammar in WSN format:

Being that for the STEP/EXPRESS mapping it is better to have the grammar in WSN format, this section contains the description of the UOL grammar in WSN format.

Keywords:

- action = 'action'.
- actions = 'actions'.
- activity = 'activity'.
- actor = 'actor'.
- 'adaptation = adaptation'.
- addonly = 'addonly'.
- after = 'after'.
- all = 'all'.
- alternative = 'alternative'.
- and = 'and'.
- any = 'any'.
- as = 'as'.
- attached = 'attached'.
- BIT = 'BIT'.
- branch = 'branch'.
- by = 'by'.
- call = 'call'.
- class = 'class'.
- collaboration = 'collaboration'.
- component = 'component'.
- composite = 'composite'.
- concurrent = 'concurrent'.
- constraiend = 'constrained'.
- constraint = 'constraint'.
- course = 'course'.
- creates = 'creates'.
- current = 'current'.
- deep = 'deep'.
- deferred = 'deferred'.
- diagrams = 'diagrams'.
- else = 'else'.

UOL grammar in WSN format:

(UOL 1.2)

- end = 'end'.
- entry = 'entry'.
- event = 'event'.
- exception = 'exception'.
- exit = 'exit'.
- expanded = 'expanded'.
- export = 'export'.
- extend = 'extend'.
- extension = 'extension'.
- false = 'false'.
- feature = 'feature'.
- final = 'final'.
- flow = 'flow'.
- fork = 'fork'.
- form = 'from'.
- frozen = 'frozen'.
- history = 'history'.
- implements = 'implements'.
- implies = 'implies'.
- import = 'import'.
- in = 'in'.
- infix = 'infix'.
- inherit = 'inherit'.
- initial = 'initial'.
- inout = 'inout'.
- interface = 'interface',
- is = 'is'.
- join = 'join'.
- like = 'like'.
- link = 'link'.
- machine = 'machine'.
- model ='model'.
- node = 'node'.
- none = 'none'.
- not = 'not'.
- of = 'of'.
- or = 'or'.
- out = 'out'.
- package = 'package'.
- partitioned = 'partitioned'.
- postcondition = 'postcondition'.
- precondition = 'precondition'.
- prefix = 'prefix'.
- raise = 'raise'.

- redefine = 'redefine'.
- relation = 'relation'.
- rename = 'rename'.
- request = 'request'.
- result = 'result'.
- select = 'select'.
- shallow = 'shallow'.
- signal = 'signal'.
- simple = 'simple'.
- state = 'state'.
- static = 'static'.
- stereotype = 'stereotype'.
- stereotyped = 'stereotyped'.
- subactivity = 'subactivity'.
- submachine = 'submachine'.
- subsystem = 'subsystem'.
- synchronous = 'synchronous'.
- tag = 'tag'.
- then = 'then'.
- to = 'to'.
- transition = 'transition'.
- trigger = 'trigger'.
- true = 'true'.
- undefine = 'undefine'.
- unique = 'unique'.
- use = 'use'.
- usecase = 'usecase'.
- values = 'values'.
- viewed = 'viewed'.
- when = 'when'.
- with = 'with'.
- xor = 'xor'.

UOL grammar in WSN format:

(UOL 1.2)

Lexical definitions:

- character_lower = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'.
- character_upper = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'.
- digit_excl_null = '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'.
- digit = digit_excl_null | null.
- null = '0'.
- underscore = '_'.
- OCLname = character_lower [{character_lower | underscore | underscore | digit}].
- OCLtypeName = character_upper [{character_lower | underscore | character_upper | digit}].
- Dec = (digit_excl_null [{digit}]) | null.
- quotation_mark = "".
- special_character_a = '!' | quotation_mark | '#' | '$' | '%' | '%' | special_character_q | '(' | ')' | '*' | '+' | ',' | '-' | '.' | '/' | digit | ':' | ';' | '<' | '=' | '>' | '?' | '@' | character_upper | '[' | '\' | ']' '^' | underscore | '`' | character_lower | '{' | '|' | '}' | '~'.
- special_character_n = \n //newline
- special_character_q = '''. //aphostrophe mark
- special_character_s = ' '. //space
- special_character_o = NUL | TC1/SOH | TC2/STX | TC3/ETX | TC4/EOT | TC5/ENQ | TC6/ACK | BEL | FE0/BS | Tab | FE2/LF | FE3/VT | FE4/FF | FE5/CR | SO | SI | TC7/DLE | DC1 | DC2 | DC3 | DC4 | TC8/NAK | TC9/SYN | TC10/ETB | CAN | EM | SUB | ESC | DT/FS/IS4 | PT/GS/IS3 | RS/IS2.
- CChar = quotation_mark ['\'] (special_character_a | special_character_q | special_character_s | special_character_o).
- Tab = FE1/HT.
- Range = {digit} [{Tab}] '..' [{Tab}] ({digit} | '*').
- Float0 = {digit} '.' [{digit}] [('e' | 'E') ('+' | '-') {digit}].
- MAYBE = '?' [{digit}].
- OBrack = '['.
- CBrack = ']'.
- OParen = '('.
- CParen = ')'.
- OCurly = '}'.
- CCurly = '{'.
- Comma = ','.
- Colon = ':'.
- Receives = ':='.
- Path = '::'.
- SemiColon = ';'.
- Dot = '.'.
- Plus = '+'.
- UPlus = Plus.

- Minus = '-'.
- UMinus = Minus.
- Div = '/'.
- DD = '//'.
- DS = '\\'.
- Percent = '%'.
- LT = '<'.
- GT = '>'.
- AngleBL = '<<'.
- AngleRR = '>>'.
- LTE = '<='.
- GTE = '>='.
- Equal = '='.
- NotEqual = '/='.
- DIF = '<>'.
- Power = '^'.
- Arrow = '->'.
- Exclamation = '!'.
- DoubleDot = '..'.
- Dollar = '$'.
- Star = '*'.
- AL = '#'.
- At = '@'.
- DES = '|'.

UOL grammar in WSN format:

(UOL 1.2)

Grammar productions

- Act_concurrent_state_list = (concurrent Act_state_list)[{(concurrent Act_state_list)}].
- Act_simple_state = (Action_state) | (Object_flow_state).
- Act_state_kind = (Activity_body) | (Act_simple_state) | (Pseudostate) | (Subactivity).
- Act_state_list = (Act_state_kind Partition_opt end)[{(Act_state_kind Partition_opt end)}].
- Action_def_list = [{(Action_definition)}].
- Action_definition = (Synchronous_opt action Name Recurrence_opt Script_opt Object_set_expression_opt Request_opt Action_kind).
- Action_kind = [(call Operation_use) | (creates identifier) | (TextMultiline)].
- Action_list = (identifier)[{',' identifier)}].
- Action_opt = [(actions EntryExit_action_opt)].
- Action_sequence_opt = [(actions Action_list)].
- Action_state = (state Name Viewed_with_opt Constraint_use_def_opt Action_opt).
- Activity_body = (Activity_state Transition_list Action_def_list).
- Activity_model = (activity Name Viewed_with_opt Constraint_use_def_opt Activity_body end).
- Activity_state = (composite State_definition Partition_opt Act_concurrent_state_list) | (composite State_definition Partition_opt Act_state_list).
- Actor_or_exception = (actor) | (exception).
- Actual = (Expression).
- Actual_generics = ('[' Type_list ']').
- Actual_generics_opt = [(Actual_generics)].
- Actual_list = (Actual)[{',' Actual)}].
- Actuals = ('(' Actual_list ')').
- Actuals_opt = [(Actuals)].
- Alias = (Element_name).
- Anchor = (current) | (identifier).
- Anchored = (like Anchor).
- As_alias = [(as Alias)].
- Assoc_entity_opt = [(with Classifier_name)].
- Attach_or_view = (attached to Name) | (viewed with View_element_name_list).
- Attribute_declaration = (Attribute_signature Attribute_single_or_multi Extension_use).
- Attribute_rest = (Invariant_opt Type_mark is Initial_value) | (Invariant_opt Type_mark) | (Invariant_opt ':' unique '{' Classifier_list '}') | (',' Feature_name_list Type_mark).
- Attribute_signature = (addonly Signature_rest) | (frozen Signature_rest) | (Signature_rest).
- Attribute_single_or_multi = (Attribute_rest) | (Cardinality2 Invariant_opt Type_mark Initial_multi_value_opt).
- Attribute_value = (identifier is Expression).
- Base_class = (Classifier_name).
- Binary = (and) | (DD) | (DS) | (GTE) | (implies) | (LTE) | (or) | (xor) | ('<') | ('>') | ('+') | ('-') | ('*') | ('/') | ('^').

- Binary_expression = (Expression Binary Expression).
- Bit_type = (BIT Constant).
- Boolean_constant = (false) | (true).
- Boolean_expression = (TextMultiline).
- Call = (Parenthesized_qualifier_opt Call_chain).
- Call_chain = (Unqualified_call)[{('.' Unqualified_call)}].
- Cardinality = (Range_list Range_last).
- Cardinality_opt = [(Cardinality2)].
- Cardinality2 = ('[' Cardinality ']').
- Class_body = [(In_or_st_or_fe)].
- Class_declaration = (Class_header Formal_generics_opt Viewed_with_opt Extension_declaration_opt Extension_use Class_body end).
- Class_header = (Deferred_opt class Class_name).
- Class_list = (Class_name)[{(',' Class_name)}].
- Class_name = (Classifier_name).
- Class_or_intf_or_rel_decl_list = [{(Class_declaration) | (Element_name) | (Interface_declaration) | (Relation_declaration)}].
- Class_type = (Class_name Actual_generics_opt).
- Class_type_expanded = (expanded Class_type).
- Classifier_list = (Classifier_name)[{(',' Classifier_name)}].
- Classifier_name = (Element_name).
- Classifier_or_operation = (Element_name).
- Clients = ('{' Class_list '}').
- Collaboration_declaration = (collaboration Collaboration_name Formal_generics_opt Viewed_with_opt Implementation_of_opt Class_or_intf_or_rel_decl_list Action_def_list Message_list_opt end).
- Collaboration_name = (identifier).
- Comment_definition = (CommentMultiline Attach_or_view).
- Comparision = (NOTEQUAL) | ('=').
- Component_or_node = (component) | (node).
- Composite_state = (composite State_definition Concurrent_state_list) | (composite State_definition State_list).
- Concurrent_state_list = (concurrent State_list)[{(concurrent State_list)}].
- Condition_opt = [('{' PrePost)].
- Constant = (Entity) | (Manifest_constant).
- Constraint = (constraint Constraint_def_list end).
- Constraint_def_list = (Constraint_name is '{' Constraint_expression '}')[{(Constraint_name is '{' Constraint_expression '}')}].
- Constraint_expression = (OCL_expression) | (TextMultiline).
- Constraint_name = (identifier).
- Constraint_use_def_opt = [(Constraint Invariant_opt) | (Use_of_constraint)].
- Deferred_event_list = (event Name)[{(event Name)}].
- Deferred_events_opt = [(deferred Deferred_event_list)].
- Deferred_opt = [(deferred)].

(UOL 1.2)

- Dependency = (Element_path to Element_path).
- Dependency_description_opt = [(is TextMultiline)].
- Dependency_list = (Dependency)[{(',' Dependency)}].
- Discriminator_opt = [('(' Name ')')].
- Element_name = (identifier).
- Element_path = (Element_name)[{(Path Element_name)}].
- Element_path_opt = [(Element_path)].
- Entity = (current) | (identifier) | (result).
- Entity_declaration_group = (Parameter_kind_opt Feature_name_list Type_mark Initial_value_opt).
- Entity_declaration_list = (Entity_declaration_group)[{(';' Entity_declaration_group)}].
- EntryExit_action_opt = (entry Action_list exit Action_list) | (entry Action_list) | (exit Action_list).
- Equality = (Expression Comparision Expression).
- Exception_list = (raise Exception_use)[{(raise Exception_use)}].
- Exception_name_list = (Identifier_list).
- Exception_opt = [(Exception_list)].
- Exception_raise_opt = [(raise Exception_name_list)].
- Exception_use = (Name from Name_comma_list).
- Expression = (Call) | (Equality) | (Manifest_array) | (Manifest_constant) | (Operator_expression).
- Expression_list = (Expression)[{(',' Expression)}].
- Extension_declaration = (Constraint) | (Stereotype) | (Tagged_values).
- Extension_declaration_opt = [{(Extension_declaration)}].
- Extension_name = (identifier).
- Extension_point = (String).
- Extension_point_list = (String_list).
- Extension_use = (Use_of_stereotype_opt List_use_consr_tag).
- Feature_declaration = (Attribute_declaration) | (Constraint) | (Method_declaration) | (Operation_declaration) | (Use_of_constraint) | (Use_of_tagged_value).
- Feature_declaration_attrib_or_Oper = (Attribute_declaration) | (Constraint) | (Operation_declaration) | (Use_of_constraint) | (Use_of_tagged_value).
- Feature_list = (Feature_declaration) | (Feature_list_nul Feature_declaration).
- Feature_list_attrib_or_Oper = (Feature_declaration_attrib_or_Oper) | (Feature_list_nul_attrib_or_Oper Feature_declaration_attrib_or_Oper).
- Feature_list_nul = (Feature_declaration ';')[{(Feature_declaration ';')}].
- Feature_list_nul_attrib_or_Oper = (Feature_declaration_attrib_or_Oper ';')[{(Feature_declaration_attrib_or_Oper ';')}].
- Feature_name = (identifier) | (Infix) | (Prefix).
- Feature_name_list = (Identifier_list).
- Feature_or_redef = (Feature_list) | (Redefine_with_list).
- Feature_rest = (Feature_list) | (Use_of_stereotype ';' Feature_list) | (Use_of_stereotype).
- Feature_rest_attrib_or_Oper = (Feature_list_attrib_or_Oper) | (Use_of_stereotype ';' Feature_list_attrib_or_Oper) | (Use_of_stereotype).

- Feature_set = (all) | (Identifier_list).
- Features = [{(feature '{' Visibility '}' Feature_rest end)}].
- Features_attrib_or_Oper = [{(feature '{' Visibility '}' Feature_rest_attrib_or_Oper end)}].
- Float_constant = ('FLOATINGconstant0').
- Formal_arguments_opt = ('(' Entity_declaration_list ')') | ('(' ')').
- Formal_generic = (Element_name Invariant_opt).
- Formal_generic_list = (Formal_generic)[{',' Formal_generic)}].
- Formal_generics = ('[' Formal_generic_list ']').
- Formal_generics_opt = [(Formal_generics)].
- Guard_expression = (Expression) | (TextMultiline).
- Guard_expression_opt = [(When_or_after)].
- Icon_opt = [(viewed as String)].
- identifier = (MAYBE) | (OCLtypeOrName).
- Identifier_list = (identifier)[{',' identifier)}].
- Implementation_of_opt = [(implements Classifier_or_operation)].
- In_or_st_or_fe = (Features feature '{' Visibility '}' Feature_rest end Invariant_opt) | (Features State_machine Invariant_opt) | (Inheritance Rest Invariant_opt).
- Infix = (infix '(' Infix_operator ')').
- Infix_operator = (Binary) | (identifier).
- Inheritance = (inherit Parent_list).
- Inheritance_opt = [(Inheritance)].
- Initial_multi_value_opt = [(is '{' Expression_list '}')].
- Initial_value = (Expression).
- Initial_value_opt = [(is Initial_value)].
- Int_or_star = (Integer_constant) | (Star).
- Interface_declaration = (Interface_header Formal_generics_opt Viewed_with_opt Extension_declaration_opt Extension_use Inheritance_opt Interface_operation_declaration_opt Invariant_opt end).
- Interface_header = (interface Feature_name).
- Interface_operation_declaration_opt = [{(feature '{' Visibility '}' Operation_list end)}].
- Internal_transition_list = [{(transition When_or_after Trigger_opt Action_sequence_opt)}].
- Invariant_opt = [(Use_of_constraint)].
- Light_body = (Name Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Features Invariant_opt end).
- Link_list = (link Type_or_dependency Assoc_entity_opt).
- Link_list_opt = [(Link_list)].
- List_use_consr_tag = [{(Use_of_tagged_value)}].
- Machine_body = (Composite_state Transition_list Action_def_list).
- Manifest_array = (ANGLEBL Expression_list ANGLERR).
- Manifest_constant = (Boolean_constant) | (Character_constant) | (Float_constant) | (Integer_constant) | (String).
- Message = (actions Action_list to Classifier_name from Classifier_name).

UOL grammar in WSN format:

(UOL 1.2)

- Message_list = (Message)[{(Message)}].
- Message_list_opt = [(Message_list)].
- Method_body = (OM_body Method_rest).
- Method_declaration = (Method_header Method_body).
- Method_header = (Signature) | (Signature_rest).
- Method_rest = (like Feature_name Method_routine) | (Method_routine) | (Specification Method_routine).
- Method_routine = (is Routine).
- Model_declaration = (model Model_name View_element_decl_list_opt Package_decl_list_opt end).
- Model_name = (identifier).
- MorePost = [('{' postcondition ':' Constraint_expression '}')].
- Name = (identifier).
- Name_comma_list = (Name)[{(',' Name)}].
- Name_inherit_list = (Name Discriminator_opt)[{(',' Name Discriminator_opt)}].
- New_export_item = (Clients Feature_set).
- New_export_list = (New_export_item)[{(';' New_export_item)}].
- New_exports = (export New_export_list).
- New_exports_opt = [(New_exports)].
- Object_body = (is Attribute_value) [{(';' Attribute_value)}].
- Object_declaration = (Object_name Formal_generics_opt instance of Element_path Extension_use Viewed_with_opt Object_body Invariant_opt end).
- Object_flow_state = (State_definition flow Name '[' Name ']' Invariant_opt).
- Object_name = (identifier).
- Object_set_expression = (Name)[{(',' Name)}].
- Object_set_expression_opt = [(to Object_set_expression)].
- OCL_expression = (OCLexpression).
- OCLactualParameterList = (OCLexpressionListing).
- OCLactualParameterListOpt = [(OCLactualParameterList)].
- OCLadditiveExpression = (OCLmultiplicativeExpression OCLadditiveExpressionList).
- OCLadditiveExpressionList = [{(OCLaddOperator OCLmultiplicativeExpression)}].
- OCLaddOperator = ('+') | ('-').
- OCLcollectionKind = (Bag) | (Collection) | (Sequence) | (Set).
- OCLdeclarator = (OCLnameList OCLsimpleTypeSpecifierOpt '|').
- OCLdeclaratorOpt = [(OCLdeclarator)].
- OCLdotOrArrow = (Arrow) | ('.').
- OCLenumerationType = (enum '{' '#' OCLnameList '}').
- OCLexpression = (OCLlogicalExpression).
- OCLexpressionList = (',' OCLexpression)[{(',' OCLexpression)}].
- OCLexpressionListing = (OCLexpression)[{(',' OCLexpression)}].
- OCLexpressionListOrRange = (OCLexpression OCLexpressionRestOpt).
- OCLexpressionListOrRangeOpt = [(OCLexpressionListOrRange)].
- OCLexpressionRestOpt = [(DoubleDot OCLexpression) | (OCLexpressionList)].

- OCLfeatureCall = (OCLpathName OCLtimeExpressionOpt OCLqualifiersOpt OCLfeatureCallParametersOpt).
- OCLfeatureCallList = [{(OCLdotOrArrow OCLfeatureCall)}].
- OCLfeatureCallParameters = ('(' OCLdeclaratorOpt OCLactualParameterListOpt ')').
- OCLfeatureCallParametersOpt = [(OCLfeatureCallParameters)].
- OCLifExpression = (if OCLexpression then OCLexpression else OCLexpression endif).
- OCLliteral = (OCLnumber) | (OCLSTRING) | ('#' OCLname).
- OCLliteralCollection = (OCLcollectionKind '{' OCLexpressionListOrRangeOpt '}').
- OCLlogicalExpression = (OCLrelationalExpression OCLlogicalExpressionList).
- OCLlogicalExpressionList = [{(OCLlogicalOperator OCLrelationalExpression)}].
- OCLlogicalOperator = (and) | (implies) | (or) | (xor).
- OCLmultiplicativeExpression = (OCLunaryExpression OCLmultiplicativeExpressionList).
- OCLmultiplicativeExpressionList = [{(OCLmultiplyOperator OCLunaryExpression)}].
- OCLmultiplyOperator = ('*') | ('/').
- OCLnameList = (OCLname)[{(',' OCLname)}].
- OCLnumber = (Dec).
- OCLpathName = (Element_path).
- OCLpathTypeName = (OCLtypeName)[{(Path OCLtypeName)}].
- OCLpostfixExpression = (OCLprimaryExpression OCLfeatureCallList).
- OCLprimaryExpression = (OCLifExpression) | (OCLliteralCollection) | (OCLliteral) | (OCLpathName OCLtimeExpressionOpt OCLqualifiersOpt OCLfeatureCallParametersOpt) | ('(' OCLexpression ')').
- OCLqualifiers = ('[' OCLactualParameterList ']').
- OCLqualifiersOpt = [(OCLqualifiers)].
- OCLrelationalExpression = (OCLadditiveExpression OCLrelationalExpressionOpt).
- OCLrelationalExpressionOpt = [(OCLrelationalOperator OCLadditiveExpression)].
- OCLrelationalOperator = (DIF) | (GTE) | (LTE) | ('<') | ('>') | ('=').
- OCLsimpleTypeSpecifier = (OCLenumerationType) | (OCLpathTypeName).
- OCLsimpleTypeSpecifierOpt = [(':' OCLsimpleTypeSpecifier)].
- OCLSTRING = ('STRINGstart' 'STRINGliteral' 'STRINGend').
- OCLtimeExpression = ('@' OCLname).
- OCLtimeExpressionOpt = [(OCLtimeExpression)].
- OCLtypeOrName = (OCLname) | (OCLtypeName).
- OCLunaryExpression = (OCLpostfixExpression) | (OCLunaryOperator OCLpostfixExpression).
- OCLunaryOperator = (not) | ('-' UMinus).
- OM_body = (Formal_arguments_opt Type_mark_opt Exception_raise_opt Condition_opt Extension_use Invariant_opt).
- Operation_body = (OM_body Operation_rest).
- Operation_declaration = (Signature Operation_body).
- Operation_list = (Operation_declaration)[{(';' Operation_declaration)}].
- Operation_or_signal = (Operation_use) | (Signal_definition).
- Operation_rest = (is Specification).

(UOL 1.2)

- Operation_use = (Name '.' Name '(' Expression_list ')').
- Operator_expression = (Binary_expression) | (Parenthesized) | (Unary_expression).
- Package_decl_list = (Package_or_subsystem_declaration)[{(Package_or_subsystem_declaration)}].
- Package_decl_list_opt = [(Package_decl_list)].
- Package_declaration = (package Package_name Viewed_with_opt Extension_use Package_inheritance_opt Package_import_optPackage_element_decl_list_opt Invariant_opt end).
- Package_element_decl = (Activity_model) | (Actor_or_exception Light_body) | (Class_declaration) | (Collaboration_declaration) | (Comment_definition) | (Object_declaration)|(Component_or_node Ultra_light_body) | (Extension_declaration)| (Interface_declaration) | (Package_declaration) | (Relation_declaration) | (Usecase_abstraction).
- Package_element_decl_list = (Visibility_opt Package_element_decl)[{(Visibility_opt Package_element_decl)}].
- Package_element_decl_list_opt = [(is Package_element_decl_list)].
- Package_import_elem = (Visibility_opt Element_path_opt As_alias from Package_name).
- Package_import_list = (Package_import_elem)[{(',' Package_import_elem)}].
- Package_import_opt = [(import Package_import_list)].
- Package_inheritance_opt = [(inherit Package_name_list)].
- Package_name = (identifier).
- Package_name_list = (Package_name Discriminator_opt)[{(',' Package_name Discriminator_opt)}].
- Package_or_subsystem_declaration = (Package_element_decl) | (Subsystem_declaration) | (Use_of_constraint) | (Use_of_stereotype) | (Use_of_tagged_value).
- Parameter_kind_opt = [(inout) | (in) | (out)].
- Parent = (Class_type Discriminator_opt).
- Parent_list = (Parent)[{(';' Parent)}].
- Parent_relation = (Relation_type Relation_feature_adaptation_opt).
- Parent_relation_list = (Parent_relation)[{(';' Parent_relation)}].
- Parenthesized = ('(' Expression ')' UPlus).
- Parenthesized_qualifier = (Parenthesized '.').
- Parenthesized_qualifier_opt = [(Parenthesized_qualifier)].
- Partition_opt = [(partitioned in Name)].
- Path_name = (Name)[{(Path Name)}].
- Path_name_or_final = (final) | (Path_name).
- Path_name_or_initial = (initial) | (Path_name).
- Position = [('(' Dec ',' Dec Third_dimension ')')].
- Post_opt = ('}' MorePost).
- Prefix = (prefix '(' Prefix_operator ')').
- Prefix_operator = (Unary identifier).

- PrePost = (postcondition ':' Constraint_expression '}') | (precondition ':' Constraint_expression Post_opt).
- Property = (LT Extension_name GT) | (LT Extension_name ',' Expression GT).
- Property_list = (Property)[{',' Property)}].
- Pseudostate = (Pseudostate_kind NameConstraint_use_def_opt Action_opt).
- Pseudostate_kind = (branch) | (deep history) | (final) | (fork) | (initial) | (join) | (shallow history).
- Range_last = (Int_or_star) | (Range).
- Range_list = [{(Range_mid)}].
- Range_mid = (Integer_constant ',') | (Range ',').
- Reception_opt = [(to Name_comma_list)].
- Recurrence_opt = [('(' Expression ')')].
- Redefine_pair = (Feature_name with Feature_name).
- Redefine_with_list = (Redefine_pair)[{',' Redefine_pair)}].
- Relation_declaration = (relation Relation_name Extension_declaration_opt Extension_use Relation_inheritance_opt Link_list_opt Features_attrib_or_Oper Invariant_opt end).
- Relation_feature_adaptation = (Rename_opt New_exports_opt Undefine_opt Relation_redefine_opt Select_opt end).
- Relation_feature_adaptation_opt = [(adaptation Relation_feature_adaptation)].
- Relation_inheritance = (inherit Parent_relation_list).
- Relation_inheritance_opt = [(Relation_inheritance)].
- Relation_name = (identifier).
- Relation_path = (Element_path).
- Relation_redefine = (redefine Feature_or_redef).
- Relation_redefine_opt = [(Relation_redefine)].
- Relation_type = (Relation_path).
- Rename = (rename Rename_list).
- Rename_list = (Rename_pair)[{',' Rename_pair)}].
- Rename_opt = [(Rename)].
- Rename_pair = (Feature_name as Feature_name).
- Request_opt = [(request Operation_or_signal)].
- Rest = (Features State_declaration_opt).
- Routine = (TextMultiline).
- Script_opt = [(String)].
- Select = (select Feature_list).
- Select_opt = [(Select)].
- Signal_definition = (signal Name Reception_opt Exception_opt).
- Signal_or_time_or_change = (after Time_expression) | (Signal_definition) | (when Boolean_expression).
- Signature = (deferred Signature_rest).
- Signature_rest = (Feature_name) | (static Feature_name).
- Simple_state = (simple State_definition).
- Specification = (TextMultiline).

(UOL 1.2)

- Start_production = (Model_declaration) | (Package_declaration).
- State_declaration_opt = [(State_machine)].
- State_definition = (state Name Viewed_with_opt Constraint_use_def_opt Action_opt Internal_transition_list Deferred_events_opt).
- State_kind = (Machine_body end) | (Pseudostate end) | (Simple_state end) | (Submachine end).
- State_list = (State_kind)[{(State_kind)}].
- State_machine = (state machine Name Viewed_with_opt Constraint_use_def_opt Machine_body end).
- Stereotype = (stereotype Extension_name of Base_class Icon_opt Stereotype_parent_opt Stereotype_extension_dec end).
- Stereotype_extension_dec = [{(Constraint) | (Tagged_values)}].
- Stereotype_parent_list = (Extension_name Discriminator_opt)[{(',' Extension_name Discriminator_opt)}].
- Stereotype_parent_opt = [(inherit Stereotype_parent_list)].
- String = ('STRINGstart' 'STRINGliteral' 'STRINGend').
- String_list = (String)[{(',' String)}].
- Subactivity = (subactivity Name Viewed_with_opt Constraint_use_def_opt Activity_body).
- Submachine = (submachine Name Viewed_with_opt Constraint_use_def_opt Machine_body).
- Subsystem_declaration = (Subsystem_header Formal_generics_opt Extension_use Viewed_with_opt Inheritance_optPackage_import_opt Interface_operation_declaration_opt Package_element_decl_list_opt Invariant_opt end).
- Subsystem_header = (Deferred_opt subsystem Subsystem_name).
- Subsystem_name = (identifier).
- Synchronous_opt = [(synchronous)].
- Tagged_values = (tag values Tagged_values_def_list end).
- Tagged_values_def = (Extension_name is Initial_value) | (Extension_name).
- Tagged_values_def_list = (Tagged_values_def)[{(Tagged_values_def)}].
- TextMultiline_opt = [(TextMultiline)].
- Third_dimension = [(',' Dec)].
- Time_expression = (Integer_constant String).
- Transition_definition = (transition Name from Path_name_or_initial to Path_name_or_final Guard_expression_opt Trigger_opt Action_sequence_opt).
- Transition_list = [{(Transition_definition)}].
- Trigger_expression = (call Operation_use) | (trigger Signal_or_time_or_change).
- Trigger_opt = [(Trigger_expression)].
- Type = (Anchored) | (Bit_type) | (Class_type) | (Class_type_expanded).
- Type_link = (Classifier_name).
- Type_link_list = (Type_link)[{(',' Type_link)}].
- Type_link_two_list = (Type_link Cardinality_opt ',' Type_link_list Cardinality_opt).
- Type_list = (Type)[{(',' Type)}].

- Type_mark = (':' Classifier_name) | (':' like Feature_name).
- Type_mark_opt = [(Type_mark)].
- Type_or_dependency = (Dependency_list Dependency_description_opt) | (Type_link_two_list).
- Ultra_light_body = (Name Formal_generics_opt Extension_use Viewed_with_opt Inheritance_opt Identifier_list Invariant_opt end).
- Unary = (not) | ('+' UPlus) | ('-' UMinus).
- Unary_expression = (Unary Expression).
- Undefine = (undefine Feature_list).
- Undefine_opt = [(Undefine)].
- Unqualified_call = (Entity Actuals_opt).
- Use_of_constraint = (constrained by '{' Constraint_expression '}').
- Use_of_stereotype = (stereotyped with Extension_name).
- Use_of_stereotype_opt = [(Use_of_stereotype)].
- Use_of_tagged_value = (with tag values '(' Property_list ')').
- Usecase_abstraction = (Usecase_definition) | (Usecase_extension) | (Usecase_instance).
- Usecase_actor_opt = [(actor Identifier_list)].
- Usecase_alt_course_opt = [(alternative course TextMultiline)].
- Usecase_definition = (usecase Name Formal_generics_opt Usecase_inherit_opt Usecase_use_opt Usecase_actor_opt Features_attrib_or_Oper TextMultiline_opt Usecase_alt_course_opt Usecase_extension_point_opt end).
- Usecase_extension = (extend Usecase_path with Usecase_path_list in Extension_point).
- Usecase_extension_point_opt = [(extension in Extension_point_list)].
- Usecase_inherit_opt = [(inherit Name_inherit_list)].
- Usecase_instance = (usecase Name Formal_arguments_opt Type_mark_opt is Usecase_method_list end).
- Usecase_method = (identifier is Expression).
- Usecase_method_list = (Usecase_method)[{(Usecase_method)}].
- Usecase_path = (Name)[{(Path Name)}].
- Usecase_path_list = (Usecase_path)[{(',' Usecase_path)}].
- Usecase_use_opt = [(use Identifier_list)].
- View_element_decl_list = (View_element_declaration)[{(';' View_element_declaration)}].
- View_element_decl_list_opt = [(diagrams View_element_decl_list)].
- View_element_declaration = (View_element_name_list ':' View_element_kind).
- View_element_kind = (identifier).
- View_element_name = (identifier).
- View_element_name_list = (View_element_name Position)[{(',' View_element_name Position)}].
- Viewed_with_opt = [(viewed with View_element_name_list)].
- Visibility = (any) | (Classifier_list) | (none).
- Visibility_opt = [('{' Visibility '}')].
- When_or_after = (after Time_expression) | (when Guard_expression).

UOL grammar in WSN format:

(UOL 1.2)

E ASCII, UNICODE and ISO 10646 Character Set

This section shows the code charts of the four formats more known:

Hex	ISO 10646	ASCII/UNICODE[1]
00	0/0	NUL
01	0/1	TC1/SOH
02	0/2	TC2/STX
03	0/3	TC3/ETX
04	0/4	TC4/EOT
05	0/5	TC5/ENQ
06	0/6	TC6/ACK
07	0/7	BEL
08	0/8	FE0/BS
09	0/9	FE1/HT
0A	0/10	FE2/LF
0B	0/11	FE3/VT
0C	0/12	FE4/FF
0D	0/13	FE5/CR
0E	0/14	SO
0F	0/15	SI
10	1/0	TC7/DLE
11	1/1	DC1
12	1/2	DC2
13	1/3	DC3
14	1/4	DC4
15	1/5	TC8/NAK
16	1/6	TC9/SYN
17	1/7	TC10/ETB
18	1/8	CAN
19	1/9	EM
1A	1/10	SUB
1B	1/11	ESC
1C	1/12	DT/FS/IS4
1D	1/13	PT/GS/IS3
1E	1/14	RS/IS2
1F	1/15	US/IS1
20	2/0	SPACE
21	2/21	!
22	2/2	“
23	2/3	#
24	2/4	currency / $
25	2/5	%

[1] The first 256 characters have the same value with ASCII and with UNICODE.

26	2/6	&
27	2/7	′
28	2/8	(
29	2/9	)
2A	2/10	*
2B	2/11	+
2C	2/12	,
2D	2/13	-
2E	2/14	.
2F	2/15	/
30	3/0	0
31	3/1	1
32	3/2	2
33	3/3	3
34	3/4	4
35	3/5	5
36	3/6	6
37	3/7	7
38	3/8	8
39	3/9	9
3A	3/10	:
3B	3/11	;
3C	3/12	<
3D	3/13	=
3E	3/14	>
3F	3/15	?
40	4/0	@
41	4/1	A
42	4/2	B
43	4/3	C
44	4/4	D
45	4/5	E
46	4/6	F
47	4/7	G
48	4/8	H
49	4/9	I
4A	4/10	J
4B	4/11	K
4C	4/12	L
4D	4/13	M
4E	4/14	N
4F	4/15	O
50	5/0	P
51	5/1	Q
52	5/2	R
53	5/3	S
54	5/4	T
55	5/5	U
56	5/6	V
57	5/7	W
58	5/8	X
59	5/9	Y
5A	5/10	Z

5B	5/11	[
5C	5/12	\
5D	5/16	]
5E	5/14	^
5F	5/15	_
60	6/0	`
61	6/1	a
62	6/2	b
63	6/3	c
64	6/4	d
65	6/5	e
66	6/6	f
67	6/7	g
68	6/8	h
69	6/9	i
6A	6/10	j
6B	6/11	k
6C	6/12	l
6D	6/13	m
6E	6/14	n
6F	6/15	o
70	7/0	p
71	7/1	q
72	7/2	r
73	7/3	s
74	7/4	t
75	7/5	u
76	7/6	v
77	7/7	w
78	7/8	x
79	7/9	y
7A	7/10	z
7B	7/11	{
7C	7/12	|
7D	7/13	}
7E	7/14	~
7F	7/15	DEL