THALES

Research & Technology

Report / Technical Document

MARTE FOR RATIONAL SOFTWARE ARCHITECT

UML profile documentation

DEPARTMENT / SERVICE	DOCUMENT NUMBER	PAGE
STI / LSE	61565273 305 1	1/30
	B A -	REVISION

DOCUMENT CONTROL

	- the : 08/08/07	A the:11/10/07	B the: 30/10/07	C the:	D the:
Written by Signature	N. Vienne	N. Vienne	S. Demathieu		
Approved par Signature					

Revision index	Modifications
A	Update to profile version 0.0.7 - Bugs and spelling issues fixed - Shapes and icons added
B	Document finalisation - Updated copyright notice - Added references to the MARTE Beta 1 document - Miscellaneous
C	
D	

CONTENTS

1. PURPOSE

A UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) has been adopted by the OMG. This initiative meets the needs of several Thales divisions, which develop real-time and embedded systems. Thales is an active contributor to the MARTE submission, through the ProMARTE consortium, as well as an expected end user of the profile when the latter will be implemented.

In that context, Thales Research & Technology has started a demonstrator of MARTE using Rational Software Architect 7.0 (RSA). That's why we have implemented the UML profile for MARTE. This implementation provided us useful feedback on the limitation of the tool, with respect to the UML 2.1 and MARTE specifications. This document keeps track of the design choices and encountered issues when implementing the profile. It gives a snapshot, as of October 2007, based on voted second revised version of MARTE. Changes may happen both in the MARTE specification and in the next versions of RSA.

2. DOCUMENTS

2.1. MANDATORY

[N/A] Not applicable.

2.2. REFERENCE

[MARTE] UML profile for MARTE, Beta 1 (http://www.omg.org/cgi-bin/doc?ptc/2007-08-04)

3. PROFILE IMPLEMENTATION

This section documents the implementation of the MARTE profile version 0.0.7 in Rational Software Architect 7.0 (RSA). The second revised submission [MARTE] is considered as the reference at this time. Implementing the profile required to adapt the specifications to the RSA metamodel. The following subsections highlight the differences between the MARTE specifications and the actual RSA profile. Details on particular problems encountered with RSA can be found in section 4.

3.1. Implementation notes

The MARTE profile was implemented for Rational Software Architect 7.0.0.

Once released, RSA does not allow a profile to be modified if the changes are not consistent with the released version. E.g. the entities can't be renamed. That can lead to some issues in case of mistakes discovered after the release of the profile. Anyway, one can modify directly the profile as plain text either to make small modifications or to unlock the released profile. Be very cautious while doing such changes : the profile must remain consistent.

To unlock the profile, open it with a text editor and find the section looking like :

```
<eAnnotations xmi:id="_UeYT4Dm8EdyFLpm0OTOjhA" source="uml2.versions">
   <details xmi:id="_0HaqEDwgEdy-A8rawEsbRQ" key="3" value="0.0.4"/>
   <details xmi:id="_EAzlgEAPEdyorvmbla6gFQ" key="4" value="0.0.6"/>
</eAnnotations>
```

Each details tag is a release of the profile. The key attributes matches the profile version that is released and the value attributes the corresponding release labels.

Delete the details tags from newer to older until the profile is unreleased up to the version you wish to alter. Then save the changes, open the file with the Profile Editor and make the desired changes.

While re-opening a model that has this profile applied, you should be asked to update the profile application since a newer version exists. Some errors can occur, note them carefully because you will have to fix them manually.

Example : to rename a stereotype <<*MySteretypeWithError*>> that was defined in the 2nd version of the profile, both details tags of the previous example has to be deleted. Then renaming <<*MySteretypeWithError*>> to <<*MyStereotypeWithoutError*>> is allowed. The model which uses this profile will give an error if <<*MySteretypeWithError*>> is applied. Just open the *emx* model file, find the *MySteretypeWithError* tags and rename them to *MyStereotypeWithoutError*.

If for some reasons, a released profile must be altered, keep in mind that the new version could be inconsistent with the previously released profiles.

3.2. Global changes

Due to technical limitations of the RSA tool and its metamodel, the MARTE profile was not implemented as defined in the second revised submission. This section aims at presenting the global changes that affected the entire profile.

3.2.1. Profile organization

Since RSA does not support:

- Hierarchical profiles,
- Applying a profile on itself,
- Packages in profiles,

To keep the hierarchical organization of MARTE, we had to split the profile into its sub-profile parts. Each one is defined as an independent profile in its own *epx* file. It allows us to apply or import profiles on other parts.

In each of these parts, the profile has been flattened because sub-packages are not supported. We can force a profile to have packages but the entities defined into can't be used while applying the profile.

For instance in the HRM profile,

- HwLogical,
- HwLogical \ HwGeneral,
- HwLogical \ HwComputing,
- HwLogical \ HwCommunication,

- HwLogical \ HwTiming,
- HwLogical \ HwStorage,
- HwLogical \ HwStorage \ HwMemory,
- HwLogical \ HwStorage \ HwStorageManager,
- HwLogical \ HwDevice,
- HwPhysical,
- HwPhysical \ HwGeneral,
- HwPhysical \ HwLayout,
- HwPhysical \ HwPower

Are merged at the root of the HRM profile.

Each profile <*profile_name*> is defined in a file *MARTE_*<*profile_name*>*.epx*. For instance, *RTEMoCC* is defined in *MARTE_RTEMoCC.epx*.

3.2.2. MARTE Library

Since RSA does not support:

- Model library imports into profiles,
- Other definitions than stereotype, class, enumeration in profiles (especially DataType and PrimitiveType),
- Packages in profiles,

We have chosen the following organization for the MARTE Library:

- A flat profile containing classes and enumerations which can be imported into profiles where needed. This profile is called InternalMarteLibrary.
- A full MARTE compliant implementation of the MARTE Library in a model library.
  This model library is called MARTELibrary.

The first part defines the names of the structure used in the profiles. It is only a partial implementation of the model library, only the entities used in the profiles are defined and nothing more. No properties are available.

Due to the above limitations of RSA, the PrimitiveTypes and the DataTypes are mapped to classes with the same names. Theses replacement classes have only one attribute named *value* of type *String* (UML Primitive Type). We chose this workaround for the external tools (e.g. an editor of properties) to have acces to the data. How the *value* field is used must be decided by each one of the external tools. The end user must not use this part.

The second part is the real implementation designed to be imported and used in the end-user models. Since it is a model and not a profile, no limitations prevent us from implementing the packages, DataTypes and PrimitiveTypes as defined in the second revised submission [MARTE].

It is important to notice that the links between the internal library (profile) and the MARTE Library entities are their names and not their IDs.

The following classes have been defined in the InternalMarteLibrary :

ArrivalPattern	Boolean	DateTime	Integer
IntegerMatrix	IntegerVector	IntervalNFP_Duration	IntervalNFP_Frequency
IntervalNFP_Integer	IntervalNFP_Natural	IntervalNFP_Real	IntervalReal
NFP_Area	NFP_Boolean	NFP_BoundedDuration	NFP_DataSize
NFP_DataTxRate	NFP_DateTime	NFP_Duration	NFP_Energy
NFP_Frequency	NFP_Integer	NFP_Interval	NFP_Length
NFP_Natural	NFP_Percentage	NFP_Power	NFP_Price
NFP_Real	NFP_String	NFP_Weight	Real
SchedParameters	ShapeSpecification	String	TilerSpecification
UnlimitedNatural	UtilityType	IntervalInteger	

The following enumerations have also been defined:

EventKind	ProtectProtocolKind	SchedPolicykind	TimeInterpretationKind
TimeNatureKind	TimeStandardKind	TransmModeKind	

3.2.3. Intervals

Since RSA does not support the use of template elements in profiles, the *Interval* class in the InternalMarteLibrary (see 3.2.2 for further explanations) could not be defined.

To address this issue, the *IntervalNFP_Frequency*, *IntervalNFP_Integer*, *IntervalNFP_Natural* and *IntervalReal* classes have been created to replace the intervals typed *NFP_Frequency*, *NFP_Integer*, *NFP_Natural* and *Real* that are used in the MARTE Profile. Moreover, it seemed also useful to add the *IntervalNFP_Duration*, the *IntervalNFP_Real* and the *IntervalInteger* classes.

Each one of theses classes defines only one attribute named *value* of type *String* (UML Primitive Type), as for the other classes in the internal library,

In order for the libraries to remain consistent, datatypes with the same name were added to the MARTE Library. These datatypes are similar to the Interval datatype :

- According to the datatype, a *bound* attribute with the correct type is defined,
- The datatype is stereotyped by <<IntervalType>>,
- The *bound* attribute is referenced in the *intervalAttrib* of the <<intervalType>> stereotype.

3.2.4. Stereotype attribute inheritance

Children elements do not inherit of stereotype attributes values. In MARTE, this mechanism is used in the *BasicNFP_Types* model library, thus stereotype attributes values have been redefined in the concerned classes.

Here is the list of the elements that redefine the stereotype attribute *exprAttrib = expr* (inherited from *NFP_CommonType*) :

NFP_Boolean	NFP_BoundedDuration	NFP_Natural	NFP_String	NFP_Real
NFP_Integer	NFP_DateTime			
NFP_Duration	NFP_DataTxRate	NFP_Frequency	NFP_Power	NFP_DataSize
NFP_Energy	NFP_Length	NFP_Area	NFP_Percentage	

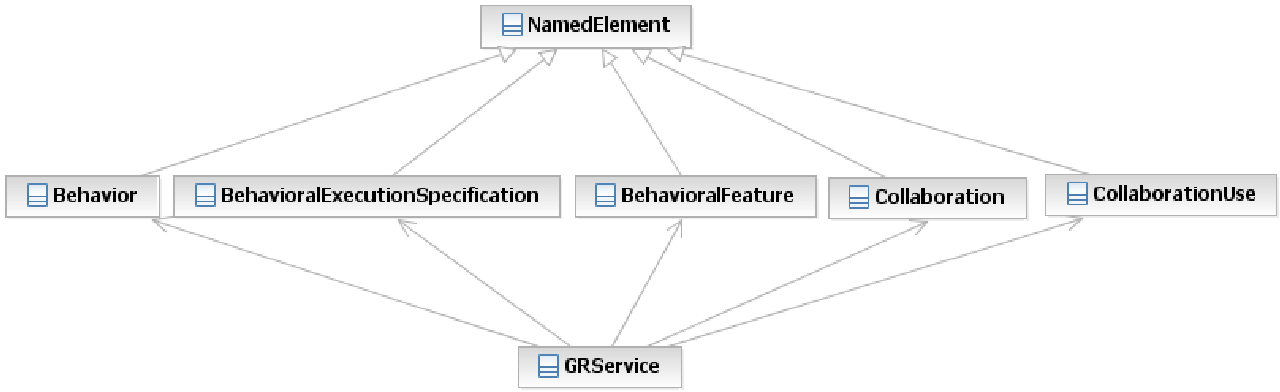
Here is the list of the elements that redefine the stereotype attribute *valueAttrib = value* (inherited from *NFP_Real*) :

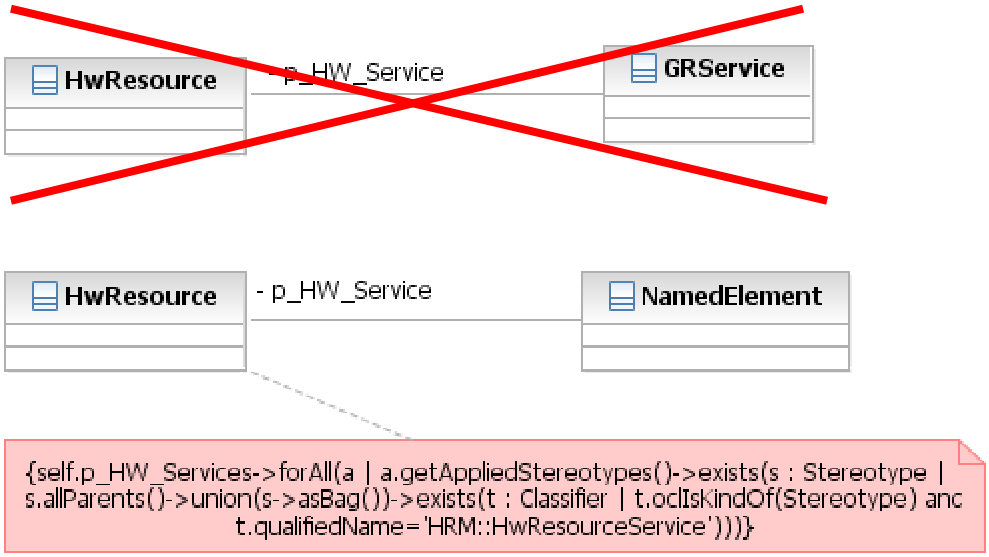
NFP_Duration	NFP_DataTxRate	NFP_Frequency	NFP_Power	NFP_DataSize
NFP_Energy	NFP_Length	NFP_Area	NFP_Percentage	

3.2.5. Stereotype aggregation

Since RSA does not support the use of stereotype as end of an aggregation link, we had to use the closest common ancestor in the tree inheritance of the extension metaclasses for these stereotypes.

For instance, the stereotype *<<GRService>>*, defined in the GRM profile, extends the metaclasses *BehavioralExecutionSpecification*, *BehavioralFeature*, *Behavior*, *Collaboration*, *CollaborationUse*. Each one of theses metaclasses inherits from the metaclass *NamedElement* and if a metaclass is a parent of all theses metaclasses, then *NamedElement* inherits from this one. That's why, the aggregation *p_HW_Services* that links the stereotype *HwResource* with the stereotype *HwResourceService* (that inherits from *GRService*) has been changed to a metaclass association with *NamedElement*.





Constraints are available to check that the entity pointed by the metaclass association has the correct stereotype applied.

These constraints are similar to :

```
self.attribute->forAll(a | a.getAppliedStereotypes()->exists(s : Stereotype |
    s.allParents()->union(s->asBag())->exists(t : Classifier |
        t.oclIsKindOf(Stereotype) and
        t.qualifiedName=TargetStereotype)))
```

where *attribute* is the name of the attribute or association and *TargetStereotype* is the qualified name (string) of the stereotype that have been replaced.

For instance, for the above example there is the following constraint:

```
self.p_HW_Services->forAll(a | a.getAppliedStereotypes()->exists(s : Stereotype |
    s.allParents()->union(s->asBag())->exists(t : Classifier | t.oclIsKindOf(Stereotype) and
    t.qualifiedName='HRM::HwResourceService')))
```

The following attributes / associations were modified:

Profile	Stereotype	Attribute / Association name	Target stereotype	Replacement metaclass
NFP	<<Unit>>	baseUnit	Unit	EnumerationLiteral
Time	<<Clock>>	type	ClockType	Class
		unit	Unit	EnumerationLiteral
	<<TimedElement>>	on	Clock	InstanceSpecification
GRM	<<GRService>>	owner	Resource	NamedElement
	<<MutualExclusionResource>>	scheduler	Scheduler	NamedElement
	<<ProcessingResource>>	mainScheduler	Scheduler	NamedElement
	<<ResourceUsage>>	usedResources	Resource	NamedElement

		subUsages	ResourceUsage	NamedElement
	<<SchedulableResource>>	dependentScheduler	SecondaryScheduler	NamedElement
		host	Scheduler	NamedElement
	<<Scheduler>>	host	ComputingResource	NamedElement
		processingUnits	ProcessingResource	NamedElement
		protectedSharedResources	MutualExclusionResource	NamedElement
		schedulableResources	SchedulableResource	NamedElement
	<<SecondaryScheduler>>	virtualProcessingUnits	SchedulableResource	NamedElement
Alloc	<<Allocate>>	impliedConstraint	NfpConstraint	Constraint
	<<ApplicationAllocationEnd>>	allocatedTo	ExecutionPlatformAllocationEnd	NamedElement
	<<ExecutionPlatformAllocationEnd>>	allocatedFrom	ApplicationAllocationEnd	NamedElement
RTEMoCC	<<rtf>>	tRef	TimedInstantObservation	TimeObservation
HRM	<<HwArbiter>>	controlledMedias	HwMedia	NamedElement
	<<HwBridge>>	sides	HwMedia	NamedElement
	<<HwComponent>>	subComponents	HwComponent	NamedElement
	<<HwDMA>>	drivenBy	HwProcessor	NamedElement
	<<HwDrive>>	buffer	HwRAM	NamedElement
	<<HwEndPoint>>	connectedTo	HwMedia	NamedElement
	<<HwMedia>>	arbiters	HwArbiter	NamedElement
	<<HwMMU>>	ownedTLBs	HwCache	NamedElement
	<<HwPLD>>	blocksComputing	HwComputingResource	NamedElement
		blocksRAM	HwRAM	NamedElement
	<<HwProcessor>>	predictors	HwBranchPredictor	NamedElement
		caches	HwCache	NamedElement
		ownedMMUs	HwMMU	NamedElement
		ownedISAs	HwISA	NamedElement
	<<HwResource>>	ownedHW	HwResource	NamedElement
		p_HW_Services	HwResourceService	NamedElement
		r_HW_Services	HwResourceService	NamedElement
		endPoints	HwEndPoint	NamedElement
	<<HwStorageManager>>	managedMemories	HwMemory	NamedElement
	<<HwTimer>>	inputClock	HwClock	NamedElement
GQAM	<<GaAcqStep>>	acqRes	Resource	NamedElement
	<<GaAnalysisContext>>	platform	GaResourcesPlatform	Classifier

		workload	GaWorkloadBehavior	NamedElement
	<<GaEventTrace>>	stream	GaWorkloadEvent	NamedElement
	<<GaRelStep>>	relRes	Resource	NamedElement
	<<GaResourcesPlatform>>	resources	Resource	NamedElement
	<<GaScenario>>	cause	GaWorkloadEvent	NamedElement
		steps	Step	NamedElement
		root	GaStep	NamedElement
		timing	GaTimingObs	Constraint
	<<GaStep>>	behavior	GaScenario	NamedElement
		concurRes	SchedulableResource	NamedElement
		host	GaExecHost	NamedElement
		servDemand	GaRequestedService	NamedElement
	<<GaTimingObs>>	startEvent	TimedInstantObservation	InstanceSpecification
		endEvent	TimedInstantObservation	InstanceSpecification
	<<GaWorkloadBehavior>>	behavior	GaScenario	NamedElement
		demand	GaWorkloadEvent	NamedElement
	<<GaWorkloadEvent>>	generator	GaWorkloadGenerator	NamedElement
		trace	GaEventTrace	NamedElement
SAM	<<SaEnd2EndFlow>>	timing	GaTimingObs (Instead of TimingObserver)	Constraint
PAM	<<PaCommStep>>	concurResource	SchedulableResource	NamedElement
	<<PaResPassStep>>	resource	Resource	NamedElement
	<<PaRunTInstance>>	instance	SchedulableResource	NamedElement
		host	GaExecHost	NamedElement
	<<PaStep>>	behavDemands	GaScenario	NamedElement

3.2.6. OCL rules and constraints

No constraints defined in the second revised submission [MARTE] are implemented in the profile. The end user has to check by himself the validity of the model and the data he entered.

3.2.7. Implicit information

All the implicit information or undefined points of MARTE have been let to the default values of RSA. It concerns mainly the multiplicities. In some parts of [MARTE], every attribute has its count made explicit, but in

other parts no data are provided. We can infer the values from the rest of the specification but as it is not expressed we choose to keep the default values of RSA.

3.2.8. Shapes and icons

This profile includes all the drawing defined in [MARTE]. These shapes concern SRM and HRM. Some other custom shapes for RTEMoCC, Time and GRM were added. All corresponding icons are included.

The following stereotypes are concerned by the addition of custom images:

PpUnit	RtUnit	Clock	ClockType	TimedDomain
ClockResource	CommunicationEndPoint	CommunicationMedia	ComputingResource	ConcurrencyResource
DeviceResource	MutualExclusionResource	Resource	ProcessingResource	SchedulableResource
Scheduler	SecondaryScheduler	StorageResource	SynchronizationResource	TimerResource

3.3. Punctual Changes

We have to consider two kinds of causes for the changes made to MARTE in order for it to be suitable for the implementation and the use on RSA:

- The limitations of RSA
- The issues of the MARTE specification

3.3.1. Caused by the limitations of RSA

3.3.1.1. Profile HRM

3.3.1.1..1 CacheStructure

Since RSA does not allow the definition and the use of DataTypes in profile, *`Cache Structure`* was defined as a Class.

3.3.1.1..2 Env_Condition

Since RSA does not allow the definition and the use of DataTypes in profile, *`Env_Condition`* was defined as a Class.

The type of the attribute *`range`* was set to *`IntervalReal`*, class defined in the InternalMarteLibrary, instead of *`Interval<T->Real>`*, because RSA does not allow the use of templates in profiles.

3.3.1.1..3 <<HwComponent>>

The type of the attribute *`position`* was set to *`IntervalNFP_Natural`*, class defined in the InternalMarteLibrary, instead of *`Interval<NFP_Natural>`*, because RSA does not allow the use of templates in profiles.

3.3.1.1..4 <<HwComputingResource>>

The type of the attribute *op_frequencies* was set to *IntervalNFP_Frequency*, class defined in the InternalMarteLibrary, instead of *Interval<NFP_Frequency>*, because RSA does not allow the use of templates in profiles.

3.3.1.1..5 <<HwI/O>>

This stereotype has been renamed *HwIO* because "/" are forbidden in RSA names.

3.3.1.1..6 MemoryOrganization

Since RSA does not allow the definition and the use of DataTypes in profile, *MemoryOrganization* was defined as a Class.

3.3.1.1..7 PLD_Organization

Since RSA does not allow the definition and the use of DataTypes in profile, *PLD_Organization* was defined as a Class.

3.3.1.1..8 Timing

Since RSA does not allow the definition and the use of DataTypes in profile, *Timing* was defined as a Class.

3.3.1.2. MARTE_Library

3.3.1.2..1 TimedValueType

The template type *TUK* is set to Class.

3.3.1.2..2 Osek/VDX

All the names containing *Osek/VDX* are renamed *OsekVDX* because RSA does not allow the use of "/" in names.

3.3.2. Caused by the issues of the MARTE profile

3.3.2.1. Profile GCM

3.3.2.1..1 <<MsgPort>>

This stereotype extends *Port*, as defined in the schema 11.6 of [MARTE] since no extensions or generalizations are provided in its definition.

3.3.2.1..2 <<ServiceSpecification>>

This stereotype extends *Interface*, as defined in the schema 11.6 of [MARTE] since no extensions or generalizations are provided in its definition.

3.3.2.1..3 <<SignalSpecification>>

This stereotype extends *Interface*, as defined in the schema 11.6 of [MARTE] since no extensions or generalizations are provided in its definition.

3.3.2.2. Profile GQAM

3.3.2.2..1 <<GaAnalysisContext>>

There is a conflict in this stereotype definition between the text and the schema. To remain consistent with the use of this stereotype, we kept the schema definition: the attributes *platform* (type : *GaResourcesPlatform*) and *workload* (type : *GaWorkloadBehavior*) are defined.

3.3.2.2..2 <<GaScenario>>

There is a conflict in this stereotype definition between the text and the schema. To remain consistent with the use of this stereotype, we kept the schema definition: the attributes *cause* (type : *GaWorkloadEvent*) and *timing* (type : *GaTimingObs*) are defined.

3.3.2.3. Profile HRM

3.3.2.3..1 <<HwPLD>>

Types of *nbLUTs*, *nbLUTs_Inputs*, *nbFlipFlops* set to *NFP_Natural* since they are not defined in the specification.

3.3.2.3..2 <<HwResourceService>>

This stereotype inherits from *GRService* instead of *ResourceService*, as the latter is not defined.

3.3.2.4. Profile SAM

3.3.2.4..1 <<SaEnd2EndFlow>>

The timing attribute is typed *GaTimingObs* instead of *TimingObserver* as the latter is not defined.

3.3.2.4..2 <<SaCommStep>>

This stereotype inherits from *GaCommStep* instead of *CommStep*, as the latter is not defined.

3.3.2.4..3 <<SaExecHost>>

The type of the attribute *ISRprioRange* was set to *IntervalInteger*, class defined in the InternalMarteLibrary, instead of *NFP_IntegerInterval*, because this type is not defined.

3.3.2.4..4 <<SaSchedObs>>

This stereotype inherits from *GaTimingObs* instead of *TimingObs*, as the latter is not defined.

3.3.2.5. Profile VSL

3.3.2.5..1 <<IntervalType>>

There is a conflict in this stereotype definition between the text and the schema. To remain consistent with the use of this stereotype in the MARTELibrary, we kept the schema definition: the attribute *intervalAttrib* is defined, but *minAttrib* and *maxAttrib* are not.

3.3.2.6. MARTE_Library

3.3.2.6..1 NFP_Weight & NFP_Price

These classes are defined in the InternalMarteLibrary for compatibility reasons but not in the MarteLibrary, as defined in [MARTE].

3.3.2.6..2 FrequencyUnitKind

All *baseUnit* are set to *Hz* instead of *W*.

3.3.2.6..3 LengthUnitKind

The *baseUnit* of the *mm* literal is set to *m* instead of *bits*.

3.3.2.6..4 AreaUnitKind

The *baseUnit* of the *um2* literal is set to *mm2* instead of *bits*.

3.3.2.6..5 Interval

The type of the template *T* is *Class* instead of *StringExpression*.

3.3.2.6..6 TimeLibrary::TimeUnitKind & TimeLibrary::LogicalTimeUnitKind

This two enumerations are not defined since are not defined since *MeasurementUnits::TimeUnitKind* is already defined.

3.3.2.6..7 OsekVDX_Platform::OsekVDXLibrary::Concurrency::ConcurrencyCore::Interrupt

The undefined stereotype attributes are left blank.

3.3.2.6..8 ARINCLibrary, ConcurrencyDataTypes

The *name* attribute of the DataType *ProcessAttribute_Type* is typed *ProcessNameType* instead of *Process*.

3.3.2.6..9 ARINCLibrary, CommunicationCore

The attributes of *Buffer*, *SamplingPort* and *QueuingPort* are set static to remain consistent with the attributes of the other classes.

4. KNOWN BUG WITH RSA

When targeting an element from an association of a stereotype, in some conditions, the targeted element could have the same stereotype applied onto it. This behavior is abnormal and is considered as a bug. The range of affected elements has been reduced but some stereotypes remain concerned with this issue in the current implementation.

Further explanations of this bug is given in ANNEXE A.

ANNEXE A "Stereotype applied to end of association" bug

1. BUG DESCRIPTION, CONTEXT AND CONSEQUENCES

This bug has two forms with different level of consequences.

The major form can occur when targeting an element from an association of a stereotype : the targeted element can – under certain conditions – have the stereotype applied onto it without requesting this action and the application can't be undone without un-applying the stereotype from the initial element.

For instance, if we consider the model of the Figure 1, setting *MyClockInstance* in the *on* association field of the *<<GaScenario>>* stereotype of *MyClass* will apply the *<<GaScenario>>* stereotype to *MyClockInstance*. The changes made to the *<<GaScenario>>* stereotype of *MyClockInstance* are reported to the one on *MyClass*.

The minor version can occur in the same conditions but, when targeting an element, the list of the association remains empty or at least out of date until a force refresh occurs. Using the same example with the minor version will give : when setting *MyClockInstance* in the *on* association field of *<<GaScenario>>* the list remains empty until selecting another element (e.g. *MyClock*) and coming back to *MyClass*.

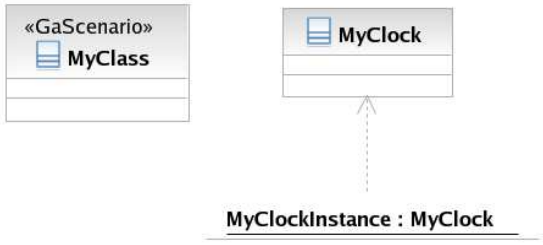


Figure 1 - Example model for bug expression

Unfortunately, it seems that this bug's expression is favored by the « Stereotype Aggregation » changes. See 3.2.5 for further information.

As far as we know the minor version of this bug does not have serious consequences. Considering the major one, there is two main points :

- Unexpected stereotype applications can occur.
- They can occur on forbidden or at least undesirable spots.

2. BUG ANALYSIS

The major version of this bug was first found in the MARTE Library. Within the *ARINCLibrary* the *QueuingPort* class is defined in the *Interaction::Communication* package. This class is stereotyped

<<MessageComResource>>. It was noticed that the *createServices* field was forcing the *createQueuingPort* operation to have the stereotype applied. Later, similar behaviors were noticed with the *on* field of *<<GaScenario>>*.

First we concentrate on the known expression configurations of the bug. We were able to produce the following diagrams where a red stereotype denotes a present bug and a green one a normal behavior.

Note : the *createServices* (inherited from *SwResource*) and the *on* (inherited from *TimedElement*) fields were concerned.

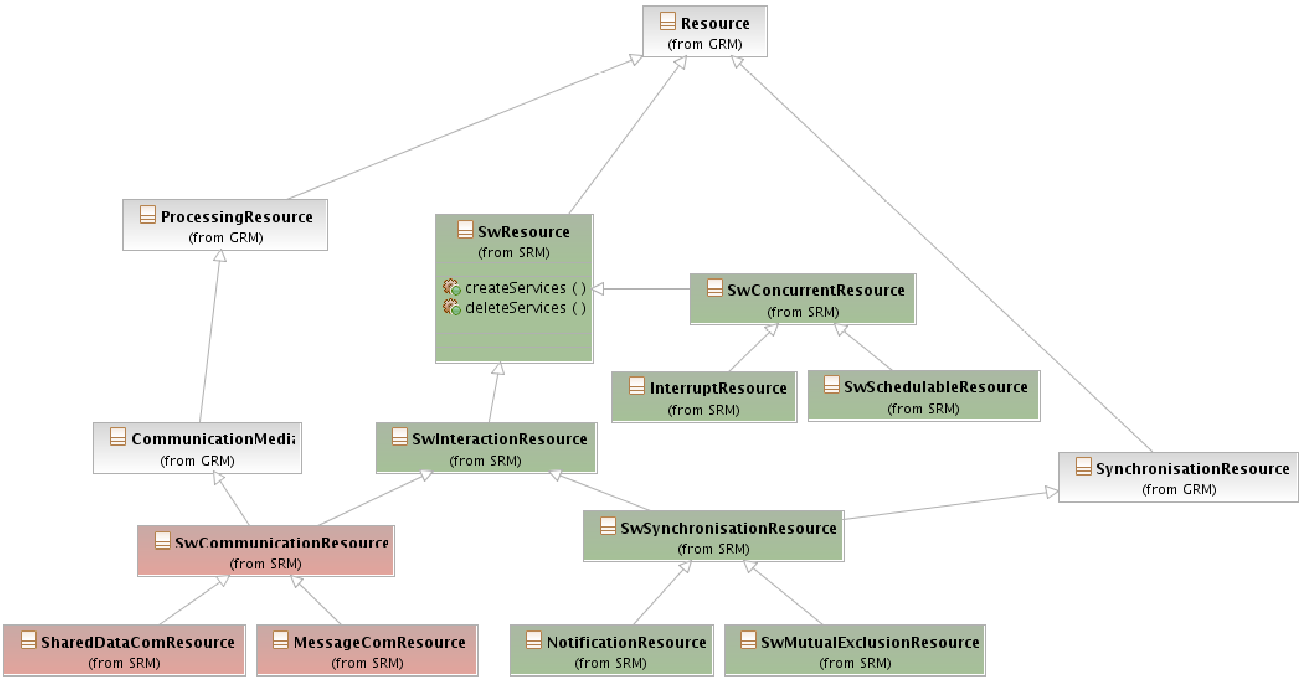


Figure 2 - The <<SwCommunicationResource>> configuration

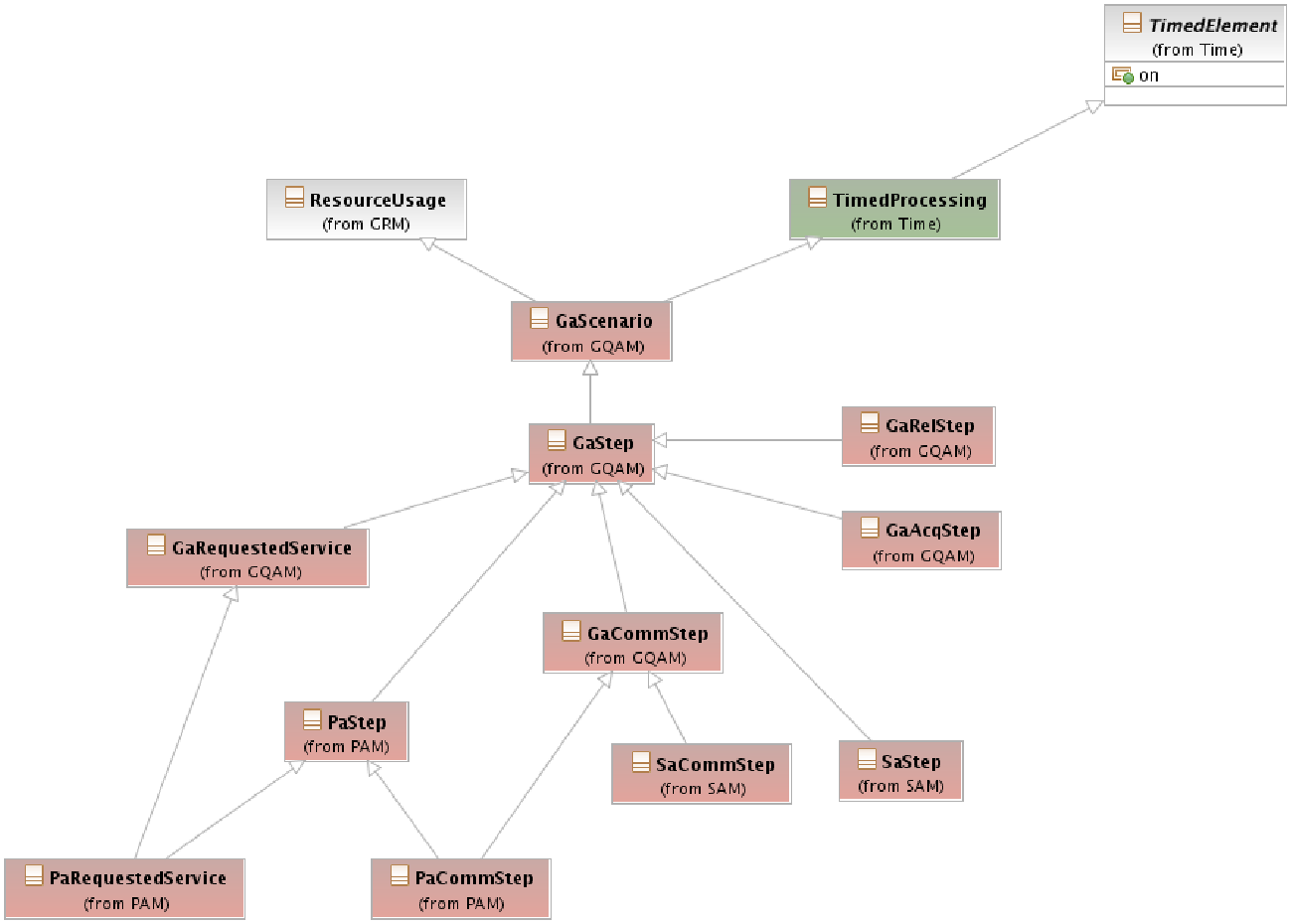


Figure 3 - The <<GaScenario>> configuration

Using the highlighted similarities of the situations, many different profiles were build to first reproduce the bug occurrence and then to understand how it worked.

All theses profiles were based upon the following configuration :

Various combinations of metaclass extensions, metaclass associations, multiplicities, generalizations, etc. were produced and checked for bug occurrences.

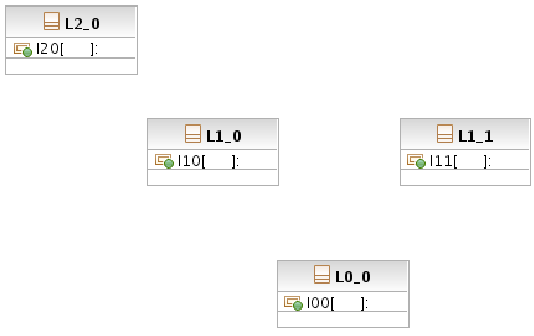


Figure 4 - Analysis base model

It was declined into several versions to identify the influencing factor, that's why no particular information are shown

Caution :

- Many facts were noticed but often all characteristics are not fully expressed.
- All data collected are based upon observations and thus are not rigorously demonstrated.

The value of these observations is measured by the validity of the predictions based upon them and of the bug fixes issued from them.

The following facts were noticed :

- If the bug occurs, all stereotypes inheriting from it are concerned.
- The bug occurs on stereotypes that own several generalizations. The name “branch” is used to refer to the whole inheritance tree (all parents stereotypes) leading to a particular generalization. More precisely:
    - One generalization : the branch is thought to be « healthy »
    - Two generalizations : the first defined branch can be affected whereas the second branch is thought to be « healthy ». The first defined branch is the last one in the generalization list of the UML properties dialog box. It is also the last one in the *epx* file.
    - More generalizations : no precise rule are known. The generalization tree, the positions of the metaclass associations and of the other members, the metaclass extensions have an influence on the affected branches.
- On an affected branch, the metaclass associations with an infinite multiplicity can be affected, and :
    - If only one association of this kind is present, the major version occurs.
    - In case of multiple associations of this kind or associations mixed with other elements there is no rule to determine which ones remain unaffected or which one are affected by the bug. E.g in Figure 2, from the *<<SwCommunicationResource>>* stereotype point-of-view only *createServices* has the major version of the bug, other associations either are clean or are affected by the minor version.
    - When a metaclass association is affected, there is no precise rule to determine which version occurs. E.g. see Figure 2, *<<SwSynchronizationResource>>* could have been affected as well as *<<SwCommunicationResource>>* but there is no traces of the bug. A lot of factors seem to have an influence and we failed to find a working rule.
- The bug seems to interfere with the internal representation of the model in RSA :
    - Modifying the unexpected stereotype will report the changes to the initial stereotype
    - Listing the stereotype applied by using the API will give the unexpected stereotype whereas the *emx* file remains clean.

3. BUG TRACKING & FIXING

The observations made were used to predict the bug occurrences and when possible to fix it.

3.1. General case

A python script was written to detect the spots where the bug could occur and if it can easily be fixed directly from the epx files. The most common way to fix a bug occurrence is to switch the order of the generalizations when the other branch contains no potential expression spots. See 4.1.4.1 for further information.

The following table shows the spots (stereotype fields) that were pointed out by the script and the ones that were fixed by “branch switching”. Only the first stereotypes in the generalization tree are listed below: e.g. <<*GaScenario*>> appear once but has 10 affected children (see Figure 3). There is a special notice for <<*PaRequestedServices*>> and <<*PaCommStep*>> because they inherit from <<*GaScenario*>> but can also be affected as well since every appearance conditions are fulfilled (see Figure 3).

Profile	Stereotype	Association name	Association type	Final state (after correction)
SRM	<<SwCommunicationResource>>	waitingPolicyElements	TypedElement	-
		identifierElements	TypedElement	-
		stateElements	TypedElement	-
		createServices	BehavioralFeature	-
		deleteServices	BehavioralFeature	-
		initializeServices	BehavioralFeature	-
	<<SwSynchronisationResource>>	waitingPolicyElements	TypedElement	-
		identifierElements	TypedElement	-
		stateElements	TypedElement	-
		createServices	BehavioralFeature	-
		deleteServices	BehavioralFeature	-
		initializeServices	BehavioralFeature	-
GQAM	<<GaScenario>>	**on**	**InstanceSpecification**	**Major**
Time	<<TimedConstraint>>	on	InstanceSpecification	-
HRM	<<HwComputingResource>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
	<<HwMemory>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-

		endPoints	NamedElement	-
	<<HwTimingResource>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
	<<HwStorageManager>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
	<<HwMemory>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
	<<HwDMA>>	managedMemories	NamedElement	Minor
		ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
	<<HwMedia>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
	<<HwEndPoint>>	ownedHW	NamedElement	-
		r_HW_Services	NamedElement	-
		p_HW_Services	NamedElement	-
		endPoints	NamedElement	-
RSM	<<Distribute>>	impliedConstraint	Constraint	-
PAM	<<PaRequestedService>>	behavDemands	NamedElement	Minor
		servDemand	NamedElement	-
		steps	NamedElements	-
		timing	Constraint	-
		on	**InstanceSpecification**	**Major**
		usedResources	NamedElement	-
		subUsages	NamedElement	-

	<<PaCommStep>>	behavDemands	NamedElement	-
		servDemand	NamedElement	-
		steps	NamedElement	-
		timing	Constraint	-
		on	**InstanceSpecification**	**Major**
		usedResources	NamedElement	-
		subUsages	NamedElement	-

In short, initially, of the 62 expressions spots that could be concerned:

- 27 were clean,
- 27 were affected by the minor version,
- 8 were affected by the major version (5 different associations).

The bug analysis led to fix 25 minor and 5 major occurrences. Some expression spots cannot be fixed by "branch switching" since it could lead to make things worse, of the 20 that remain :

- 17 are clean
- 2 are affected by the minor version
- 3 are affected by the major version (one different association : see the *on* case in 4.1.3.2.1).

The accuracy of the script is low (around 56% of effective presence of the bug in the case of the initial profiles) but we do not know the exact working rules for this bug, especially concerning the affecting form. It is important to keep in mind that that the made observations could also have led to incomplete detections.

3.2. Special cases

3.2.1. <<TimedElement>> and the on association

The association named *on* of the <<*TimedElement*>> stereotype occurs 3 times as a major version of the bug expression and remains unfixed. It concerns the stereotypes <<*GaScenario*>>, <<*PaRequestedServices*>>, <<*PaCommStep*>> and their children (see Figure 3).

This major occurrence cannot be fixed by "branch switching" since this way things are made worse : the *subUsages* and *usedResources* associations of the <<*ResourceUsage*>> stereotype are also affected by this version of the bug.

Workaround:

No way to fix the bug in the profile was found, but if you are concerned you can use this workaround :

Define the instances that should be pointed by the on association (and their base class) in an separate model without importing it into the real model. After closing the two models, re open the separate model first, it should be spared by the bug occurrence.

3.2.2. <<PaRequestedService>> and <<PaCommStep>>

These stereotypes are affected by the case 4.1.3.2..1 and the workaround depicted there can be used. But they also present other bug opportunities since both of them inherits from two stereotypes (see Figure 3).

The same scenario as in 4.1.3.2..1 happens here: switching the branches does not improve anything. Since the only major concerns are the on association, nothing more was done for this special case.

4. RESOURCES

4.1. Analysis script

As already said, a small script was written to perform a small analysis of the models, find the spots where the bug could occur.

The command syntax is :

```
bbbb-check.py [-a] [files]
```

(where *bbbb-check.py* is the name of the script).

If files are specified (their extension should be *.epx*) the script will perform it's analysis only on the specified files, otherwise all profile files in the current working directory will be concerned.

The *–a* switch indicates that a more advanced analysis should be performed : all branches are analyzed and some clue about fixing it (by "branch switching") are displayed. Some data are also extracted from the profiles but not used for the time being. These data could be used latter to give more precise information on the bug occurrences.

The source code is available in annex A. The python language was chosen for the ease of its use (especially the reading of *epx* files), its portability and its adaptation faculties.

Here is some examples of the results obtained. The potential spots for the bug to be present are highlighted with >>>.

In the case of the <<SwCommunicationResource>> :

```
PROFILE SRM


  <<SwCommunicationResource>>
    ( i ) Generalization branch: GRM/CommunicationMedia (MARTE_GRM.epx#_l4oV4DUqEdyDId_-RZ-V8A)
    Generalization : CommunicationMedia (_l4oV4DUqEdyDId_-RZ-V8A)
      has generalization : ProcessingResource (_XGUYIDUqEdyDId_-RZ-V8A)
    Generalization : ProcessingResource (_XGUYIDUqEdyDId_-RZ-V8A)
      has generalization : Resource (_gD8-UDUmEdyDId_-RZ-V8A)
    Generalization : Resource (_gD8-UDUmEdyDId_-RZ-V8A)
    Found spot(s) : 0
    / ! \ Potential problem found with generalization branch: SwInteractionResource
(_L7PzkDU6EdyDId_-RZ-V8A)
    Generalization : SwInteractionResource (_L7PzkDU6EdyDId_-RZ-V8A)
>>>   has association : waitingPolicyElements -> TypedElement
      has generalization : SwResource (_gadcADU1EdyDId_-RZ-V8A)
    Generalization : SwResource (_gadcADU1EdyDId_-RZ-V8A)
```

```
>>>   has association : identifierElements -> TypedElement
>>>   has association : stateElements -> TypedElement
>>>   has association : createServices -> BehavioralFeature
>>>   has association : deleteServices -> BehavioralFeature
>>>   has association : initializeServices -> BehavioralFeature
      has generalization : Resource (MARTE_GRM.epx#_gD8-UDUmEdyDId_-RZ-V8A)
    Generalization : Resource (_gD8-UDUmEdyDId_-RZ-V8A)
    Found spot(s) : 6
    Analysis :
      The bug could be present in the previous configuration.
      Since in the other branch there is no expression spots, consider switching the branchs
```

In the case of the <<GaScenario>> :

```
PROFILE GQAM


  <<GaScenario>>
    ( i ) Generalization branch: GRM/ResourceUsage (MARTE_GRM.epx#_OcjloDUtEdyDId_-RZ-V8A)
    Generalization : ResourceUsage (_OcjloDUtEdyDId_-RZ-V8A)
>>>   has association : usedResources -> NamedElement
>>>   has association : subUsages -> NamedElement
    Found spot(s) : 2
    / ! \ Potential problem found with generalization branch: Time/TimedProcessing
(MARTE_Time.epx#_zYzvoDUlEdyDId_-RZ-V8A)
    Generalization : TimedProcessing (_zYzvoDUlEdyDId_-RZ-V8A)
      has generalization : TimedElement (_4rGQADUiEdyDId_-RZ-V8A)
    Generalization : TimedElement (_4rGQADUiEdyDId_-RZ-V8A)
>>>   has association : on -> InstanceSpecification
    Found spot(s) : 1
    Analysis :
      The bug could be present in the previous configuration.
      Try switching the branchs at your own risks !
```

4.2. Python script sources

Here is the content of bbbb-check.py :

```
#!/usr/bin/python
# bbbb-check
# Script for checking the "end of association" bugs in RSA profiles
# Copyright 2007 Thales Research and Technology
# By Nicolas Vienne

# Imports
import xml.dom.minidom          # Parsing the EPX files
import os                       # Directory, file operations
import sys                      # Command line

# Global Data
report=[]         # The report is produced as a list of strings (for exporting to file)
```

```
option_list = ["-a"]     # recognized switches for the cmd line

# User defined exception
fe = "Fatal Error"

########################################################################
# Tool functions

# Delimiters
def delim():
   report.append("\n\n")

# Load the epx file
def loadEPX(f):
   return xml.dom.minidom.parse(f)

# Convert a reference to a name
def ref2name(s):
   return s.split('#')[0]

# Generic attribute getter
def getElementAttribute(element, att):
   return element.attributes.get(att).nodeValue

# Id getter
def getElementId(element):
   return getElementAttribute(element, 'xmi:id')

# Name getter
def getElementName(element):
        return getElementAttribute(element, 'name')

# Retrieve elements by id, open - if necessary - the corresponding profile
def getElementById(selfdoc, full_id):
   id = full_id
   doc = selfdoc
   full_id2 = full_id.split('#')
   if(len(full_id2) == 2):
          id = full_id2[1]
          doc = loadEPX(full_id2[0])
   packagedElements = doc.getElementsByTagName('packagedElement')
   for p in packagedElements:
          if p.attributes.get("xmi:id").nodeValue == id:
                 return p
   raise fe, "Element " + full_id + " not found !"

# Retrieve the information of a generalization : the parent name and its id.
def getGenInfo(g):
```

```
   parent_name=""
   parent_id=""
   if g.hasAttribute('general'):        # Heritage interne
                parent_id = g.attributes.get('general').nodeValue
                parent_name = getElementName(getElementById(g.ownerDocument, parent_id))
   else:                                # Heritage externe
                gentags = g.getElementsByTagName('general')
                if len(gentags) == 1:
                        href = gentags[0].attributes.get('href').nodeValue
                        href_split = href.split('?')
                        parent_name = href_split[1]
                        parent_id = href_split[0]
                else:
                        raise fe, "Generalization error in <<"+name+">>"
   return [parent_name, parent_id]

#########################################################################
# Processing functions

# Build some data for further analysis (not used)
# Parent stereotype lists, the associations list and the links between the two.
def buildAnalysis(linklist):
   stereotypes={}
   st_li={}
   links=[]

   for link in linklist:
          links.append([link[2], link[3], link[4]])
          stereotypes[link[0]] = link[1]
          if st_li.has_key(link[0]):
                 st_li[link[0]].append(link[2])
          else:
                 st_li[link[0]]=[link[2]]

# Association management
# Retrieve all associations in a branch where the bug could happen
def handleLinks(s, p_id):
   linklist = []
   elem = getElementById(s.ownerDocument, p_id)# Retrieving the root parent of the branch
   todolist = [elem]                           # List of the parents to analyse
   doneids = []                                # IDs of the parents already processed
   problem_count = 0;                          # Number of problems found in this branch
   while(len(todolist) > 0):
          elem = todolist.pop()                # Removing the current element of the todo list
          id = getElementId(elem)              # Retrieving ID and Name
          name = getElementName(elem)
          report.append("    Generalization : " + name + " ("+id+")")
          if(id not in doneids):               # If the element has already been processed, ignoring it
```

```
                doneids.append(id)            # The element has been done
                # Analyzing associations
                oa = elem.getElementsByTagName("ownedAttribute")
                for e in oa: # For each owned attributes
                        if e.hasAttribute("association"):
                                if(len(e.getElementsByTagName("upperValue"))==1):
                                        if
e.getElementsByTagName("upperValue")[0].attributes.get('value').nodeValue=="*":
                                                # if it is an association with infinite multiplicity
                                                link_name = getElementName(e)
                                                link_id = getElementId(e)
                                                link_type = ""
                                                types = e.getElementsByTagName("type")
                                                if(len(types)==1):
                                                        t = types[0].attributes.get('href').nodeValue
                                                        link_type = t.split('#')[1]
                                                # It could be a problem
                                                # > adding it to the report and the problems count
                                                linklist.append([id, name, link_id, link_name, link_type])
                                                report.append(">>>   has association : " + link_name + " -> "
+ link_type)
                                                problem_count += 1
                # Finding parents
                ps = elem.getElementsByTagName("generalization")
                for g in ps: # For each generalizations
                         parent_id=""
                        # Retrieving the parent_id
                        if g.hasAttribute('general'):        # (profile) internal generalization
                                parent_id = g.attributes.get('general').nodeValue
                        else:
                                gentags = g.getElementsByTagName('general')
                                if len(gentags) == 1:        # (profile) external generalization
                                        parent_id =
gentags[0].attributes.get('href').nodeValue.split('?')[0]
                                else:
                                         raise fe, "Generalization error in <<"+name+">>"
                        curr_parent = getElementById(elem.ownerDocument, parent_id)       # Retrieving the
parent
                        todolist.append(curr_parent)                                      # Adding it to
the todo list + to the report
                        report.append("      has generalization : " + getElementName(curr_parent) + "
("+parent_id+")")
   return [linklist, problem_count]

# Branch management
# Format report + branch analysis
def handleProblem(s, parent_name, parent_id, isProblem=True):
   if isProblem:
           report.append("    / ! \ Potential problem found with generalization branch: " + parent_name + "
(" + parent_id + ")")
   else:
```

```
            report.append("    ( i ) Generalization branch: " + parent_name + " (" + parent_id + ")")
      ll = handleLinks(s, parent_id)                        # Branch analysis for the parent with id parent_id of
stereotype s
      # buildAnalysis(ll[0])                                # Further analysis disabled
      report.append("    Found spot(s) : "+str(ll[1]))      # Printing the problem count to the report
      return ll[1]


# Stereotype management
# Analyze the banches and possibly make a small report
def handleStereotype(s):
      name = s.attributes.get('name').nodeValue
      gens = s.getElementsByTagName('generalization')     # Retrieving the stereotypes
      if len(gens) >= 2:                                  # Two or more associations : Potential problems
            report.append("\n  <<"+name+">>")
            if len(gens) > 2:                             # If more than 2 associations : no rules known for
determining the affected branches, all are considered as a problem
                  report.append("\n    Unable to determine the affected branches !")
                  for k in gens:
                        i = getGenInfo(k)
                        handleProblem(s,i[0], i[1])
            else:                                         # Two branches, the first defined is affected, the
other is clean. It is the reverse order in the epx file ! branchX = count of potentially affected spots in
the branch n°X
                  branch1 = 999;
                  if "-a" in sys.argv:                    # If -a, analysis the clean branch to find the
potential problems for switching
                        i = getGenInfo(gens[0])
                        branch1 = handleProblem(s,i[0], i[1], False)
                  i = getGenInfo(gens[1])                 # Analysis of the affected branch
                  branch2 = handleProblem(s,i[0], i[1])
                  if "-a" in sys.argv:                    # Analysis report
                        report.append("    Analysis :")
                        if branch2 > 0:
                              report.append("      The bug could be present in the previous
configuration.")
                              if branch1 == 0:
                                    report.append("      Since in the other branch there is no
expression spots, consider switching the branchs !")
                              elif branch1 < branch2:
                                    report.append("      Switching the branchs could be worth a try but
do not expect too much of it.")
                              else:
                                    report.append("      Try switching the branchs at your own risks !")


                        else:
                              report.append("      No expression spots were found ! Everything is OK !")



# Retrieving the stereotypes in the file
def handlePackagedElements(pe):
      type = pe.attributes.get('xmi:type').nodeValue
```

```
   if type=="uml:Stereotype":
          handleStereotype(pe)

def handleProfile(p):
   packelems = p.getElementsByTagName("packagedElement")
   for e in packelems:
          handlePackagedElements(e)
   delim()

def handleXMI(doc):
   profiles = doc.getElementsByTagName("uml:Profile")
   for p in profiles:
          profile_line = "PROFILE "+p.attributes.getNamedItem('name').nodeValue
          report.append(profile_line)
          handleProfile(p)

########################################################################
# MAIN
# Analyzing the parameter files (determining if the entire directory must be analyzed)
hasFileParam = False
for k in sys.argv[1:]:
   if k not in option_list:
          hasFileParam = True
          handleXMI(loadEPX(k))

# If the whole dir is concerned, finding the epx file and analyzing them
if not hasFileParam:
   for fichier in os.listdir("."):
               if(os.path.isfile(fichier) and os.path.splitext(fichier)[1]==".epx"):
                      handleXMI(loadEPX(fichier))

# Printing report
print "========================= REPORT =========================\n"
for l in report:
   print l
```