THALES

Research & Technology

Report / Technical Document

MARTE TO CHEDDAR TRANSFORMATION USING ATL

DEPARTMENT / SERVICE	DOCUMENT NUMBER	PAGE
SWRG/LSE	61565546-179	1/80
	-	REVISION

DOCUMENT CONTROL

	- the : 17/10/2007	A the:	B the:	C the:	D the:
Written by Signature	Eric MAES, Nicolas VIENNE				
Approved by Signature					

Revision index	Modifications
A	
B	
C	
D	

CONTENTS

Pages

1. PURPOSE

This document explains how can MARTE concepts can be matched to cheddar concepts in order to do analysis on models and proposes model transformation solutions using ATL

Implementation of the transformation has been done under the following configuration :

Name	Version	Date
Cheddar (http://beru.univ-brest.fr/~singhoff/cheddar/Cheddar-2.0-win32-bin.zip)	2.0	12/02/2007
ATL (http://www.sciences.univ-nantes.fr/lina/atl/www/atldemo/adt_am3.zip)		16/02/2007
RSA	7.0.0	
MARTE Profile for Rational Software Architect (RSA) 7.0 (http://www.omgmarte.org/Documents/MARTE4RSA/MARTE_RSA_1.0.0.zip)	1.0.0	01/11/2007

2. DOCUMENTS

2.1. MANDATORY

2.2. REFERENCE

Cheddar site	http://beru.univ-brest.fr/~singhoff/cheddar/index-fr.html
Cheddar user's guide	http://beru.univ-brest.fr/~singhoff/cheddar/ug/cheddar-r2.html
Eclipse ATL site	http://www.eclipse.org/m2m/atl/
Eclipse.modeling.m2m newsgroup	http://dev.eclipse.org/newslists/news.eclipse.modeling.m2m/maillist.html
ATL User manual	http://www.eclipse.org/m2m/atl/doc/ATL_User_Manual%5Bv0.7%5D.pdf
ATL Transformations	http://www.eclipse.org/m2m/atl/atlTransformations/
UML2 Javadoc	http://download.eclipse.org/modeling/mdt/uml2/javadoc/2.1.0/
MARTE profile 4 RSA	http://www.omgmarte.org/Documents/MARTE4RSA/MARTE_RSA_1.0.0.zip

2.3. CONTACT

Eric MAES mailto:eric.maes@thalesgroup.com

3. CONCEPTS MAPPING

3.1. Root

3.1.1. General concept

Concept	Cheddar concept	MARTE concept
Name	***Root***	***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***
Description		For each kind of analysis, there is one analysis context in a UML model. This class identifies elements (diagrams) that are of interest for the given analysis. Global parameters of the Context. An analysis context is the root concept to collect relevant quantitative information for performing a specific analysis scenario. Starting with the analysis context and its elements, a tool can follow the links of the model to extract the information that it needs to perform the model analysis. Analysis contexts are also known as *real-time situations* in the schedulability analysis domain (SaAnalysisContext).

Rule #1	A ***Root*** concept is mapped to an ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***
Check #1	An error is raised if there is no ***Element*** stereotyped ***GaResourcePlatform*** in the ***platform*** stereotype attribute of the ***Elements*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***
Check #2	An error is raised if duplicate names are found.
Check #3	A warning is raised if there is no ***Element*** stereotyped ***GaWorkloadBehavior*** in the ***workload*** stereotype attribute of the ***Elements*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***
Check #4	A warning is raised if there is no ***Element*** stereotyped ***GaScenario*** in the ***behavior*** stereotype attribute of the first ***Element*** stereotyped ***GaWorkloadBehavior*** in the ***workload*** stereotype attribute of the ***Elements*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***

3.1.2. Related concepts

3.1.2.1. processor

Field	Cheddar	MARTE
Name	***processor***	***any Element that is stereotyped by GQAM::GaExecHost*** or SAM::SaExecHost ***or GQAM::GaExecHost*** or SAM::SaExecHost ***in the resources stereotype attribute***

		value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)

Rule #2	***processor*** concept is mapped to a list of ***any Element that is stereotyped by GQAM::GaExecHost or SAM::SaExecHost or GQAM::GaExecHost or SAM::SaExecHost in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)*** of the ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***. Each of the element of the ***processor*** list will be mapped regarding the ***processor*** rules.
Check #5	An error is raised if no processors can be found.

3.1.2.2. address_space

Field	Cheddar	MARTE
Name	***address_space***	***any Element that is stereotyped by MARTE::SRM::Sw_Concurrency::MemoryPartition in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #3	***address_space*** concept is mapped to a list of ***any Element that is stereotyped by MARTE::SRM::Sw_Concurrency::MemoryPartition in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)*** of the ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***. Each of the element of the ***address_space*** list will be mapped regarding the ***address_space*** rules.
Check #6	An error is raised if no address spaces can be found.

3.1.2.3. task

Field	Cheddar	MARTE
Name	***task***	***any Element that is stereotyped by SRM::SwSchedulableResource in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #4	***task*** concept is mapped to a list of ***any Element that is stereotyped by SRM::SwSchedulableResource in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)*** of the ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***. Each of the element of the ***task*** list will be mapped regarding the ***task*** rules.
Check #7	An error is raised if no tasks can be found.

3.1.2.4. resource

Field	Cheddar	MARTE
Name	***resource***	***any Element that is stereotyped by SRM::SwMutualExclusionResource in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #5	***resource*** concept is mapped to a list of ***any Element that is stereotyped by SRM::SwMutualExclusionResource in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)*** of the ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***. Each of the element of the ***resource*** list will be mapped regarding the ***resource*** rules.

3.1.2.5. buffer

Field	Cheddar	MARTE
Name	***buffer***	***any Element that is stereotyped by GRM::StorageResource in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #6	***buffer*** concept is mapped to a list of ***any Element that is stereotyped by GRM::StorageResource in the resources stereotype attribute value (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)*** of the ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***. Each of the element of the ***buffer*** list will be mapped regarding the ***buffer*** rules.

3.1.2.6. message

Field	Cheddar	MARTE
Name	***message***	***any Element in the steps stereotype attribute values of all ements that are stereotyped by SAM::SaCommStep in the behavior stereotype attribute values of all elements that are stereotyped by GaWorkloadBehavior in the workload stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #7	***message*** concept is mapped to a list of ***any Element in the steps stereotype attribute values of all ements that are stereotyped by SAM::SaCommStep in the behavior stereotype attribute values of all elements that are stereotyped by GaWorkloadBehavior in the workload stereotype attribute values (GQAM::GaAnalysisContext)*** of the ***Element*** stereotyped by ***GQAM::GaAnalysisContext or SAM::SaAnalysisContext***. Each of the element of the ***message*** list will be mapped regarding the ***message*** rules.

3.1.2.7. event_analyzer

Field	Cheddar	MARTE
Name	***event_analyzer***	***any Element in the contextParams stereotype attribute values (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #8	***event_analyzer*** concept is mapped to a list of ***any Element in the contextParams stereotype attribute values (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext).*** Each of the element of the ***event_analyzer*** list will be mapped regarding the ***event_analyzer*** rules.

3.1.2.8. dependency

Field	Cheddar	MARTE
Name	***dependency***	***any UML::Dependency in the steps stereotype attribute values of all Elements that are stereotyped by GQAM::GaScenario in the behavior stereotype attribute values of all elements that are stereotyped by GaWorkloadBehavior in the workload stereotype attribute values (GQAM::GaAnalysisContext)***

Rule #9	***dependency*** concept is mapped to a list of ***any UML::Dependency in the steps stereotype attribute values of all Elements that are stereotyped by GQAM::GaScenario in the behavior stereotype attribute values of all elements that are stereotyped by GaWorkloadBehavior in the workload stereotype attribute values (GQAM::GaAnalysisContext).*** Each of the element of the ***dependency*** list will be mapped regarding the ***dependency*** rules.

3.2. processor

3.2.1. General concept

Concept	Cheddar concept	MARTE concept
Name	***processor***	***GQAM::GaExecHost or SAM::SaExecHost***
Description	A processor is composed of : • its name • the scheduler hosted • if the scheduler is preemptive • network (not used in this cheddar version) • the quantum value associated with the scheduler • the file name of a file which contains the source code of a user-defined scheduler	A CPU or other device which executes functional steps.

Rule #10	A ***processor*** is mapped to any ***Element*** stereotyped by ***GQAM::GaExecHost or SAM::SaExecHost***

3.2.2. Related concepts

3.2.2.1. name

Field	Cheddar	MARTE
Name	***name***	***name***
Description	A processor name can be any combination of literal characters including underscore. Space is forbidden. Each processor must have a unique name.	

Rule #11	The ***processor name*** is mapped to the ***name*** of the ***Element*** stereotyped by ***GQAM::GaExecHost or SAM::SaExecHost***

3.2.2.2. scheduler

Field	Cheddar	MARTE
Name	***scheduler***	***mainScheduler (GRM::ProcessingResource)***
Description	The **scheduler** hosted by the processor.	scheduler that share the processing capacity brought in by the processing resource.

Rule #12	The ***processor scheduler*** is mapped to the ***mainScheduler (GRM::ProcessingResource)*** stereotype attribute value (if not null) of the ***Element*** stereotyped by ***GQAM::GaExecHost or SAM::SaExecHost***. If the ***mainScheduler (GRM::ProcessingResource)*** stereotype attribute value is null, then the ***processor scheduler*** policy value is equal to NO_SCHEDULING_PROTOCOL. ***mainScheduler (GRM::ProcessingResource)*** Element will be mapped regarding the ***scheduler*** rules.
Check #8	A warning is raised if ***mainScheduler (GRM::ProcessingResource)*** is not defined.

3.2.2.3. network_link

Field	Cheddar	MARTE
Name	***network_link***	
Description	In this cheddar version, the network field is not used (planned to be used in order to simulate message scheduling).	

Rule #13	The value of ***network_link*** is “No_Network”.

3.3. scheduler

3.3.1. General concept

	Cheddar concept	MARTE concept
Name	***scheduler***	***GRM::Scheduler***
Description	The scheduler hosted by the processor	A ***GRM::Scheduler*** is defined as a kind of ResourceBroker that brings access to its broked ProcessingResource or resources following a certain scheduling policy.

Rule #14	The ***scheduler*** is mapped to any ***Element*** stereotyped by ***GRM::Scheduler***

3.3.2. Related concepts

3.3.2.1. policy

Field	Cheddar	MARTE
Name	***policy***	***schedPolicy (GRM::Scheduler)***
Description	Basically, you can choose from a various set of schedulers such as (to get a detailed description on these schedulers, see your preferred real time books of the publications provided with this program) : • "Earliest Deadline First (or EDF)". Tasks can be periodic or not and are scheduled according to their deadline. • "Least Laxity First (or LLF)". Tasks can be periodic or not and are scheduled according to their laxity. • "Rate Monotonic (or RM, or RMA, or RMS)". Tasks have to be periodic, and deadline must be equal to period. Tasks are scheduled according to their period. You have to be aware that the value of the priority field of the tasks is ignored here. • "Deadline Monotonic (or DM)". Tasks have to be periodic and are scheduled according to their deadline. You have to be aware that the value of the priority field of the tasks is ignored here. • "POSIX 1003.1b scheduler ". Tasks can be periodic or not. Tasks are scheduled according to the priority and the policy of the tasks. (Rate Monotonic and Deadline Monotonic use the same scheduler engine except that priorities are automatically computed from task period or deadline). POSIX 1003.1b scheduler supports SCHED_RR, SCHED_FIFO and SCHED_OTHERS queueing policies. SCHED_OTHERS is a time sharing policy.	scheduling policy implemented by the scheduler.

SCHED_RR and SCHED_FIFO tasks must have priorities ranging between 255 and 1. Priority level 0 is reserved for SCHED_OTHERS tasks. The highiest priority level is 255.

- "Time sharing based on waiting time (which is a Linux-like scheduler)" and "Time sharing based on cpu usage". These two schedulers provide a way to share the processor as on a time sharing operatong system. With the first scheduler, the more a ready task waits for the processor and the more its priority increases. With the second scheduler, the more a ready task uses the processor and the more its priority decreases.

- Round robin (with quantum). The processor is regulary shared between all the tasks. A quantum (which is a bound on the time a task keeps the processor) can be given.

- "Maximum Urgency First based on laxity" and "Maximum Urgency First based on deadline". Such schedulers are based on an hybrid priority assignment : a task priority is made of a fixed part and a dynamic part (see).

- "D-Over/Stable EDF". This scheduler is an EDF like but which is work fine when the processor is over-loaded. When the processor is over-loaded, D-Over is always able to predict which tasks will miss its deadline (in contrary to EDF).

- User-defined schedulers ("Pipeline user-defined scheduler", "Automata user-defined scheduler" or "Compiled user-defined scheduler"). These schedulers allow users to define their own scheduler into Cheddar.

Rule #15

The ***scheduler*** value is mapped to the ***schedPolicy (GRM::Scheduler)*** of the ***Element*** stereotyped by ***GRM::Scheduler*** following the corresponding mapping :

Cheddar policy_kind	MARTE::GRM::schedPolicyKind
EARLIEST_DEADLINE_FIRST_PROTOCOL	**EarliestDeadlineFirst**
LEAST_LAXITY_FIRST_PROTOCOL	**LeastLaxityFirst**
RATE_MONOTONIC_PROTOCOL	**Other**
DEADLINE_MONOTONIC_PROTOCOL	**Other**
POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL	**FixedPriority**
TIME_SHARING_BASED_ON_WAIT_TIME_PROTOCOL	**Other**
ROUND_ROBIN_PROTOCOL	**RoundRobin**
MAXIMUM_URGENCY_FIRST_BASED_ON_LAXITY_PROTOCOL	**Other**
MAXIMUM_URGENCY_FIRST_BASED_ON_DEADLINE_PROTOCOL	**Other**
D_OVER_PROTOCOL	**Other**

	<table><tr><td>**D_OVER_PROTOCOL**</td><td>**Other**</td></tr><tr><td>**TIME_SHARING_BASED_ON_CPU_USAGE_PROTOCOL**</td><td>**Other**</td></tr><tr><td>**PIPELINE_USER_DEFINED_PROTOCOL**</td><td>**Undef**</td></tr></table> In the case that the ***schedPolicy (GRM::Scheduler)*** stereotype attribute value is equal to '**Other**', then the scheduler value is considered to be filled as the ***otherSchedPolicy (GRM::Scheduler)*** stereotype attribute using the cheddar format : ▪ **RATE_MONOTONIC_PROTOCOL** ▪ **DEADLINE_MONOTONIC_PROTOCOL** ▪ **TIME_SHARING_BASED_ON_WAIT_TIME_PROTOCOL** ▪ **MAXIMUM_URGENCY_FIRST_BASED_ON_LAXITY_PROTOCOL** ▪ **MAXIMUM_URGENCY_FIRST_BASED_ON_DEADLINE_PROTOCOL** ▪ **D_OVER_PROTOCOL** ▪ **TIME_SHARING_BASED_ON_CPU_USAGE_PROTOCOL** If no cheddar formatted attribute is found in this case, the scheduling policy is set to **NO_PROTOCOL.**
Check #9	An error is raised if the policy is not well defined. E.g. if the ***schedPolicy*** is set to '**Other**' and the ***otherSchedPolicy (GRM::Scheduler)*** does not match any of the previous litterals.

3.3.2.2. is_preemptive

Field	Cheddar concept	MARTE concept
Name	***is_preemptive***	***isPreemptible (GRM::Scheduler)***

Rule #16	The ***is_preemptive*** attribute is mapped to the ***isPreemptible (GRM::Scheduler)*** stereotype attribute of the Element stereotyped by ***GRM::Scheduler*** following the corresponding mapping. <table><tr><th>Cheddar is_preemptive_kind</th><th>MARTE</th></tr><tr><td>**PREEMPTIVE**</td><td>**true**</td></tr><tr><td>**NOT_PREEMPTIVE**</td><td>**false**</td></tr></table>
Check #10	A warning is raised if isPreemptible is not defined.

3.3.2.3. quantum

Field	Cheddar concept	MARTE concept
Name	***quantum***	***attribute cheddar_quantum of the object stereotyped with GRM::Scheduler***
Description	The **quantum** value associated with the scheduler. This information is useful if a scheduler has to manage several tasks with the same dynamic or static priority : in this case, the simulator has to choose how to share the processor between these tasks. The quantum is a bound on the delay a task can hold the processor (if the quantum is equal to zero, there is no bound on the processor holding time). At the time we're speaking, the quantum value can be used with the POSIX 1003.1b scheduler	

	(only with SCHED_RR tasks) and the round robin scheduler. With POSIX 1003.1b, two SCHED_RR tasks with the same priority level should share the processor with a POSIX round-robin policy. In this case, the quantum value is the time slot of this round-robin scheduler. Finally, the quantum value could also be used for user-defined scheduler.	

Rule #17	The quantum attribute is mapped to integer default value of the attribute named ***cheddar_quantum*** (type Integer) of the ***Element*** stereotyped by ***GRM::Scheduler***

3.3.2.4. parametric_filename

Field	Cheddar concept	MARTE concept
Name	***parametric_filename***	***otherSchedPolicy (GRM::Scheduler)***
Description	it's the name of a file which contains the source code of a user-defined scheduler	is used to annotate a scheduling policy that is not included among the values of the schedPolicyKind enumerated type.

Rule #18	In the case the ***schedPolicy (GRM::Scheduler)*** stereotype attribute value of the ***Element*** stereotyped by ***GRM::Scheduler*** is equal to '**Undef**', then the ***parametric_filename*** attribute is mapped to the ***otherSchedPolicy (GRM::Scheduler)*** stereotype attribute.
Check #11	An error is raised if the ***schedPolicy (GRM::Scheduler)*** stereotype attribute value of the ***Element*** stereotyped by ***GRM::Scheduler*** is equal to '**Undef**' and if the ***otherSchedPolicy (GRM::Scheduler)*** stereotype attribute is not defined.

3.4. network

	Cheddar concept	MARTE concept
Name	***network***	
Description	In this cheddar version, the network field is not used (planned to be used in order to simulate message scheduling).	

Rule #19	The value of ***network*** is "No_Network".

3.5. address_space

3.5.1. General concept

	Cheddar concept	MARTE concept
Name	***address_space***	***MARTE::SRM::Sw_Concurrency::MemoryPartition***

Description		***MARTE::SRM::Sw_Concurrency::MemoryPartition*** represents a virtual address space. It insures that each concurrent resource associated to a specific memory partition can only access its own memory space.

Rule #20	An ***address_space*** is mapped to any ***Element*** stereotyped by ***MARTE::SRM::Sw_Concurrency::MemoryPartition***

3.5.2. Related concepts

3.5.2.1. name

Field	Cheddar	MARTE
Name	***name***	***name***
Description	An address space name can be any combination of literal characters including underscore. Space is forbidden. Each address space name has to be unique.	

Rule #21	The ***address_space name*** is mapped to the ***name*** of the element stereotyped by ***SRM::MemoryPartition***

3.5.2.2. cpu_name

Field	Cheddar	MARTE
Name	***cpu_name***	***first Element of concurrentResources (SRM::MemoryPartition) whose type is stereotyped GQAM::GaExecHost or SAM::SaExecHost***
Description	**A processor name.** This is the processor which hosts the address space.	

Rule #22	The ***cpu_name*** is mapped to the ***first Element of concurrentResources (SRM::MemoryPartition) whose type is stereotyped GQAM::GaExecHost or SAM::SaExecHost***. This ***Element*** must be a valid processor.
Check #12	An error is raised if no ***Element*** of ***concurrentResources (SRM::MemoryPartition)*** has it's type ***stereotyped GQAM::GaExecHost or SAM::SaExecHost***.
Check #13	An error is raised if the first of theses elements does no match any existing cheddar processors.

3.5.2.3. text_memory_size

Field	Cheddar	MARTE
Name	***text_memory_size***	***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and***

		that is named text_memory_size
Description	The fields related to memory size will be used in the next Cheddar's release in order to perform a global memory analysis.	

Rule #23	The ***text_memory_size*** is mapped to the ***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named text_memory_size*** of the ***Element*** stereotyped by ***MARTE::SRM::Sw_Concurrency::MemoryPartition***

3.5.2.4. heap_memory_size

Field	Cheddar	MARTE
Name	***heap_memory_size***	***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named heap_memory_size***
Description	The fields related to memory size will be used in the next Cheddar's release in order to perform a global memory analysis.	

Rule #24	The ***heap_memory_size*** is mapped to the ***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named heap_memory_size*** of the ***Element*** stereotyped by ***MARTE::SRM::Sw_Concurrency::MemoryPartition***

3.5.2.5. stack_memory_size

Field	Cheddar	MARTE
Name	***stack_memory_size***	***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named stack_memory_size***
Description	The fields related to memory size will be used in the next Cheddar's release in order to perform a global memory analysis.	

Rule #25	The ***stack_memory_size*** is mapped to the ***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named stack_memory_size*** of the ***Element*** stereotyped by ***MARTE::SRM::Sw_Concurrency::MemoryPartition***

3.5.2.6. data_memory_size

Field	Cheddar	MARTE
Name	***data_memory_size***	***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named data_memory_size***
Description	The fields related to memory size will be used in the next Cheddar's release in order to perform a global memory analysis.	

Rule #26	The ***data_memory_size*** is mapped to the ***elementSize (MARTE::GRM::StorageResource) of an Element of concurrentResources (MARTE::SRM::Sw_Concurrency::MemoryPartition) that is stereotyped by MARTE::GRM::StorageResource and that is named data_memory_size*** of the ***Element*** stereotyped by ***MARTE::SRM::Sw_Concurrency::MemoryPartition***

3.5.2.7. scheduler

Field	Cheddar	MARTE
Name	***scheduler***	***first Element of concurrentResources (SRM::MemoryPartition) whose type is stereotyped GRM::Scheduler***
Description	As for the processor objects, an address space may include **a scheduler** (in the context of hierarchical scheduling algorithms for example). In this case, you can also specify a scheduler name, its quantum, a file name if the scheduler is a user-defined one and finally the option if your scheduler is preemptive.	

Rule #27	The ***scheduler*** is mapped to the ***first Element of concurrentResources (SRM::MemoryPartition) whose type is stereotyped GRM::Scheduler***. This ***Element*** must be a valid scheduler
Check #14	An error is raised if no ***Element*** of ***concurrentResources (SRM::MemoryPartition)*** has it's type ***stereotyped GRM::Scheduler***.
Check #15	An error is raised if the first of theses elements does no match any existing cheddar schedulers.

3.6. task

3.6.1. General concept

	Cheddar concept	MARTE concept

Name	***task***	***SRM::SwSchedulableResource***
Description	At least, a task is defined by a **name** (the task name should be unique), a **capacity** (bound on its execution time) and a place to run it (a **processor name** and an **address space name**). The other parameters are optional but can be required for a particular scheduler	It is an active resource able to perform actions using the processing capacity brought from a processing resource by the scheduler that manages it.

Rule #28	A ***task*** is mapped to any ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2. Related concepts

3.6.2.1. x

Field	Cheddar	MARTE
Name	***x***	

Rule #29	The value of ***x*** is 0.

3.6.2.2. y

Field	Cheddar	MARTE
Name	***y***	

Rule #30	The value of ***y*** is 0.

3.6.2.3. name

Field	Cheddar	MARTE
Name	***name***	***name***
Description	the task name should be unique	

Rule #31	The ***task name*** is mapped to the ***name*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.4. task_type

Field	Cheddar	MARTE
Name	***task_type***	***type (SRM:: SwConcurrentResource)***
Description	It describes the way the task is activated. An aperiodic task is only activated once. A periodic task is activated many times and the delay between two activations is a	

	fixed one. A poisson process task is activated many times and the delay between two activations is a random delay : the random law used to generated these delays is an exponential one (poisson process). a sporadic task is a task which is activated many times with a minimal delay between two succesive activations. If the task type is "user-defined", the task activation law is defined by the user	

Rule #32	The ***task_type*** value is mapped to the ***type (SRM:: SwConcurrentResource*** stereotype attribute type of the ***Element*** stereotyped by ***SRM::SwSchedulableResource*** following the corresponding mapping : **task_type is mapped to the 'type' (SRM::SwConcurrentResource) attribute (as String)**
Check #16	An error is raised if the type is not defined or if none of the above task type are found.

Cheddar task_type_kind	MARTE ArrivalPattern
PERIODIC_TYPE	**PeriodicPattern**
APERIODIC_TYPE	**AperiodicPattern**
SPORADIC_TYPE	**SporadicPattern**
POISSON_TYPE	**IrregularPattern**
PARAMETRIC_TYPE	**ClosedPattern**

3.6.2.5. cpu_name

Field	Cheddar	MARTE
Name	***cpu_name***	***host attribute value (GRM::Scheduler) of the host attribute value (GRM::SchedulableResource) of the Element stereotyped by SRM::SwSchedulableResource***

Rule #33	The ***cpu_name*** value is mapped to the ***host attribute value (GRM::Scheduler) of the host attribute value (GRM::SchedulableResource) of the Element stereotyped by SRM::SwSchedulableResource***
Note	***GRM::SchedulableResource*** is not a generalization of ***SRM::SwSchedulableResource*** as defined in the current version of the MARTE specification, whereas it should. To replace missing stereotype attributes, ***GRM::SchedulableResource*** must be applied to the element too. When this bug will be fixed, this won't be necessary anymore.
Check #17	An error is raised if ***GRM::SchedulableResource*** is not applied to the element
Check #18	An error is raised if the ***host attribute value (GRM::SchedulableResource)*** is not defined
Check #19	An error is raised if the ***host attribute value (GRM::SchedulableResource)*** is not a valid scheduler
Check #20	An error is raised if the ***host attribute value (GRM::Scheduler)*** is not defined
Check #21	An error is raised if the ***host attribute value (GRM::Scheduler)*** is not a valid execution host

3.6.2.6. address_space_name

Field	Cheddar	MARTE
Name	***address_space_name***	***type of the addressSpace Element (SRM::SwConcurrentResource) (stereotyped SRM::MemoryPartition)***

Rule #34	The ***address_space_name*** is mapped to the ***type of the addressSpace Element (SRM::SwConcurrentResource) (stereotyped SRM::MemoryPartition***.
Check #22	An error is raised if ***address_space_name*** is not defined.

3.6.2.7. capacity

Field	Cheddar	MARTE
Name	***capacity***	***activationCapacity (SRM::SwConcurrentResource)***
Description	Bound on the task execution time	

Rule #35	The ***capacity*** is mapped to ***activationCapacity (SRM::SwConcurrentResource)*** stereotype attribute value of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***
Check #23	An error is raised if ***activationCapacity (SRM::SwConcurrentResource)*** is not defined or < 1.

3.6.2.8. period

Field	Cheddar	MARTE
Name	***period***	***the Element of periodElements (SRM::SwConcurrentResource) that is named period***
Description	It is the time between two task activations. The period is a constant delay for a periodic task. It's an average delay for a poisson process task. If you have selected a processor that owns a Rate Monotonic or a Deadline Monotonic scheduler, you have to give a period for each of its tasks.	

Rule #36	The ***period*** is mapped to the integer default value of ***the Element of periodElements (SRM::SwConcurrentResource)*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***
Check #24	An error is raised if ***period*** is not defined for tasks whose type are : • **PERIODIC_TYPE** • **SPORADIC_TYPE** • **POISSON_TYPE**

3.6.2.9. start_time

Field	Cheddar	MARTE
Name	***start_time***	***the Element of periodElements (SRM::SwConcurrentResource) that is named start_time***
Description	It is the time when the task arrives in the system (its first activation time).	

Rule #37	The ***start_time*** is mapped to the integer default value of ***the Element of periodElements (SRM::SwConcurrentResource) that is named start_time*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.10. deadline

Field	Cheddar	MARTE
Name	***deadline***	***the Element of deadlineElements (SRM::SwSchedulableResource) that is named deadline***
Description	The task must end its activation before its deadline. A deadline is a relative information : to get the absolute date at which a task must end an activation, you should add the time when the task was awoken/activated to the task deadline. **Warning** : the deadline must be equal to the period if you define a Rate Monotonic scheduler.	

Rule #38	The ***deadline*** is mapped to ***the Element of deadlineElements (SRM::SwSchedulableResource) that is named deadline*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.11. priority

Field	Cheddar	MARTE
Name	***priority***	***the Element of priorityElements (SRM::SwConcurrentResource) that is named priority***
Description	This parameter is dedicated to the POSIX 1003.1b/Highest Priority First scheduler. Priority is the fixed priority of a task. **Warning** : the **priority** is ignored by a Rate Monotonic and a Deadline Monotonic scheduler.	

Rule #39	The ***priority*** is mapped to ***the Element of priorityElements (SRM::SwConcurrentResource) that is named priority*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***
Check #25	An error is raised if ***priority*** is not defined of < 1.

3.6.2.12. policy

Field	Cheddar	MARTE
Name	***policy***	***schedPolicy stereotype attribute value (GRM::Scheduler) of the host stereotype attribute value (GRM:: SchedulableResource)***
Description	This parameter is dedicated to the POSIX 1003.1b/Highest Priority First scheduler. Policy can be SCHED_RR, SCHED_FIFO or SCHED_OTHERS and describes how the scheduler chooses a task when several tasks have the same priority level. **Warning** : the **policy** is ignored by a Rate Monotonic and a Deadline Monotonic scheduler.	

Rule #40	The ***policy*** is mapped to the ***schedPolicy stereotype attribute value (GRM::Scheduler) of the host stereotype attribute value (GRM:: SchedulableResource)*** attribute value of the ***Element*** stereotyped by ***SRM::SwSchedulableResource*** following the corresponding mapping : Cheddar task_policy_kind → MARTE::GRM::schedPolicyKind: **SCHED_FIFO** → **EarliestDeadlineFirst**; **SCHED_FIFO** → **FixedPriority**; **SCHED_RR** → **RoundRobin**; **SCHED_OTHERS** → **Other**; **SCHED_OTHERS** → **Other** **policy is mapped to the 'schedPolicy' (GRM::Scheduler) attribute (as String)**
Note	***GRM::SchedulableResource*** is not a generalization of ***SRM::SwSchedulableResource*** as defined in the current version of the MARTE specification, whereas it should. To replace missing stereotype attributes, ***GRM::SchedulableResource*** must be applied to the element too. When this bug will be fixed, this won't be necessary anymore.
Check #26	An error is raised if ***GRM::SchedulableResource*** is not applied to the element

Cheddar task_policy_kind	MARTE::GRM::schedPolicyKind
SCHED_FIFO	**EarliestDeadlineFirst**
SCHED_FIFO	**FixedPriority**
SCHED_RR	**RoundRobin**
SCHED_OTHERS	**Other**
SCHED_OTHERS	**Other**

3.6.2.13. jitter

Field	Cheddar	MARTE
Name	***jitter***	***the Element of periodElements (SRM::SwConcurrentResource) that is named jitter***
Description	The jitter is a maximum lateness on the task wake up time. This information can be used to express task precedencies and to applied method such as the Holistic task response time method.	

Rule #41	The ***jitter*** is mapped to ***the Element of periodElements (SRM::SwConcurrentResource) that is named jitter*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.14. blocking_time

Field	Cheddar	MARTE
Name	***blocking_time***	***the Element of periodElements (SRM::SwConcurrentResource) that is named blocking_time***
Description	It's a bound on shared resource waiting time. This delay could be set by the user but could also be computed by Cheddar if you described how shared resources are accessed.	

Rule #42	The ***blocking_time*** is mapped to the integer default value of ***the Element of periodElements (SRM::SwConcurrentResource) that is named blocking_time*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.15. activation_rule

Field	Cheddar	MARTE
Name	***activation_rule***	***the Element of stateElements (SRM::SwResource) that is named activation_rule***
Description	The name of the rule which defines the way the task should be activated. Only used with user-defined task.	

Rule #43	The ***activation_rule*** is mapped to the string default value of ***the Element of stateElements (SRM::SwResource) that is named activation_rule*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***
Check #27	An error is raised if ***activation_rule*** is not defined and task type is ***PARAMETRIC_TYPE***

3.6.2.16. criticality

Field	Cheddar	MARTE
Name	***criticality***	***the Element of deadlineElements (MARTE::SRM::Sw_Concurrency::SwSchedulableResource) that is named criticality***
Description	The field indicates how the task is critical. Currently used by the MUF scheduler or any user-defined schedulers.	

Rule #44	The ***criticality*** is mapped to the integer default value of ***the Element of deadlineElements (MARTE::SRM::Sw_Concurrency::SwSchedulableResource) that is named criticality*** of the ***Element*** stereotyped by

	SRM::SwSchedulableResource

3.6.2.17. seed

Field	Cheddar	MARTE
Name	***seed***	***element named "seed" in the "stateElements" (SRM::SwResource) attribute***
Description	If you define a poisson process task or a user-defined task, you can set here how random activation delay should be generated (in a deterministic way or not). The "Seed" button proposes you a randomly generated seed value but of course, you can give any seed value. This seed value is used only if the **Predictable** button is pushed. If the **Unpredictable** button is pushed instead, the seed is initialized at simulation time with "gettimeofday".	

Rule #45	The ***seed*** is mapped to the integer default value of the ***element named "seed" in the "stateElements" (SRM::SwResource) attribute*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.18. predictable_seed

Field	Cheddar	MARTE
Name	***predictable_seed***	***element named "predictable_seed" in the "stateElements" (SRM::SwResource) attribute***

Rule #46	The ***predictable_seed*** is mapped to the integer default value of the ***element named "predictable_seed" in the "stateElements" (SRM::SwResource) attribute*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.19. text_memory_size

Field	Cheddar	MARTE
Name	***text_memory_size***	***"elementSize" attribute of the first element stereotyped StorageResource (GRM::StorageResource) and named "text_memory_size" in the "stateElements" (SRM::SwResource) attribute***
Description	The fields related to task memory size will be used in the next Cheddar's release in order to perform memory requirement analysis.	

Rule #47	The ***text_memory_size*** is mapped to the integer default value of the ***"elementSize" attribute of the first element stereotyped StorageResource (GRM::StorageResource) and named "text_memory_size" in the "stateElements" (SRM::SwResource) attribute*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.20. stack_memory_size

Field	Cheddar	MARTE
Name	***stack_memory_size***	***"elementSize" attribute of the first element stereotyped StorageResource (GRM::StorageResource) and named "stack_memory_size" in the "stateElements" (SRM::SwResource) attribute***
Description	The fields related to task memory size will be used in the next Cheddar's release in order to perform memory requirement analysis.	

Rule #48	The ***stack_memory_size*** is mapped to the integer default value of the ***"elementSize" attribute of the first element stereotyped StorageResource (GRM::StorageResource) and named "stack_memory_size" in the "stateElements" (SRM::SwResource) attribute*** of the ***Element*** stereotyped by ***SRM::SwSchedulableResource***

3.6.2.21. offsets

Field	Cheddar	MARTE
Name	***offsets***	**-**

Rule #49	The ***offset*** are not mapped

3.6.2.22. parameters

Field	Cheddar	MARTE
Name	***parameters***	
Description		

Rule #50	The ***parameters*** are not mapped

3.7. offset

3.7.1. General concept

	Cheddar concept	MARTE concept
Name	***offset***	
Description	An offset stores two information : an activation number and a value. It allows to change the wake up time of a task on a given activation number. For an activation number, the task wake up time will be delayed by the amount of time given by the	

	value field.	

Rule #51	The ***offset*** concept is not mapped

3.7.2. Related concepts

3.7.2.1. activation

Field	Cheddar	MARTE
Name	***activation***	

Rule #52	The ***activation*** is not mapped

3.7.2.2. value

Field	Cheddar	MARTE
Name	***value***	

Rule #53	The ***value*** is not mapped

3.8. parameter

3.8.1. General concept

	Cheddar concept	MARTE concept
Name	***parameter***	***Element of an Element which type is either EDFParameters, FixedPriorityParameters, PeriodicServerParameters or PoolingParameters***
Description	A parameter is similar to the deadline, the period, the capacity ... but used by user-defined schedulers. A user can define new task parameters. A user-defined task parameter has a value, a name and a type. The types currently available to defined user-defined task parameters are : string, integer boolean and double.	Values given to a SchedulableResource to quantify its merits to receive processing capacity in comparison with others scheduled under the same scheduler.

Rule #54	***parameter*** concept is mapped to any ***Element of an Element which type is either EDFParameters, FixedPriorityParameters, PeriodicServerParameters or PoolingParameters***.
Note	Due to limitations in the VSL support, this concept is not used for now.

3.8.2. Related concepts

3.8.2.1. name

Field	Cheddar	MARTE
Name	***name***	***name***

Rule #55	The ***parameter name*** is mapped to the ***name*** of the ***Element of an Element which type is either EDFParameters, FixedPriorityParameters, PeriodicServerParameters or PoolingParameters***

3.8.2.2. type

Field	Cheddar	MARTE
Name	***type***	***type***

Rule #56	The ***type*** attribute is mapped to the ***type*** of the ***Element of an Element which type is either EDFParameters, FixedPriorityParameters, PeriodicServerParameters or PoolingParameters***. following the corresponding mapping.

Cheddar parameter_type_kind	MARTE data type
integer	**Integer**
	NFP_Duration
	PeriodicServerKind
double	
string	
boolean	

3.8.2.3. value

Field	Cheddar	MARTE
Name	***value***	***priority, backgroundPriority or maxPending Replenish***

Rule #57	The ***value*** attribute is mapped to the ***priority, backgroundPriority or maxPending Replenish*** value of the ***Element of an Element which type is either EDFParameters, FixedPriorityParameters, PeriodicServerParameters or PoolingParameters*** according to the its ***name***.

3.9. resource

3.9.1. General concept

	Cheddar concept	MARTE concept

Name	***resource***	***SRM::SwMutualExclusionResource***
Description		A ***SRM::SwMutualExclusionResource*** represents the kind of protected resources that serve as the mechanisms used to arbitrate concurrent execution flows, and in particular the mutual excusive access to shared resources.

Rule #58	A ***resource*** is mapped to any ***Element*** stereotyped by ***SRM::SwMutualExclusionResource***

3.9.2. Related concepts

3.9.2.1. name

Field	Cheddar	MARTE
Name	***name***	***name***

Rule #59	The ***resource name*** is mapped to the ***name*** of the ***Element*** stereotyped by ***SRM::SwMutualExclusionResource***

3.9.2.2. state

Field	Cheddar	MARTE
Name	***state***	***mechanism (SRM:SwMutualExclusionResource)***
Description	An initial **value/state** (simular to a semaphore initial value). During a scheduling simulation, at a given time, if a resource value is equal or less than zero, the requesting tasks are blocked until the semaphore/shared resource is released. An initial value equal to 1 allows you to design a shared resource that is initially free and that can be used by only one task at a given time.	

Rule #60	The ***state*** is mapped to the ***mechanism (SRM:SwMutualExclusionResource)*** attribute of the ***Element*** stereotyped by ***SRM::SwMutualExclusionResource*** with the following equivalents :
Check #28	A warning is raised if the mechanism is set to CountSemaphore and cheddar_state is not defined

Cheddar state	MARTE mecanism
0	BooleanSemaphore
1	Mutex
integer default value of the element named ‘cheddar_state’ in stateElements (SRM::SwResource) if defined	CountSemaphore
0 else	

Check #29	An error is raised if the mechanism is not defined

3.9.2.3. protocol

Field	Cheddar	MARTE
Name	***protocol***	***concurrentAccessProtocol (SRM::SwMutualExclusionResource)***
Description	Currently, you can choose between PCP (for Priority Ceiling Protocol), PIP (for Priority Inheritance Protocol) or "No protocol". With PCP or PIP, accessing shared resources may change task priorities. The "No protocol" just means that no task prioriy will be changed at accessing the shared resource.	

Rule #61	The ***protocol*** is mapped to ***concurrentAccessProtocol (SRM::SwMutualExclusionResource)*** of the ***Element*** stereotyped by ***SRM::SwMutualExclusionResource*** with the following equivalents :

Cheddar protocol_kind	MARTE concurrentAccessProtocol
NO_PROTOCOL	**NoPreemption**
PCP	**PCP**
PIP	**PIP**
IPCP	**Other and the element has an attribute named 'cheddar_IPCP'**
NO_PROTOCOL (default)	**Other cases:** - **Undef** - **Field left blank** - **Other without the attribute cheddar_IPCP**

Check #30	An error is raised if the ***concurrentAccessProtocol (SRM::SwMutualExclusionResource)*** is missing or undefined
Check #31	An error is raised if the ***concurrentAccessProtocol (SRM::SwMutualExclusionResource)*** is set to "Other" and no attribute **'cheddar_IPCP'** is found

3.9.2.4. cpu_name

Field	Cheddar	MARTE
Name	***cpu_name***	
Description	Each shared resource has to be hosted by a given processor.	

Rule #62	If ***address_space_name*** is defined, ***cpu_name*** is mapped to it's ***cpu_name*** field.

3.9.2.5. address_space_name

Field	Cheddar	MARTE
Name	***address_space_name***	***first supplying element stereotyped Memory Partition in the dependencies of this element***

Rule #63	The ***address_space_name*** is mapped to ***first supplying element stereotyped Memory Partition in the dependencies of this element***
Check #32	An error is raised if the resource depends on no known and valid address_space.

3.9.2.6. resource_used_by

Field	Cheddar	MARTE
Name	***resource_used_by***	***client elements of the dependencies stereotyped GRM::ResourceUsage***

Rule #64	The ***resource_used_by*** is mapped to the list of ***client elements of the dependencies stereotyped GRM::ResourceUsage*** of this element

3.10. resource_user

3.10.1. General concept

	Cheddar concept	MARTE concept
Name	***resource_user***	***UML::Dependency stereotyped GRM::ResourceUsage***
Description	tasks that need the resource. Tasks hold resources in critical section. Each critical section has to be defined by : • The task name requiring the shared resource. • The start time of the critical section. • The end time of the critical section. Of course, you can define several critical sections for a given task of a given shared resource.	

Rule #65	A ***resource_user*** is mapped to any ***UML::Dependency stereotyped GRM::ResourceUsage*** of this element

3.10.2. Related concepts

3.10.2.1. task_name

Field	Cheddar	MARTE

Name	***task_name***	***client name of the UML::Dependency stereotyped GRM::ResourceUsage***

Rule #66	The ***task_name*** is mapped to the first ***client name of the UML::Dependency stereotyped GRM::ResourceUsage*** that is an existing Cheddar ***task***
Check #33	An error is raised if the client of the dependency is not an existing task

3.10.2.2. start_time

Field	Cheddar	MARTE
Name	***start_time***	***value of the constraint stereotyped TimedConstraint that is attached to the dependency link***

Rule #67	The ***start_time*** is mapped to the ***value of the constraint stereotyped TimedConstraint that is attached to the dependency link***
Note	The stereotype TimedConstraint is not currently required
Check #34	An error is raised if no constraint are attached to the dependency link
Check #35	An error is raised if the constraint is invalid or does not produce an integer as a result

3.10.2.3. end_time

Field	Cheddar	MARTE
Name	***end_time***	***execTime stereotype attribute (GRM::ResourceUsage)***

Rule #68	The ***end_time*** is mapped to the ***execTime stereotype attribute (GRM::ResourceUsage)*** reduced by the ***start_time***
Note	The end instant is replaced by the data of the duration
Check #36	An error is raised if the ***execTime stereotype attribute (GRM::ResourceUsage)*** is undefined or invalid

3.11. dependency

3.11.1. General concept

	Cheddar concept	MARTE concept
Name	***dependency***	***UML::Dependency***
Description	The ***dependency*** concept is used to define precedencies between tasks, messages, buffers.	

Rule #69	The ***dependency*** is mapped to any ***UML::Dependency***

3.11.2. Related concepts

3.11.2.1. from_type

Field	Cheddar	MARTE
Name	***from_type***	***type of the supplier of the UML::Dependency***

Rule #70	The ***from_type*** is mapped to the ***type of the supplier of the UML::Dependency*** :

Cheddar ***from_type***	Cheddar equivalent of MARTE ***type of the supplier of the UML::Dependency***	MARTE ***type of the supplier of the UML::Dependency***
task	***task***	***SRM::SwSchedulableResource***
buffer	***buffer***	***GRM::StorageResource***
message	***message***	***SAM::SaCommStep***

3.11.2.2. to_type

Field	Cheddar	MARTE
Name	***to_type***	***type of the client of the UML::Dependency***

Rule #71	The ***to_type*** is mapped to the ***type of the client of the UML::Dependency*** :

Cheddar ***to_type***	Cheddar equivalent of MARTE ***type of the client of the UML::Dependency***	MARTE ***type of the client of the UML::Dependency***
task	***task***	***SRM::SwSchedulableResource***
buffer	***buffer***	***GRM::StorageResource***
message	***message***	***SAM::SaCommStep***

3.11.2.3. from

Field	Cheddar	MARTE
Name	***from***	***supplier of the UML::Dependency***

Rule #72	The ***from*** is mapped to the ***supplier of the UML::Dependency***
Check #37	An error is raised if the ***supplier of the UML::Dependency*** is not a known element.

3.11.2.4. to

Field	Cheddar	MARTE
Name	***to***	***client of the UML::Dependency***

Rule #73	The ***to*** is mapped to the ***client of the UML::Dependency***
Check #38	An error is raised if the ***client of the UML::Dependency*** is not a known element.

3.12. buffer

3.12.1. General concept

	Cheddar concept	MARTE concept
Name	***buffer***	***GRM::StorageResource***
Description	A buffer has a unique **name**, **size** and is hosted by a **processor** and an **address space**.	A StorageResource represents memory, and its capacity is expressed in number of elements; the size of an individual element in bits must be given. The reference clock in this kind of resources corresponds to the pace at which data is updated in it, and hence it determines the time it takes to access to one individual memory element. The level of granularity in the amount of storage resources represented is up to the model designer. For example, if the storage resource represents a hard disk drive, the element could be a block or a sector, and the speed of the clock to access such element would be directly related to the disk rotation speed. The services provided by a storage resource are intended to move data between memory and a processing unit, which in this case can be a computing resource or a communication endpoint.

Rule #74	A ***buffer*** is mapped to any ***Element*** stereotyped by ***GRM::StorageResource***

3.12.2. Related concepts

3.12.2.1. x

Field	Cheddar	MARTE
Name	***x***	

Rule #75	The value of ***x*** is 0.

3.12.2.2. y

Field	Cheddar	MARTE
Name	***y***	

Rule #76	The value of ***y*** is 0.

3.12.2.3. name

Field	Cheddar	MARTE
Name	***name***	***name***

Rule #77	The ***buffer name*** is mapped to the ***name*** of the ***Element*** stereotyped by ***GRM::StorageResource***

3.12.2.4. size

Field	Cheddar	MARTE
Name	***size***	***elementSize (GRM::StorageResource)***

Rule #78	The ***buffer size*** is mapped to the ***elementSize (GRM::StorageResource)*** of the ***Element*** stereotyped by ***GRM::StorageResource***
Check #39	An error is raised if ***elementSize (GRM::StorageResource)*** is not defined or invalid.

3.12.2.5. cpu_name

Field	Cheddar	MARTE
Name	***cpu_name***	

Rule #79	If ***address_space_name*** is defined, ***cpu_name*** is mapped to it's ***cpu_name*** field.

3.12.2.6. address_space_name

Field	Cheddar	MARTE
Name	***address_space_name***	***first supplying element stereotyped Memory Partition in the dependencies of this element***

Rule #80	The ***address_space_name*** is mapped to ***first supplying element stereotyped Memory Partition in the dependencies of this element***
Check #40	An error is raised if the buffer depends on no known and valid address_space.

3.12.2.7. qs

Field	Cheddar	MARTE
Name	***qs***	***attribute named cheddar_qs***
Description	A **queueing system model** is assigned to each buffer. This queueing system model describes the way buffer read and write operations will be done at simulation time. This information is also used to apply buffer feasibility tests.	

Rule #81	The ***qs*** is mapped to the string default value of the ***attribute named cheddar_qs***: Cheddar qs_kind → MARTE: **QS_MDS** → **QS_MDS**; **QS_MM1** → **QS_MM1**; **QS_MD1** → **QS_MD1**; **QS_MP1** → **QS_MP1**; **QS_MG1** → **QS_MG1**; **QS_MMS** → **QS_MMS**; **QS_MDS** → **QS_MDS**; **QS_MPS** → **QS_MPS**; **QS_MGS** → **QS_MGS**; **QS_MM1N** → **QS_MM1N**; **QS_MD1N** → **QS_MD1N**; **QS_MP1N** → **QS_MP1N**; **QS_MG1N** → **QS_MG1N**; **QS_MMSN** → **QS_MMSN**; **QS_MDSN** → **QS_MDSN**; **QS_MPSN** → **QS_MPSN**; **QS_MGSN** → **QS_MGSN**
Check #41	An error is raised if the ***attribute named cheddar_qs*** is not one of the previous strings.

Rule #81 mapping table:

Cheddar qs_kind	MARTE
QS_MDS	**QS_MDS**
QS_MM1	**QS_MM1**
QS_MD1	**QS_MD1**
QS_MP1	**QS_MP1**
QS_MG1	**QS_MG1**
QS_MMS	**QS_MMS**
QS_MDS	**QS_MDS**
QS_MPS	**QS_MPS**
QS_MGS	**QS_MGS**
QS_MM1N	**QS_MM1N**
QS_MD1N	**QS_MD1N**
QS_MP1N	**QS_MP1N**
QS_MG1N	**QS_MG1N**
QS_MMSN	**QS_MMSN**
QS_MDSN	**QS_MDSN**
QS_MPSN	**QS_MPSN**
QS_MGSN	**QS_MGSN**

3.12.2.8. buffer_used_by

Field	Cheddar	MARTE
Name	***buffer_used_by***	***client elements of the dependencies stereotyped GRM::ResourceUsage***
Description	**A list of tasks** which access to the buffer (read or write operations).	

Rule #82	The ***buffer_used_by*** is mapped to the list of ***client elements of the dependencies stereotyped GRM::ResourceUsage*** of this element

3.13. buffer_user

3.13.1. General concept

	Cheddar concept	MARTE concept
Name	***buffer_user***	***UML::Dependency stereotyped GRM::ResourceUsage***
Description	Two type of tasks can access a buffer : **producers** and **consumers** . We suppose that a producer/consumer writes/reads a fixed size of information in the buffer. For each producer or consumer, the size of the information produced or consummed have to be defined. The time of the read/write operation is also given : this time is relative to the task capacity (eg. if T2 consumes a message at time 2, it means that the message will be removed from the buffer when T2 run the 2nd unit of time of its capacity).	

Rule #83	A ***buffer_user*** is mapped to any ***UML::Dependency stereotyped GRM::ResourceUsage*** of this element

3.13.2. Related concepts

3.13.2.1. buffer_role

Field	Cheddar	MARTE
Name	***buffer_role***	

Rule #84

The ***buffer_role*** is mapped the the Cheddar ***size*** of the buffer :

Cheddar buffer_role_kind	Cheddar size
consumer	**< 0**
producer	**> 0**

3.13.2.2. task_name

Field	Cheddar	MARTE
Name	***task_name***	***client name of the UML::Dependency stereotyped GRM::ResourceUsage***

Rule #85	The ***task_name*** is mapped to the first ***client name of the UML::Dependency stereotyped GRM::ResourceUsage*** that

	is an existing Cheddar ***task***
Check #42	An error is raised if the client of the dependency is not an existing task

3.13.2.3. time

Field	Cheddar	MARTE
Name	***time***	***value of the constraint stereotyped TimedConstraint that is attached to the dependency link***

Rule #86	The ***time*** is mapped to the ***value of the constraint stereotyped TimedConstraint that is attached to the dependency link***
Note	The stereotype TimedConstraint is not currently required
Check #43	An error is raised if no constraint are attached to the dependency link
Check #44	An error is raised if the constraint is invalid or does not produce an integer as a result

3.13.2.4. size

Field	Cheddar	MARTE
Name	***size***	***msgSize stereotype attribute (GRM::ResourceUsage)***

Rule #87	The ***size*** is mapped to the ***msgSize stereotype attribute (GRM::ResourceUsage)***
Check #45	An error is raised if the ***msgSize stereotype attribute (GRM::ResourceUsage)*** is undefined or invalid

3.14. message

3.14.1. General concept

	Cheddar concept	MARTE concept
Name	***message***	***SAM::SaCommStep***
Description		

Rule #88	A ***message*** is mapped to any ***Element*** stereotyped by ***SAM::SaCommStep***

3.14.2. Related concepts

3.14.2.1. x

Field	Cheddar	MARTE
Name	***x***	

Rule #89	The value of ***x*** is 0.

3.14.2.2. y

Field	Cheddar	MARTE
Name	***y***	

rule #90	The value of ***y*** is 0.

3.14.2.3. name

Field	Cheddar	MARTE
Name	***name***	***name***

rule #91	The ***message name*** is mapped to the ***name*** of the ***Element*** stereotyped by ***SAM::SaCommStep***

3.14.2.4. jitter

Field	Cheddar	MARTE
Name	***jitter***	***attribute named "cheddar_jitter"***

rule #92	The ***jitter*** is mapped to the integer default value of ***attribute named "cheddar_jitter***.

3.14.2.5. deadline

Field	Cheddar	MARTE
Name	***deadline***	***deadline (SAM::SaCommStep)***

Rule #93	The ***message deadline*** is mapped to the ***deadline (SAM::SaCommStep)*** stereotype attribute value of the ***Element*** stereotyped by ***SAM::SaCommStep***

3.14.2.6. period

Field	Cheddar	MARTE
Name	***period***	***interOccT (GQAM::GaScenario)***

Rule #94	The ***message period*** is mapped to the ***interOccT (GQAM::GaScenario)*** stereotype attribute value of the ***Element*** stereotyped by ***SAM::SaCommStep***

3.14.2.7. size

Field	Cheddar	MARTE
Name	***size***	***msgSize (GQAM::GaCommStep)***

Rule #95	The ***message size*** is mapped to the ***msgSize (GQAM::GaCommStep)*** stereotype attribute value of the ***Element*** stereotyped by ***SAM::SaCommStep***
Check #46	***msgSize (GQAM::GaCommStep)*** must be defined
Check #47	***msgSize (GQAM::GaCommStep)*** must be > 0

3.14.2.8. response_time

Field	Cheddar	MARTE
Name	***response_time***	***respT (GQAM::GaScenario)***

Rule #96	The ***message response_time*** is mapped to the ***respT (GQAM::GaScenario)*** stereotype attribute value of the ***Element*** stereotyped by ***SAM::SaCommStep***

3.14.2.9. communication_time

Field	Cheddar	MARTE
Name	***communication_time***	***execTime (GRM::ResourceUsage)***

Rule #97	The ***message communication_time*** is mapped to the ***execTime (GRM::ResourceUsage)*** stereotype attribute value of the ***Element*** stereotyped by ***SAM::SaCommStep***

3.15. event_analyzer

3.15.1. General concept

	Cheddar concept	MARTE concept
Name	***event_analyzer***	***any Element in the contextParams stereotype attribute values (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***
Description		

Rule #98	The ***event_analyzer*** concept is mapped to ***any Element in the contextParams stereotype attribute values (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

3.15.2. Related concepts

3.15.2.1. filename

Field	Cheddar	MARTE
Name	***filename***	***contextParams***

Rule #99	The ***filename*** is mapped to the string value of ***contextParams*** of ***any Element in the contextParams stereotype attribute values (GQAM::GaResourcesPlatform) of all the platform stereotype attribute values (GQAM::GaAnalysisContext)***

ANNEXE A Modeling with MARTE for CHEDDAR

#	Rule concerned	Check concerned	
#1			
#2			
#3			
#4			
#5			
#6			
#7			
#8			
#9			
#10			
#11			
#12			
#13			
#14			
#15			
#16			
#17			
#18			
#19			
#20			

ANNEXE B   Cheddar metamodel

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="cheddar">
  <eClassifiers xsi:type="ecore:EClass" name="DependencyElement" abstract="true"
eSuperTypes="#//Element"/>
  <eClassifiers xsi:type="ecore:EClass" name="processor" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EReference" name="scheduler" lowerBound="1"
        eType="#//scheduler" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="network_link" eType="#//network"
        defaultValueLiteral="" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="scheduler" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="is_preemptive" lowerBound="1"
        eType="#//Library/is_preemptive_kind" defaultValueLiteral="PREEMPTIVE"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="quantum" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"
        defaultValueLiteral="0"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="policy" eType="#//Library/policy_kind"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="parametric_filename" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="address_space" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EReference" name="cpu_name" lowerBound="1"
        eType="#//processor"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="text_memory_size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="heap_memory_size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="stack_memory_size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="data_memory_size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="scheduler" eType="#//scheduler"
        containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="task" eSuperTypes="#//DependencyElement #//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="task_type"
eType="#//Library/task_type_kind"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="cpu_name" lowerBound="1"
        eType="#//processor"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="address_space_name" lowerBound="1"
        eType="#//address_space"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="x" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="y" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="capacity" lowerBound="1"
        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="period" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="start_time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="deadline" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="priority" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="policy"
eType="#//Library/task_policy_kind"/>
```

```
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="jitter" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="blocking_time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="activation_rule" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="criticality" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="seed" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="predictable_seed" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="text_memory_size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="stack_memory_size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="offsets" upperBound="-1"
        eType="#//offset" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="parameters" upperBound="-1"
        eType="#//parameter" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="parameter" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="parameter_type"
eType="#//Library/parameter_type_kind"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" lowerBound="1"
eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="event_analyzer" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="parametric_filename" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="buffer" eSuperTypes="#//DependencyElement #//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="x" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="y" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="cpu_name" lowerBound="1"
        eType="#//processor"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="address_space_name" lowerBound="1"
        eType="#//address_space"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="buffer_used_by" upperBound="-1"
        eType="#//buffer_user" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="size" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="qs" eType="#//Library/qs_kind"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="buffer_user" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="buffer_role"
eType="#//Library/buffer_role_kind"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="task_name" eType="#//task"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="dependency" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="to_type"
eType="#//Library/to_from_type_kind"
        defaultValueLiteral="task"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="from_type"
eType="#//Library/to_from_type_kind"
        defaultValueLiteral="task"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="from" lowerBound="1"
eType="#//DependencyElement"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="to" lowerBound="1"
eType="#//DependencyElement"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="message" eSuperTypes="#//DependencyElement #//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="x" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
```

```
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="y" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="jitter" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="deadline" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="period" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="size" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="response_time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="communication_time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="resource_user" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EReference" name="task_name" lowerBound="1"
        eType="#//task"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="start_time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="end_time" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="network" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral="No Network"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="resource" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EReference" name="cpu_name" eType="#//processor"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="address_space_name"
eType="#//address_space"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="resource_used_by" upperBound="-1"
        eType="#//resource_user" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" lowerBound="1" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="state" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"
        defaultValueLiteral="0"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="protocol"
eType="#//Library/protocol_kind"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="offset" eSuperTypes="#//Element">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="activation" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" eType="ecore:EDataType
http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Element" abstract="true"/>
  <eClassifiers xsi:type="ecore:EClass" name="Root">
    <eStructuralFeatures xsi:type="ecore:EReference" name="processor" upperBound="-1"
        eType="#//processor" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="address_space" upperBound="-1"
        eType="#//address_space" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="task" upperBound="-1" eType="#//task"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="buffer" upperBound="-1"
        eType="#//buffer" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="message" upperBound="-1"
        eType="#//message" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="dependency" upperBound="-1"
        eType="#//dependency" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="resource" upperBound="-1"
        eType="#//resource" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="event_analyzer" upperBound="-1"
        eType="#//event_analyzer" containment="true"/>
  </eClassifiers>
  <eSubpackages name="Library">
    <eClassifiers xsi:type="ecore:EEnum" name="task_type_kind">
      <eLiterals name="PERIODIC_TYPE" literal="PERIODIC_TYPE"/>
      <eLiterals name="PARAMETRIC_TYPE" value="4"/>
      <eLiterals name="APERIODIC_TYPE" value="1"/>
      <eLiterals name="SPORADIC_TYPE" value="2"/>
      <eLiterals name="POISSON_TYPE" value="3"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="parameter_type_kind">
      <eLiterals name="integer" literal="integer"/>
      <eLiterals name="double" value="1"/>
      <eLiterals name="string" value="2"/>
      <eLiterals name="boolean" value="3"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="buffer_role_kind">
      <eLiterals name="consumer" literal="consumer"/>
      <eLiterals name="producer" value="1"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="to_from_type_kind">
      <eLiterals name="buffer"/>
      <eLiterals name="message" value="1"/>
      <eLiterals name="task" value="2"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="is_preemptive_kind">
      <eLiterals name="PREEMPTIVE" value="1" literal="PREEMPTIVE"/>
      <eLiterals name="NOT_PREEMPTIVE"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="policy_kind">
      <eLiterals name="EARLIEST_DEADLINE_FIRST_PROTOCOL"/>
      <eLiterals name="LEAST_LAXITY_FIRST_PROTOCOL" value="1"/>
      <eLiterals name="RATE_MONOTONIC_PROTOCOL" value="2"/>
      <eLiterals name="DEADLINE_MONOTONIC_PROTOCOL" value="3"/>
      <eLiterals name="POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL" value="4"/>
      <eLiterals name="TIME_SHARING_BASED_ON_WAIT_TIME_PROTOCOL" value="5"/>
      <eLiterals name="ROUND_ROBIN_PROTOCOL" value="6"/>
      <eLiterals name="MAXIMUM_URGENCY_FIRST_BASED_ON_LAXITY_PROTOCOL" value="7"/>
      <eLiterals name="MAXIMUM_URGENCY_FIRST_BASED_ON_DEADLINE_PROTOCOL" value="8"/>
      <eLiterals name="D_OVER_PROTOCOL" value="9"/>
      <eLiterals name="TIME_SHARING_BASED_ON_CPU_USAGE_PROTOCOL" value="10"/>
      <eLiterals name="PIPELINE_USER_DEFINED_PROTOCOL" value="11"/>
      <eLiterals name="NO_SCHEDULING_PROTOCOL" value="12"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="protocol_kind">
      <eLiterals name="NO_PROTOCOL"/>
      <eLiterals name="PCP" value="1"/>
      <eLiterals name="PIP" value="2"/>
      <eLiterals name="IPCP" value="3"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="task_policy_kind">
      <eLiterals name="SCHED_FIFO"/>
      <eLiterals name="SCHED_RR" value="1"/>
      <eLiterals name="SCHED_OTHERS" value="2"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EEnum" name="qs_kind">
      <eLiterals name="QS_PP1" literal="QS_PP1"/>
      <eLiterals name="QS_MM1" value="1" literal=""/>
      <eLiterals name="QS_MD1" value="2"/>
      <eLiterals name="QS_MP1" value="3"/>
      <eLiterals name="QS_MG1" value="4"/>
      <eLiterals name="QS_MMS" value="5"/>
      <eLiterals name="QS_MDS" value="6"/>
      <eLiterals name="QS_MPS" value="7"/>
      <eLiterals name="QS_MGS" value="8"/>
      <eLiterals name="QS_MM1N" value="9"/>
      <eLiterals name="QS_MD1N" value="10"/>
      <eLiterals name="QS_MP1N" value="11"/>
      <eLiterals name="QS_MG1N" value="12" literal="QS_MG1N"/>
      <eLiterals name="QS_MMSN" value="13"/>
      <eLiterals name="QS_MDSN" value="14"/>
      <eLiterals name="QS_MPSN" value="15"/>
      <eLiterals name="QS_MGSN" value="16"/>
    </eClassifiers>
  </eSubpackages>
</ecore:EPackage>
```

ANNEXE C XML metamodel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore">
  <ecore:EPackage name="XML">
    <eClassifiers xsi:type="ecore:EClass" name="Node" abstract="true">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="startLine" ordered="false" unique="false"
eType="/1/Integer"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="startColumn" ordered="false"
unique="false" eType="/1/Integer"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="endLine" ordered="false" unique="false"
eType="/1/Integer"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="endColumn" ordered="false" unique="false"
eType="/1/Integer"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" unique="false"
lowerBound="1" eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" ordered="false" unique="false"
lowerBound="1" eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="parent" ordered="false"
eType="/0/Element" eOpposite="/0/Element/children"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Attribute" eSuperTypes="/0/Node"/>
    <eClassifiers xsi:type="ecore:EClass" name="Text" eSuperTypes="/0/Node"/>
    <eClassifiers xsi:type="ecore:EClass" name="Element" eSuperTypes="/0/Node">
      <eStructuralFeatures xsi:type="ecore:EReference" name="children" upperBound="-1" eType="/0/Node"
containment="true" eOpposite="/0/Node/parent"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Root" eSuperTypes="/0/Element"/>
  </ecore:EPackage>
  <ecore:EPackage name="PrimitiveTypes">
    <eClassifiers xsi:type="ecore:EDataType" name="Boolean"/>
    <eClassifiers xsi:type="ecore:EDataType" name="Integer"/>
    <eClassifiers xsi:type="ecore:EDataType" name="String"/>
  </ecore:EPackage>
</xmi:XMI>
```

ANNEXE D ATL files

LIBRARY	FILE	UML2	MARTE	

ATL FILE MARTE4CHEDDAR

This library contains some helpers for MARTE

```
--------------------------------------------------
-- Nicolas VIENNE
-- 09/2007
--------------------------------------------------
-- This library contains some helpers for MARTE
--------------------------------------------------
library MARTE4CHEDDAR;

uses UML2;
uses MARTE;

----
-- Generic helpers
----
helper context UML!Element def :
getStereoElemsKindOfStereoAttrib(stereotype:String,attributeName:String, stereotype2:String) :
Sequence(UML!Element) =
       self.getStereotypeAttributeValue(stereotype, attributeName)->select(e2 |
e2.hasStereotypeKindOf(stereotype2));

helper context UML!Element def :
getStereoElemsTypeOfStereoAttrib(stereotype:String,attributeName:String, stereotype2:String) :
Sequence(UML!Element) =
       self.getStereotypeAttributeValue(stereotype, attributeName)->select(e2 |
e2.hasStereotypeTypeOf(stereotype2));

helper context UML!Element def : getDefaultValueAttributeByName(attribute_name:String) : OclAny =
       if not self.getOwnedAttributes()->select(e|e.name=attribute_name).first().oclIsUndefined() then
              if not self.getOwnedAttributes()-
>select(e|e.name=attribute_name).first().getDefaultValue().oclIsUndefined() then
                     self.getOwnedAttributes()-
>select(e|e.name=attribute_name).first().getDefaultValue().getValue()
              else OclUndefined endif
       else OclUndefined endif;

helper context UML!Element def : getStereoAttribDefaultValueByName(stereotype : String, attribute:
String, name:String) : UML!Element =
       let a : UML!Element = self.getStereotypeAttributeValue(stereotype, attribute)
                     ->select(a | a.name=name)->first() in
                     if not a.oclIsUndefined() then a.getDefaultValue() else OclUndefined endif;

helper context UML!Element def : getIntSADVBN(stereotype : String, attribute: String, name:String) :
UML!Element =
       let a : UML!Element = self.getStereoAttribDefaultValueByName(stereotype, attribute, name)
       in if not a.oclIsUndefined() then a.integerValue() else OclUndefined endif;

helper context UML!Element def : getStrSADVBN(stereotype : String, attribute: String, name:String) :
UML!Element =
       let a : UML!Element = self.getStereoAttribDefaultValueByName(stereotype, attribute, name)
       in if not a.oclIsUndefined() then a.stringValue() else OclUndefined endif;

helper context UML!Element def : getBoolSADVBN(stereotype : String, attribute: String, name:String) :
UML!Element =
       let a : UML!Element = self.getStereoAttribDefaultValueByName(stereotype, attribute, name)
       in if not a.oclIsUndefined() then a.booleanValue() else OclUndefined endif;

-- getFirstStereotypedTypesOfStereotypeAttribute
helper context UML!Element def : getFirstStereotypedTypeOfStereotypeAttribute(stereotype : String,
attribute : String, type_stereotype : String): UML!Element = self.getStereotypeAttributeValue(
                      stereotype, attribute).asSequence()
                      -> collect(e1 | e1.getType())
                      -> select(e2 | not e2.oclIsUndefined())
                      -> select(e3 | e3.hasStereotypeKindOf(type_stereotype)).first();
-- Sequence version
helper context UML!Element def : getFirstStereotypedTypeOfStereotypeAttribute_Seq(stereotype : String,
attribute : String, stereotypes : Sequence(String)): UML!Element = self.getStereotypeAttributeValue(
                      stereotype, attribute).asSequence()
                      -> collect(e1 | e1.getType())
                      -> select(e2 | not e2.oclIsUndefined())
                      -> select(e3 | stereotypes->select(s | e3.hasStereotypeKindOf(s))-
>notEmpty()).first();


helper context UML!Element def : getStorageResourceElementSize(stereotype : String, attribute: String,
name : String) : Integer =
       let elem : UML!Element = self.getStereoElemsKindOfStereoAttrib(stereotype, attribute,
self.StorageResource())
                      ->select(el | el.name = name)
                      .first()
              in if not elem.oclIsUndefined() then
                      elem.marteGetAttributeValue(self.StorageResource(),
self.StorageResource_elementSize()).toInteger()
              else 0 endif;


helper context UML!Element def : getSuppliersKindOf(stereotype : String) : Sequence(UML!Element) =
       self.getClientDependencies()->collect(e | e.getSuppliers())->flatten()->select(sup |
sup.marteIsKindOf(stereotype));

helper context UML!Element def : getSupplyingDependencies() : Sequence(UML!Element) =
        UML!Dependency.allInstances()->select(dep | not dep.getSupplier(self.name).oclIsUndefined());

-- Analysis Context Version
helper context UML!Element def : ACV() : String = if self.marteIsTypeOf(self.GaAnalysisContext()) then
self.GaAnalysisContext() else self.SaAnalysisContext() endif;
```

INPUT		OUTPUT		LIBRARY
MODEL	METAMODEL	MODEL	METAMODEL	FILE
IN	UML	OUT	CHEDDAR	UML2
				MARTE
				MARTE4CHEDDAR
				XMLPARAMETERS

ATL FILE MARTE2cheddar

This ATL file transforms a well formed MARTE model to Cheddar metamodel (scheduling analysis)

```
-- Thales MARTE to Cheddar (Copyright (c) THALES 2007 All rights reserved) is free software; you can
redistribute itand/or modify
-- it under the terms of the Eclipse Public License as published in http://www.eclipse.org/legal/epl-
v10.html
--
-- Thales MARTE to Cheddar is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied
-- warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the Eclipse Public License for
more details.

--------------------------------------------------
-- Nicolas VIENNE
-- Eric MAES
--------------------------------------------------
-- Transforms a well formed MARTE model to Cheddar metamodel for scheduling analysis
--------------------------------------------------
module MARTE2cheddar;
create  OUT:CHEDDAR from IN:UML, parameters:XML;

----
-- Libraries
----
uses MARTE;
uses UML2;
uses MARTE4CHEDDAR;
uses XMLPARAMETERS;

----
-- Check if the element is the element targeted by parameters
----
helper context UML!Element def : isTarget() : Boolean =
       if thisModule.hasParameter('targetElement') then
                      if self.oclIsKindOf(UML!NamedElement) then
                             if thisModule.getParameter('targetElement') = self.getQualifiedName()
then
                                    true
                             else false endif
                      else true endif
               else true endif;

----
-- Rules
----

lazy rule scheduler {
       from
               e:UML!Element (e.marteIsTypeOf(e.Scheduler()))
       to
               scheduler : CHEDDAR!scheduler
       do {
               if( not e.oclIsUndefined() ) { -- Scheduler defini ...

                      -- Scheduling Policy
                      scheduler.policy <- let sp : String = e.marteGetAttributeValue(e.Scheduler(),
e.Scheduler_schedPolicy()).toString() in
                      if sp='IN!'+e.SchedPolicyKind_EarliestDeadlineFirst() then
```

```
                              #EARLIEST_DEADLINE_FIRST_PROTOCOL
                       else if sp='IN!'+e.SchedPolicyKind_LeastLaxityFirst() then
                              #LEAST_LAXITY_FIRST_PROTOCOL
                       else if sp='IN!'+e.SchedPolicyKind_FixedPriority() then
                              #POSIX_1003_HIGHEST_PRIORITY_FIRST_PROTOCOL
                       else if sp='IN!'+e.SchedPolicyKind_RoundRobin() then
                               #ROUND_ROBIN_PROTOCOL
                       else if sp='IN!'+e.SchedPolicyKind_Other() then
                              if e.marteHasAttributeValue(e.Scheduler(),
e.Scheduler_otherSchedPolicy()) then
                                     let sp2 : String = e.marteGetAttributeValue(e.Scheduler(),
e.Scheduler_otherSchedPolicy()) in
                                     if sp2 = 'RATE_MONOTONIC_PROTOCOL' then
                                            #RATE_MONOTONIC_PROTOCOL
                                     else if sp2 = 'DEADLINE_MONOTONIC_PROTOCOL' then
                                            #DEADLINE_MONOTONIC_PROTOCOL
                                     else if sp2 = 'TIME_SHARING_BASED_ON_WAIT_TIME_PROTOCOL' then
                                            #TIME_SHARING_BASED_ON_WAIT_TIME_PROTOCOL
                                     else if sp2 = 'MAXIMUM_URGENCY_FIRST_BASED_ON_LAXITY_PROTOCOL'
then
                                            #MAXIMUM_URGENCY_FIRST_BASED_ON_LAXITY_PROTOCOL
                                     else if sp2 = 'MAXIMUM_URGENCY_FIRST_BASED_ON_DEADLINE_PROTOCOL'
then
                                            #MAXIMUM_URGENCY_FIRST_BASED_ON_DEADLINE_PROTOCOL
                                     else if sp2 = 'D_OVER_PROTOCOL' then
                                            #D_OVER_PROTOCOL
                                     else if sp2 = 'TIME_SHARING_BASED_ON_CPU_USAGE_PROTOCOL' then
                                            #TIME_SHARING_BASED_ON_CPU_USAGE_PROTOCOL
                                     else
                                            #NO_SCHEDULING_PROTOCOL -- policy=Other and otherSched
invalid
                                     endif
                                     endif
                                     endif
                                     endif
                                     endif
                                     endif
                                     endif
                              else
                                     #NO_SCHEDULING_PROTOCOL -- policy=Other and otherSched undef
                              endif
                       else if sp='IN!'+e.SchedPolicyKind_Undef() then
                              #PIPELINE_USER_DEFINED_PROTOCOL
                       else
                              #NO_SCHEDULING_PROTOCOL -- Should never happen
                       endif
                       endif
                       endif
                       endif
                       endif
                       endif;

                       if (scheduler.policy.toString() = 'NO_SCHEDULING_PROTOCOL') {
                              e.errorMessage(e.name + ' -> ' + 'Scheduler::schedPolicy and/or
Scheduler::otherSchedPolicy seems to be invalid');
                       }

                       -- quantum
                       if (not e.getDefaultValueAttributeByName('cheddar_quantum').oclIsUndefined()) {
                              scheduler.quantum <- e.getDefaultValueAttributeByName('cheddar_quantum');
                       }

                       -- Preemptibility
                       if(e.marteHasAttributeValue(e.Scheduler(), e.Scheduler_isPreemptible())) {
                              if(e.marteGetAttributeValue(e.Scheduler(), e.Scheduler_isPreemptible()) =
'true') {
                                     scheduler.is_preemptive <- #PREEMPTIVE;
                              } else if(e.marteGetAttributeValue(e.Scheduler(),
e.Scheduler_isPreemptible()) = 'false') {
                                     scheduler.is_preemptive <- #NOT_PREEMPTIVE;
                              } else {
                                     e.errorMessage(e.name + ' -> ' + 'Scheduler::isPreemtible
invalid');
                              }
```

```
                        } else {
                                e.warningMessage(e.name + ' -> ' + 'Scheduler::isPreemtible undefined');
                        }

                        -- parametric_filename
                        if (e.marteGetAttributeValue(e.Scheduler(),
e.Scheduler_schedPolicy()).toString() = 'IN!'+e.SchedPolicyKind_Undef()) {
                                if( e.marteHasAttributeValue(e.Scheduler(),
e.Scheduler_otherSchedPolicy())) {
                                        scheduler.parametric_filename <-
e.marteGetAttributeValue(e.Scheduler(), e.Scheduler_otherSchedPolicy()).toString();
                                } else {
                                        e.errorMessage('otherSchedPolicy (parametric_filename) undefined
with scheduling policy Undef');
                                }
                        }

                } else {                                             -- Scheduler undef, using default
                        e.warningMessage('Scheduler is missing, using default');
                        scheduler.policy <- #NO_SCHEDULING_PROTOCOL;

                }
        }
}


rule processor(root : CHEDDAR!Root,  e:UML!Element) {
        to
        nw : CHEDDAR!network (
                value <- 'No network'
        ),

        proc: CHEDDAR!processor (
                name <- e.getQName(),
                network_link <- nw
        )

        do {
                e.verboseMessage('processor ' + e.getQName());
                proc.scheduler <- thisModule.scheduler(e.getStereotypeAttributeValue(if
e.marteIsTypeOf(e.SaExecHost()) then e.SaExecHost() else e.GaExecHost()
endif,e.ProcessingResource_mainScheduler()));

                root.processor <- root.processor->append(proc);
        }
}

rule address_space (root : CHEDDAR!Root,  e:UML!Element) {
        to as: CHEDDAR!address_space (
                -- name
                name <- e.getQName(),
                -- text memory size
                text_memory_size <- e.getStorageResourceElementSize(e.MemoryPartition(),
e.MemoryPartition_concurrentResources(),'text_memory_size'),
                -- heap memory size
                heap_memory_size <- e.getStorageResourceElementSize(e.MemoryPartition(),
e.MemoryPartition_concurrentResources(),'heap_memory_size'),
                -- stack memory size
                stack_memory_size <- e.getStorageResourceElementSize(e.MemoryPartition(),
e.MemoryPartition_concurrentResources(),'stack_memory_size'),
                -- data memory size
                data_memory_size <- e.getStorageResourceElementSize(e.MemoryPartition(),
e.MemoryPartition_concurrentResources(),'data_memory_size'),

                -- cpu_name
                cpu_name <- let proc : UML!Element =
e.getFirstStereotypedTypeOfStereotypeAttribute_Seq(
                        e.MemoryPartition(), e.MemoryPartition_concurrentResources(),
                        Sequence {e.GaExecHost(), e.SaExecHost()})
                in if not proc.oclIsUndefined() then
                        CHEDDAR!processor.allInstances()->select(elem | elem.name = proc.getQName())-
>first()
                else OclUndefined endif,
		-- scheduler
		scheduler <- let sched : UML!Element = e.getFirstStereotypedTypeOfStereotypeAttribute(
			e.MemoryPartition(), e.MemoryPartition_concurrentResources(), e.Scheduler())
		in if not sched.oclIsUndefined() then
			thisModule.scheduler(sched)
		else OclUndefined endif
	)

	do {
		e.verboseMessage('adress_space '+e.getQName());

		if(as.cpu_name.oclIsUndefined()) {
			e.errorMessage('cpu_name undefined !');
		}

		if(as.scheduler.oclIsUndefined()) {
			e.errorMessage('scheduler undefined !');
		}

		root.address_space <- root.address_space->append(as);
	}
}



rule task (root : CHEDDAR!Root,  e:UML!Element) {
	to t: CHEDDAR!task (
		name <- e.getQName(),
		x <- 0,
		y <- 0,

		--
		-- /!\
		--
		-- Need VSL functions to parse fields
		-- see do{} section for solution ...
		task_type <- OclUndefined,
		jitter <- OclUndefined,
		period <- OclUndefined,
		parameters <- Sequence {}, -- + <<SchedulableResource>> ...

		--
		-- /!\
		--
		-- Missing inheritance link
		-- <<SwSchedulableResource>> does not inherit from <<SchedulableResource>>
		-- see do{} section for solution ...
		cpu_name <- OclUndefined,
		policy <- OclUndefined,


		-- address_space is matched by name
		address_space_name <- let as : UML!Element =
e.getFirstStereotypedTypeOfStereotypeAttribute(e.SwSchedulableResource(),
e.SwConcurrentResource_addressSpace(), e.MemoryPartition())
			in if not as.oclIsUndefined() then
				CHEDDAR!address_space.allInstances()->select(i | i.name =
as.getQName()).first()
			else OclUndefined endif,

		-- capacity
		capacity <- if e.marteHasAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_activationCapacity()) then
			e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_activationCapacity()).toInteger() else 0 endif,

		-- deadline
		deadline <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwSchedulableResource_deadlineElements(),'deadline'),
		-- criticality
		criticality <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwSchedulableResource_deadlineElements(),'criticality'),

		-- priority
```

```
               priority <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_priorityElements(), 'priority'),
               -- start time
               start_time <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_periodElements(), 'start_time'),
               -- blocking time
               blocking_time <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_periodElements(), 'blocking_time'),

               -- activation rule
               activation_rule <- e.getStrSADVBN(e.SwSchedulableResource(),
e.SwResource_stateElements(), 'activation_rule'),
               -- seed
               seed <- e.getStrSADVBN(e.SwSchedulableResource(), e.SwResource_stateElements(),
'seed'),
               -- predictable seed
               predictable_seed <- e.getBoolSADVBN(e.SwSchedulableResource(),
e.SwResource_stateElements(), 'predictable_seed'),
               -- text memory size
               text_memory_size <- e.getStorageResourceElementSize(e.SwSchedulableResource(),
e.SwResource_stateElements(), 'text_memory_size'),
               -- stack memory size
               stack_memory_size <- e.getStorageResourceElementSize(e.SwSchedulableResource(),
e.SwResource_stateElements(), 'stack_memory_size'),

               -- offsets
               offsets <- Sequence {} -- Mapping to precise
       )

       do {
               e.verboseMessage('task '+e.getQName());

               -- task type
               --
               -- /!\
               --
               -- WARNING : VSL Workaround
               t.task_type <- if e.marteHasAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type())
               then let ty : String = e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type()) in
                       if ty = 'PERIODIC_TYPE' then #PERIODIC_TYPE
                       else if ty = 'APERIODIC_TYPE' then #APERIODIC_TYPE
                       else if ty = 'SPORADIC_TYPE' then #SPORADIC_TYPE
                       else if ty = 'POISSON_TYPE' then #POISSON_TYPE
                       else if ty = 'PARAMETRIC_TYPE' then #PARAMETRIC_TYPE
                       else
                               OclUndefined                        -- default value ...
                       endif endif endif endif endif
               else OclUndefined endif;                     -- default value ...

               -- period
               t.period <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_periodElements(), 'period');
               -- jitter
               t.jitter <- e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_periodElements(), 'jitter');

               -- cpu name
               --
               -- /!\
               --
               -- WARNING : <<SchedulableResourceInheritance>> Workaround
               if(e.hasStereotypeKindOf(e.SchedulableResource())) {
                       if(e.hasStereotypeAttributeValue(e.SchedulableResource(),
e.SchedulableResource_host())) {
                               -- CPU_NAME
                               t.cpu_name <- let exechost : UML!Element =
                                       let scheduler : UML!Element =
e.getStereotypeAttributeValue(e.SchedulableResource(), e.SchedulableResource_host())
                                       in if not scheduler.oclIsUndefined() then
                                               if scheduler.hasStereotypeKindOf(e.Scheduler()) then

       scheduler.getStereotypeAttributeValue(e.Scheduler(), e.Scheduler_host())
```

```
                                             else OclUndefined endif
                                     else OclUndefined endif
                              in if not exechost.oclIsUndefined() then
                                     CHEDDAR!processor.allInstances()->select(a |
a.name=exechost.getQName())->first()
                              else OclUndefined endif;
                              if(t.cpu_name.oclIsUndefined()) {
                                     e.errorMessage('cpu_name is undefined');
                              }

                              -- policy
                              t.policy <- let scheduler : UML!Element =
e.getStereotypeAttributeValue(e.SchedulableResource(), e.SchedulableResource_host())
                                     in if not scheduler.oclIsUndefined() then
                                             let sp : String =
scheduler.marteGetAttributeValue(e.Scheduler(), e.Scheduler_schedPolicy()).toString() in
                                                    if
sp='IN!'+e.SchedPolicyKind_EarliestDeadlineFirst() then    #SCHED_FIFO
                                                    else if sp='IN!'+e.SchedPolicyKind_FixedPriority()
then    #SCHED_FIFO
                                                    else if sp='IN!'+e.SchedPolicyKind_RoundRobin()
then           #SCHED_RR
                                                    else
                                                           #SCHED_OTHERS
                                                    endif endif endif
                                     else OclUndefined endif;
                       }
               } else {
                       e.errorMessage('<<SchedulableResource>> is not applied, workaround won\'t
work');
               }

               root.task <- root.task->append(t);
       }
}


rule resource_user (root : CHEDDAR!resource, link:UML!Dependency) {
       to
               ru : CHEDDAR!resource_user (
                       -- task name
                       task_name <- CHEDDAR!task.allInstances()->select(t | t.name =
link.getClients().first().getQName())->first(),
                       -- start time
                       start_time <- let mc : UML!Constraint = UML!Constraint.allInstances()->select(c
| c.getConstrainedElements()->exists(ce | ce=link))->first()
                              in if not mc.oclIsUndefined() then mc.getSpecification().value() else
OclUndefined endif, -- TODO improve this because value is specific to opaque expressions
                       -- end time
                       end_time <- if link.marteHasVSLInteger(link.ResourceUsage(),
link.ResourceUsage_execTime(),Sequence{'value'}) and ru.start_time <> 0
                              then link.marteGetVSLInteger(link.ResourceUsage(),
link.ResourceUsage_execTime(),Sequence{'value'}) + ru.start_time - 1
                              else 0 endif
               )
       do {
               link.verboseMessage('resource_user ' +ru.task_name.name);
               root.resource_used_by <- root.resource_used_by->append(ru);
       }
}


rule resource (root : CHEDDAR!Root,  e:UML!Element) {
       to r: CHEDDAR!resource (
               -- name
               name <- e.getQName(),
               -- protocol
               protocol <- let cap: String  =
e.getStereotypeAttributeValue(e.SwMutualExclusionResource(),e.SwMutualExclusionResource_concurrentAcce
ssProtocol()).toString()
                              in             if cap = 'IN!'+e.ConcurrentAccessProtocolKind_PIP() then
#PIP
                                                    else    if cap =
'IN!'+e.ConcurrentAccessProtocolKind_PCP() then #PCP
```

```
                                                    else    if cap =
'IN!'+e.ConcurrentAccessProtocolKind_NoPreemption() then #NO_PROTOCOL
                                                    else    if cap =
'IN!'+e.ConcurrentAccessProtocolKind_Other() then
                                                                   if not e.getOwnedAttributes()-
>select(e|e.name='cheddar_IPCP').first().oclIsUndefined() then #IPCP else #NO_PROTOCOL endif
                                                    else #NO_PROTOCOL endif endif endif endif,
               -- state
               state <-  let m : String =
e.getStereotypeAttributeValue(e.SwMutualExclusionResource(),e.SwMutualExclusionResource_mechanism()).t
oString()
                              in             if m = 'IN!'+
e.MutualExclusionResourceKind_BooleanSemaphore() then 0
                              else           if m = 'IN!'+ e.MutualExclusionResourceKind_Mutex() then 1
                              else           if m = 'IN!'+
e.MutualExclusionResourceKind_CountSemaphore() then
                                     e.getIntSADVBN(e.SwMutualExclusionResource(),
e.SwResource_stateElements(), 'cheddar_state')
                              else 0 endif endif endif,
               -- address space name
               address_space_name <- let mas : UML!Element = CHEDDAR!address_space.allInstances()
                      ->select(as | e.getSuppliersKindOf(e.MemoryPartition())->exists(a | a.getQName()
= as.name))->first()
                      in if not mas.oclIsUndefined() then  mas     else OclUndefined endif,
               -- cpu name
               cpu_name <- if not r.address_space_name.oclIsUndefined() then
r.address_space_name.cpu_name else OclUndefined endif,
               -- resource used by
               resource_used_by <- Sequence {}
       )

       do {
               e.verboseMessage('resource ' + e.getQName());

               for(d in e.getSupplyingDependencies()->select(dep |
dep.marteIsKindOf(e.ResourceUsage()))) {
                      if(CHEDDAR!task.allInstances()->exists(t |
t.name=d.getClients().first().getQName())) {
                              thisModule.resource_user(r,d);
                      }
               }

               root.resource <- root.resource->append(r);
       }
}

rule buffer_user (root : CHEDDAR!buffer, link:UML!Dependency) {
       to
               bu : CHEDDAR!buffer_user (
                      -- task name
                      task_name <- CHEDDAR!task.allInstances()->select(t | t.name =
link.getClients().first().getQName())->first(),
                      -- time
                      time <- let mc : UML!Constraint = UML!Constraint.allInstances()->select(c |
c.getConstrainedElements()->exists(ce | ce=link))->first()
                              in if not mc.oclIsUndefined() then mc.getSpecification().value() else
OclUndefined endif, -- TODO improve this because value is specific to opaque expressions
                      -- size
                      size <- if
link.marteHasVSLInteger(link.ResourceUsage(),link.ResourceUsage_msgSize(),Sequence{'value'})
                              then

       link.marteGetVSLInteger(link.ResourceUsage(),link.ResourceUsage_msgSize(),Sequence{'value'})
                              else
                                     0
                              endif,
                      -- buffer role
                      buffer_role <- if bu.size > 0 then #producer else #consumer endif,
                      -- size
                      size <- if bu.size < 0 then 0-bu.size else bu.size endif
               )
       do {
               link.verboseMessage('buffer_user ' +bu.task_name.name);
```

```
                root.buffer_used_by <- root.buffer_used_by->append(bu);
        }
}


rule buffer (root : CHEDDAR!Root,e:UML!Element ) {
        to b: CHEDDAR!buffer (
                -- name
                name <- e.getQName(),
                -- x
                x <- 0,
                -- y
                y <- 0,
                -- qs
                qs <- let s: String = e.getDefaultValueAttributeByName('cheddar_qs') in
                        if s = 'QS_PP1' then #QS_PP1
                        else if s = 'QS_MM1' then #QS_MM1
                        else if s = 'QS_MD1' then #QS_MD1
                        else if s = 'QS_MP1' then #QS_MP1
                        else if s = 'QS_MG1' then #QS_MG1
                        else if s = 'QS_MMS' then #QS_MMS
                        else if s = 'QS_MDS' then #QS_MDS
                        else if s = 'QS_MPS' then #QS_MPS
                        else if s = 'QS_MGS' then #QS_MGS
                        else if s = 'QS_MM1N' then #QS_MM1N
                        else if s = 'QS_MD1N' then #QS_MD1N
                        else if s = 'QS_MP1N' then #QS_MP1N
                        else if s = 'QS_MG1N' then #QS_MG1N
                        else if s = 'QS_MMSN' then #QS_MMSN
                        else if s = 'QS_MDSN' then #QS_MDSN
                        else if s = 'QS_MPSN' then #QS_MPSN
                        else if s = 'QS_MGSN' then #QS_MGSN
                        else OclUndefined endif endif endif endif endif endif
                        endif endif endif endif endif endif endif endif endif,
                -- size
                size <- if e.marteHasAttributeValue(e.StorageResource(),
e.StorageResource_elementSize()) then
                        e.marteGetAttributeValue(e.StorageResource(),
e.StorageResource_elementSize()).toInteger()
                        else 0 endif,
                -- address space name
                address_space_name <- let mas : UML!Element = CHEDDAR!address_space.allInstances()
                        ->select(as | e.getSuppliersKindOf(e.MemoryPartition())->exists(a | a.getQName()
= as.name))->first()
                        in if not mas.oclIsUndefined() then  mas    else OclUndefined endif,
                -- cpu name
                cpu_name <- if not b.address_space_name.oclIsUndefined() then
b.address_space_name.cpu_name else OclUndefined endif,
                -- buffer used by
                buffer_used_by <- Sequence {}
        )

        do {
                e.verboseMessage('buffer ' + e.getQName());

                for(d in e.getSupplyingDependencies()->select(dep |
dep.marteIsTypeOf(e.ResourceUsage())
                        and CHEDDAR!task.allInstances()->exists(t |
t.name=dep.getClients().first().getQName())
                        )) {
                        thisModule.buffer_user(b,d);
                }
                root.buffer <- root.buffer->append(b);
        }
}

rule message (root : CHEDDAR!Root,  e:UML!Element) {
        to t: CHEDDAR!message (
                -- name
                name <- e.getQName(),
                -- x
                x <- 0,
                -- y
                y <- 0,
               -- jitter
               jitter <- e.getDefaultValueAttributeByName('cheddar_jitter'),
               -- deadline
               deadline <- if
e.marteHasVSLInteger(e.SaCommStep(),e.SaCommStep_deadline(),Sequence{'value'})
                      then e.marteGetVSLInteger(e.SaCommStep(),
e.SaCommStep_deadline(),Sequence{'value'})
                      else 0 endif,
               -- period
               period <- if e.marteHasVSLInteger(e.SaCommStep(),
e.GaScenario_interOccT(),Sequence{'value'})
                      then e.marteGetVSLInteger(e.SaCommStep(),
e.GaScenario_interOccT(),Sequence{'value'})
                      else 0 endif,
               -- response time
               response_time <- if e.marteHasVSLInteger(e.SaCommStep(),
e.GaScenario_respT(),Sequence{'value'})
                      then e.marteGetVSLInteger(e.SaCommStep(),
e.GaScenario_respT(),Sequence{'value'})
                      else 0 endif,
               -- size
               size <- if
e.marteHasVSLInteger(e.SaCommStep(),e.GaCommStep_msgSize(),Sequence{'value'})
                      then
e.marteGetVSLInteger(e.SaCommStep(),e.GaCommStep_msgSize(),Sequence{'value'})
                      else 0 endif,
               -- communication time
               communication_time <- if
e.marteHasVSLInteger(e.SaCommStep(),e.ResourceUsage_execTime(),Sequence{'value'})
                      then
e.marteGetVSLInteger(e.SaCommStep(),e.ResourceUsage_execTime(),Sequence{'value'})
                      else 0 endif
       )

       do {
               e.verboseMessage('message '+e.getQName());
               root.message <- root.message->append(t);
       }
}

rule event_analyzer (root : CHEDDAR!Root, e : String) {
       to t: CHEDDAR!event_analyzer (
               -- parametric filename
               parametric_filename <- e
       )

       do {
               e.verboseMessage('event_analyzer '+e);
               root.event_analyzer <- root.event_analyzer->append(t);
       }
}

rule dependency (root : CHEDDAR!Root, e: UML!dependency) {
       to t: CHEDDAR!dependency (
               -- to
               to <- let elem:UML!Element = CHEDDAR!DependencyElement.allInstances()
                                    ->select(a | a.name = e.getClients()->first().getQName())-
>first() in
                      if not elem.oclIsUndefined() then
                             elem
                      else OclUndefined endif,
               -- from
               from <- let elem:UML!Element = CHEDDAR!DependencyElement.allInstances()
                                    ->select(a | a.name = e.getSuppliers()->first().getQName())-
>first() in
                      if not elem.oclIsUndefined() then
                             elem
                      else OclUndefined endif,
               -- from type
               from_type <- let f : CHEDDAR!DependencyElement = t.from in
                             if not f.oclIsUndefined() then
                                    if f.oclType() = CHEDDAR!task then #task else
                                    if f.oclType() = CHEDDAR!buffer then #buffer else
                                    #message endif endif else OclUndefined endif,
               -- to type
               to_type <- let f : CHEDDAR!DependencyElement = t.to in
                              if not f.oclIsUndefined() then
                                     if f.oclType() = CHEDDAR!task then #task else
                                     if f.oclType() = CHEDDAR!buffer then #buffer else
                                     #message endif endif else OclUndefined endif
       )

       do {
               e.verboseMessage('dependency '+t.from.name + ' -> ' + t.to.name);
               root.dependency <- root.dependency->append(t);
       }
}

----
-- Root rule
----
rule AnalysisContext {
       from e:UML!NamedElement ((e.marteIsTypeOf(e.GaAnalysisContext()) or
e.marteIsTypeOf(e.SaAnalysisContext())) and e.isTarget())

       to root : CHEDDAR!Root

       do {
               e.verboseMessage('MARTE 2 Cheddar transformation for '+e.getName());

               -- processors
               for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
                      for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.GaExecHost())){
                              thisModule.processor(root, elem2);
                      }
               }

               -- address spaces
               for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
                      for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.MemoryPartition())){
                              thisModule.address_space(root, elem2);
                      }
               }

               -- tasks
               for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
                      for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.SwSchedulableResource())){
                              thisModule.task(root, elem2);
                      }
               }

               -- resources
               for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
                      for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.SwMutualExclusionResource())){
                              thisModule.resource(root, elem2);
                      }

               }

               -- buffer
               for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
                        -- buffer
                        for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.StorageResource())){
                                thisModule.buffer(root, elem2);
                        }
                }

                -- event_analyzers
                for( elem in e.marteGetAttributeValueSequence(if e.marteIsTypeOf(e.GaAnalysisContext())
then e.GaAnalysisContext() else e.SaAnalysisContext() endif, e.GaAnalysisContext_contextParams())) {
                        thisModule.event_analyzer(root, elem);
                }

                -- messages
                for(elem in e.getStereoElemsKindOfStereoAttrib(e.ACV(), e.GaAnalysisContext_workload(),
e.GaWorkloadBehavior())) {
                        for(elem2 in elem.getStereoElemsKindOfStereoAttrib(e.GaWorkloadBehavior(),
e.GaWorkloadBehavior_behavior(), e.GaScenario()) ) {
                                for(msg in elem2.getStereoElemsKindOfStereoAttrib(e.GaScenario(),
e.GaScenario_steps(), e.SaCommStep())){
                                        thisModule.message(root, msg);
                                }
                        }
                }

                -- dependencies
                for(elem in e.getStereoElemsKindOfStereoAttrib(e.ACV(), e.GaAnalysisContext_workload(),
e.GaWorkloadBehavior())) {
                        for(elem2 in elem.getStereoElemsKindOfStereoAttrib(e.GaWorkloadBehavior(),
e.GaWorkloadBehavior_behavior(), e.GaScenario()) ) {
                                for(step in elem2.getStereotypeAttributeValue(e.GaScenario(),
e.GaScenario_steps())) {
                                        if(step.oclType() = UML!Dependency and
step.marteIsKindOf(step.GaStep())) {
                                                thisModule.dependency(root, step);
                                        }
                                }
                        }
                }

        }
}
```

INPUT		OUTPUT		LIBRARY
MODEL	METAMODEL	MODEL	METAMODEL	FILE
IN	UML	OUT	PROBLEM	UML2
				MARTE
				MARTE4CHEDDAR
				XMLPARAMETERS

ATL FILE CheckMARTE4cheddar

Check if the MARTE model is well formed for MARTE2Cheddar transform

```
-- Thales MARTE to Cheddar (Copyright (c) THALES 2007 All rights reserved) is free software; you can
redistribute itand/or modify
-- it under the terms of the Eclipse Public License as published in http://www.eclipse.org/legal/epl-
v10.html
--
-- Thales MARTE to Cheddar is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied
-- warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the Eclipse Public License for
more details.


--------------------------------------------------
-- Nicolas VIENNE
-- Eric MAES
--------------------------------------------------
-- Check the if MARTE model is well formed for MARTE2Cheddar transform ...
--------------------------------------------------
module CheckMARTE4cheddar;
create  OUT:PROBLEM from IN:UML, parameters:XML;

----
-- Libraries
----
uses MARTE;
uses UML2;
uses MARTE4CHEDDAR;
uses XMLPARAMETERS;

----
-- Check if the element is the element targeted by parameters
----
helper context UML!Element def : isTarget() : Boolean =
        if thisModule.hasParameter('targetElement') then
                        if self.oclIsKindOf(UML!NamedElement) then
                                if thisModule.getParameter('targetElement') = self.getQualifiedName()
then
                                        true
                                else false endif
                        else true endif
                else true endif;

helper def : getErrorTarget(): String =
        if thisModule.hasParameter('targetElement')  then
                ' (' + thisModule.getParameter('targetElement') + ')'
        else '' endif;

----
-- Helping Rules
----

rule critic(e:UML!NamedElement, s:String) {
        to p:PROBLEM!Problem (
                severity <- #critic,
                location <- e.getQualifiedName() + thisModule.getErrorTarget(),
                description <-s
        )
}
rule error(e:UML!NamedElement, s:String) {
        to p:PROBLEM!Problem (
                severity <- #error,
                location <- e.getQualifiedName() + thisModule.getErrorTarget(),
                description <-s
        )
}
rule warning(e:UML!NamedElement, s:String) {
        to p:PROBLEM!Problem (
                severity <- #warning,
                location <- e.getQualifiedName() + thisModule.getErrorTarget(),
                description <-s
        )
}

rule info(loc: String, descr:String) {
        to p:PROBLEM!Problem (
                severity <- #warning,
                location <- loc + thisModule.getErrorTarget(),
                description <- 'INFO: '+descr
        )
}


rule unkerror2(e:UML!NamedElement,cont:String) {
        to p:PROBLEM!Problem (
                severity <- #error,
                location <- cont + thisModule.getErrorTarget(),
                description <-e.getQualifiedName() + ' is not a known element (it may be missing in a
resources list)'
        )
}

rule undeferror(e:UML!NamedElement, s:String) {
        do {
                thisModule.error(e,s+' is not defined');
        }
}
rule unkerror(e:UML!NamedElement,e2:UML!NamedElement) {
        do {
                thisModule.error(e,e2.getQualifiedName() + ' is not a known element (it may be missing
in a resources list)');
        }
}
rule unkwarning(e:UML!NamedElement,e2:UML!NamedElement) {
        do {
                thisModule.warning(e,e2.getQualifiedName() + ' is not a known element (it may be
missing in a resources list)');
        }
}
rule undefwarning(e:UML!NamedElement, s:String) {
        do {
                thisModule.warning(e,s+' is not defined');
        }
}

----
-- Rules
----
helper def : KnownNames : Sequence(String) = Sequence {};
helper def : ProcessorNames : Sequence(String) = Sequence {};
helper def : TaskNames : Sequence(String) = Sequence {};
helper def : AddressSpaceNames : Sequence(String) = Sequence {};
helper def : ResourceNames : Sequence(String) = Sequence {};
helper def : MessageNames : Sequence(String) = Sequence {};
helper def : BufferNames : Sequence(String) = Sequence {};
helper def : EANames : Sequence(String) = Sequence {};
helper def : AnalysisContextCount : Integer = 0;


----
-- Rules
```

```
----

rule scheduler(e:UML!Element) {
	do {
		-- Scheduling Policy
		let sp : String = e.marteGetAttributeValue(e.Scheduler(),
e.Scheduler_schedPolicy()).toString() in
		if sp='IN!'+e.SchedPolicyKind_EarliestDeadlineFirst() then
			0
		else if sp='IN!'+e.SchedPolicyKind_LeastLaxityFirst() then
			0
		else if sp='IN!'+e.SchedPolicyKind_FixedPriority() then
			0
		else if sp='IN!'+e.SchedPolicyKind_RoundRobin() then
			0
		else if sp='IN!'+e.SchedPolicyKind_Other() then
			if e.marteHasAttributeValue(e.Scheduler(), e.Scheduler_otherSchedPolicy()) then
				let sp2 : String = e.marteGetAttributeValue(e.Scheduler(),
e.Scheduler_otherSchedPolicy()) in
				if sp2 = 'RATE_MONOTONIC_PROTOCOL' then
					0
				else if sp2 = 'DEADLINE_MONOTONIC_PROTOCOL' then
					0
				else if sp2 = 'TIME_SHARING_BASED_ON_WAIT_TIME_PROTOCOL' then
					0
				else if sp2 = 'MAXIMUM_URGENCY_FIRST_BASED_ON_LAXITY_PROTOCOL' then
					0
				else if sp2 = 'MAXIMUM_URGENCY_FIRST_BASED_ON_DEADLINE_PROTOCOL' then
					0
				else if sp2 = 'D_OVER_PROTOCOL' then
					0
				else if sp2 = 'TIME_SHARING_BASED_ON_CPU_USAGE_PROTOCOL' then
					0
				else
							thisModule.error(e,'Invalid scheduling policy,
Scheduler::schedPolicy and/or Scheduler::otherSchedPolicy seems to be invalid')
				endif
				endif
				endif
				endif
				endif
				endif
				endif
			else
						thisModule.error(e,'Invalid scheduling policy,
Scheduler::schedPolicy and/or Scheduler::otherSchedPolicy seems to be invalid')
			endif
		else if sp='IN!'+e.SchedPolicyKind_Undef() then
			#PIPELINE_USER_DEFINED_PROTOCOL
		else
			thisModule.error(e,'Invalid scheduling policy, Scheduler::schedPolicy and/or
Scheduler::otherSchedPolicy seems to be invalid')
		endif
		endif
		endif
		endif
		endif
		endif;


		-- Preemptibility
		if(e.marteHasAttributeValue(e.Scheduler(), e.Scheduler_isPreemptible())) {
			if(not (e.marteGetAttributeValue(e.Scheduler(), e.Scheduler_isPreemptible()) =
'true') and not (e.marteGetAttributeValue(e.Scheduler(), e.Scheduler_isPreemptible()) = 'false')) {
				thisModule.error(e, 'Scheduler::isPreemtible invalid');
			}
		} else {
			thisModule.undefwarning(e, 'Scheduler::isPreemtible');
		}

		-- parametric_filename
		if (e.marteGetAttributeValue(e.Scheduler(), e.Scheduler_schedPolicy()).toString() =
'IN!'+e.SchedPolicyKind_Undef()) {
			if( not e.marteHasAttributeValue(e.Scheduler(), e.Scheduler_otherSchedPolicy()))
{
                              thisModule.error(e, 'otherSchedPolicy (parametric_filename) undefined
with scheduling policy Undef');
                       }
               }
       }
}


rule processor( e:UML!Element) {

       do {
               thisModule.verboseMessage('processor ' + e.getQName());
               let s : UML!Element = e.getStereotypeAttributeValue(if e.marteIsTypeOf(e.SaExecHost())
then e.SaExecHost() else e.GaExecHost() endif,e.ProcessingResource_mainScheduler())
               in if s.oclIsUndefined() then thisModule.undefwarning(e, 'mainScheduler')
               else thisModule.scheduler(s) endif;

               -- check for duplicate names
               if(thisModule.KnownNames->exists(s | s = e.getQName())) {
                       thisModule.error(e, 'Duplicate name found');
               } else {
                       thisModule.KnownNames <- thisModule.KnownNames->append(e.getQName());
                       thisModule.ProcessorNames <- thisModule.ProcessorNames->append(e.getQName());
               }
       }
}



rule address_space (e:UML!Element) {

       do {
               thisModule.verboseMessage('address_space '+e.getQName());

                let proc : UML!Element = e.getFirstStereotypedTypeOfStereotypeAttribute_Seq(
                       e.MemoryPartition(), e.MemoryPartition_concurrentResources(),
                       Sequence {e.GaExecHost(), e.SaExecHost()})
               in if not proc.oclIsUndefined() then
                       if thisModule.ProcessorNames->exists(s | s = proc.getQName()) then OclUndefined
-- the processor referenced exists
                       else thisModule.unkerror(e,proc) endif
               else thisModule.error(e, 'No <<GaExecHost>> or <<SaExecHost>> found in
concurrentResources') endif;

               let sched : UML!Element = e.getFirstStereotypedTypeOfStereotypeAttribute(
                       e.MemoryPartition(), e.MemoryPartition_concurrentResources(), e.Scheduler())
               in if not sched.oclIsUndefined() then
                       thisModule.scheduler(sched)
               else thisModule.error(e, 'No <<Scheduler>> found in concurrentResources') endif;

               -- check for duplicate names
               if(thisModule.KnownNames->exists(s | s = e.getQName())) {
                       thisModule.error(e, 'Duplicate name found');
               } else {
                       thisModule.KnownNames <- thisModule.KnownNames->append(e.getQName());
                       thisModule.AddressSpaceNames <- thisModule.AddressSpaceNames-
>append(e.getQName());
               }
       }
}


rule task ( e:UML!Element) {

       do {
               thisModule.verboseMessage('task '+e.getQName());

               -- address_space (matched by name)
               let as : UML!Element =
e.getFirstStereotypedTypeOfStereotypeAttribute(e.SwSchedulableResource(),
e.SwConcurrentResource_addressSpace(), e.MemoryPartition())
                       in if not as.oclIsUndefined() then
                              if thisModule.AddressSpaceNames->exists(s | s = as.getQName()) then
```

```
OclUndefined -- the referenced address_space exists
                              else thisModule.unkerror(e,as) endif
                      else thisModule.error(e,'No address_space defined (addressSpace should contain a
element which type is an existing <<MemoryPartition>> element)') endif;


               -- activation capacity
               if(not e.marteHasAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_activationCapacity())) {
                      thisModule.undeferror(e,'activationCapacity');
               } else {
                      if( e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_activationCapacity())='') {
                              thisModule.undeferror(e,'activationCapacity');
                      } else  if(not( e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_activationCapacity()).toInteger() > 0)) {
                              thisModule.error(e,'activationCapacity must be > 0');
                      }
               }

               -- priority
               if(e.getIntSADVBN(e.SwSchedulableResource(), e.SwConcurrentResource_priorityElements(),
'priority').oclIsUndefined()) {
                      thisModule.undeferror(e,'priority');
               } else {
                      if(not (e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_priorityElements(), 'priority') > 0)) {
                      thisModule.error(e,'priority must be > 0');
                      }
               }
               --
               -- /!\
               --
               -- WARNING : VSL Workaround
                if(e.marteHasAttributeValue(e.SwSchedulableResource(), e.SwConcurrentResource_type()))
{
                      if(e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type()) =  'PERIODIC_TYPE'
                                     or e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type()) =  'SPORADIC_TYPE'
                                     or e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type()) =  'POISSON_TYPE' ) {
                              if(e.getIntSADVBN(e.SwSchedulableResource(),
e.SwConcurrentResource_periodElements(), 'period').oclIsUndefined()) {
                                     thisModule.error(e,'a period must be defined for this task\'s
kind');
                              }
                      } else  if(e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type()) =  'PARAMETRIC_TYPE') {
                              if(e.getStrSADVBN(e.SwSchedulableResource(),
e.SwResource_stateElements(), 'activation_rule').oclIsUndefined()) {
                                     thisModule.error(e,'an activation rule must be defined for this
task\'s kind');
                              }
                      } else  if(e.marteGetAttributeValue(e.SwSchedulableResource(),
e.SwConcurrentResource_type()) =  'APERIODIC_TYPE') {
                              OclUndefined; --
                      } else {
                              thisModule.undeferror(e,'type');
                      }
                } else {
                      thisModule.undeferror(e,'type');
                }

               --
               -- /!\
               --
               -- WARNING : <<SchedulableResourceInheritance>> Workaround
               if(e.hasStereotypeKindOf(e.SchedulableResource())) {
                      if(e.hasStereotypeAttributeValue(e.SchedulableResource(),
e.SchedulableResource_host())) {
                              -- CPU_NAME
                               let exechost : UML!Element =
                                     let scheduler : UML!Element =
e.getStereotypeAttributeValue(e.SchedulableResource(), e.SchedulableResource_host())
                                     in if not scheduler.oclIsUndefined() then
                                            if scheduler.hasStereotypeKindOf(e.Scheduler()) then

        scheduler.getStereotypeAttributeValue(e.Scheduler(), e.Scheduler_host())
                                            else  thisModule.error(e,'host is not a valid
scheduler')endif
                                     else  thisModule.undeferror(e,'host') endif
                              in if not exechost.oclIsUndefined() then
                                     if thisModule.ProcessorNames->exists(s | s = exechost.getQName())
then OclUndefined
                                     else thisModule.unkerror(e,exechost) endif
                              else thisModule.undeferror(e, 'ExecHost of host scheduler') endif;
                      }
               } else {
                      thisModule.error(e,'<<SchedulableResource>> must be applied.');
               }

               -- check for duplicate names
               if(thisModule.KnownNames->exists(s | s = e.getQName())) {
                      thisModule.error(e, 'Duplicate name found');
               } else {
                      thisModule.KnownNames <- thisModule.KnownNames->append(e.getQName());
                      thisModule.TaskNames <- thisModule.TaskNames->append(e.getQName());
               }
        }
}


rule resource_user (link:UML!Dependency) {
        do {
               thisModule.verboseMessage('resource_user ' +link.getClients().first().getQName());

               if thisModule.TaskNames->exists(s | s =  link.getClients().first().getQName()) then
OclUndefined
               else thisModule.unkwarning(link.getSuppliers().first(),link.getClients().first())
endif;

               let mc : UML!Constraint = UML!Constraint.allInstances()->select(c |
c.getConstrainedElements()->exists(ce | ce=link))->first()
               in if not mc.oclIsUndefined() then
                      if not mc.getSpecification().isIntegral() then
                              thisModule.error(link.getSuppliers().first(), 'Invalid constraint on
resource used by' + link.getClients().first().getQualifiedName())
                      else OclUndefined endif
               else
                      thisModule.error(link.getSuppliers().first(), 'No start time constraint on
resource used by' + link.getClients().first().getQualifiedName())
               endif;

               if link.marteHasVSLInteger(link.ResourceUsage(),
link.ResourceUsage_execTime(),Sequence{'value}'})
               then
                      OclUndefined
               else
                      thisModule.error(link.getSuppliers().first(), 'execTime is required on resource
used by' + link.getClients().first().getQualifiedName())
               endif;

        }
}


rule resource ( e:UML!Element) {

        do {
               thisModule.verboseMessage('resource ' + e.getQName());

               let cap: String  =
e.getStereotypeAttributeValue(e.SwMutualExclusionResource(),e.SwMutualExclusionResource_concurrentAcce
ssProtocol()).toString()
               in     if cap = 'IN!'+e.ConcurrentAccessProtocolKind_PIP() then OclUndefined
                      else    if cap = 'IN!'+e.ConcurrentAccessProtocolKind_PCP() then  OclUndefined
                      else    if cap = 'IN!'+e.ConcurrentAccessProtocolKind_NoPreemption() then
```

```
OclUndefined
                      else    if cap = 'IN!'+e.ConcurrentAccessProtocolKind_Other() then
                                     if not e.getOwnedAttributes()-
>select(e|e.name='cheddar_IPCP').first().oclIsUndefined() then OclUndefined
                                     else thisModule.error(e,'element must have a cheddar_IPCP
attribute if the concurrent access protocol is Other') endif
                      else thisModule.undeferror(e,'concurrentAccessProtocol') endif endif endif
endif;


               let m : String =
e.getStereotypeAttributeValue(e.SwMutualExclusionResource(),e.SwMutualExclusionResource_mechanism()).t
oString()
               in             if m = 'IN!'+ e.MutualExclusionResourceKind_BooleanSemaphore() then 0
                      else           if m = 'IN!'+ e.MutualExclusionResourceKind_Mutex() then 1
                      else           if m = 'IN!'+ e.MutualExclusionResourceKind_CountSemaphore() then
                              if      e.getIntSADVBN(e.SwMutualExclusionResource(),
e.SwResource_stateElements(), 'cheddar_state').oclIsUndefined() then
                                     thisModule.warning(e,'cheddar_state is missing, using default
value 0 for count semaphore initial state')
                              else OclUndefined endif
                      else thisModule.error(e,'mechanism is invalid') endif endif endif;


               let mas : String = thisModule.AddressSpaceNames->select(as |
e.getSuppliersKindOf(e.MemoryPartition())->exists(a | a.getQName() = as))->first()
                      in if  mas.oclIsUndefined() then thisModule.undeferror(e,'address space') else
OclUndefined   endif;

               for(d in e.getSupplyingDependencies()->select(dep |
dep.marteIsKindOf(e.ResourceUsage()))) {
                      thisModule.resource_user(d);
               }

               -- check for duplicate names
               if(thisModule.KnownNames->exists(s | s = e.getQName())) {
                      thisModule.error(e, 'Duplicate name found');
               } else {
                      thisModule.KnownNames <- thisModule.KnownNames->append(e.getQName());
                      thisModule.ResourceNames <- thisModule.ResourceNames->append(e.getQName());
               }
       }
}


rule buffer_user (link:UML!Dependency) {
       do {
               thisModule.verboseMessage('buffer_user ' + link.getClients().first().getQName());

               if thisModule.TaskNames->exists(s | s =  link.getClients().first().getQName()) then
OclUndefined
               else thisModule.unkwarning(link.getSuppliers().first(),link.getClients().first())
endif;

               let mc : UML!Constraint = UML!Constraint.allInstances()->select(c |
c.getConstrainedElements()->exists(ce | ce=link))->first()
               in if not mc.oclIsUndefined() then
                      if not mc.getSpecification().isIntegral() then
                              thisModule.error(link.getSuppliers().first(), 'Invalid constraint on
buffer used by ' + link.getClients().first().getQualifiedName())
                      else OclUndefined endif
               else
                      thisModule.error(link.getSuppliers().first(), 'No time constraint on buffer used
by ' + link.getClients().first().getQualifiedName())
               endif;

               if link.marteHasVSLInteger(link.ResourceUsage(),
link.ResourceUsage_msgSize(),Sequence{'value'})
               then
                      OclUndefined
               else
                      thisModule.error(link.getSuppliers().first(), 'msgSize is required on buffer
used by' + link.getClients().first().getQualifiedName())
               endif;
```

```
        }
}


rule buffer (e:UML!Element ) {

        do {
                thisModule.verboseMessage('buffer ' + e.getQName());

                let s: String = e.getDefaultValueAttributeByName('cheddar_qs') in
                        if s <> 'QS_PP1' and s <> 'QS_MM1' and s <> 'QS_MD1'  and s <> 'QS_MP1'
                        and s <> 'QS_MG1'  and s <> 'QS_MMS'  and s <> 'QS_MDS' and s <> 'QS_MPS'
                        and s <> 'QS_MGS'     and s <> 'QS_MM1N'  and s <> 'QS_MD1N' and s <> 'QS_MP1N'
                        and s <> 'QS_MG1N' and s <> 'QS_MMSN' and s <> 'QS_MDSN'  and s <> 'QS_MPSN'
and s <> 'QS_MGSN'
                        then thisModule.error(e,'cheddar_qs is invalid') else OclUndefined endif;

                if e.marteHasAttributeValue(e.StorageResource(), e.StorageResource_elementSize()) then
                        if e.marteGetAttributeValue(e.StorageResource(),
e.StorageResource_elementSize())='' then
                                thisModule.error(e,'elementSize is invalid')
                        else OclUndefined endif
                        else thisModule.undeferror(e,'elementSize') endif;

                let mas : String = thisModule.AddressSpaceNames->select(as |
e.getSuppliersKindOf(e.MemoryPartition())->exists(a | a.getQName() = as))->first()
                in if  mas.oclIsUndefined() then thisModule.undeferror(e,'address space') else
OclUndefined   endif;


                for(d in e.getSupplyingDependencies()->select(dep |
dep.marteIsTypeOf(e.ResourceUsage()))) {
                        thisModule.buffer_user(d);
                }

                -- check for duplicate names
                if(thisModule.KnownNames->exists(s | s = e.getQName())) {
                        thisModule.error(e, 'Duplicate name found');
                } else {
                        thisModule.KnownNames <- thisModule.KnownNames->append(e.getQName());
                        thisModule.BufferNames <- thisModule.BufferNames->append(e.getQName());
                }
        }
}



rule message ( e:UML!Element) {

        do {
                thisModule.verboseMessage('message '+e.getQName());

                if e.marteHasVSLInteger(e.SaCommStep(), e.GaCommStep_msgSize(),Sequence{'value'}) then
                        if e.marteGetVSLInteger(e.SaCommStep(),
e.GaCommStep_msgSize(),Sequence{'value'}) > 0 then OclUndefined
                        else  thisModule.error(e,'msgSize must be >0') endif
                else thisModule.undeferror(e,'msgSize') endif;

                -- check for duplicate names
                if(thisModule.KnownNames->exists(s | s = e.getQName())) {
                        thisModule.error(e, 'Duplicate name found');
                } else {
                        thisModule.KnownNames <- thisModule.KnownNames->append(e.getQName());
                        thisModule.MessageNames <- thisModule.MessageNames->append(e.getQName());
                }
        }
}


rule event_analyzer (e : String) {
        do {
                thisModule.verboseMessage('event_analyzer '+e);
                -- check for duplicate names
                if(thisModule.KnownNames->exists(s | s = e)) {
                        thisModule.error(e, 'Duplicate name found');
                } else {
                        thisModule.KnownNames <- thisModule.KnownNames->append(e);
                        thisModule.EANames <- thisModule.EANames->append(e);
                }
        }
}



rule dependency (link: UML!dependency) {
        do {
                thisModule.verboseMessage('dependency '+link.getSuppliers()->first().getQName() + ' ->
' + link.getClients()->first().getQName());

                if (thisModule.TaskNames->exists(s | s =  link.getClients().first().getQName())
                        or thisModule.BufferNames->exists(s | s =  link.getClients().first().getQName())
                        or thisModule.MessageNames->exists(s | s =
link.getClients().first().getQName()))
                then OclUndefined
                else thisModule.unkerror2(link.getClients().first(), 'dependency') endif;

                if (thisModule.TaskNames->exists(s | s =  link.getSuppliers().first().getQName())
                        or thisModule.BufferNames->exists(s | s =
link.getSuppliers().first().getQName())
                        or thisModule.MessageNames->exists(s | s =
link.getSuppliers().first().getQName()))
                then OclUndefined
                else thisModule.unkerror2(link.getSuppliers().first(),'dependency') endif;
        }

}


----
-- Root rule
----
rule AnalysisContext {
        from e:UML!Element ((e.marteIsTypeOf(e.GaAnalysisContext()) or
e.marteIsTypeOf(e.SaAnalysisContext()))  and e.isTarget())

        do {
                thisModule.verboseMessage('Checking for errors ...');

                thisModule.AnalysisContextCount <- thisModule.AnalysisContextCount + 1;

                if(thisModule.AnalysisContextCount > 1) {
                        thisModule.info('','Multiple analysis context are defined.');
                }

                if(thisModule.hasParameter('targetElement')) {
                        if(e.oclIsKindOf(UML!NamedElement)){
                                if (thisModule.getParameter('targetElement') = e.getQualifiedName()) {
                                        thisModule.info('', 'Analyzing targeted element : ' +
e.getQualifiedName() );
                                }
                        }
                }

                if(e.getStereoElemsKindOfStereoAttrib(if e.marteIsTypeOf(e.GaAnalysisContext()) then
e.GaAnalysisContext() else e.SaAnalysisContext() endif, e.GaAnalysisContext_platform(),
e.GaResourcesPlatform())->size()=0) {
                        thisModule.error(e, 'No platform found');
                }

                -- processors
                if(not e.getStereoElemsKindOfStereoAttrib(if e.marteIsTypeOf(e.GaAnalysisContext())
then e.GaAnalysisContext() else e.SaAnalysisContext() endif, e.GaAnalysisContext_platform(),
e.GaResourcesPlatform())
                                        -
>exists(elem|elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resou
rces(), e.GaExecHost())->size()<>0)) {
                        thisModule.error(e, 'No processor(s) declared');
                }
```

```
		for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
			for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.GaExecHost())){
				thisModule.processor(elem2);
			}
		}

		-- address spaces
		if(not e.getStereoElemsKindOfStereoAttrib(if e.marteIsTypeOf(e.GaAnalysisContext())
then e.GaAnalysisContext() else e.SaAnalysisContext() endif, e.GaAnalysisContext_platform(),
e.GaResourcesPlatform())
			-
>exists(elem|elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resou
rces(), e.MemoryPartition())->size()<>0)) {
			thisModule.error(e, 'No address space(s) declared');
		}
		for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
			for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.MemoryPartition())){
				thisModule.address_space(elem2);
			}
		}

		-- tasks
		if(not e.getStereoElemsKindOfStereoAttrib(if e.marteIsTypeOf(e.GaAnalysisContext())
then e.GaAnalysisContext() else e.SaAnalysisContext() endif, e.GaAnalysisContext_platform(),
e.GaResourcesPlatform())
			-
>exists(elem|elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resou
rces(), e.SwSchedulableResource())->size()<>0)) {
			thisModule.error(e, 'No task(s) declared');
		}
		for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
			for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.SwSchedulableResource())){
				thisModule.task( elem2);
			}
		}

		-- resources
		for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {
			for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.SwMutualExclusionResource())){
				thisModule.resource(elem2);
			}
		}

		-- buffer
		for( elem in e.getStereoElemsKindOfStereoAttrib(if
e.marteIsTypeOf(e.GaAnalysisContext()) then e.GaAnalysisContext() else e.SaAnalysisContext() endif,
e.GaAnalysisContext_platform(), e.GaResourcesPlatform())) {

			for(elem2 in
elem.getStereoElemsKindOfStereoAttrib(e.GaResourcesPlatform(),e.GaResourcesPlatform_resources(),
e.StorageResource())){
				thisModule.buffer(elem2);
			}
		}

		-- event_analyzers
		for( elem in e.marteGetAttributeValueSequence(if e.marteIsTypeOf(e.GaAnalysisContext())
then e.GaAnalysisContext() else e.SaAnalysisContext() endif, e.GaAnalysisContext_contextParams())) {
```

```
                        thisModule.event_analyzer(elem);
                }

                -- messages
                if(e.getStereoElemsKindOfStereoAttrib(e.ACV(), e.GaAnalysisContext_workload(),
e.GaWorkloadBehavior())->first().oclIsUndefined()) {
                        thisModule.warning(e, 'no workload found');
                }
                for(elem in e.getStereoElemsKindOfStereoAttrib(e.ACV(), e.GaAnalysisContext_workload(),
e.GaWorkloadBehavior())) {
                        if( elem.getStereoElemsKindOfStereoAttrib(e.GaWorkloadBehavior(),
e.GaWorkloadBehavior_behavior(), e.GaScenario())->first().oclIsUndefined()) {
                                thisModule.warning(elem, 'no behavior found');
                        }

                        for(elem2 in elem.getStereoElemsKindOfStereoAttrib(e.GaWorkloadBehavior(),
e.GaWorkloadBehavior_behavior(), e.GaScenario()) ) {
                                for(elem3 in
elem2.getStereoElemsKindOfStereoAttrib(e.GaScenario(),e.GaScenario_steps(), e.SaCommStep())){
                                        thisModule.message(elem3);
                                }
                        }
                }

                -- dependencies
                for(elem in e.getStereoElemsKindOfStereoAttrib(e.ACV(), e.GaAnalysisContext_workload(),
e.GaWorkloadBehavior())) {
                        for(elem2 in elem.getStereoElemsKindOfStereoAttrib(e.GaWorkloadBehavior(),
e.GaWorkloadBehavior_behavior(), e.GaScenario()) ) {

                                for(step in elem2.getStereotypeAttributeValue(e.GaScenario(),
e.GaScenario_steps())) {
                                        if(step.oclType() = UML!Dependency and
step.marteIsKindOf(step.GaStep())) {
                                                thisModule.dependency( step);
                                        }
                                }
                        }
                }
        }
}
```

INPUT		OUTPUT		LIBRARY
MODEL	**METAMODEL**	**MODEL**	**METAMODEL**	**FILE**
IN	CHEDDAR	OUT	XML	

ATL FILE cheddar2XML

Transforms a Cheddar model to the XML metamodel for further XML injection

```
-- Thales MARTE to Cheddar (Copyright (c) THALES 2007 All rights reserved) is free software; you can
redistribute itand/or modify
-- it under the terms of the Eclipse Public License as published in http://www.eclipse.org/legal/epl-
v10.html
--
-- Thales MARTE to Cheddar is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied
-- warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the Eclipse Public License for
more details.

--------------------------------------------------
-- Eric MAES, Nicolas VIENNE
--------------------------------------------------
-- Transforms a Cheddar model to the XML metamodel
--------------------------------------------------
module cheddar2XML;
create OUT : XML from IN : CHEDDAR;

-- Toolkit functions
helper def: add (list:Sequence(XML!Element),element:XML!Element) : Sequence(XML!Element) =
	if (element.oclIsUndefined()) then
		list
	else
		list->append(element)
	endif;

-- Toolkit rules
rule Text (text:String,element:XML!Element) {
	to
		t : XML!Text (
			value <- text,
			parent <- element
		)
	do {
		element.children <- thisModule.add(element.children,t);
	}
}

rule Attribute(attributeName:String,attributeValue:String,element:XML!Element) {
	to
		a : XML!Attribute (
			name <- attributeName,
			value <- attributeValue,
			parent <- element
		)
	do {
		element.children <- thisModule.add(element.children,a);
	}
}

rule ElementText (elementName:String,elementValue:String,element:XML!Element) {
	to
		e : XML!Element (
			name <- elementName,
			parent <- element
		)
	do {
		thisModule.Text(elementValue,e);
		element.children <- thisModule.add(element.children,e);
	}
}
```

```
rule Scheduler(scheduler:CHEDDAR!scheduler,host:CHEDDAR!Element) {
        to
                element : XML!Element (
                        name <- 'scheduler',
                        parent <- host,
                        children <- Sequence {}
                )
        do {
                if (not scheduler.parametric_filename.oclIsUndefined()) {

        thisModule.Attribute('parametric_filename',scheduler.parametric_filename,element);
                }
                if (not scheduler.is_preemptive.oclIsUndefined()) {

        thisModule.Attribute('is_preemptive',scheduler.is_preemptive.toString(),element);
                }
                if (not scheduler.quantum.oclIsUndefined()) {
                        thisModule.Attribute('quantum',scheduler.quantum.toString(),element);
                }

                if (not scheduler.policy.oclIsUndefined()) {
                        thisModule.Text(scheduler.policy.toString(),element);
                }
                thisModule.add(host.children,element);
        }
}

rule Network (network:CHEDDAR!network,host:CHEDDAR!Element) {
        to
                element : XML!Element (
                        name <- 'network_link',
                        parent <- host,
                        children <- Sequence {}
                )
        do {
                if (not network.value.oclIsUndefined()) {
                        thisModule.Text(network.value,element);
                }
                thisModule.add(host.children,element);
        }
}

rule Parameter (parameter:CHEDDAR!parameter,host:CHEDDAR!Element) {
        to
                element : XML!Element (
                        name <- 'parameter',
                        parent <- host,
                        children <- Sequence {}
                )
        do {
                if (not parameter.parameter_type.oclIsUndefined()) {

        thisModule.Attribute('parameter_type',parameter.parameter_type.toString(),element);
                }
                if (not parameter.name.oclIsUndefined() and not parameter.value.oclIsUndefined()) {
                        thisModule.Text(parameter.name+' '+parameter.value.toString(),element);
                }
                thisModule.add(host.children,element);
        }
}

rule Offset (offset:CHEDDAR!offset,host:CHEDDAR!Element) {
        to
                element : XML!Element (
                        name <- 'offset',
                        parent <- host,
                        children <- Sequence {}
                )
        do {
                if (not offset.activation.oclIsUndefined() and not offset.value.oclIsUndefined()) {
                        thisModule.Text(offset.activation.toString()+'
'+offset.value.toString().toString(),element);
```

```
                }
                thisModule.add(host.children,element);
        }
}

rule ResourceUser (resource_user:CHEDDAR!resource_user,host:CHEDDAR!Element) {
        to
                element : XML!Element (
                        name <- 'resource_user',
                        parent <- host,
                        children <- Sequence {}
                )
        do {
                if (not resource_user.task_name.oclIsUndefined() and not
resource_user.start_time.oclIsUndefined() and not resource_user.end_time.oclIsUndefined()) {
                        thisModule.Text(resource_user.task_name.name+'
'+resource_user.start_time.toString()+' '+resource_user.end_time.toString(),element);
                }
                thisModule.add(host.children,element);
        }
}

rule BufferUser (buffer_user:CHEDDAR!buffer_user,host:CHEDDAR!Element) {
        to
                element : XML!Element (
                        name <- 'buffer_user',
                        parent <- host,
                        children <- Sequence {}
                )
        do {

                if (not buffer_user.buffer_role.oclIsUndefined()) {
                        thisModule.Attribute('buffer_role',buffer_user.buffer_role.toString(),element);
                }
                if (not buffer_user.task_name.oclIsUndefined() and not
buffer_user.time.oclIsUndefined() and not buffer_user.size.oclIsUndefined()) {
                        thisModule.Text(buffer_user.task_name.name+'  '+buffer_user.size.toString()+'
'+buffer_user.time.toString()+' ',element);
                }
                thisModule.add(host.children,element);
        }
}

rule Processors(root:XML!Root) {
        to
                processors : XML!Element (
                        name <- 'processors',
                        children <- CHEDDAR!processor.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,processors);
        }
}

rule AddressSpaces(root:XML!Root) {
        to
                address_spaces : XML!Element (
                        name <- 'address_spaces',
                        children <- CHEDDAR!address_space.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,address_spaces);
        }
}

rule Tasks(root:XML!Root) {
        to
                tasks : XML!Element (
                        name <- 'tasks',
                        children <- CHEDDAR!task.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,tasks);
        }
```

```
}

rule Resources(root:XML!Root) {
        to
                resources : XML!Element (
                        name <- 'resources',
                        children <- CHEDDAR!resource.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,resources);
        }
}

rule Buffers(root:XML!Root) {
        to
                buffers : XML!Element (
                        name <- 'buffers',
                        children <- CHEDDAR!buffer.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,buffers);
        }
}

rule Dependencies(root:XML!Root) {
        to
                dependencies : XML!Element (
                        name <- 'dependencies',
                        children <- CHEDDAR!dependency.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,dependencies);
        }
}

rule Messages(root:XML!Root) {
        to
                messages : XML!Element (
                        name <- 'messages',
                        children <- CHEDDAR!message.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,messages);
        }
}

rule EventAnalyzers(root:XML!Root) {
        to
                event_analyzers : XML!Element (
                        name <- 'event_analyzers',
                        children <- CHEDDAR!event_analyzer.allInstances()
                )
        do {
                root.children <- thisModule.add(root.children,event_analyzers);
        }
}

rule Parameters(task:CHEDDAR!task,host:XML!Element) {
        to
                parameters : XML!Element (
                        name <- 'parameters',
                        children <- Sequence{}
                )
        do {
                for (ts in task.parameters)  {
                        thisModule.Parameter(ts,parameters);
                }
                host.children <- thisModule.add(host.children,parameters);
        }
}

rule Offsets(task:CHEDDAR!task,host:XML!Element) {
        to
                offsets : XML!Element (
                        name <- 'offsets',
                        children <- Sequence{}
                )
        do {
                for (ts in task.offsets)  {
                        thisModule.Offset(ts,offsets);
                }
                host.children <- thisModule.add(host.children,offsets);
        }
}

rule ResourceUsedBy(resource:CHEDDAR!resource,host:XML!Element) {
        to
                resource_used_by : XML!Element (
                        name <- 'resource_used_by',
                        children <- Sequence{}
                )
        do {
                for (rs in resource.resource_used_by)  {
                        thisModule.ResourceUser(rs,resource_used_by);
                }
                host.children <- thisModule.add(host.children,resource_used_by);
        }
}

rule BufferUsedBy(buffer:CHEDDAR!buffer,host:XML!Element) {
        to
                buffer_used_by : XML!Element (
                        name <- 'buffer_used_by',
                        children <- Sequence{}
                )
        do {
                for (bs in buffer.buffer_used_by)  {
                        thisModule.BufferUser(bs,buffer_used_by);
                }
                host.children <- thisModule.add(host.children,buffer_used_by);
        }
}

rule Root {
        from
                cheddar : CHEDDAR!Root
        to
                root : XML!Root (
                        name <- 'cheddar',
                        children <- Sequence{}
                )

                do {

                        if (CHEDDAR!processor.allInstances()->notEmpty()) {
                                thisModule.Processors(root);
                        }
                        if (CHEDDAR!address_space.allInstances()->notEmpty()) {
                                thisModule.AddressSpaces(root);
                        }
                        if (CHEDDAR!task.allInstances()->notEmpty()) {
                                thisModule.Tasks(root);
                        }
                        if (CHEDDAR!message.allInstances()->notEmpty()) {
                                thisModule.Messages(root);
                        }
                        if (CHEDDAR!resource.allInstances()->notEmpty()) {
                                thisModule.Resources(root);
                        }
                        if (CHEDDAR!buffer.allInstances()->notEmpty()) {
                                thisModule.Buffers(root);
                        }
                        if (CHEDDAR!dependency.allInstances()->notEmpty()) {
                                thisModule.Dependencies(root);
                        }
                        if (CHEDDAR!event_analyzer.allInstances()->notEmpty()) {
                                thisModule.EventAnalyzers(root);
                        }
                }
}

rule Processor {
        from
                processor : CHEDDAR!processor
        to
                element : XML!Element (
                        name <- 'processor',
                        children <- Sequence{}
                )
        do {
                if (not processor.name.oclIsUndefined()) {
                        thisModule.ElementText('name',processor.name,element);
                }
                if (not processor.scheduler.oclIsUndefined()) {
                        thisModule.Scheduler(processor.scheduler,element);
                }
                if (not processor.network_link.oclIsUndefined()) {
                        thisModule.Network(processor.network_link,element);
                }
        }
}

rule AddressSpace {
        from
                address_space : CHEDDAR!address_space
        to
                element : XML!Element (
                        name <- 'address_space',
                        children <- Sequence{}
                )
        do {
                if (not address_space.name.oclIsUndefined()) {
                        thisModule.ElementText('name',address_space.name,element);
                }
                if (not address_space.cpu_name.oclIsUndefined()) {
                        thisModule.ElementText('cpu_name',address_space.cpu_name.name,element);
                }
                if (not address_space.text_memory_size.oclIsUndefined()) {

        thisModule.ElementText('text_memory_size',address_space.text_memory_size.toString(),element);
                }
                if (not address_space.stack_memory_size.oclIsUndefined()) {

        thisModule.ElementText('stack_memory_size',address_space.stack_memory_size.toString(),element);
                }
                if (not address_space.data_memory_size.oclIsUndefined()) {

        thisModule.ElementText('data_memory_size',address_space.data_memory_size.toString(),element);
                }
                if (not address_space.heap_memory_size.oclIsUndefined()) {

        thisModule.ElementText('heap_memory_size',address_space.heap_memory_size.toString(),element);
                }
                if (not address_space.scheduler.oclIsUndefined()) {
                        thisModule.Scheduler(address_space.scheduler,element);
                }
        }
}

rule Task {
        from
                task : CHEDDAR!task
        to
                element : XML!Element (
                        name <- 'task',
                        children <- Sequence{}
                )
        do {
                if (not task.x.oclIsUndefined()) {
                        thisModule.Attribute('x',task.x.toString(),element);
                }
                if (not task.y.oclIsUndefined()) {
```

```
				thisModule.Attribute('y',task.y.toString(),element);
			}
			if (not task.task_type.oclIsUndefined()) {
				thisModule.Attribute('task_type',task.task_type.toString(),element);
			}
			if (not task.cpu_name.oclIsUndefined()) {
				thisModule.ElementText('cpu_name',task.cpu_name.name,element);
			}
			if (not task.address_space_name.oclIsUndefined()) {

		thisModule.ElementText('address_space_name',task.address_space_name.name,element);
			}
			if (not task.name.oclIsUndefined()) {
				thisModule.ElementText('name',task.name,element);
			}
			if (not task.capacity.oclIsUndefined()) {
				thisModule.ElementText('capacity',task.capacity.toString(),element);
			}
			if (not task.start_time.oclIsUndefined()) {
				thisModule.ElementText('start_time',task.start_time.toString(),element);
			}
			if (not task.policy.oclIsUndefined()) {
				thisModule.ElementText('policy',task.policy.toString(),element);
			}
			if (not task.deadline.oclIsUndefined()) {
				thisModule.ElementText('deadline',task.deadline.toString(),element);
			}
			if (not task.criticality.oclIsUndefined()) {
				thisModule.ElementText('criticality',task.criticality.toString(),element);
			}
			if (not task.blocking_time.oclIsUndefined()) {
				thisModule.ElementText('blocking_time',task.blocking_time.toString(),element);
			}
			if (not task.priority.oclIsUndefined()) {
				thisModule.ElementText('priority',task.priority.toString(),element);
			}
			if (not task.text_memory_size.oclIsUndefined()) {

		thisModule.ElementText('text_memory_size',task.text_memory_size.toString(),element);
			}
			if (not task.stack_memory_size.oclIsUndefined()) {

		thisModule.ElementText('stack_memory_size',task.stack_memory_size.toString(),element);
			}
			if (not task.period.oclIsUndefined()) {
				thisModule.ElementText('period',task.period.toString(),element);
			}
			if (not task.jitter.oclIsUndefined()) {
				thisModule.ElementText('jitter',task.jitter.toString(),element);
			}
			if (not task.parameters.oclIsUndefined() and task.parameters.notEmpty()) {
				thisModule.Parameters(task,element);
			}
			if (not task.offsets.oclIsUndefined() and task.offsets.notEmpty()) {
				thisModule.Offsets(task,element);
			}
		}
}

rule Resource {
		from
			resource : CHEDDAR!resource
		to
			element : XML!Element (
				name <- 'resource',
				children <- Sequence{}
			)
		do {
			if (not resource.cpu_name.oclIsUndefined()) {
				if (not resource.cpu_name.name.oclIsUndefined()) {
					thisModule.ElementText('cpu_name',resource.cpu_name.name,element);
				}
			}
			if (not resource.address_space_name.oclIsUndefined()) {
```

```
                        if (not resource.address_space_name.name.oclIsUndefined()) {

        thisModule.ElementText('address_space_name',resource.address_space_name.name,element);
                        }
                }
                if (not resource.name.oclIsUndefined()) {
                        thisModule.ElementText('name',resource.name,element);
                }
                if (not resource.state.oclIsUndefined()) {
                        thisModule.ElementText('state',resource.state.toString(),element);
                }
                if (not resource.protocol.oclIsUndefined()) {
                        thisModule.ElementText('protocol',
                                let m:String=resource.protocol.toString()
                                in if m='PCP' then 'PRIORITY_CEILING_PROTOCOL'
                                        else if m='PIP' then 'PRIORITY_INHERITANCE_PROTOCOL'
                                        else if m='IPCP' then 'IMMEDIATE_PRIORITY_CEILING_PROTOCOL'
                                        else 'NO_PROTOCOL' endif endif endif
                                ,element);
                }
                if (not resource.resource_used_by.oclIsUndefined() and
resource.resource_used_by.notEmpty()) {
                        thisModule.ResourceUsedBy(resource,element);
                }
        }
}

rule EventAnalyzer {
        from
                event_analyzer : CHEDDAR!event_analyzer
        to
                element : XML!Element (
                        name <- 'event_analyzer',
                        children <- Sequence{}
                )
        do {
                if (not event_analyzer.parametric_filename.oclIsUndefined()) {

        thisModule.Attribute('parametric_filename',event_analyzer.parametric_filename,element);
                }
        }
}

rule Buffer {
        from
                buffer : CHEDDAR!buffer
        to
                element : XML!Element (
                        name <- 'buffer',
                        children <- Sequence{}
                )
        do {
                if (not buffer.x.oclIsUndefined()) {
                        thisModule.Attribute('x',buffer.x.toString(),element);
                }
                if (not buffer.y.oclIsUndefined()) {
                        thisModule.Attribute('y',buffer.y.toString(),element);
                }
                if (not buffer.cpu_name.oclIsUndefined()) {
                        thisModule.ElementText('cpu_name',buffer.cpu_name.name,element);
                }
                if (not buffer.address_space_name.oclIsUndefined()) {

        thisModule.ElementText('address_space_name',buffer.address_space_name.name,element);
                }
                if (not buffer.name.oclIsUndefined()) {
                        thisModule.ElementText('name',buffer.name,element);
                }
                if (not buffer.size.oclIsUndefined()) {
                        thisModule.ElementText('size',buffer.size.toString(),element);
                }
                if (not buffer.qs.oclIsUndefined()) {
                        thisModule.ElementText('qs',buffer.qs.toString(),element);
                }
```

```
			if (not buffer.buffer_used_by.oclIsUndefined() and buffer.buffer_used_by.notEmpty()) {
				thisModule.BufferUsedBy(buffer,element);
			}
		}
}

rule Dependency {
		from
			dependency : CHEDDAR!dependency
		to
			element : XML!Element (
				name <- 'dependency',
				children <- Sequence{}
			)
		do {
			if (not dependency.to_type.oclIsUndefined()) {
				thisModule.Attribute('to_type',dependency.to_type.toString(),element);
			}
			if (not dependency.from_type.oclIsUndefined()) {
				thisModule.Attribute('from_type',dependency.from_type.toString(),element);
			}
			if (not dependency.from.oclIsUndefined() and not dependency.to.oclIsUndefined()) {
				thisModule.Text(dependency.from.name+' '+dependency.to.name,element);
			}
		}
}

rule Message {
		from
			message : CHEDDAR!message
		to
			element : XML!Element (
				name <- 'message',
				children <- Sequence{}
			)
		do {
			if (not message.x.oclIsUndefined()) {
				thisModule.Attribute('x',message.x.toString(),element);
			}
			if (not message.y.oclIsUndefined()) {
				thisModule.Attribute('y',message.y.toString(),element);
			}
			if (not message.name.oclIsUndefined()) {
				thisModule.ElementText('name',message.name,element);
			}
			if (not message.jitter.oclIsUndefined()) {
				thisModule.ElementText('jitter',message.jitter.toString(),element);
			}
			if (not message.deadline.oclIsUndefined()) {
				thisModule.ElementText('deadline',message.deadline.toString(),element);
			}
			if (not message.period.oclIsUndefined()) {
				thisModule.ElementText('period',message.period.toString(),element);
			}
			if (not message.size.oclIsUndefined()) {
				thisModule.ElementText('size',message.size.toString(),element);
			}
			if (not message.response_time.oclIsUndefined()) {

		thisModule.ElementText('response_time',message.response_time.toString(),element);
			}
			if (not message.communication_time.oclIsUndefined()) {

		thisModule.ElementText('communication_time',message.communication_time.toString(),element);
			}
		}
}
```

ANNEXE E Cheddar DTD

```
<?xml-stylesheet type="text/xsl" href="cheddar_project.xsl"?>
<!DOCTYPE cheddar [
<!ELEMENT name (#PCDATA) >
<!ELEMENT scheduler (#PCDATA) >
<!ELEMENT parameter (#PCDATA) >
<!ELEMENT network_link (#PCDATA) >
<!ELEMENT offset (#PCDATA) >
<!ELEMENT period (#PCDATA) >
<!ELEMENT capacity (#PCDATA) >
<!ELEMENT deadline (#PCDATA) >
<!ELEMENT blocking_time (#PCDATA) >
<!ELEMENT policy (#PCDATA) >
<!ELEMENT priority (#PCDATA) >
<!ELEMENT cpu_name (#PCDATA) >
<!ELEMENT address_space_name (#PCDATA) >
<!ELEMENT jitter (#PCDATA) >
<!ELEMENT seed (#PCDATA) >
<!ELEMENT predictable_seed (#PCDATA) >
<!ELEMENT activation_rule (#PCDATA) >
<!ELEMENT state (#PCDATA) >
<!ELEMENT protocol (#PCDATA) >
<!ELEMENT size (#PCDATA) >
<!ELEMENT qs (#PCDATA) >
<!ELEMENT time (#PCDATA) >
<!ELEMENT buffer_user (#PCDATA) >
<!ELEMENT resource_user (#PCDATA) >
<!ELEMENT event_Analyzer (#PCDATA) >
<!ELEMENT type (#PCDATA) >
<!ELEMENT text_memory_size (#PCDATA) >
<!ELEMENT heap_memory_size (#PCDATA) >
<!ELEMENT stack_memory_size (#PCDATA) >
<!ELEMENT data_memory_size (#PCDATA) >
<!ELEMENT cheddar (processors, (address_spaces)?, (tasks)?, ((event_analyzers)?|(networks)?|
(buffers)?|(resources)?|(messages)?),(dependencies)?  )  >
<!ELEMENT processors (processor)+ >
<!ELEMENT processor (name|scheduler|network_link) >
<!ELEMENT address_spaces (address_space)+ >
<!ELEMENT address_space (name|text_memory_size|data_memory_size|stack_memory_size|heap_memory_size) >
<!ELEMENT networks (network)+ >
<!ELEMENT network (name|type) >
<!ELEMENT tasks (task)+>
<!ELEMENT task
(name|cpu_name|address_space_name|capacity|start_time|(stack_memory_size)?|(text_memory_size)?|(period)?|
(deadline)?|(parameters)?|(offsets)?|(jitter)?|(policy)?|(priority)?|(predictable_seed)?|
(blocking_time)?|(seed)?
|(activation_rule)?) >
<!ELEMENT offsets (offset)+ >
<!ELEMENT parameters (parameter)+ >
<!ELEMENT messages (message)+ >
<!ELEMENT message (name|size|(period)?|(deadline)?|(jitter)?) >
<!ELEMENT buffers (buffer)+ >
<!ELEMENT buffer (cpu_name|address_space_name|qs|name|size|(buffer_used_by)?) >
<!ELEMENT buffer_used_by (buffer_user)+ >
<!ELEMENT resources (resource)+ >
<!ELEMENT resource (cpu_name|address_space_name|name|protocol|(state)?|(resource_used_by)?) >
<!ELEMENT resource_used_by (resource_user)+ >
<!ELEMENT dependencies (dependency)+ >
<!ELEMENT event_analyzers (event_analyzer)+ >
<!ELEMENT dependency (#PCDATA)  >
<!ATTLIST scheduler
        quantum CDATA  "0"
        is_preemptive CDATA  "PREEMPTIVE"
        parametric_file_name CDATA "" >
<!ATTLIST event_analyzer
        parametric_file_name CDATA "" >
<!ATTLIST task
        task_type CDATA  "APERIODIC_TYPE"
        x CDATA "0"
        y CDATA "0" >
<!ATTLIST buffer
        x CDATA "0"
        y CDATA "0" >
<!ATTLIST message
        x CDATA "0"
        y CDATA "0" >
<!ATTLIST buffer_user
        buffer_role CDATA  "producer" >
<!ATTLIST dependency
        from_type CDATA "task"
        to_type CDATA "task" >
<!ATTLIST parameter
        parameter_type CDATA  "integer" >
] >
```