Full-stack of implemented OCL rules

April 4, 2022

This document shows a complete set of OCL rules for validating the models of our Model4_DataCTrack profile. As you can see, some of these rules could also be applied for validating other very similar properties. This document is divided into three sections. Section 1 is devoted to the OCL rules focusing on extending the UML base constraints to provide structural correctness to the profile models. Section 2 presents the subset of rules created to introduce the constraints based on the GDPR legislation, which should be fulfilled in the profile models. Finally, section 3 collects the OCL rules defined as warnings, introduced to inform the tool's users when certain parameters have undesirable values .

1 Structure Consistency Rules

This section presents the complete set of OCL rules implemented to validate the model's structural coherence and correctness. These rules validate some additional constraints regarding the modeling that cannot be checked with base UML. For instance, it is possible to force that an attribute must be provided a value at the time of defining, by establishing a multiplicity of one on this attribute. However, UML is unable to recognize whether or not it is sensible that such value is negative or zero. Therefore, the following rules (see tables 1, 2, 3, and 4) are created to provide that additional layer of structural validation.

Table 1: OCL Rules defined for structure consistency (I).

Attributes	**Value**
Cons. Rule 1	**all_machines_must_contain_data_to_update**
Severity	ERROR
Context	upDate
Description	This rule validates that the set of data to be updated, *Data* parameter on the *update* message, is found on all of the machines indicated by the *machines* parameter of that message.
Specification	`self.machines->forAll(m | m.storage.data->includes(self.data))`
Cons. Rule 2	**newLocation_machine_must_be_under_sla_with_controller**
Severity	ERROR
Context	ControllerCP
Description	This rule validates that the new data processor (where it is copied or moved) has signed an SLA with the controller of such data to regulate the processing. For this purpose, the controller's *AccessLog* is checked for matching SLAs in both the processor's and controller's SLA lists.
Specification	`self.accesslog->forAll(log | self.sla->exists(sla | log.newLocation.sla->includes(sla)))`
Cons. Rule 3	**no_empty_racks**
Severity	ERROR
Context	Rack
Description	This rule checks that the number of boards and machines per board on a rack is greater than zero.
Specification	`self.machinesPerBoard>0 and self.boards>0`
Cons. Rule 4	**no_empty_datacenters**
Severity	ERROR
Context	RackElement
Description	This rule validates that the number of racks of any data center is greater than zero.
Specification	`self.numberOfracks>0`
Cons. Rule 5	**no_empty_infra**
Severity	ERROR
Context	DataCenterElement
Description	This rule checks that the number of data centers in the infrastructure is greater than zero.
Specification	`self.numberOfDataCenters>0`
Cons. Rule 6	**cpu_cores_and_and_flops_greater_than_0**
Severity	ERROR
Context	CPU
Description	Similar to the previous rules, this rule validates that any *CPU* instance is not initialized with a core with FLOPS equal to 0 or negative.
Specification	`self.cores>0 and self.FLOPs>0`

Table 2: OCL Rules defined for structure consistency (II).

Attributes	**Value**
Cons. Rule 7	**latency_name_not_empty**
Severity	ERROR
Context	Latency
Description	This rule produces an error when the name attribute of an instance of the *Latency* stereotype is an empty string (""), i.e its length is zero.
Specification	`self.name.size()>0`
Cons. Rule 8	**size_value_greater_than_0**
Severity	ERROR
Context	Size
Description	This simple rule checks that the value for the size of a memory is greater than 0
Specification	`self.value>0`
Cons. Rule 9	**time_value_greater_than_0**
Severity	ERROR
Context	Time
Description	In the same way as the previous rule, this one checks that the value for any instance of *Time* is greater than zero.
Specification	`self.value>0`
Cons. Rule 10	**bandwidth_value_greater_than_0**
Severity	ERROR
Context	Bandwidth
Description	Like the previous two rules, this one validates that the value for any instance of *Bandwidth* (i.e. any attribute of such type) is greater than zero.
Specification	`self.value>0`
Cons. Rule 11	**numberOfDrivers_greater_than_0**
Severity	ERROR
Context	Storage
Description	This one validates that no instance of *Storage* is parameterized with a value for its number of drivers attribute negative or equal to zero.
Specification	`self.numberOfDrivers>0`

Table 3: OCL Rules defined for structure consistency (III).

Attributes	**Value**
Cons. Rule 12	**sendData_maxTime_value_greater_than_0**
Severity	ERROR
Context	sendData
Description	This rule checks that the value for the time established as maximum data processing time in an instance of *sendData* is greater than zero.
Specification	`self.maxTime.value>0`
Cons. Rule 13	**combine_maxTime_greater_than_0**
Severity	ERROR
Context	combineData
Description	This rule works exactly as the previous one does but for the data processing time in a *combineData* message.
Specification	`self.maxTime.value>0`
Cons. Rule 14	**maxSubTime_greater_than_0**
Severity	ERROR
Context	subscribe
Description	This rule checks that the time in a *subscribe* message, which establishes the time for which a controller subscribes to know any breaches or changes on a processor, is greater than zero.
Specification	`self.maxSubscriptionTime.value>0`
Cons. Rule 15	**machine_contains_data_to_rectify**
Severity	ERROR
Context	newData
Description	This rule is analogous to the first consistency rule for the *newData* message. It checks that all the machines in the destination list contain the data referred to by the message.
Specification	`self.machines->forAll(m | m.storage.data->includes(self.data))`
Cons. Rule 16	**machine_contains_data_to_erase**
Severity	ERROR
Context	eraseData
Description	This checks that all the machines in the destination machines list of *eraseData* message contain the data.
Specification	`self.machines->forAll(m | m.storage.data->includes(self.data))`

Table 4: OCL Rules defined for structure consistency (IV).

Attributes	Value
Cons. Rule 17	**machine_contains_data_to_subscribe_to**
Severity	ERROR
Context	subscribe
Description	This rule checks that the data, in a *subscribe* message, exists in all the machines to which the controller wants to subscribe.
Specification	`self.machines->forAll(m | m.storage.data->includes(self.data))`
Cons. Rule 18	**l1_machine_must_be_under_SLA_with_controller**
Severity	ERROR
Context	ControllerCP
Description	This rule checks the storage location of a log record, indicating where the data accessed are, belongs to a machine that is under SLA with the controller of data.
Specification	`self.machines->forAll(m | m.storage.data->includes(self.data))`

2 GDPR-based Rules

This section introduces the rules based on the GDPR principles considered for our profile. These rules are meant to ensure congruity with those principles through OCL, since base UML constraints fall short to check such laws, as it already struggles with some of the constraints needed for section 1. These rules are all collected in tables 5, 6 and 7.

Table 5: OCL rules derived from GDPR (I).

<table>
<tr><th>Attributes</th><th>Value</th></tr>
<tr><td><b>GDPR Rule 1</b></td><td><b>upDate_destinantion_machines_comply_with_GDPR</b></td></tr>
<tr><td>Severity</td><td>ERROR</td></tr>
<tr><td>Context</td><td>upDate</td></tr>
<tr><td>Description</td><td>This rule verifies that all machines to which the <i>upDate</i> message is sent have been evaluated as compliant with GDPR standards.</td></tr>
<tr><td>Specification</td><td>self.machines->forAll(m | m.GDPRCompliance=true)</td></tr>
<tr><td><b>GDPR Rule 2</b></td><td><b>allowed_access_purpose</b></td></tr>
<tr><td>Severity</td><td>ERROR</td></tr>
<tr><td>Context</td><td>StickyPolicy</td></tr>
<tr><td>Description</td><td>This rule forces all accesses in StickyPolicy's <i>accessHistory</i> list to have a purpose included in the policy's list of purposes.</td></tr>
<tr><td>Specification</td><td>self.accessHistory->forAll(his | his.purpose->forAll(p | self.purpose-> includes(p)))</td></tr>
<tr><td><b>GDPR Rule 3</b></td><td><b>tp_in_history_given_permissions</b></td></tr>
<tr><td>Severity</td><td>ERROR</td></tr>
<tr><td>Context</td><td>AccessLog</td></tr>
<tr><td>Description</td><td>This rule checks that all third parties (TPs) in StickyPolicy's <i>accessHistory</i> list have previously received the consent for that access. For this purpose, the information about these accesses stored in the controller log is used to validate it.</td></tr>
<tr><td>Specification</td><td>self.accessHistory->forAll(his | AccessLog.allInstances->exists(log | log.tp = his.tp and log.action = his.actionPerformed))</td></tr>
<tr><td><b>GDPR Rule 4</b></td><td><b>log_access_match_sp_access</b></td></tr>
<tr><td>Severity</td><td>ERROR</td></tr>
<tr><td>Context</td><td>AccessLog</td></tr>
<tr><td>Description</td><td>This rule forces the access action and the third party registered in the controller log to match those in the <i>accessHistory</i> list of the StickyPolicy included in such a log.</td></tr>
<tr><td>Specification</td><td>AccessLog.allInstances()->forAll(log | log.sp.accessHistory->exists(access | access.tp = log.tp and access.actionPerformed=log.action)</td></tr>
</table>

Table 6: OCL rules derived from GDPR (II).

Attributes	**Value**
GDPR Rule 5	**no_access_permission_given_without_user_consent**
Severity	ERROR
Context	permission
Description	This rule throws an error when a *permission* message is passed from a controller to a third party without the required previous messages to retrieve the user's corresponding consent and affirmative response.
Specification	`permission.allInstances()->forAll(ok.allInstances()-> exists(okmsg | self.purpose->forAll(p | okmsg.purpose-> includes(p)) and okmsg.permissionType=self.permissionType) and consentInfo.allInstances()->exists(consentmsg | self.purpose->forAll(p | consentmsg.purpose->includes(p)) and consentmsg.action=self.permissionType and consentmsg.tp=StatelessAppCTP.allInstances()-> select(tp | tp.base_Lifeline.coveredBy-> includes(self.base_Message.receiveEvent))))`
GDPR Rule 6	**no_empty_rectify_fields**
Severity	ERROR
Context	rectifyData
Description	This rule ensures that data entered in *rectify* messages does not violate the GDPR data accuracy principle by entering empty fields.
Specification	`self.newData->forAll(f | f.value.size()>0)`
GDPR Rule 7	**no_empty_newData_fields**
Severity	ERROR
Context	newData
Description	This rule provides consistency with the GDPR data accuracy principle, in the same way as the previous rule, but for the *newData* messages.
Specification	`self.newData->forAll(f | f.value.size()>0)`
GDPR Rule 8	**no_empty_write_fields**
Severity	ERROR
Context	writeData
Description	This rule also provides consistency with the GDPR data accuracy principle, preventing empty fields written with any *writeData* messages.
Specification	`self.newContent->forAll(f | f.value.size()>0)`

Table 7: OCL rules derived from GDPR (III).

Attributes	**Value**
GDPR Rule 9	**newData_destinantion_machines_comply_with_GDPR**
Severity	ERROR
Context	newData
Description	As can be inferred by the name, this rule works similarly to the GDPR rule 1, although this time it checks the list of destinations of any *newData* message.
Specification	`self.machines->forAll(m | m.GDPRCompliance=true)`
GDPR Rule 10	**eraseData_destinantion_machines_comply_with_GDPR**
Severity	ERROR
Context	newData
Description	As can be inferred by the name, this rule works similarly to the GDPR rule 1, although this time it checks for compliance with the list of destinations of any *newData* message.
Specification	`self.machines->forAll(m | m.GDPRCompliance=true)`
GDPR Rule 11	**subscribe_destinantion_machines_comply_with_GDPR**
Severity	ERROR
Context	subscribe
Description	This rule checks for compliance in the list of destinations as well, this time, for the instances of any *subscribe* message.
Specification	`self.machines->forAll(m | m.GDPRCompliance=true)`
GDPR Rule 12	**notify_destinantion_machines_comply_with_GDPR**
Severity	ERROR
Context	notify
Description	Just like the previous two, this rule checks the list of destinations of any *notify* message, validating that all the machines are compliant with the GDPR.
Specification	`self.machines->forAll(m | m.GDPRCompliance=true)`
GDPR Rule 13	**consent_machine_complies_with_GDPR**
Severity	ERROR
Context	consent
Description	This rule is similar to the previous ones, in this case, *consent* messages have only one destination instead of a list, so it just checks that said machine has been evaluated as GDPR compliant.
Specification	`self.machines->forAll(m | m.GDPRCompliance=true)`

3 Warning Rules

Finally, this section concerns the rules defined as OCL warnings. These rules, rather than verifying model correctness, are aimed at informing the users of the MDCT tool that some values introduced are not oscillating in ranges considered standard. These rules can be found in Table 8.

Table 8: Table of warning OCL rules.

Attributes	**Value**
Warning Rule 1	**latency_not_in_us_or_ns**
Severity	WARNING
Context	Latency
Description	This rule warns users that the units for the delay of latency are larger orders of magnitude than what is considered standard.
Specification	`self.time.unit=TimeUnit::us or self.time.unit=TimeUnit::ns`
Warning Rule 2	**sendData_timeunit_not_days_or_hours_or_minutes**
Severity	WARNING
Context	sendData
Description	This rule informs the user if the periods of storage for data in a *sendData* message are shorter than usual.
Specification	`self.maxTime.unit=TimeUnit::days or self.maxTime.unit=TimeUnit::h or self.maxTime.unit=TimeUnit::min`
Warning Rule 3	**pasteData_timeunit_not_days_or_hours_or_minutes**
Severity	WARNING
Context	pasteData
Description	This warning has the same conditions and specifications as the previous rule, this time with messages of kind *pasteData*.
Specification	`self.maxTime.unit=TimeUnit::days or self.maxTime.unit=TimeUnit::h or self.maxTime.unit=TimeUnit::min`
Warning Rule 4	**combineData_timeunit_not_days_or_hours_or_minutes**
Severity	WARNING
Context	sendData
Description	This rule informs the user if the periods of storage for data in a *combineData* message are shorter than usual.
Specification	`self.maxTime.unit=TimeUnit::days or self.maxTime.unit=TimeUnit::h or self.maxTime.unit=TimeUnit::min`