[CGI] 15/06/2018. Adding comments from analysing mission PLIDs.

[CGI-1] The model allows constraints on Activities, but not on field values. The mission XML schemas express field value restrictions, such as limits on field length, range of values, allowed values (enumerations). This is data validation, is it part of model?

*RST: This is a very good point. While we have the means to express such a constraint, we currently do not have a specific constraint type that would allow this, other than the placeholder for a Complex Constraint based on a full expression language syntax. We can express this type of constraint on a Resource value, but not on other “field” values, such as the arguments of an Activity or Event. I would propose that we extend the model to support constraints on the value of arguments.*

[CGI-2] Planning Request: passing literals to Activities is convoluted. It uses *ArgumentConstructor (CGI edit: should be ActivityConstructor)* and ArgSpecs, which is deferred Argument initialisation. There is no shorthand for the simple case (passing Argument literals).

*RST: I don’t follow the issue here. In this case the ArgumentConstructor (CGI edit: should be ActivityConstructor) resolves to simply the Argument Name and its value as a literal (an expression can be just the literal value) – not sure how much more of a shorthand it could be.*

*CGI: Updated the comment: it was meant ActivityConstructor in place of ArgumentConstructor. In this case the ArgSpec inside the ActivityConstructor becomes Argument. OK.*

[CGI-3] Lists have a count attribute (number of elements in list) that is not expressed in MO model. Example: GAIA ParameterList in Occurrence. The count attribute may be used as a verification on expected number of elements in list.

*RST: We had previously discussed in the WG whether we wanted to support complex types, including Lists, for arguments, and had decided not to do so at this stage. This can be reconsidered if the WG wishes to extend it. I would note that support for lists/arrays is probably simpler than custom structures. See also CGI-13 and 15.*

[CGI-4] Lists/complex types not supported in Arguments. Example: GAIA ParameterList in Occurrence contains elements of type “Parameter”:

* Name (string)
* Position (positiveInteger)
* Description (string)
* Unit (string)
* Value (RelativeToEventTypeValue or CRFParameterValue, both are structures).

*RST: See previous response – you can represent the above example as multiple discrete arguments. If some of the information is static (Description, Unit) it would be straightforward to add these as optional elements of the Argument Definition.*

*CGI: Static items are part of the ArgumentDefinition, makes sense. Flattening a list into discrete elements is serialization, which requires extra knowledge. If there is a list of structures, then serialization would look something like this: write number of elements, first field of element1, 2nd field of element1, etc.*

[CGI-5] How to represent delta time? Example: GAIA Delta time: [-][DDD.]hh:mm:ss[.mmm].

*RST: A delta time is a Duration. Negative values are allowed.*

*CGI: Duration is not yet defined in model. It appears in DurationExpression. The deltaTime in use cases appears as an attribute. It would be convenient if Duration was a primitive data type.*

[CGI-6] GAIA, EXM, Euclid. Constraint type required to handle “**RelativeValidityEvents**”. List of fields:

* **baseEvent** (string) Mnemonic of the event that this time is based on.
* **baseCount** (nonNegativeInteger) Event count of the event the time is based on.
* **delta** (Time) Delta/Relative Time. [-][DDD.]hh:mm:ss[.mmm] format.
* **propagationFactor** (integer) Used to specify whether the occurrence's time is to be set with +/- a number of propagation delays. If no value is specified, 0 will be used as the default, i.e. no delays added.

*RST: I’m not familiar with the usage of the above, but can this not be formulated as a temporal expression of the form* Ta±Td\*/N *where Ta is a reference to the Event, Td is the Delta and N is the propagation factor? If this is the case then it is a simple time constraint.*

*I am unclear what the baseCount is – but it seems to be identifying the Event instance, rather than the Event definition. Possibly the implication it is the Nth occurrence of the Event – we cannot currently express this in what we have identified for an Object Expression.*

*CGI: baseEvent relates to the Event definition, and baseCount to the Nth occurence of the Event. Delta is a Duration and propagationFactor another Duration that is part of the TimeExpression. So the „RelativeValidityEvents“ is formulated as TimeExpression. The missing part is the Nth occurence.*

 [CGI-7] GAIA, EXM, Euclid. Constraint type required to handle a window of **RelativeValidityEvents** from above point.

*RST: needs further explanation – discuss. However, if you can express the RelativeValidityEvents as a time expression, then presumably the time window structure would support this.*

CGI: *The „RelativeValidityEvents“ is a datatype for „start“ and „end“, which are part of a request. (request has a header section with „start“, „end“ (ie validity periof of a request), etc, and a list of Activities.) My initial thought was that „start“ and „end“ go into Constraints. Now I think „start“ and „end“ go to Arguments, where they are TimeExpressions, as in CGI-6.*

[CGI-8] Constraints for Planning Requests required. PlanningRequest has Constraints, and Activity has Constraints, but Constraint allows only Activities, there is no PlanningRequest reference. Use case: validityRange in CRF headers.

*RST: needs further explanation – what is a CRF header? What aspect of a PlanningRequest do we need to express a constraint against? The only issue I can see we may have is a need to express constraints against the value of the arguments of a Planning Request (which goes together with CGI-1).*

*CGI: CRF is a Command Request File. The CRF file has a header section with „start“, „end“ (ie validity periof of a request), etc, and a list of Activities. Use case: the CRF file has a validity range. How to express that?*

[CGI-9] An insertOrDeleteFlag “Indicates whether the request is to insert or delete the occurrence.” Is cancelling/deleting an Activity handled by the information model or by interactions?

*RST: What is the context of your quoted text? – it is difficult for me to see to what you are referring. If you mean that the Request service interface may include operations to cancel a Request, then I would assume we will deal with this in the operations and associated messages of the service. As noted in the model, we have not yet addressed the specific operations to directly modify entries in a plan. These may also be dealt with as the operations and associated messages of the service. It is a matter for debate whether we need to extend the information model to support the representation of these operations. I am not convinced this is necessary unless we need to represent some additional information such as the fact that a request or activity has been cancelled/deleted in a subsequent Plan or status provision service/history.*

*The Revision Status element of a Plan already enables the identification of insertions/deletions/modifications of items with respect to a previous instance of the plan.*

*CGI: insertOrDeleteFlag identifies the occurrence which is to be added or deleted. In the use case this information is part of the data model. In MP is it going to be part of the data model (in submitPlanningRequest) or part of some other interaction?*

[CGI-10] Constraint type required to handle “**executionTime**” (GAIA). List of fields is summarised below. ExecutionTime is one of the following:

1. Based on time:
	1. **propagationFactor** (time) Used to specify whether the occurrence's time is to be set with +/- a number of propagation delays. If no value is specified, 0 will be used as the default, i.e. no delays added.
	2. **actionTime** (time) The time at which the occurrence is to be actioned.
2. Based on event:
	1. **propagationFactor** (time) Used to specify whether the occurrence's time is to be set with +/- a number of propagation delays. If no value is specified, 0 will be used as the default, i.e. no delays added.
	2. **eventID** This field permits the specification of an event mnemonic relative to which the operation is to be released
	3. **deltaTime** This field permits the specification of an offset time to the event specified in 'event ID' at which the occurrence should be scheduled. If not specified then the occurrence will be scheduled at exactly the time of the event
	4. **separation** (time) This field is used to specify the separation between repeated occurrences
	5. **repeat** (positive integer) minInclusive value="2"
	6. **eventCount** This field identifies a specific instance of the event specified in 'event ID'
	7. **eventCount2** This field is used in order to specify a range of event instances to be triggered

*RST: These seem to be specific representations of a Temporal Constraint. Can these not be reformulated and expressed using the constructs we have already identified for Temporal Constraints? If not, can you highlight the item that cannot currently be expressed? We want to avoid proliferating constraint types to suit a specific mission’s existing presentation of the information and use generic structures wherever possible.*

*CGI: 1) is a TimeExpression. 2) is a TimeExpression with repetitions, which is not covered by the model.*

[CGI-11] Constraint type required to handle “**releaseTime**” (GAIA). List of fields is summarised below:

* **propagationFactor** (time) Used to specify whether the occurrence's time is to be set with +/- a number of propagation delays. If no value is specified, 0 will be used as the default, i.e. no delays added.
* **earliestOffset** (delta time) The offset of the start of the window for the occurrence, which is only applied to the 'actionTime' or the 'event ID' if specified
* **latestOffset** (positive delta time) The offset of the end of the window for the occurrence, which is only applied to the 'actionTime' or the 'event ID' if specified
* value, one of the following:
	+ Based on time:
		- **actionTime** (time) The time at which the occurrence is to be actioned.
	+ Based on event:
		- **eventID** This field permits the specification of an event mnemonic relative to which the operation is to be released
		- **deltaTime** This field permits the specification of an offset time to the event specified in 'event ID' at which the occurrence should be scheduled. If not specified then the occurrence will be scheduled at exactly the time of the event
		- **separation** (time) This field is used to specify the separation between repeated occurrences
		- **repeat** (positive integer) minInclusive value="2"start
		- **eventCount** This field identifies a specific instance of the event specified in 'event ID'
		- **eventCount2** This field is used in order to specify a range of event instances to be triggered

*RST: See responses to previous and next comments.*

[CGI-12] How to express both executionTime and releaseTime in constraints? For example TimeConstraint specifies a TimeExpression for an ActivityRef, but it does not specify the semantics of the time constraint, which is assumed to be the execution time of the Activity.

*RST: This is a fair point. We really only currently support the concept of execution time. Do we want to extend the concept of an Activity to support the concept of release time – and then by implication extend the temporal constraint to support reference to either? Or do we want to keep a simpler structure to the model? Note that an alternative approach is to consider the Release and Execution as two separate activities (which they are, and may be executed by different entities within the mission system), that are linked through a common parent activity and associated temporal and sequential constraints. In my view this approach helps us simplify the structure of the model – and may better suit any subsequent automated execution of the plan. It is also more extensible to the case that there are more than two stages to execution of an activity.*

[CGI-13] BC: Cannot specify profile as part of PRQ (if that is a supported use case in the model). Sequence Occurrence specifies a list of Profiles. Profile is a structure:

* timeOffset ([-]hh:mm:ss)
* value (double)
* type (“X\_Data, “Ka\_Data, “X\_DV”, “KA\_DV”)

*RST: Not sure I have understood your point. Seems to be some overloading of the term “profile” – given that this has specific meaning in our model. If you are referring to a structure of the above combination of parameters, then is this not just a specific case of the more generic point in CGI-3?*

*CGI: A case of complex Arguments then.*

[CGI-14] Multiple errors not handled in Request Response. RequestUpdate allows specifying single error using ErrCode (Integer) and ErrInfo (String). GAIA allows multiple errors. First it says the number of errors, and for each error there is reference (line nr. in file) and string value.

*RST: Discuss. You can report multiple errors, if you can provide multiple responses – we have not yet defined the interactions in the service. Do we want to extend to allow for multiple errors?*

[CGI-15] There are no Arguments that support complex types in request response. Example: EnMAP AcquisitionRequestStatus. It is a structure with two complex elements:

* swathItem
* datatakeStatusEntry

*RST: Is this not the same point as CGI-3 and CGI-13? Either we extend to support complex types, or we do not. In my view it will add complexity, as to support custom structures, you will also need to support their definition. MAL requires us to break down the messages into fields of simple data types – so you would have to define the structure in terms of these anyway. Note that you can pass a complex structure as a Blob, if the planning process itself treats it as a black box.*

*CGI: This means the request response is the ActivityUpdate object, with the structures part of Arguments.*