Report of the
UCM 1.3 RTF Revision Task Force
to the
OMG Platform Technical Committee

17 May 2021

Document Number: ptc/21-05-01
Task Force Chair(s): Thomas Vergnaud, THALES
(thomas.vergnaud@thalesgroup.com)
Chartered: 21 June 2019 — Amsterdam, The Netherlands
Comments Due: 1 March 2021
Expiry Due: 28 June 2021
JIRA Project Prefix: UCM13

JIRA Task Force Home: https://issues.omg.org/browse/UCM13
Table of Contents

- Deliverables
  - Publication Directions
  - Specification
  - Supporting Documents
  - Machine-readable documents

- IPR Mode
- RTF Membership
- Disposition Summary
- Voting Record
- Summary of Changes Made

- Disposition: Resolved
  - OMG Issue UCM13-1
    Title: Named entities have no ID in the meta-model
  - OMG Issue UCM13-2
    Title: Attribute values of parts cannot be set in composite component implementation, while configuration parameter values of connections can
  - OMG Issue UCM13-3
    Title: Allow specific configuration for the ports and policies managed by a given technical policy
  - OMG Issue UCM13-5
    Title: problems in the XML schema and the DTD
  - OMG Issue UCM13-8
    Title: Inconsistency in number of bindings for technical policies
  - OMG Issue UCM13-10
    Title: rename “executor” into “context” in the container model
  - OMG Issue UCM13-11
    Title: define port types for “pull-pull” interactions
  - OMG Issue UCM13-13
    Title: broken link in the document
  - OMG Issue UCM13-16
    Title: Introduce structured configuration parameters
  - OMG Issue UCM13-17
    Title: Unexpected mention of “monolithic” component implementation
  - OMG Issue UCM13-21
    Title: Rephrase some explanations
  - OMG Issue UCM13-23
    Title: List XML tags names and clarify the rules for name references

- Disposition: Deferred
- Disposition: Transferred
- Disposition: Closed; No Change
  - OMG Issue UCM13-4
    Title: Add a method in the comp_trace policy to log messages with a default severity

- Disposition: Closed; Out Of Scope
- Disposition: Duplicate or Merged
Deliverables

Publication Directions

This section contains information about how OMG staff should publish the resulting specification on the OMG formal specification web page. The chosen acronym and version number determine the URL to be used; for instance, if the acronym chosen is "ABCD" and the version number is "1.2", the normative specification documents will be available under the URL https://www.omg.org/spec/ABCD/1.2/.

Documents in this section are classified as one of:

Normative: A document that specifies how a conforming implementation shall behave (but may contain specifically-labelled non-normative sections).

Informative: Non-normative material aimed at users of the specification, such as examples or non-normative guidelines.

Ancillary: Material intended to help OMG Members reviewing the report or creating future revisions of the specification, such as a change-tracked specification showing issue resolutions applied, editable source of models stored in proprietary formats, or programs used to build parts of the specification from machine-readable sources. The specification shall be usable without reference to the ancillary files, which should not be given URLs.

Specification

Chosen acronym: UCM
Version number: 1.3

<table>
<thead>
<tr>
<th>Title:</th>
<th>Revised specification (clean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc Number:</td>
<td>ptc/21-05-02</td>
</tr>
<tr>
<td>Status:</td>
<td>Normative</td>
</tr>
<tr>
<td>URL:</td>
<td><a href="https://www.omg.org/spec/UCM">https://www.omg.org/spec/UCM</a> /1.3</td>
</tr>
</tbody>
</table>

Title: UCM 1.3 convenience document with change bars
Doc Number: ptc/21-05-03
Status: Informative
## Supporting Documents

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Doc Number</th>
<th>Filename</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCM 1.3 ancillary attachment file (Zip File)</td>
<td></td>
<td>ptc/21-05-09</td>
<td>UCM13-1/base_classes.svg (zip entry)</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Issue UCM13-1 Attachment base_classes.svg</td>
<td></td>
<td>ptc/21-05-09</td>
<td>UCM13-1/base_classes.svg (zip entry)</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Issue UCM13-2 Attachment ucm_base.xsd</td>
<td></td>
<td>ptc/21-05-09</td>
<td>UCM13-2/ucm_base.xsd (zip entry)</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Issue UCM13-3 Attachment ucm_base.xsd</td>
<td></td>
<td>ptc/21-05-09</td>
<td>UCM13-3/ucm_base.xsd (zip entry)</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Issue UCM13-3 Attachment ucm_component_technical_policies.svg</td>
<td></td>
<td>ptc/21-05-09</td>
<td>UCM13-3/ucm_component_technical_policies.svg (zip entry)</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Issue UCM13-3 Attachment ucm_component_technical_policies.svg</td>
<td></td>
<td>ptc/21-05-09</td>
<td>UCM13-3/ucm_component_technical_policies.svg (zip entry)</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Doc Number:</td>
<td>ptc/21-05-09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filename:</td>
<td>UCM13-5/ucm_base.dtd (zip entry)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-5 Attachment ucm_base.xsd</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-5/ucm_base.xsd (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-10 Attachment UCMContainerStructure.svg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-10/UCMContainerStructure.svg (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-10 Attachment UCM_ComponentInterfaces.svg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-10/UCM_ComponentInterfaces.svg (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-11 Attachment ucm_ext_interac.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-11/ucm_ext_interac.xml (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-16 Attachment ucm_annotations.svg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-16/ucm_annotations.svg (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-16 Attachment ucm_base.dtd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-16/ucm_base.dtd (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-16 Attachment ucm_base.xsd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-16/ucm_base.xsd (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-16 Attachment ucm_configuration_parameters.svg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-16/ucm_configuration_parameters.svg (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
<tr>
<td>Title:</td>
<td>Issue UCM13-16 Attachment ucm_structured_parameters.svg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc Number:</th>
<th>ptc/21-05-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename:</td>
<td>UCM13-16/ucm_structured_parameters.svg (zip entry)</td>
</tr>
<tr>
<td>Status:</td>
<td>Ancillary</td>
</tr>
</tbody>
</table>
Title: Issue UCM13-17 Attachment policy_fragmentation.svg
Doc Number: ptc/21-05-09
Filename: UCM13-17/policy_fragmentation.svg (zip entry)
Status: Ancillary

**Machine Readable documents**

This section contains one entry for each machine-readable file accompanying the submission.

Authors should assign a version stamp to each new normative or informative machine-readable file that they create. Version stamps comprise 8 digits, and each newly-assigned version stamp for a particular specification must be greater than any previous version stamp for the same specification. One convenient way to achieve this is to concatenate the current year & two-digit month number with a two-digit sequence number within the month. Hence the sixth version created during May 2013 would have version stamp 20130506. Multiple files with different names should share the same version stamp where appropriate.

If many files are supplied, it may be convenient to package them in a single ZIP archive with a single OMG document number; however, each separate file should nevertheless still be given its own entry in this section.

For each machine-readable file, list any other files on which it depends (e.g. by inclusion), and whether it's normative, informative or ancillary (see above).

Each normative or informative machine-readable file is given a URL that allows it to be unambiguously referenced on the OMG formal specification web site. Ancillary files do not have URLs - they are not usually placed on the formal specification page.

**Description:** UCM 1.3 core file for the IDL mapping
Doc Number: ptc/21-05-05
Status: Normative
URL: https://www.omg.org/spec/UCM/20210619/ucm.idl

**Description:** UCM 1.3 reference implementation of the meta-model
Doc Number: ptc/21-05-07
Status: Ancillary

**Description:** UCM 1.3 meta-model XMI file
Doc Number: ptc/21-05-08
Status: Normative
URL: https://www.omg.org/spec/UCM/20210619/ucm_base.xmi

**Description:** UCM 1.3 XML schema and DTD (Zip File)
Doc Number: ptc/21-05-04
Status: Normative

**Description:** DTD
Doc Number: ptc/21-05-04
Filename: ucm_base.dtd (zip entry)
Status: Normative
URL: https://www.omg.org/spec/UCM/20210619/ucm_base.dtd

**Description:** XML schema
<table>
<thead>
<tr>
<th>Doc Number</th>
<th>Filename</th>
<th>Status</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> UCM 1.3 XML files for the standard libraries (Zip File)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doc Number</td>
<td>ptc/21-05-06</td>
<td>Normative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ucm_core.xml (zip entry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>standard core library</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> standard core library</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doc Number</td>
<td>ptc/21-05-06</td>
<td>Normative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ucm_ext_exec.xml (zip entry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>additional policies</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> additional policies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doc Number</td>
<td>ptc/21-05-06</td>
<td>Normative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ucm_ext_interac.xml (zip entry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>additional interactions</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> additional interactions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doc Number</td>
<td>ptc/21-05-06</td>
<td>Normative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ucm_std_ann.xml (zip entry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>standard properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> standard properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doc Number</td>
<td>ptc/21-05-06</td>
<td>Normative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ucm_timer.xml (zip entry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>advanced timer services</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> advanced timer services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doc Number</td>
<td>ptc/21-05-06</td>
<td>Normative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ucm_lang--cpp.xml (zip entry)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>programming language declarations</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> programming language declarations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Under OMG’s IPR policy, every specification published after April 2013 must include a declaration of the terms under which anyone who contributed text to the specification must agree to license any patents that they control, and which they claim are essential to implementing the specification. This is known as the "IPR Mode" of the specification. It must be chosen from the list specified in the IPR policy, and once OMG’s Board has selected the IPR Mode for a specification, it cannot subsequently be changed for any later RTF or FTF for that specification.

For RTFs or FTFs chartered before April 2013, a Legacy IPR mode must be selected for the specification being revised/finalised using the procedure in ipr/12-11-01 section 3.3.3, and listed below.

For RTFs or FTFs chartered after April 2013 for a specification without an IPR mode, the charter includes a Legacy IPR mode selected by the same procedure - that mode is listed below.

For a full, definitive statement of OMG's IPR policy, see: http://doc.omg.org/ipr.

**IPR mode of base specification: Non-Assert**
<table>
<thead>
<tr>
<th>Member</th>
<th>Organization</th>
<th>Role</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manfred Koethe</td>
<td>88solutions</td>
<td>Voting Member</td>
<td>Charter</td>
</tr>
<tr>
<td>Gerardo Pardo-Castellote</td>
<td>Real-Time Innovations</td>
<td>Voting Member</td>
<td>Charter</td>
</tr>
<tr>
<td>Julio Medina</td>
<td>Universidad de Cantabria</td>
<td>Voting Member</td>
<td>Charter</td>
</tr>
<tr>
<td>Ansgar Radermacher</td>
<td>Commissariat a l Energie Atomique-CEA</td>
<td>Voting Member</td>
<td>Charter</td>
</tr>
<tr>
<td>Mark Hayman</td>
<td>Northrop Grumman</td>
<td>Voting Member</td>
<td>Charter</td>
</tr>
<tr>
<td>Charlotte Wales</td>
<td>MITRE</td>
<td>Voting Member</td>
<td>Removed Apr 1, 2021</td>
</tr>
<tr>
<td>Thomas Vergnaud</td>
<td>THALES</td>
<td>Chair</td>
<td>Charter</td>
</tr>
</tbody>
</table>
## Disposition Summary

<table>
<thead>
<tr>
<th>Disposition</th>
<th>Number Of Issues</th>
<th>Meaning Of Disposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolved</td>
<td>12</td>
<td>The RTF/FTF agreed that there is a problem that needs fixing, and has proposed a resolution (which may or may not agree with any resolution the issue submitter proposed)</td>
</tr>
<tr>
<td>Deferred</td>
<td>0</td>
<td>The RTF/FTF agrees that there is a problem that needs fixing, but did not agree on a resolution and deferred its resolution to a future RTF/FTF.</td>
</tr>
<tr>
<td>Transferred</td>
<td>0</td>
<td>The RTF/FTF agrees to move the issue to another RTF/FTF.</td>
</tr>
<tr>
<td>Closed; No Change</td>
<td>1</td>
<td>The RTF/FTF decided that the issue report does not, in fact, identify a problem with this (or any other) OMG specification.</td>
</tr>
<tr>
<td>Closed; Out Of Scope</td>
<td>0</td>
<td>The RTF/FTF decided that the issue report is an enhancement request, and therefore out of scope for this or any future FTF or RTF working on this major version of the specification. The RTF/FTF has closed the issue without making any specification changes, but RFP or RFC submission teams may like to consider these enhancement requests when proposing future new major versions of the specification.</td>
</tr>
<tr>
<td>Duplicate or Merged</td>
<td>0</td>
<td>This issue is either an exact duplicate of another issue, or very closely related to another issue: see that issue for disposition.</td>
</tr>
</tbody>
</table>
Note: Issue numbers shown in the tables below should be prefixed with the JIRA Project Prefix (UCM13-) to form fully qualified issue numbers.

<table>
<thead>
<tr>
<th>Ballot No.</th>
<th>Closing Date</th>
<th>Number of Issues</th>
<th>Issues Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14 April 2021</td>
<td>13</td>
<td>1, 2, 3, 4, 5, 8, 10, 11, 13, 16, 17, 21, 23</td>
</tr>
</tbody>
</table>

The following tables detail the vote for each member in each ballot. The vote is described as follows:

- **Y** Yes - Positive Vote
- **N** No - Negative Vote
- **A** Abstain
- **-** Member Did Not Vote

**Voting Record for Ballot No. 1**

<table>
<thead>
<tr>
<th>Voter</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>11</th>
<th>13</th>
<th>16</th>
<th>17</th>
<th>21</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manfred Koethe</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ansgar Radermacher</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Charlotte Wales</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mark Hayman</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Gerardo Pardo-Castellote</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Thomas Vergnaud</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Julio Medina</td>
<td>Y</td>
<td>Y</td>
<td>A</td>
<td>Y</td>
<td>A</td>
<td>Y</td>
<td>A</td>
<td>Y</td>
<td>Y</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>
Summary of Changes Made

The FTF made changes that:

- Corrected errors in the specification
- Increased the clarity of the specification
- Corrected features in order to improve implementability and adoption of the standard

Here is the FTF's categorization of the resolutions applied to the specification according to their impact on the clarity and precision of the specification:

Note: Issue numbers shown in the tables below should be prefixed with the JIRA Project Prefix (UCM13-) to form fully qualified issue numbers.

<table>
<thead>
<tr>
<th>Extent of Change</th>
<th>Number of Issues</th>
<th>Issues Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical/Urgent — Fixed problems with normative parts of the specification which prevented implementation work.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Significant — Fixed problems with normative parts of the specification that raised concern about implementability.</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Minor — Fixed minor problems with normative parts of the specification.</td>
<td>6</td>
<td>1, 2, 3, 5, 11, 16</td>
</tr>
<tr>
<td>Support Text — Changes to descriptive, explanatory, or supporting material.</td>
<td>6</td>
<td>8, 10, 13, 17, 21, 23</td>
</tr>
</tbody>
</table>
Disposition: Resolved

OMG Issue: UCM13-1

Title:
Named entities have no ID in the meta-model

Summary:
The UCM DTD and XML schema specify an ID field for each named entity. Yet, such ID do not exist in the INamed metaclass of the ucm_base metamodel. Adding IDs in INamed entities would ease the importation of XML files into existing UCM models by enabling model mergers.

Source:
ptc-19-04-03 — Chapter/Section: 9.1.4.1 — Page Number/s: 16

Resolution Summary:
Add an ID field in class INamed
add a field “id” to class INamed

Revised Text:
In section 9.1.4 (Common meta-model definitions), replace the diagram with file base_classes.svg.
In section 9.1.4.1 (INamed), insert the following item between item “name” and item “comment”:
- id: String [0…1] (owned)

Also add the following text at the end of section 9.1.4.1:
The name of each INamed entity shall be unique within its name scope. The name scope of an INamed entity is the nearest parent of type INamed.

Attribute id is optional. If set, it shall be unique in the whole model.

Extent Of Change:
Minor

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Thu, 5 Dec 2019 10:37 GMT on UCM 1.2

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
meta-model

Discussion:
UCM13-1 - https://issues.omg.org/browse/UCM13-1

Attachments:
- base_classes.svg - ptc-21-05-09/UCM13-1/base_classes.svg 27 kB (image/svg+xml)
OMG Issue: UCM13-2

Title:

Attribute values of parts cannot be set in composite component implementation, while configuration parameter values of connections can

Summary:

In CompositeComponentImplementation, the ucm_base meta-model allows the specification of configuration parameter values in Connection and ConnectionEnd, but it does not allow the specification of attribute values in AssemblyPart. The meta-model would be more consistent if it supported the specification of attribute values in AssemblyPart.

For example, if we have the following component declaration:

```
<componentModule name="comp">
  <compType name="C1">
    <attribute mode="read" name="attr1" type="::dataTypes::Int32_t"/>
  </compType>

  <atomic name="C1_i1" lang="::ucm_lang::cpp::CPP11_typed" type="C1"/>
</componentModule>
```

We should be able to write something like this:

```
<componentModule name="comp2">
  <compType name="C2"/>

  <composite name="C2_i1" type="C2">
    <part name="c1a" ref="::comp::C1_i1">
      <attrVal ref="attr1" value="23"/>
    </part>
    <part name="c1b" ref="::comp::C1_i1">
      <attrVal ref="attr1" value="12"/>
    </part>
  </composite>
</componentModule>
```

Attribute attr1 shall be initialized to 23 for part c1a and to 12 for c1b.

Source:

ptc-19-04-03 — Chapter/Section: 9.5.6.3 — Page Number/s: 49

Resolution Summary:

Set attribute values in assembly parts

Add a new metaclass AttributeValue that allows the specification of a value for an attribute. This metaclass is referenced by metaclass AssemblyPart.

Also update the DTD and XML schema.

Revised Text:

In section 9.5.6 (composite component implementations), replace diagram 32 (UCM Composite component implementations) with file
In section 9.5.6.3 (AssemblyPart), add the following item to the list:

- attributeValue: AttributeValue [0…*] (owned)

Also insert the following text at the end of section 9.5.6.3:

The component definition referenced by an assembly part may define attributes. The assembly part can specify values for these attributes. The values specified by the assembly part override the previously defined values of the considered attributes. In particular, if the assembly part references a composite component definition and if the assembly part specifies a value for an attribute that is delegated, then the specified value overrides the previously specified values of the attributes pointed by the delegation.

Create a new section 9.5.6.8 with title “AttributeValue (Valued)”. The section contains the following text:

Attribute value specifies a value for a given attribute.

- attribute: Attribute [1]

Extent Of Change:

Minor

Disposition:

Resolved - Approved on Ballot #1

Reporter:

Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:

Thu, 5 Dec 2019 12:26 GMT on UCM 1.2

Updated:

Thu, 15 Apr 2021 00:45 GMT

Components:

meta-model

Discussion:


Attachments:

- ucm_base.xsd - ptc-21-05-09/UCM13-2/ucm_base.xsd 40 kB (text/xml)
- ucm_composite_component_implementations.svg - ptc-21-05-09/UCM13-2/ucm_composite_component_implementations.svg 55 kB (image/svg+xml)
OMG Issue: UCM13-3

Title: Allow specific configuration for the ports and policies managed by a given technical policy

Summary:
Technical policies can have configuration parameters, and can manage component features (i.e. they can intercept them). However, it is not possible to associate a specific configuration with a given component feature.

Current state
Technical policy definitions can specify configuration parameters. In the following example, technical policy definition watchdog specifies a configuration parameter for the timeout. It also declares an interface named watchdog_intf that must be provided by components. The policy calls method timeout if no activity has been detected on the intercepted component features for some time.

```xml
<policyModule name="example">
  <contractModule name="api">
    <interface name="watchdog_intf">
      <method name="timeout"/>
    </interface>
  </contractModule>
  <technicalAspect constraint="any_number" name="asp"/>
  <policyDef applicability="on_some_ports" aspect="asp" name="watchdog">
    <comment>This policy intercepts some component ports and policies. It invokes method timeout if no activity is detected on one of these ports and policies after max_delay. </comment>
    <portElement interface="api::watchdog_intf" kind="provided" name="wdg"/>
    <configParam name="max_delay" type="::ucm_ext_exec::contracts::ucm_duration_t"/>
  </policyDef>
</policyModule>
```

One can set the value max_delay when declaring such a policy. In the following, we declare an atomic component named calculation_impl that has two input ports input1 and input2. It is associated with technical policy watcher that monitors both ports. Policy watcher triggers an alert after 2 seconds of inactivity on both ports.

```xml
<componentModule name="comp">
  <contractModule name="data_types">
    <integer name="int32_t" kind="int32"/>
  </contractModule>
  <bindingSet name="bindings">
    <binding abstract="::ucm_core::messages::api::transm_data_t" actual="data_types::int32_t"/>
  </bindingSet>
  <compType name="calculation">
    <port name="input1" type="::ucm_core::message::msg_recv_pt" bindings="bindings"/>
    <port name="input2" type="::ucm_core::message::msg_recv_pt" bindings="bindings"/>
  </compType>
</componentModule>
```
Limitation

The problem is, it is impossible to set a specific timeout value for each intercepted port. Thus, one cannot specify that policy watcher shall trigger an alert after 2 seconds of inactivity on port input1, or 3 seconds of inactivity on port input2.

Having two separate policies (one for each intercepted port) is not convenient because the component business code would then have to implement two different methods `timeout` (one for each policy).

What should be added

Technical policy definition should be extended to declare configuration parameters that apply to particular managed component features. Thus, the declaration of watchdog may be something like this:

```
<policyDef applicability="on_some_ports" aspect="asp" name="watchdog">
  <comment>This policy intercepts some component ports and policies. It invokes method timeout if no activity is detected on one of these ports and policies after max_delay. </comment>
  <portElement interface="api::watchdog_intf" kind="provided" name="wdg"/>
  <configParam name="default_max_delay" type=":ucm_ext_exec::contracts::ucm_duration_t"/>
  <specificConf name="specific_delays">
    <configParam name="max_delay" type=":ucm_ext_exec::contracts::ucm_duration_t"/>
  </specificConf>
</policyDef>
```

An additional section named `featureConf` allows the declaration of per-feature configuration parameters. Technical policy watcher may then be declared as follows:

```
<policy name="watcher" def=":example::watchdog">
  <componentRef ref="calculation_impl"/>
  <managedPort ref="input1"/>
  <managedPort ref="input2"/>
  <configValue name="default_max_delay" value="[2, s]"/>
  <featureConf name="input2_config">
    <managedPort ref="input2"/>
    <configValue name="max_delay" value="[3, s]"/>
  </featureConf>
</policy>
```

Here, the policy triggers an alert after 2 seconds of inactivity on input1
Allow the specification of configuration parameters that are associated to particular ports or policies

Create two new meta-classes:

- SpecificPolicyParameter enables the declaration of configuration parameters that may be associated to some managed ports or policies
- SpecificPolicyConfigurationValue actually associates configuration parameter values to managed ports or policies

Provide an example for a better understanding.

Update the XML schema and DTD with the new syntactic elements.

Revised Text:

In section 9.4.2 (Nonfunctional aspect module), replace the diagram of figure 25 (Main classes of UCM technical policies package) with file ucm_technical_policies.svg.

In section 9.4.3.2 (TechnicalPolicyDefinition), add an item to the list:

- specificParam: SpecificPolicyParameter [0…*] (owned)

In the same section, replace the paragraph “A technical policy definition may have configuration...” with the following text:

A technical policy definition may have configuration parameters to specify nonfunctional settings (e.g. execution period). These configuration parameters apply to the technical policy itself. A technical policy may also define configuration parameters that are specific to some of the ports or policies it applies to. Configuration parameters of specific policy parameters shall be unique in the scope of the technical policy definition.

Create a new section 9.4.3.3 with title “SpecificPolicyParameter (IConfigurable)”. Put the following text in the new section:

A specific policy parameter enables the declaration of configuration parameter that may apply to some of the ports or policies that are managed by a technical policy. The concept is similar to the connector port configuration (see section 9.3.4.5).

In section 9.5.5 (Atomic component implementations and technical policies), replace the diagram of figure 31 (UCM technical policies) with file ucm_component_technical_policies.svg.

In section 9.5.5.2 (TechnicalPolicy), add the following item to the list:

- specificConfigurationValue: SpecificPolicyConfigurationValue [0…*] (owned)

Add a new section 9.5.5.3 with title “SpecificPolicyConfigurationValue (IConfigured)”. 

(default policy configuration) or after 3 seconds of inactivity on input2 (specific configuration).

Source:
ptc-19-04-03 — Chapter/Section: 9.4, 9.5 — Page Number/s: 40, 47
Resolution Summary:
Put the following text in the new section:

Specific policy configuration values are used to associate configuration values to some managed ports or policies. It is thus possible to have a technical policy configuration that manages ports or policies according to different parameter values. See section 17.4.4 for an example.

- managedPort: Port [0…*]
- managedPolicy: TechnicalPolicy [0…*]

The ports and policies managed by a specific policy configuration value shall be managed by the technical policy that contains the specific policy configuration value.

A technical policy can have several specific policy configuration values. The whole set of the configuration parameter values of all the specific policy configuration values shall match the whole set of the configuration parameters defined in the specific policy parameters that are contained in the definition of the technical policy.

Make “section 17.4.4” an hyperlink to section 17.4.4. Instructions to create section 17.4.4 are given below.

Add a new section 17.3.3 with title “Technical policy definition with specific configuration”. Put the following text in the new section:

```xml
<policyModule name="fdir">
  <comment>fault detection, isolation and recovery</comment>
  <contractModule name="contracts">
    <integer kind="uint8" name="group_t"/>

    <enum indexType="uint32" name="importance_t">
      <value index="0" name="MINOR"/>
      <value index="1" name="MAJOR"/>
      <value index="3" name="CRITICAL"/>
    </enum>

    <interface name="alrt_intf">
      <method name="alert">
        <param dir="in" name="importance" type="importance_t"/>
        <param dir="in" name="group" type="group_t"/>
      </method>
    </interface>

    <interface name="ctrl_intf">
      <method name="reset"/>
    </interface>
  </contractModule>
</policyModule>

<technicalAspect constraint="any_number" name="fdir_asp"/>

<policyDef applicability="on_all_ports" aspect="fdir_asp" name="fault_detection">
  <comment>Monitors activity on all ports and policies, and reports an alert in case of inactivity. The default timeout is infinite.</comment>
  <portElement interface="contracts::alrt_intf" kind="provided" name="fault"/>
  <portElement interface="contracts::ctrl_intf" kind="required" name="control"/>
  <configParam max="1" min="1" name="default_importance" type="contracts::importance_t"/>
  <configParam max="1" min="1" name="default_group" type="ucm_ext_exec::contracts::group_t"/>
  <specificParam>
    <configParam max="1" min="1" name="timeout" type="ucm_ext_exec::contracts::ucm_duration_t"/>
    <configParam max="1" min="1" name="group" type="ucm_ext_exec::contracts::group_t"/>
  </specificParam>
</policyDef>
```
Technical policy definition fault_detection manages all the ports and policies. If there is no activity (i.e. no method invocation) on a given port or policy for a too long time, the policy shall call method “alert”.

The technical policy definition requires two mandatory configuration parameters to set the default importance and group of the alerts. Specific importance and groups can also be set for particular managed ports or policies. This is optional, but if one chooses to set a particular group, he or she shall also set a timeout. One can set specific importance separately.

Add a new section 17.4.4 with title “Technical policy with specific configuration”. Put the following text in the new section:

Atomic component implementation “Detector_atomic_fdir” is another implementation of component type “Detector”. It involves two periodic processings driven by policies “trigger1” and “trigger2”. It is monitored by a fault detection policy “fdir” that raises an alert in case of inactivity for a too long time on its ports and policies.
Policy “fdir” is configured as follows:

- if policy “trigger1” is late of more than 3 seconds, then “fdir” calls method “alert” with importance MINOR and group 1;
- if policy “trigger2” is late of more than 29 seconds (because the component execution takes too much time in processing), then “fdir” calls method “alert” with importance MAJOR and group 2;
- if port “detector_in” is late of more than 29 seconds (because the component execution takes too much time in processing), then “fdir” calls method “alert” with importance MINOR and group 1;
- port “detector_out” and policy “fdir” are not monitored because the default timeout is infinite; yet, they are associated with group 0 and importance MINOR.

**Extent Of Change:**
Minor

**Disposition:**
Resolved - Approved on Ballot #1

**Reporter:**
Thomas Vergnaud, THALES (thomas.vergnau@thalesgroup.com)

**Reported:**
Tue, 7 Jan 2020 15:17 GMT on UCM 1.2

**Updated:**
Thu, 15 Apr 2021 00:45 GMT

**Components:**
- meta-model
- syntax

**Discussion:**
UCM13-3 - https://issues.omg.org/browse/UCM13-3

**Attachments:**
- ucm_base.xsd - ptc-21-05-09/UCM13-3/ucm_base.xsd 41 kB (text/xml)
- ucm_component_technical_policies.svg - ptc-21-05-09/UCM13-3/ucm_component_technical_policies.svg 26 kB (image/svg+xml)
- ucm_technical_policies.svg - ptc-21-05-09/UCM13-3/ucm_technical_policies.svg 42 kB (image/svg+xml)
OMG Issue: UCM13-5

Title:
problems in the XML schema and the DTD

Summary:

In the DTD, tag "comment" is missing in the declaration of element componentModule, line 283.

In the XML schema, elements "componentModule" and "contractModule" are inverted with respect to the DTD, lines 672 and 673.

Source:
ptc/19-04-06

Resolution Summary:

Fix the problems in the DTD and the XML schema

Fix the problems

Revised Text:

No change in the document.

Extent Of Change:
Minor

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Tue, 3 Mar 2020 13:49 GMT on UCM 1.2b1

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
syntax

Discussion:

Attachments:

- ucm_base.xsd - ptc-21-05-09/UCM13-5/ucm_base.xsd 44 kB (text/xml)
Title:
Inconsistency in number of bindings for technical policies

Summary:
Field “bindings” of technical policies is specified with cardinality [0…*] in the text, while it is displayed with cardinality [0…1] in diagram 31.

Source:
formal/20-04-02 — Chapter/Section: 9.5.5.2 — Page Number/s: 47

Resolution Summary:
change cardinality of field “bindings” of class TechnicalPolicy
The correct cardinality is [0…1]. Change the text accordingly.

Revised Text:
In section 9.5.5.2 (TechnicalPolicy), replace line

bindings: IBindingSet [0…*]

by

bindings: IBindingSet [0…1]

That is, replace the “*” with “1”.

Extent Of Change:
Support Text

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Mon, 7 Sep 2020 10:08 GMT on UCM 1.2

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
text

Discussion:
UCM13-8 - https://issues.omg.org/browse/UCM13-8
Title: rename “executor” into “context” in the container model

Summary:
The UCM container model specifies that a PortElementObject uses the name “executor” to refer to the ComponentObject it belongs to. The name “context” may be a better choice for this. Indeed, the methods of a PortElementObject use the ComponentObject to retrieve the references to the other PortElementObjects of the component. So, the ComponentObject can be seen as a context for the execution of PortElementObject methods. Using “context” would make UCM more consistent with CCM.

Source:
formal/20-04-02 — Chapter/Section: 14.2 — Page Number/s: 83-84

Resolution Summary:
replace “executor” by “context”

In section 13, replace “executor” by “body” (1 replacement).
In the rest of the document, replace all occurrences of “executor” by “context” (4 replacements).
Update two diagrams that describe the UCM container model.

Revised Text:

In section 13.3.2 (Index-based timers), at the end of the section, replace “is provided by the component executor” with “is provided by the component body”.

In section 14.2 (Container model), replace the content of diagram 37 (UCM container model) with file UCMContainerStructure.svg.

In section 14.2.1 (Component interfaces), replace the content of diagram 38 (UCM component object interfaces) with file UCM_ComponentInterfaces.svg.

In section 16.7.1 (Generic mapping), in the C++ code block, replace

```cpp
Filter_in_recv_pe (ComponentObjectPtr executor);
```

with

```cpp
Filter_in_recv_pe (ComponentObjectPtr context);
```

and replace

```cpp
clock_clk(FragmentObjectPtr executor);
```

with

```cpp
clock_clk (FragmentObjectPtr context);
```
In section 16.7.2 (Strongly typed mapping), in the C++ code block, replace

```cpp
Filter_in_rcvr_pe (Filter_Impl_body *executor);
```

with

```cpp
Filter_in_rcvr_pe (Filter_Impl_body *context);
```

and replace

```cpp
clock_clk(clock *executor);
```

with

```cpp
clock_clk(clock *context);
```

**Extent Of Change:**
- **Support Text**

**Disposition:**
- Resolved - Approved on **Ballot #1**

**Reporter:**
- Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

**Reported:**
- Wed, 23 Sep 2020 16:34 GMT on **UCM 1.2**

**Updated:**
- Thu, 15 Apr 2021 00:45 GMT

**Components:**
- text

**Discussion:**
- UCM13-10 - https://issues.omg.org/browse/UCM13-10

**Attachments:**
- [UCMContainerStructure.svg](https://issues.omg.org/browse/UCM13-10) 21 kB (image/svg+xml)
- [UCM_ComponentInterfaces.svg](https://issues.omg.org/browse/UCM13-10) 16 kB (image/svg+xml)
Title: define port types for “pull-pull” interactions

Summary:

The UCM core library defines the message connector in which the data producer pushes data to the connector, and the connector pushes data to the consumer. This more or less corresponds to the event interactions of CCM.

Yet, in some situations, pulling data from the producer is a good architectural choice. For example, if we want to ensure a piece of data is sent periodically, it is best to let the connector manage this, instead of the component: even if there are bugs in the component business code, that connector shall periodically fetch the data.

It might be difficult to standardize a “pull-pull” connector because there could be many different situations: periodic fetching, fetching upon request from the consumer, etc. The UCM standard should simply define port types in order to have standard APIs.

Source:
formal/20-04-02 — Chapter/Section: 13.4 — Page Number/s: 70

Resolution Summary:

define a data pull connector

Define a data pull connector in the extended interactions. This connector involves one supplier and one or more consumers. Whenever a consumer pulls a piece of data, the piece of data is fetched from the supplier.

Revised Text:

Create a new section 13.4.3 named “data pull”, with the following content:

The data pull interaction is meant to be used for data transmission between one data supplier and several data consumers. Unlike the message interaction (section 13.1.7), data is fetched from the producer upon request from a consumer. Thus, in the data pull interaction, the communication is driven by the consumers; in the message interaction, the communication is driven by the producer.

13.4.3.1 Specifications

The data pull interaction is defined in an interaction module.

One interface is defined; it is used on both sides (supplier and consumer). On one side, the supplier provides the interface to the connector, while on the other side the consumer uses the same interface, provided by the connector. The data pull interaction involves exactly one supplier and one or more consumers.

```
<interactionModule name="data_pull">
  <contractModule name="api">
    <interface name="data_pull_intf">
      <method name="pull">
        <param dir="out" name="data" type="::ucm_core::messages::api::transm_data_t"/>
        <param dir="return" name="ecode" type="::ucm_core::return_codes::comm_ecode"/>
      </method>
    </interface>
  </contractModule>
</interactionModule>
```
13.4.3.2 Semantics

On the consumer side, the port type provides a unique method `pull()` to the component. Method `pull()` is called from the component business code to get data. Method `pull()` has an out parameter named `data`, and a return type that indicates whether the method call was successful or not.

On the supplier side, the component must implement the same method `pull()` required by the port type. The supplier component stores the value to supply into parameter `data`. The return code is meant to indicate whether the supplied data is valid or not.

Upon the invocation of method `pull()` on the consumer side, the data pull connector shall call method `pull()` on the supplier side. Every call to `pull()` on the consumer side shall call `pull()` on the supplier side. If a connection involves several consumers, the piece of data pulled from the supplier is only given to the one consumer that invoked `pull()`.

Depending on the connector implementation, data may be transferred from the supplier to the consumer using data copy or direct memory access.

13.4.3.3 Equivalent IDL syntax

The equivalent IDL declarations are the following.

```idl
module data_pull<typename TRANSM_DATA_T>
{
    module api
    {
        interface data_pull_intf
        {
            ::ucm_core::return_codes::comm_ecode pull (out TRANSM_DATA_T data);
        } // end of module api
    }

    porttype data_suppl_pt
    {
        provides api::data_pull_intf suppl_pe;
    }

    porttype data_cons_pt
    {
```
uses api::data_pull_intf cons_pr;
};

collector data_pull_cnt
{
    mirrorport data_suppl_pt supplier;
    mirrorport data_cons_pt consumer;
};

};/end of module data_pull

Extent Of Change:
   Minor

Disposition:
   Resolved - Approved on Ballot #1

Reporter:
   Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
   Wed, 23 Sep 2020 16:54 GMT on UCM 1.2

Updated:
   Thu, 15 Apr 2021 00:45 GMT

Components:
   library

Discussion:
   UCM13-11 - https://issues.omg.org/browse/UCM13-11

Attachments:
   - ucm_ext_interac.xml - ptc-21-05-09/UCM13-11/ucm_ext_interac.xml 8 kB (application/xml)
Title: broken link in the document

Summary:

The content of section 17.4 “Components” is “The examples of this section correspond to the example illustrated in section Error: Reference source not found”. This is obviously a broken link.

The link should be updated to point to section 11.3 “Example”.

Resolution Summary:

fix the broken link

Update the link to section 11.3 (Example)

Revised Text:

In section 17.4 (Components), update the link and make it point to section 11.3 (Example).

Extent Of Change:
Support Text

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Tue, 29 Sep 2020 11:57 GMT on UCM 1.2

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
text

Discussion:
Title: Introduce structured configuration parameters

Summary:

It is currently not possible to gather several configuration parameters together.

For example, let us suppose we want to design a message connector that handles channels with different FIFO priorities.

Such a connector would allow the specification of a channel number for each emission port.

It would also allow the specification of a FIFO priority for each channel number in each reception port.

See the following example of what would be needed:

```xml
<interactionModule name="enhanced_fifo_connector">
  <contractModule name="data_types">
    <integer kind="int32" name="channel_id_t"/>
    <integer kind="uint8" name="fifo_priority_t"/>
  </contractModule>
  <connectorDef name="enh_fifo_msg_cnt" pattern="::ucm_core::messages::msg_intrPat">
    <extends ref="::ucm_core::messages::simple_msg_cnt"/>
    <portConf port="emitter">
      <configParam max="1" min="1" name="channel_id" type="data_types::channel_id_t"/>
    </portConf>
    <portConf port="receiver">
      <structParam max="-1" min="0" name="fifo_priorities">
        <comment>the default fifo priority is 0</comment>
        <configParam max="-1" min="1" name="emitter_channel_id" type="data_types::channel_id_t"/>
        <configParam max="1" min="1" name="fifo_priority" type="data_types::fifo_priority_t"/>
      </structParam>
    </portConf>
  </connectorDef>
</interactionModule>
```

Sets made of several channels and one priority can be specified for each port “receiver”.

Such a connector could be used as follows:

```xml
<componentModule name="components">
  <contractModule name="contracts">
    <integer kind="int16" name="payload_t"/>
  </contractModule>
  <bindingSet name="payload_binding">
    <binding abstract="::ucm_core::messages::api::transm_data_t" actual="contracts::payload_t"/>
  </bindingSet>
  <compType name="C_sender">
    <port bindings="payload_binding" name="snd" type="::ucm_core::messages::msg_emtr_pt"/>
  </compType>
  <compType name="C_receiver">
    <port bindings="payload_binding" name="rcv" type="::ucm_core::messages::msg_rcvr_pt"/>
  </compType>
  <compType name="C_appli"/>
</componentModule>
```
Composite component 'C_appli' contains 5 subcomponents: c1, c2 and c3 send messages to c4 and c5. Subcomponent c1 sends on channel 1, c2 sends on channel 2, c3 sends on channel 3.

Subcomponents c4 and c5 receive all messages, with different priorities. Thus, c4 receives data from c1 (channel 1) with priority 2, data from c2 and c3 (channels 2 and 3) with priority 9.
And c5 receives data from c1 (channel 1) with priority 5, data from c2 (channel 2) with priority 2, data from c3 (channels 3) with priority 1.

Source: formal/20-04-02
Resolution Summary:
create structured configuration parameters
Add two new concepts in the meta-model: StructuredConfigurationParameter and StructuredConfigurationParameterValue.

Update the XML schema and DTD accordingly.

Revised Text:

In section 9.2.9 (Annotations and configuration elements), replace the content of figure 20 (UCM configuration elements) with file ucm_configuration_parameters.svg.

Also change the legend of figure 20 from “UCM configuration elements” to “UCM configuration parameters”.

Just after section 9.2.9.2 (ConfigurationParameterValue), and before section 9.2.9.3 (AnnotationDefinition), insert a new figure with file ucm_annotations.svg.

The figure number should be 21. The legend is “UCM annotations”.

In order to properly create the figure, first create a frame (menu Insert → Frame → Frame...). Set the frame anchor to “as character”, so that the frame is not floating. Inside the frame, insert the image with file ucm_annotations.svg. Then insert the legend.

Just after section 9.2.9.4 (Annotation), insert a new figure with file ucm_structured_parameters.svg.

The figure number should be 22. The legend is “UCM structured configuration parameters”.

After figure 22, create a new section 9.2.9.5 with title “StructuredConfigurationParameter (IConfigurable, IConfigurationParameter)”.

Then add the following text:

A structured configuration parameter contains several IConfigurationParameter elements. It is a way to gather several parameter together.

Create a new section 9.2.9.6 with title “StructuredConfigurationParameterValue (IConfigured, IConfigurationParameterValue)”.

Add the following text:

A structured configuration parameter value gathers several configuration parameter values together. The set of configuration parameter values shall conforms with the StructuredConfigurationParameter referenced by the StructuredConfigurationParameterValue.

- parameterDefinition: StructuredConfigurationParameter [1]

Create a new section 17.2.2 with title “Connector with port configuration and structured parameters”.

Put the following text in the section:

It is possible to gather several configuration parameters together using structured configuration parameters. Connector enh_fifo_msg_cnt extends the standard message connector. It manages different message priorities in reception FIFOs, depending on the emitter.

and the following XML code block:

```xml
<interactionModule name="enhanced_fifo_connector">
  <contractModule name="data_types">
    <integer kind="int32" name="channel_id_t"/>
    <integer kind="uint8" name="fifo_priority_t"/>
  </contractModule>
  <connectorDef name="enh_fifo_msg_cnt" pattern="::ucm_core::messages::msg_intr_pat">
    <extends ref="::ucm_core::messages::simple_msg_cnt"/>
    <portConf port="emitter">
      <configParam name="channel_id" type="data_types::channel_id_t"/>
    </portConf>
    <portConf port="receiver">
      ...
    </portConf>
  </connectorDef>
</interactionModule>
```
Create a new section 17.4.5 with title “Structured configuration parameter values”. Put the following XML code block in the section:

```xml
<componentModule name="fifo_components">
<contractModule name="contracts">
<integer kind="int16" name="payload_t"/>
<contractModule>
<bindingSet name="payload_binding">
<binding abstract="::ucm_core::messages::transm_data_t" actual="contracts::payload_t"/>
</bindingSet>
<compType name="C_sender">
<port bindings="payload_binding" name="snd" type="::ucm_core::messages::msg_emtr_pt"/>
</compType>
<compType name="C_receiver">
<port bindings="payload_binding" name="rcv" type="::ucm_core::messages::msg_rcvr_pt"/>
</compType>
<compType name="C_appli"/>
<atomic lang="::ucm_lang::cpp::CPP11_typed" name="CI_sender" type="C_sender"/>
<atomic lang="::ucm_lang::cpp::CPP11_typed" name="CI_receiver" type="C_receiver"/>
<composite name="CI_appli" type="C_appli">
<part name="c1" ref="CI_sender"/>
<part name="c2" ref="CI_sender"/>
<part name="c3" ref="CI_sender"/>
<part name="c4" ref="CI_receiver"/>
<part name="c5" ref="CI_receiver"/>
</composite>
<connection name="cnx1" ref="::enhanced_fifo_connector::enh_fifo_msg_cnt">
<end name="c1_snd" part="c1" port="snd">
<config def="channel_id" value="1"/>
</end>
<end name="c2_snd" part="c2" port="snd">
<config def="channel_id" value="2"/>
</end>
<end name="c3_snd" part="c3" port="snd">
<config def="channel_id" value="3"/>
</end>
<end name="c4_rcv" part="c4" port="rcv">
<structConfig def="fifo_priorities">
<config def="emitter_channel_id" value="1"/>
<config def="fifo_priority" value="2"/>
</structConfig>
<structConfig def="fifo_priorities">
<config def="emitter_channel_id" value="2"/>
<config def="emitter_channel_id" value="3"/>
<config def="fifo_priority" value="9"/>
</structConfig>
</end>
<end name="c5_rcv" part="c5" port="rcv">
<structConfig def="fifo_priorities">
<config def="emitter_channel_id" value="1"/>
<config def="fifo_priority" value="5"/>
</structConfig>
<structConfig def="fifo_priorities">
<config def="emitter_channel_id" value="2"/>
<config def="fifo_priority" value="2"/>
</structConfig>
</end>
</connection>
</componentModule>
</contractModule>
</componentModule>
</contractModule>
```
And the following text:

Composite component CI_appli contains 5 subcomponents: c1, c2 and c3 send messages to c4 and c5. Subcomponent c1 sends on channel 1, c2 sends on channel 2, c3 sends on channel 3.

Subcomponents c4 and c5 receive all messages, with different priorities. Thus, c4 receives data from c1 (channel 1) with priority 2, data from c2 and c3 (channels 2 and 3) with priority 9. And c5 receives data from c1 (channel 1) with priority 5, data from c2 (channel 2) with priority 2, data from c3 (channels 3) with priority 1.

**Extent Of Change:**
Minor

**Disposition:**
Resolved - Approved on [Ballot #1](https://issues.omg.org/browse/UCM13-16)

**Reporter:**
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

**Reported:**
Thu, 15 Oct 2020 13:16 GMT on [UCM 1.2](https://issues.omg.org/browse/UCM13-16)

**Updated:**
Thu, 15 Apr 2021 00:45 GMT

**Components:**
- meta-model
- syntax

**Discussion:**
UCM13-16 - [https://issues.omg.org/browse/UCM13-16](https://issues.omg.org/browse/UCM13-16)

**Attachments:**

- **ucm_annotations.svg** - ptc-21-05-09/UCM13-16/ucm_annotations.svg 16 kB (image/svg+xml)
- **ucm_base.dtd** - ptc-21-05-09/UCM13-16/ucm_base.dtd 14 kB (application/xml-dtd)
- **ucm_base.xsd** - ptc-21-05-09/UCM13-16/ucm_base.xsd 44 kB (text/xml)
- **ucm_configuration_parameters.svg** - ptc-21-05-09/UCM13-16/ucm_configuration_parameters.svg 16 kB (image/svg+xml)
- **ucm_structured_parameters.svg** - ptc-21-05-09/UCM13-16/ucm_structured_parameters.svg 14 kB (image/svg+xml)
 OMG Issue: UCM13-17

Title: Unexpected mention of “monolithic” component implementation

Summary: In section 14.1.2.2 (From technical policies to fragments), diagram 35 shows a “monolithic” component implementation. Monolithic component implementations are mentioned nowhere else in the document; this “monolithic” component is should be renamed “atomic” instead.

Source: formal/20-04-02 — Chapter/Section: 14.1.2.2 — Page Number/s: 81

Resolution Summary:
rename “monolithic” into “atomic”

In the early stages of the UCM standard, atomic component implementation were called monolithic. Update diagram 35 to use word “atomic”.

Revised Text:
In section 14.1.2.2 (From technical policies to fragments), replace diagram 35 (Technical policy fragmentation example) with file policy_fragmentation.svg

Extent Of Change:
Support Text

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnau@thalesgroup.com)

Reported:
Thu, 15 Oct 2020 13:39 GMT on UCM 1.2

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
text

Discussion:
UCM13-17 - https://issues.omg.org/browse/UCM13-17

Attachments:
- policy_fragmentation.svg - ptc-21-05-09/UCM13-17/policy_fragmentation.svg 54 kB (image/svg+xml)
OMG Issue: UCM13-21

Title: Rephrase some explanations

Summary:
There are some inconsistencies in the first sections of the document.
For example, section 1.4 is entitled “Programming model” while section 14 is entitled “UCM container model”.
Also, section 9.1.2 mentions “properties”, which do not seem to be defined in the rest of the document.

Source:
formal/20-04-02 — Chapter/Section: 1, 9 — Page Number/s: 3-7, 13-14

Resolution Summary:

Rephrase some explanations

Some terms (e.g. “properties” or “programming model”) had been used in the early stages of the standard, but are have not been used in the final version of the document. Replace these obsolete terms with the correct ones.

Revised Text:

Rename the title of section 1.4 (UCM programming model) into “UCM container model”.

In this section, replace the first paragraph by the following text:

The container model of a UCM component specifies how the source code of a component is structured. It relies on the principle of decoupling the business code from the platform code. Only atomic components correspond to business logic. Composite components are simple boxes that nest subcomponents: they have no real existence in the container model.

In the second paragraph, replace “UCM framework provider” by “UCM platform provider”. Also replace “software architect” by “component designer”.

In the third paragraph, replace “The descriptions of the component” by “The descriptions of the components”. That is, append an “s”.

In section 1.5 (UCM levels of conformance), in the first paragraph, replace “The first is…” by “The first one is…”.
In the same paragraph, replace “The second is…” by “The second one is…”.

In section 9.1.2 (Configuration mechanisms), in the first paragraph, replace “properties” by “annotations”. In the second paragraph, replace “Properties” by “Annotations”.
In the fourth paragraph, replace “… platform entities: interaction patterns, connector definitions, connector implementations and technical policy definition.” by “… platform entities: connector definitions and technical policy definitions.”.
Still in the fourth paragraph, replace “a technical policy that defines” by “a technical policy that manages”. In the fifth paragraph, replace “Properties” by “Annotations”.

Extent Of Change:
Support Text

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Title:
List XML tags names and clarify the rules for name references

Summary:
Although the XML syntax of UCM is specified by a DTD and an XML schema, it would be nice to have at least a list of the XML tags in the main document, to help read the DTD and the schema. Also, the way of referencing entities is not completely clear for relative names.

Resolution Summary:
List XML tags names and clarify the rules for name references
List the main tag names of the XML syntax. Clarify relative name references.

Revised Text:
In section 10.1 (Entity names), in the paragraph, insert the following text after “if the entity is in a submodule”:

, or simply if the entity is in the same module or in a parent module.

Ce whole paragraph becomes

Basically, every UCM concept described in the meta-model (see section 9) corresponds to an XML tag. UCM elements are referenced by their name, formatted as follows:
::absolute::module::path::entity, or submodule::entity if the entity is in a submodule, or simply if the entity is in the same module or a parent module. Elements of entities (e.g. component ports) are referenced with a dot, as follows: module::component.port.

Create a subsection 10.1.1 with title “Naming scopes”.

Put the following text in this new subsection:

Each module defines a naming scope. Two entity declarations shall not have the same name within a given module. The following example is incorrect:

An entity should not have the same name as the module it is declared in. The following example is therefore correct but strongly discouraged:
An entity shall not contain two features with the same name. Entities that extend or refine other entities inherit the scope of the entities they extend or refine. The following example is incorrect:

```
<interactionModule name="incorrect2">
  <!-- the naming scope is the current module (incorrect2) -->
  <connectorDef name="cnt1" pattern="::somewhere::pattern">
    <connPort name="receive" type="::somewhere::port_type"/>
    <configParam name="protocol" type="::protocols::protocol_t"/>
  </connectorDef>
  <!-- this is incorrect because "receive" has already been defined -->
  <configParam name="receive" type="::protocols::mode_t"/>
</connectorDef>

<connectorDef name="cnt2" pattern="::somewhere::pattern">
  <extends ref="cnt1"/>
  <!-- this is incorrect because name "receive" has already been defined in cnt1. -->
  <connPort name="receive" type="::elsewhere::port_type"/>
  <!-- this is also incorrect because "protocol" has already been defined in cnt1. -->
  <configParam name="protocol" type="::other_protocols::other_protocol_t"/>
</connectorDef>
```

A feature that refines another feature can have the same name, since it replaces it. The following example is correct:

```
<componentModule name="correct1">
  <!-- the naming scope is the current module (correct1) -->
  <compType name="C1">
    <port name="p1" type="::somewhere::port_type"/>
  </compType>

  <compType name="C2">
    <refines ref="C1"/>
    <!-- this is correct because C2.p1 refines (i.e. replaces) C1.p1 -->
    <port name="p1" type="::elsewhere::port_type">
      <refines ref="C1.p1"/>
    </port>
  </compType>
</componentModule>
```

Entities declared in a module have visibility on the naming scopes of the parent modules. Consider the following example:

```
<contractModule name="M1">
  <!-- the naming scope is the current module (M1) -->
  <integer name="i" kind="uint8"/>

  <contractModule name="M2">
    <!-- the naming scope is the current module (M1::M2) -->
    <interface name="M1::intf1">
      <method name="f">
        <!-- attributes "a" and "b" have the same type, declared in ::M1 -->
      </method>
    </interface>
  </contractModule>
</contractModule>
```
Inside annotations and platform entities (e.g. technical policies and connections), configuration parameters (as well as structured configuration parameters) are referred to with respect to the naming scope of the entity definition. The following example is correct:

Create a subsection 10.1.2 with title “XML tags”.

Put the following text in this new subsection:

XML tags that correspond to a named UCM entity (i.e. an entity that inherits from INamed) have at least two attributes: name and id.

The XML tag for comment is the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleComment</td>
<td>comment</td>
</tr>
</tbody>
</table>

Create a subsection 10.1.2.1 with title “Modules”.

Put the following text in this new subsection:

The XML tags for modules are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>namespace</td>
</tr>
<tr>
<td>ContractModule</td>
<td>contractModule</td>
</tr>
</tbody>
</table>
Create a new subsection 10.1.2.2 with title “Contracts”.

Put the following text in this new subsection:

The XML tags for data types are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrimitiveInteger</td>
<td>integer</td>
</tr>
<tr>
<td>PrimitiveChar</td>
<td>char</td>
</tr>
<tr>
<td>PrimitiveFloat</td>
<td>float</td>
</tr>
<tr>
<td>PrimitiveBool</td>
<td>bool</td>
</tr>
<tr>
<td>PrimitiveOctet</td>
<td>octet</td>
</tr>
<tr>
<td>StringType</td>
<td>string</td>
</tr>
<tr>
<td>Array</td>
<td>array</td>
</tr>
<tr>
<td>Sequence</td>
<td>sequence</td>
</tr>
<tr>
<td>Struct</td>
<td>struct</td>
</tr>
<tr>
<td>Union</td>
<td>union</td>
</tr>
<tr>
<td>Enumeration</td>
<td>enum</td>
</tr>
<tr>
<td>NativeType</td>
<td>native</td>
</tr>
<tr>
<td>Alias</td>
<td>alias</td>
</tr>
<tr>
<td>AbstractDataType</td>
<td>abstractDataType</td>
</tr>
</tbody>
</table>

The XML tag for constants is

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>constant</td>
</tr>
</tbody>
</table>

The XML tags for interfaces are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>exception</td>
</tr>
<tr>
<td>Interface</td>
<td>interface</td>
</tr>
<tr>
<td>Attribute</td>
<td>attribute</td>
</tr>
<tr>
<td>Method</td>
<td>method</td>
</tr>
<tr>
<td>Parameter</td>
<td>param</td>
</tr>
<tr>
<td>AbstractInterface</td>
<td>abstractInterface</td>
</tr>
</tbody>
</table>

The XML tags for configurations and annotations are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnnotationDefinition</td>
<td>annotationDef</td>
</tr>
</tbody>
</table>
Create a new subsection 10.1.2.3 with title “Interaction”.

Put the following text in this new subsection:

The XML tags for interactions are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>InteractionPattern</td>
<td>pattern</td>
</tr>
<tr>
<td>InteractionItem</td>
<td>item</td>
</tr>
<tr>
<td>InteractionRole</td>
<td>role</td>
</tr>
<tr>
<td>PortRole</td>
<td>portRole</td>
</tr>
<tr>
<td>PortType</td>
<td>portType</td>
</tr>
<tr>
<td>ConnectorDefinition</td>
<td>connectorDef</td>
</tr>
<tr>
<td>ConnectorPort</td>
<td>connPort</td>
</tr>
<tr>
<td>ItemBinding</td>
<td>itemBinding</td>
</tr>
<tr>
<td>PortConfiguration</td>
<td>portConf</td>
</tr>
<tr>
<td>PortElement</td>
<td>portElement</td>
</tr>
</tbody>
</table>

Create a new subsection 10.1.2.4 with title “Nonfunctional aspects”.

Put the following text in this new subsection:

The XML tags for nonfunctional aspects are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>language</td>
</tr>
<tr>
<td>TechnicalAspect</td>
<td>technicalAspect</td>
</tr>
<tr>
<td>TechnicalPolicyDefinition</td>
<td>policyDef</td>
</tr>
<tr>
<td>SpecificConfigurationParameter</td>
<td>specificParam</td>
</tr>
</tbody>
</table>

Create a new subsection 10.1.2.5 with title “Components”.

Put the following text in this new subsection:

The XML tags for components are the following.

<table>
<thead>
<tr>
<th>Meta-model class</th>
<th>XML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>BindingSet</td>
<td>bindingSet</td>
</tr>
<tr>
<td>AbstractTypeBinding</td>
<td>binding</td>
</tr>
<tr>
<td>ComponentType</td>
<td>compType</td>
</tr>
<tr>
<td>Port</td>
<td>port</td>
</tr>
<tr>
<td>AtomicComponentImplementation</td>
<td>atomic</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>TechnicalPolicy</td>
<td>policy</td>
</tr>
<tr>
<td>SpecificPolicyConfigurationValue</td>
<td>specificConfig</td>
</tr>
<tr>
<td>CompositeComponentImplementation</td>
<td>composite</td>
</tr>
<tr>
<td>AssemblyPart</td>
<td>part</td>
</tr>
<tr>
<td>Connection</td>
<td>connection</td>
</tr>
<tr>
<td>ConnectionEnd</td>
<td>end</td>
</tr>
<tr>
<td>PortDelegation</td>
<td>portDelegation</td>
</tr>
<tr>
<td>AttributeDelegation</td>
<td>attrDelegation</td>
</tr>
</tbody>
</table>

Extent Of Change:
Support Text

Disposition:
Resolved - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Tue, 2 Mar 2021 17:31 GMT on UCM 1.2

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
text

Discussion:
UCM13-23 - https://issues.omg.org/browse/UCM13-23
Disposition: Deferred

No issues in this report.
No issues in this report.
Title:
Add a method in the comp_trace policy to log messages with a default severity

Summary:
The invocation of the standard technical policy for traces consists of a method log that takes two parameters:

- the severity (trace, debug, warning, etc.)
- the log message (a wide character string)

It can be tedious to always have to specify the severity. There should be an additional log method with a default severity.

Source:
ptc-19-04-03 — Chapter/Section: 13.1.4.3, 13.1.4.4 — Page Number/s: 59-60

Resolution Summary:
Do not change the comp_trace policy

No commonly used logging API (e.g. log4cpp) supports a default severity level. Therefore, such a mechanism does not correspond to current practice, and may be difficult to understand.

Platform providers that really want to deal with a default severity logging level can define their own, non standard, technical policy.

Disposition:
Closed; No Change - Approved on Ballot #1

Reporter:
Thomas Vergnaud, THALES (thomas.vergnaud@thalesgroup.com)

Reported:
Tue, 7 Jan 2020 17:13 GMT on UCM 1.2

Updated:
Thu, 15 Apr 2021 00:45 GMT

Components:
library

Discussion:
UCM13-4 - https://issues.omg.org/browse/UCM13-4
No issues in this report.
No issues in this report.