**RPS R2D2 Alternative Model: DMIM and RMIM**

v1, 4 January 2012

Keith Thomas

The DMIM and the RMIM are related such that the DMIM contains the exact same set of data objects as the RMIM, the same relationships (solid lines), **plus** the completed relationships derived from relationships conveyed by reference in the RMIM (dashed lines). The RMIM is overlaid on the DMIM.

The DMIM is a conceptual representation of a repository of RMIM (Submission Unit) instances.

*Speaking informally of the DMIM*, a set of objects connected by solid-line relationships represents one update cycle in the life of the repository (i.e. one Submission Unit). The cycles are "stacked" (as if coming out of the page) in the order of the SU time stamp.  Therefore what's "visible" is the current state of the repository. Of course there are many distinct objects of the same class: think of all the different Applications as if they were laid out side-by-side where the Application box is, each with its own history underneath; do the same for all classes (except for Submission Units, pretend there is only one stack of them).

To see the history of any object you have to "look under" it to see what it used to be, if anything. To see the status of a whole application at some previous point in time, just "peel back" the later layers of SUs until you come to the time you're interested in.

At any point in the sequence of update cycles you can start from anywhere in the hierarchy and navigate along its relationships to construct a view from that object.

In operation, there will be an XML schema *only* for the RMIM, because only RMIM instances are exchanged. It is up to each implementer to provide the functional equivalent of the conceptual repository represented by the DMIM using whatever technology they choose.

**Figure 1: DMIM Outline**

**Applicant**

**r**

[4]

**Application**

**Territorialauthority**

**C**

**C**

**C**

[4]

**Review**

**KeywordDefinition**

**Submission**

**S**

[3]

**C**

**C**

**C**

[3]

**S**

**ReviewableUnit**

**C**

**C**

**CallBackContact**

**K**

**K**

**K**

**K**

[2]

[2]

**SubmissionUnit**

**U**

**U**

**C**

**D**

 [5]

[4]

**C**

[1]

[1]

**Document**

**ContextOfUse**

**R**

**R**

 [6]

[6]

**Legend:**

Participation t relationship:

Act relationship: conveyed in RMIM by direct relationship from source to target

 conveyed in RMIM by inverse relationship from target using a reference

 conveyed in RMIM by direct relationship from source using a reference

Act relationship types:

**C** – component **D** – derived from **K** – covered by **R** – replace (doc. versions only)

**r** -- reference **S** -- subject of **U** -- uses

All act relationships are *one-to-many from source to target*, except where noted.

**Notes:**

[1] Only one of these is given for a path between a CoU and Submission (i.e. if a Reviewable Unit applies, only a reference to it is given, not to the parent Submission); these relationships carry *Priority Number*.

[2] Only one of these is given for a path between a SU and Submission (i.e. if a Reviewable Unit applies, only a reference to it is given, not to the parent Submission); these relationships carry *Sequence Number*.

[3] Only one of these is given for a given instance of the Review class.

[4] These are *many-to-many* relationships because one instance of the target class may be associated with multiple instances of the source class, and *vice versa*.

[5] This is a one-to-many relationship *from target to source* because more than one CoU may be derived from the same document object.

[6] These are *one-to-one* relationships from source to target

**TerritorialAuthority**

**Figure 2: RMIM Outline**

**Applicant**

1,\* **reference**

Application ref

 **Keyword**

* code CD CWE [1..1]

 **Keyword**

* code CD CWE [1..1]

0,1 **Replace**

1,\* **Component**

Application ref

**RPSDocument (DOC)**

* id DSET<II> [2..2]
* code CD CWE [1..1]
* title ST [0..1]
* text ED [0..1]
* statusCode CS [1..1]
* setId II [1..1]
* versionNumber INT [1..1]

1,1 **ComponentOf**

1,\* **ComponentOf**

**C**

Application ref

**KeywordDefinition**

**SubmissionUnit**

**Application**

**Territorialauthority**

1,1 **ComponentOf**

Application ref

**Submission**

* id DSET <II> [2..\*]
* code CD [1..1]
* title ST [0..1]
* statusCode CS [1..1]

1,1 **ComponentOf**

[0,1] **Subject>**

**RegulatoryStatus**

* code CD CWE [1..1]

[0,1] **<SubjectOf**

 **Group**

* id II [1..1]

[0,1] **Subject>**

**RegulatoryReviewTime**

* code CD CWE [1..1]

Submission ref

**ReviewableUnit**

Sub/RU ref

1,1 **SubjectOf**

**Review**

1,\* **ComponentOf**

[0,1] **Subject>**

 **Mode**

* code CD CWE [1..1]

Technically KeywordDefinition is not an independent class because it has no *id*, but since each instance acts independently of every other instance and is never updated, it is treated here as an independent class.

**CallBackContact**

**K**

**K**

**K**

**K**

1,\* **ComponentOf**

Sub/RU ref

**U**

Submission Message

**U**

Sub/RU ref

**ContextOfUse (DOC)**

* id DSET<II> [2..2]
* code CD CWE [1..1]
* title ST [0..1]
* statusCode CS [1..1]
* setId II [1..1]
* versionNumber INT [1..1]

1,1 **DerivedFrom**

Document ref

Document ref

0,1 **Replace**

CoU ref

Document ref

[0,\*] **Subject>**

[0,\*] **Subject>**

**Note**

Independent classes are shown as solid boxes; encapsulated classes are shown as dashed boxes inside their parent class under their relationship, unless they are reference classes, in which case they are shown outside under their relationship with the pointed side toward the target of the relationship.

For reasons of space, only *Submission, ContextOfUse* and *RPS Document* have been fully populated with attributes and encapsulated classes in this illustration.

**TerritorialAuthority**