Combining archetypes into Templates

Overview

**********NOTE: this worked example section of the wiki is now out of date as GELLO is currently at R. 2. This code will not work in later versions of the editor. We intend to update this section to R2 GELLO soon. **********

Provide a simple worked example for the MO Template Editor that shows how modular archetypes can be combined.

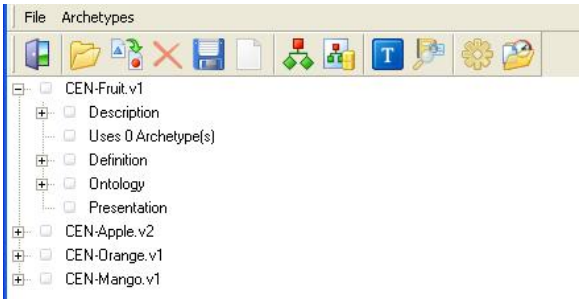
A fruity example

Archetypes may be combined into one template. The advantage of this is that archetype modules can be write once - use many times, in different report and user contexts. By way of example you may have the talked about top ten archetypes and a couple more that make up a fair chunk of the EHR content, and then choose to combine these in certain ways for certain situations. Thus a discharge summary could contain (at least) a current medication list, allergies and history of encounter; and the clinical coding department could receive an intial patient separation notice including a brief cut down history of encounter, while pharmacy could receive a list of added medications. So this example shows the use case of combined archetype modules that comprise sometimes just parts of the originating archetypes. We have heard this use case in several places now.

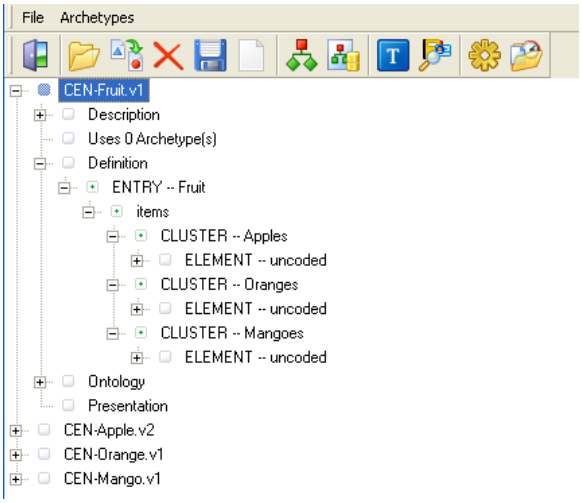
But let's start with a simple combination of archetypes into a Template. In this case - the apple archetype is joined with orange and mango archetypes and we will do some Gello enabled adding up of weight and cost.

Some necessary files are located in **CEN-Fruit.zip**

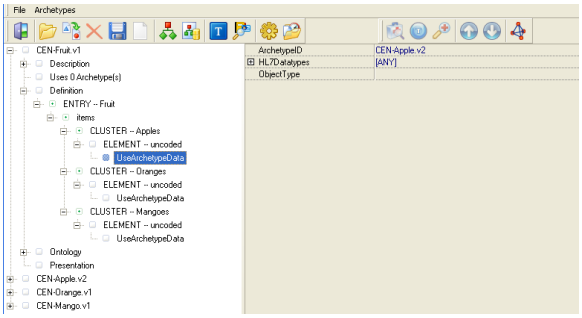
1. Create a new template in the template editor and call it 'CEN-Fruit.v1'
2. Either create two new archetypes yourself in another instance of the editor application, modelled on the mango and orange archetypes in the downloaded zip file; or choose to just bring them pre-made into the new Fruit template. In this case go **File -> Add Archetypes from File** , and then navigate to where you have your latest version of the apple archetype located. Click to add. This places the apple archetype below the 'CEN-Fruit.v1' archetype on the LHS. Repeat for 'CEN-Orange.v1.xml' and for 'CEN-Mango.v1.xml' in that order.



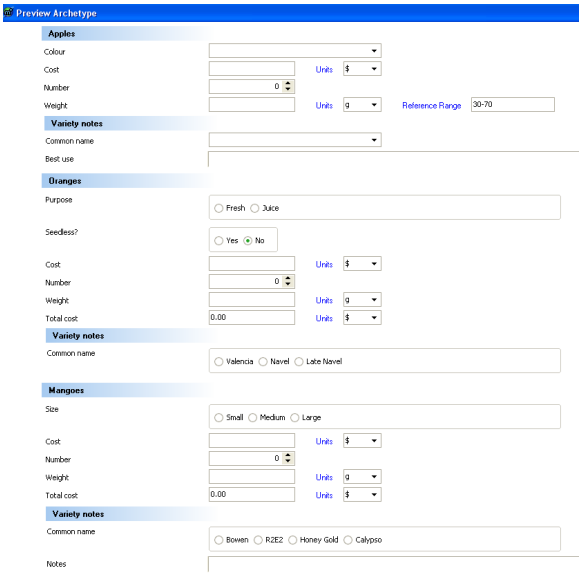
3. Open out the Definition for Fruit. Name the **ENTRY** 'Fruit' and adjust the atCode to at0000. To bring the other archetypes into the Fruit template, make three new **CLUSTERs** (by using **Add Data Group** after right clicking on items. Name them for the component fruits - Apples, Oranges, Mangoes. Add an **ELEMENT** under each of them (remember we really should have an items between the **CLUSTER** and the **ELEMENT).** This **ELEMENT** node will hold the archetypes.



4. Make the datatypes for these **ELEMENTs** each to be mtArchetype. Unfold the **ELEMENTs** to show **UseArchetypeData.** Click on this and on the RHS type in the names of the archetypes being used in the template, namely 'CEN-Apple.v2', 'CEN-Orange.v1' and then 'CEN-Mango.v1'.



5. Click on any of the nodes in the Fruit template and then on preview it. This is what you will see:



6. Save your work as 'CEN-Fruit.v1'. This will change the **uses 0 Archetype(s)** in the LHS to **Uses 3 Archetype(s)**

7. Now we will add some GELLO and some nodes that calculate the total cost and weight for the Fruit template. Make a **CLUSTER** in the Fruit template below the mangoes **CLUSTER.** Call it Grand totals for fruit. Under this put two **ELEMENTS** named 'Total Fruit Cost' and 'Total Fruit weight'. Give 'Total Fruit Cost' two decimal places. Make their data types (at **ADLMatchType** on the RHS) 'mtQuantity'. Make their units to be '$' and 'g'. (Go here if you can't remember how to do this). Add this GELLO to **isCalculated** for 'Total Fruit Cost':

```
let o:observation = parameter[1]
let ApplesCost: PhysicalQuantity =
if o.isdefined() then
o.find_observation('1.1.1.1.5').value_asPQ()
else
Factory.PhysicalQuantity(0,'$')
endif
let OrangesCost: PhysicalQuantity =
if o.isdefined() then
o.find_observation('1.1.2.1.1.6').value_asPQ()
else
Factory.PhysicalQuantity(0,'$')
endif
let MangoesCost: PhysicalQuantity =
if o.isdefined() then
o.find_observation('1.1.3.1.1.5').value_asPQ()
else
Factory.PhysicalQuantity(0,'$')
endif
ApplesCost + OrangesCost + MangoesCost
```

8. Here's some code for 'Total Fruit Weight':

```
let o:observation = parameter[1]
let ApplesWt: PhysicalQuantity =
if o.isdefined() then
o.find_observation('1.1.1.1.4').value_asPQ()
else
Factory.PhysicalQuantity(0,'g')
endif
let OrangesWt: PhysicalQuantity =
if o.isdefined() then
o.find_observation('1.1.2.1.1.5').value_asPQ()
else
Factory.PhysicalQuantity(0,'g')
endif
let MangoesWt: PhysicalQuantity =
if o.isdefined() then
o.find_observation('1.1.3.1.1.4').value_asPQ()
else
Factory.PhysicalQuantity(0,'g')
endif
ApplesWt + OrangesWt + MangoesWt
```

9. Test it in preview mode. What happens? It doesn't work (yet) as we need to set **BubbleEventsUp** to be 'True' for all the **CLUSTER** nodes in the Fruit template, as well as the **ENTRY** and any **ite ms** nodes in the component archetypes (located below where we are working in the LHS; the tags for each being located at the top on the RHS. Adding

```
Factory.PhysicalQuantity(0,'g')
```

to the OnInitialize tags on the RHS for each of the component archetypes and

```
Factory.PhysicalQuantity(0,'$')
```

similarly for their Total Cost should clean things up nicely. If you still have no action check the archetype paths for the retrieved node values in the GELLO code for the Fruit template. The zip file contains a working example.