**Proposal for Derived Relationships**  
G. Edward Roberts  
[eroberts@elparazim.com](mailto:eroberts@elparazim.com)

This paper deals with several things:

1. Proposal for Derived Relationships
2. Differentiation of purpose of Derived Relationships
3. How does the new Association between Elements and Relationships fit into the Derived Relationships
4. **Proposal for Derived Relationships:**

To get the structural rules, the following standard model simplifications are used:

* On aggregations/compositions, relationships on the part can move to the whole
* On realizations/specializations, relationships on the generalization can move to the specialization
* On assignments, relationships on the Behavior Element can be moved to the Active Structural Element.

Change the categorization and prioritization (weakest to strongest) of relationships as

**Structural:**

* Assignment
* Realize
* Aggregation
* Composition
* Specialization

**Dependency**:

* Association
* Influence
* Access
* Serving

**Dynamic** (these do not interact):

* Trigger
* Flow

Junction is not classified as a relationship.

How does one do derived rules:

1. Collapse the Structural Part of the Model, applying the weakest structural relationship possible.
2. For dependency relationships, apply the weakest relationship between the collapsed structural parts. For Access, relationship is “walked” in the direction of the Arrow. Association can be “walked” in either direction.
3. Treat all dynamic relationships separately (they do not co-mingle), and apply the single dynamic relationship to the collapsed structure.

Restated Rules: [I took these directly from Jean’s paper] (these handle all relationship classes including dynamic)

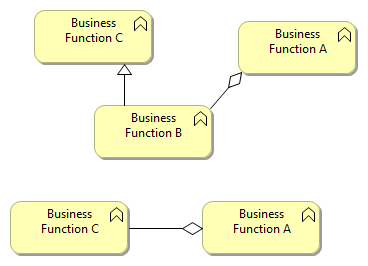
* If two relationships *r*:*R* and *s*:*S* of the same class (either structural or dependency) are permitted between elements *a*, *b*, and *c* such that *r*(*a,b*) and *s*(*b*,*c*), then a relationship *t*:*T* is also permitted, with *t*(*a*,*c*) and type *T* being the weakest of *R* and *S*.
* The Access relationship is in the direction of the Read or Write arrow (or both way if both arrow are present or if neither arrow is present)
* If there is a dependency or dynamic relationship *r* from(to) element *a* to(from) element *b*, and a structural relationship from element *c* to element *a*, a dependency or dynamic relationship *r* can be derived from(to) element *c* to(from) element *b*.

So I think that these rules are like the current derived except for the changed order of priority and Access rule about direction. Structural part are stronger than dependencies. These also remove the ability to move a dynamic relationship to a service from a function.

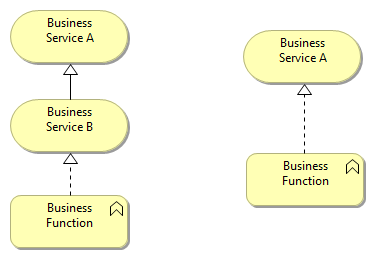
Note: this “movement” is currently in line with what is currently stated in the derived rules.

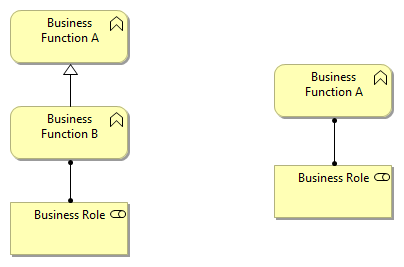
How do we get the ordering of the Structural Rules:

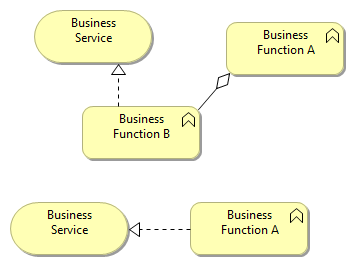
1. Specialization and Aggregation/Composition



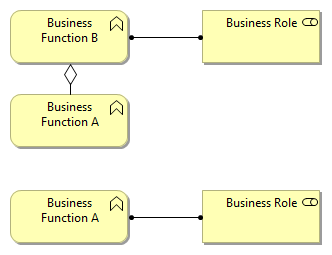
1. Specialization and Realize



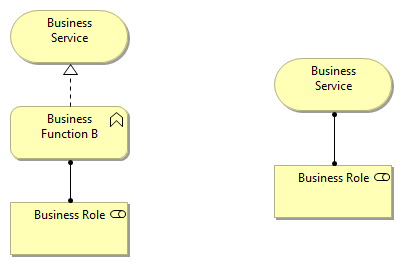
1. Specialization and Assignment  
     
   
2. Aggregation/Composition and Realization



1. Aggregation/Composition and Assignment

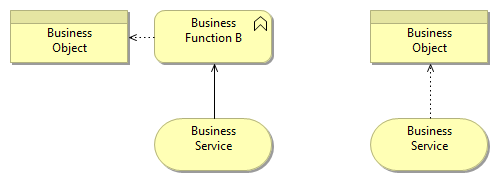


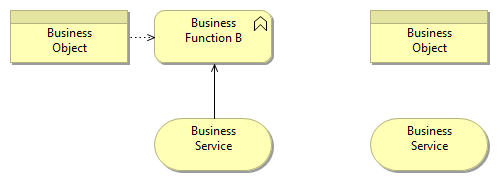
1. Realization and Assignment

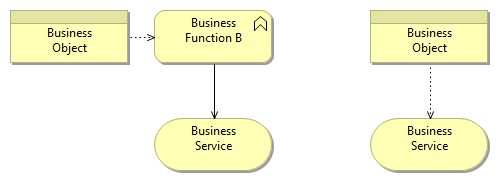


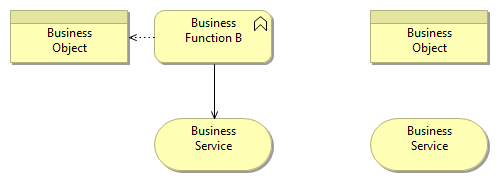
How do we get the ordering of the Dependency Rules:

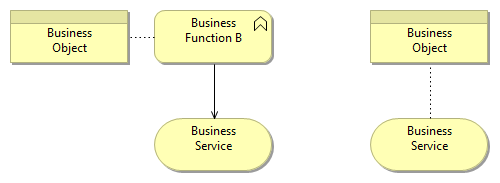
1. Association is the weakest relationship
2. Serving and Access

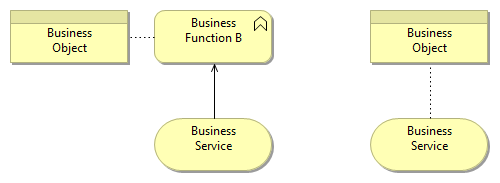






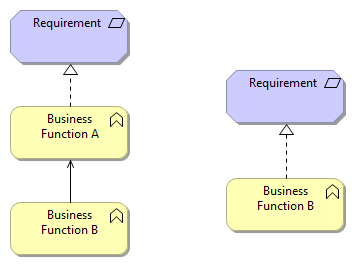




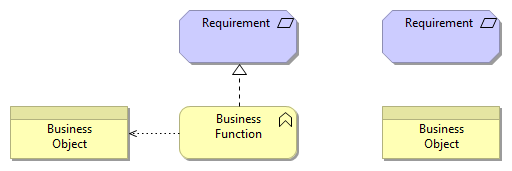


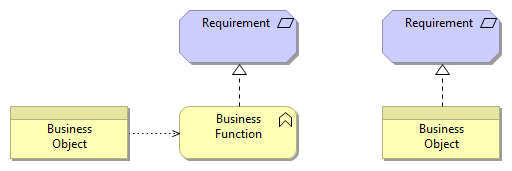
Show use Access in the direction of the arrow

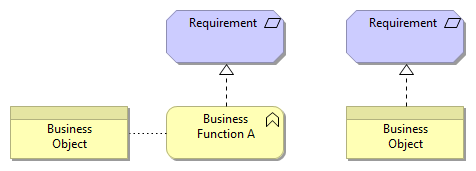
1. Serving and Influence (read realize as influence)



1. Access and Influence (read realize as influence)







The Following is an example of a model that no longer works in the same way:

Infrastructure service serves Application component realizes Application service.... Then I can't derive anything

Because there is nothing incumbent on the Application Service to be served by the Infrastructure Service....

this illustrates a really deep issue in what is meant by "architectural integrity"...

a. do we describe what is going on in a particular architecture (that has a implementation specified)... then this could be a needed relationship

b. do we describe what is architectural relevant... then Application Service does not have to be serviced by the Technology Service... and it is not relevant

personally, I like this... if you take the new derivation rules... it supports b. not a. ... and you will get things in your architecture which are only true across any implementation... in the derived rules...

1. **Purpose**

Now I have this issue, which gives me no end of headaches working with ArchiMate:

***if I have say an event A that is in relationship r to a process C which is Nested (composed or aggregated) inside of process B... then if I go to a different diagram and put event A on it and process B (and no process C), there should be the same relationship r***

This to me clearly is a use of the 2nd purpose of Derived Relationships.

Marc said in an email:

“We don’t distinguish rules for diagrams. Everything is about the models.”

Ok… I can understand the standard will not define things about diagrams only models. But he also said

“If you put a relationship on a view, it *must* be in the model too, since any view is a reflection of the model”

“ArchiMate is not a diagram-based language like UML. Its view concept always relates to a unified underlying model, and for a derived relationship to show on a view, it has to be in the model. A view is simply a visualization of a subset of that model.”

So this may be true for 2.1, but that does not mean we can not deal with the issue in Coronado.

SO…………………………………..

The question is, can we have a Derived Relationship on a Diagram which is not in the model? If not what about the problem I illustrate here?

My proposal on this is simply that a how a relationship is shown on a diagram is distinct to its presence in a model. Any relationship can be shown in a derived way on a diagram but that does not change the relationship in the model.

1. **New Considerations for Elements with Associations with Relationships**

We now have essentially

