

## Notes from iterator-composite.PDF-2

### Page 1

---

*Note 1; Label: Karl Lieberherr; Date: 11/7/98 12:04:45 PM*

I like the integration of APPCs with personalities in Mira's latest set of examples. David made a good point to adapt the generic operations in patterns, to show how to use APPCs to describe patterns and to use personalities.

I like Mira's description as a useful precursor for David's Java Beans description.

An APPC defines a one method interface with behavior for a class graph.  
A personality defines a many method interface with behavior for one class.

The ICG is now written in terms of interfaces and personalities. An interface only has an upstream interface. A personality has both an upstream interface and a downstream interface with an implementation.  
Downstream interface is given by protected methods.

*Note 2; Label: Karl Lieberherr; Date: 11/7/98 3:17:54 PM*

Luis Blando writes in his thesis:

We believe that a good way to merge the two technologies (APPCs/CGVs and personalities) would be to replace class definitions by personalities.

That is what we see happening here: An APPC is now a set of interfaces or personalities. A personality is used when there is a hotspot: behavior is required from the application or another APPC.

*Note 3; Label: Karl Lieberherr; Date: 11/7/98 11:45:48 AM*

inner(c) is a hook to extend the pattern. We would like to define such hooks on a need bases. This hook can be defined as an after method to a collection edge (-> Composite, children, Component). It is better to define this as an increment to the basiccomposite pattern, using an aspect.

How do we deal with the situation if we want to use on (-> Composite, children, Component) the visitor pattern and execute some code on a class between Composite and Component?

*Note 4; Label: Karl Lieberherr; Date: 11/7/98 12:21:14 PM*

Summing extends CompositePattern because it adds behavior to the composite pattern. The SummingPattern is a kind of CompositePattern.

The interface Leaf now turns into a personality

*Note 5; Label: Karl Lieberherr; Date: 11/7/98 2:23:20 PM*

A personality without a downstream interface? No protected method?  
Of course, there is an inherited protected method.

THIS does not look right. This code to be attached to a repetition edge.  
Where is getGross implemented for Composite?

### Page 2

---

*Note 1; Label: Karl Lieberherr; Date: 11/7/98 3:22:50 PM*

return((getCapacity() < getActualGross()));  
is simpler.

Intent: getCapacity: from class graph  
ActualGross: from another APPC

*Note 2; Label: Karl Lieberherr; Date: 11/7/98 3:36:54 PM*  
interesting: personifiedBy and not personifies. This specifies the mapping.  
Interface of Component is implemented here.

*Note 3; Label: Karl Lieberherr; Date: 11/7/98 3:34:52 PM*  
getNet is in the DI of Summing.Leaf and needs to be implemented  
setGross already implemented by Leaf personality

*Note 4; Label: Karl Lieberherr; Date: 11/7/98 3:36:08 PM*  
I like to Item: We could make it later to:  
via Something to Item

*Note 5; Label: Karl Lieberherr; Date: 11/7/98 3:43:15 PM*  
LimitedParty  
Wow: all pieces fall in place.

### **Page 3**

---

*Note 1; Label: Karl Lieberherr; Date: 11/7/98 3:51:26 PM*  
combine two APPCs

allows for partial implementation of DI

*Note 2; Label: Karl Lieberherr; Date: 11/7/98 5:34:30 PM*  
A class graph is now just viewed as an APPC.

We need a new cd for APPCs! This looks great.