Review of JSF AV Rules.

1.		**AV Rule 76** A copy constructor and an assignment operator **shall** be declared for classes that contain pointers to data items or nontrivial destructors.	Doesn't seem to fit any category cleanly, so either a category needs to be expanded to include it or a new category created. EP: this the deep-copy issue. Now [YAN]
2.		**AV Rule 77** A copy constructor **shall** copy all data members and bases that affect the class invariant (a data element representing a cache, for example, would not need to be copied).	Add to 6.43 Inheritance, or could add to a new inconsistency category. EP: this the deep-copy issue. Now [YAN]
3.		**AV Rule 78** All base classes with a virtual function **shall** define a virtual destructor.	Add to 6.15 Dangling Reference to Heap, 6.17 Using Shift Operations for Multiplication and Division EP: why on earth 6.17? Also, I see no connections to 6.15; need to read up. Now in [BKK}
4.		**AV Rule 79** All resources acquired by a class **shall** be released by the class's destructor.	Add to 6.15 Dangling Reference to Heap, 6.17 Using Shift Operations for Multiplication and Division EP: not 6.17; This is a rather obvious rule; belongs to memory leaks [XYL] 6.40
5.		**AV Rule 80** The default copy and assignment operators **will** be used for classes when those operators offer reasonable	Style issue; maybe a shallow-copy rule.

		semantics.	Now in [YAN]

6.		**AV Rule 81** The assignment operator **shall** handle self-assignment correctly
		
		AV Rule 81
		Self-assignment must be handled appropriately by the assignment operator. Example A illustrates a potential problem, whereas Example B illustrates an acceptable approach.
		Example A: Although it is not necessary to check for self-assignment in all cases, the following example illustrates a context where it would be appropriate.
		Base &operator= (const Base &rhs)
		{
		release_handle (my_handle); // Error: the resource referenced by myHandle is
		my_handle = rhs.myHandle; // erroneously released in the self-assignment case.
		*return *this;*
		}
		Example B: One means of handling self-assignment is to check for self-assignment before further processing continues as illustrated below.
		Base &operator= (const Base& rhs)
		{
		if (this != &rhs) // Check for self assignment before continuing.
		{
		release_handle(my_handle); // Release resource.
		my_handle = rhs.my_handle; // Assign members (only one member in class).
		}
		else
		{
		}
		*return *this;*
		}
			Could be a new category.

1.	6.12	**AV Rule 82 An assignment operator shall return a reference to *this.**	X EP: rather C++/Java-specific, in that assignments do not have results in all languages
1.	6.43	**AV Rule 89 A base class shall not be both virtual and non-virtual in the same hierarchy.**	X EP: a multi-inheritance style rule maybe covered in the new [BLP]
2.	6.43	**AV Rule 90 Heavily used interfaces should be minimal,**	X

		general and abstract.	EP: Style and maintenance issue
3.	6.43	**AV Rule 91 Public inheritance will be used to implement "is-a" relationships.**	X now covered in [BLP]
4.	6.43	**AV Rule 92 A subtype (publicly derived classes) will conform to the following guidelines with respect to all classes involved in the polymorphic assignment of different subclass instances to the same variable or parameter during the execution of the system:** **• Preconditions of derived methods must be at least as weak as the preconditions of the methods they override.** **• Postconditions of derived methods must be at least as strong as the postconditions of the methods they override.** **In other words, subclass methods must expect less and deliver more than the base class methods they override. This rule implies that subtypes will conform to the Liskov Substitution Principle.**	X now covered in [BLP]
5.	6.43	**AV Rule 93 "has-a" or "is-implemented-in-terms-of" relationships will be modeled through membership or non-public inheritance.**	X now covered in [BLP]
6.	6.43	**AV Rule 94 An inherited nonvirtual function shall not be redefined in a derived class.**	X EP:same issues as view conversion. By appling a different base-class op to a reference, consistency of the objects can be killed. Roughly covered in [BKK]
7.	6.43	**AV Rule 95 An inherited default parameter shall never be redefined.**	X EP:as 94 for picking the default value of a parameter (from the base, rather than the object type method); language bug? Ada has it, too.
8.	6.43	**AV Rule 96 Arrays shall not be treated polymorphically.**	X EP: very C++-specific; bug in language to allow polymorphic component types
9.	6.43, 6.53	**AV Rule 97 Arrays shall not be used in interfaces. Instead, the Array class should be used.**	X EP: very C++ specific; this is the index-check issue