Modeling Standards

For the assignment of the course "Software Architecting" (2II10) 2005/2006

ID	Name	Description	Category
1	Abstraction Level	Classes in the same package must be of the same abstraction level	Abstraction
2	Unique Names	Classes, packages and use cases must have unique names	Abstraction
3	Size of Use Cases	All use cases should cover a similar amount of functionality	Abstraction
4	Homogenity of Accessor Usage	When you specify getters/setters/constructors for a class, specify them for all classes	Balance
5	Homogenity of Visibility Usage	When you specify visibility somewhere, specify it everywhere	Balance
6	Homogenity of Method Specification	Specify methods for the classes that have methods! Don't make a difference in whether you specify or don't specify methods as long as there is not a strong difference between the classes.	Balance
7	Homogenity of Attribute Specification	Specify attributes for the classes that have attributes! Don't make a difference in whether you specify or don't specify attributes as long as there is not a strong difference between the classes.	Balance
8	Dynamic Classes	For classes with a complex internal behaviour, specify the internal behaviour using a state diagram	Completeness
9	Model Class Interaction	All classes that interact with other classes should be described in a sequence diagram	Completeness
10	Use Case Instantiation	Each Use Case must be described by at least one Sequence Diagram	Completeness
11	Specify Object Types	The type of ClassifierRoles (Objects) must be specified. (Which class in represented by the object?)	Completeness
12	Call Methods	A method that is relevant for interaction between classes should be called in a Sequence Diagram to describe how it is used for interaction.	Completeness
13	Role Names	ClassifierRoles (Objects) should have a role name	Completeness
14	Specify Message Types	Each message must correspond to a method (operation)	Consistency
15	No Abstract Leafs	Abstract classes should not be leafs (i.e. child classes should inherit from abstract classes)	Design
16	DIT at most 7	Inheritance trees should have no more than 7 levels	Design
17	Abstract-Concrete	Abstract classes should not have concrete superclasses	Design
18	High Cohesion	Classes should have high cohesion. Don't overload classes with unrelated functionality.	Design
19	Low Coupling	Your classes should have low coupling. (The number of relations between each class and other classes should be small)	Design
20	No Diagram Overload	Don't overload diagrams. Each diagram should focus on a specific concept/problem/functionality/…	Layout
21	No X-ing Lines	Diagrams should not contain crossed lines (relations)	Layout
22	Use Names	Classes, use cases, operations, attributes, packages, etc must have a name	Naming
23	Meaningful Names	Naming should use commonly accepted terminology, be non-ambigious and precisely express the function / role / characteristic of an element.	Naming