**1.1 Property Compartment**

The Property Pattern allows clients to expand properties of the state space of an object at runtime.These do not bring with them state-specific behavior. The clients treat the objects whose state space can be expanded, but according to these properties.

**Documentation**

The Figure 1.1 shows the Property Compartment class model. The data types in the model represent any data types that are not restricted here. They can be referenced more easily in method calls. There is an inheritance structure. This should not be seen as a limitation of the pattern and is only present so that the data types in method endings are the data type, the upper class can be specified. A separate parameter list of the addProperty method can be used for every possible data type.

The Compartment Type PropertyPattern allows the instantiation of the Design pattern. This class has methods for the clients' components.This happens through the addProperty method, where a client assigns a property with a specific value to an object of the GenericComponent class. This property could be generated by using a method CreateProperty.As the properties are just roles, this second method might create a role instance with a unique name that depends on the client. In addition, the handleComponent method is representative of a client's component treated according to their properties.

The design pattern also contains the class GenericComponent. Objects of this class

can play the roles of the role group Properties. Beyond this, all are the objects of assigned properties.

The role model is shown in Figure ????????. Here the properties of the roles can be found. There is an attribute in this model, which is the representative of the stored information of this property is available. It has a Role Group Constraint of 1 to infinity. This condition expresses that a component can have none, one or more properties.If a component does not have properties, then it does not play the role group.

**Figure 1.1 PropertyPattern Class Model**

**Evaluation**

This design pattern is only discussed by Dirk Riehle.

1. The design pattern allows the extension of a state space of a component at runtime.

2. The Components have any number of properties and each one represents a new part of the overall state.

3. The design pattern should be used if the exact state space is not is known.

4. The use of the design pattern makes it difficult to examine invariants on the State space of a component.

5. No specific behavior is stored in the properties.

6. The design pattern includes the following roles: Client, Component, and Property.

7. The component role manages its properties via a relation to the role Property.

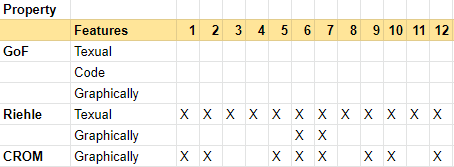
8. The Component role deletes its properties when it is deleted itself.

9. The client can generate properties.

10. The client can change the state space of a component by adding a Expand property.

11. The client treats the component according to its state space.

12. The composition constraints of the Role Relationship Matrix [Rie09, p. 19] is applied.



**Figure 1.3 Property Pattern features in 3 representations**

The characteristic 7 represents in the CROM that the GenericComponent class plays the role of Properties of the role group. The role component functionalities is expressed by the GenericComponent class. The client captures the features 6th, 9th and 10th through the methods of the PropertyPattern class. The presentation by the CROM is violated by the composition determined by Riehle constraints. A Component is allowed to play the role of properties.

?????/ Last Paragraph