**1.1 Iterator Compartment**

The iterator pattern allows access to a collection of objects without the need for the client knowledge about the structure of the collection. The iterator objects can encapsulate various traversal algorithms, which change at runtime.A client can have multiple iterator objects in different ways traversing as a collection at the same time.

**Documentation**

The class diagram contains the data types GenType and Command. The data type GenType represents an unspecified Data types whose objects are to be stored in the collections of the design pattern. Command is needed to represent an internal iterator. This is done by a Clients called the for All method on the Compartment Type IteratorPattern. When input parameters include an object of type Command, then it saves in a string from a command. The iterator that calls this method uses the method parseText of the data type and thus determines which operation on all elements of a collection should be executed.

The class IteratorPattern also allows clients to use external iterators and cursors. The methods that belong to these subsets of iterators are in the role diagram in Figure ????. Almost all methods use this class as input parameter, a pointer to an object of the class GenericIterator. This helps which iterator to work on in the design pattern.Thus, a collection can have several iterators, which have the same names. The method ColOperation is representative of the possibility that the client also directly call the methods of the collections and thus without detour over the iterator on the elements of the collection.

The GenericIterator class can handle the iterator roles. In the class model, the GenericCollection class stores the objects of type GenType and makes accessible. This is done in the model by the items attribute and the method represented as colOperation. The class can play the role of Collection.

This role model implements the function createIterator, which creates a role instance of the Iterator role and a class that plays that role. The generated role instance is referenced by the collection. This method in the role Collection is heavily influenced by the available iterator roles, not in GenericCollection class.

These roles include the iterator and the roles of the concrete iterators Role group ConcreteIterators are summarized. The role Iterator gives the interface for his concrete counterparts. This includes the reference to the first element of a collection or a Algorithm that determines the next element of this. The Gang of Four calls for the methods which belong to a minimal interface of an external iterator. The additional methods will be in the model mentioned otherOperations. The role Iterator has internal iterators the method forAll.

In the role group ConcreteIterators, the roles of the concrete iterators are shown. These are also described by the Gang of Four.There are of course several variants of each in the design pattern which can create subspecies. The ExternalIterator role represents an external iterator while InternalIterator detects the internal iterator. The scroll cursor is the third subspecies of the iterators in the model. The cursor does not have its own algorithms to

iterate over the collection. It stores only the current position in the collection. This can be made externally by the method getCurrentItem.

In the instantiation of the design pattern, there is at least one class which the role collection takes over. But a collection in the design pattern does not have to be immediately

be assigned the iterator role. That's why the iterator roles do not have to be played at all times. From this, the occurrence constraints can be derived.

Since the Iterator role should have an abstract character, so there should be no class, which plays only this role. There is a role equivalence between this and associated role group. This leads to a Role Group Constraints class that inherits the role of iterator just as exactly as a role from the role group which plays the ConcreteIterators.

**Figure 1.1 IteratorPattern Class Model**

**Evaluation**

1. The Compartment allows serial access to elements of a collection, without the client having the knowledge of the structure. This feature is also available in the Part.

2. It allows you to create a collection in several ways at the same time. It is traversing and changing the traversal type at runtime.

3. The design pattern allows the type of traversal through a uniform collection.

4. An iterator object has the capability to traverse through the the collections which is inherited by the object.

5. The collection object creates the concrete iterator.

6. This contains the following classes: Aggregates, ConcreteAggregate, Ite-rator and ConcreteIterator.

7. The class Aggregates is abstract and defines the interface for the concrete collections.

8. The class ConcreteAggregate inherits from the class Aggregates.

9. The class Iterator is abstract and defines the interface for the concrete iterators.

10. The class ConcreteIterator inherits from the class Iterator.

11. The ConcreteIterator class interacts with the collection to traverse it.

12. The ConcreteIterator class stores the current position of the iterator on the Collection.

13. A client can interact with the Collection and the iterator.

14. The minimal interface of the iterator contains the methods First, Next, IsDone and

CurrentItem. This can be extended by additional methods.

15. The class ConcreteAggregate implements the interface of the class Aggregates.

16. The ConcreteIterator class implements the interface of class Iterator.

17. New traversal types can be easily implemented without the collection object to change.

18. The design pattern allows simple classes of collections.

19. Iterators can be distinguished according to which class the traversal triggers.

a) In the case of an external iterator, the client individually calls methods of the iterator to traverse on. This solution is very flexible.

b) An internal iterator automatically passes an operation passed to each element a collection. This solution is easier to use.

20. An iterator can also be used as a cursor that only shows the current position in the collection stores. The algorithms are traversing to the collection.

21. A cursor should be used if the traversal algorithm is private Variables of the collection depends.

22. An iterator cav give elevated rights to the collection on his private Get variables. Methods with these accesses should not be excluded, but must be callable.

23. A robust iterator ensures that traversing is not caused by changes the collection is disturbed. The collection supports the iterator.

24. For recursive collections, such as tree structures, the use of external Iterators are difficult, which makes cursor and internal iterators useful.

25. An iterator that is always traversed can traverse tree structuresand can be used with leaf nodes.

26. The design pattern includes the following roles: iterator, client, aggregates and element.

27. The client role interacts with all other roles.

28. The role of iterator interacts with the roles aggregates and element.

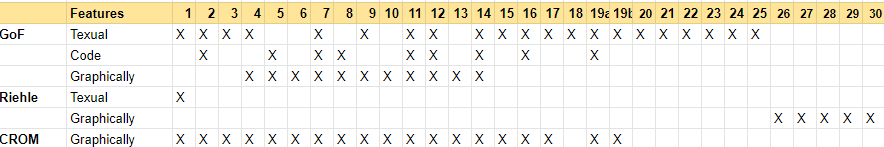
29. The role aggregates stores its elements.

30. The composition constraints of the Role Relationship Matrix [Rie09, p. 38] apply.

As per the principle, the feature 6 is partially represented by roles.The aggregate classes are represented by the GenericCollection class. The Methods from feature 14 are renamed. The features 2nd through 4th and 17th will be through the separation of the role of collections and the iterator roles. In feature 10, the inheritance structure is realized a role, a role group and their role equivalence. The class GenericCollection is concrete as. So the feature 8 will not be violated.Subspecies of collections are their abstract superclasses.

All features that are mentioned exclusively by Dirk Riehle can be described with the CROM. The role of the client is not explicitly modeled because of the capabilities those are listed in the IteratorPattern class. In addition, the role of Elements replaced by an attribute in the Collection class.

Almost all features not described are in the constituent implementation in the presentation of the Gang of Four. Dirk Riehle, on the other hand, mainly uses a graphical representation for description of the design pattern.

****

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