How to build a set of DSLs: from Theory to Practise

Xactium, TUBS, Jendrik Johannes (TUD)

Context of this work

- The present courseware has been developed within the context of the MODELPLEX European IST FP6 project (http://www.modelplex.org/).
- Co-funded by the European Commission, the MODELPLEX project involves 21 partners from 8 different countries.
- MODELPLEX aims to define and develop a coherent infrastructure specifically for the application of MDE to the development and subsequent management of complex systems within a variety of industrial domains.
- To achieve the goal of the large-scale adoption of MDE, MODELPLEX promotes the idea of the collaborative development of courseware dedicated to this domain.
- The MDE courseware provided here with the status of open-source software is produced under the EPL 1.0 licence.

Outline

- Introduction to DSLs
  - What are DSLs? (overview, with examples
  - Why are they topical? (Discussion of drivers for DSLs)
  - Who is using them? (Examples of use)
  - What is the business benefit? (The value proposition to industry)
  - Relationship within Modelplex (how they contribute to Modelplex)
  - Existing technologies (Eclipse, MS, etc)

- Architecture of a DSL
  - Core concepts (Define core concepts and terminology)
  - Structure of DSLs (Basic building blocks: abstract syntax, etc)
  - Measurements of quality (How to determine the quality of the DSL – may be a section in it own right)

Outline Continued

- Building a DSL
  - Domain analysis (Understanding the domain that is to be modelled)
  - Building an abstract syntax model (Basic steps in building a metamodel)
  - Understanding syntax (What should the interface to a DSL be?)
  - Developing a visual editor (Steps in creating a visual editor)
  - Other types of editors (For example textual)
  - Semantics (Giving the DSL meaning)
  - Modularisation for DSLs (adding modularisation support)
- An example (Take them through a small example using Eclipse)
- Building a DSL family (Examples of how to facilitate reusable DSL development)

Building a DSL: Modularisation

- Languages need modularization concepts
  - Reduce complexity
  - Improve reusability
- Challenges
  - Modularization influences syntax and semantics
  - Requires additional tooling support
- Reuseware [1][2]
  - Does not influence design of DSL syntax or semantics
    - DSL syntax can be extended at the end (but does not have to be)
  - Composes modularized models to monolithic models
    - DSL semantics do not require extension
  - Generic tooling can be used with arbitrary DSLs

[1] On Language-Independent Model Modularisation,Transactions on Aspect-Oriented Development, 2008
[2] http://reuseware.org

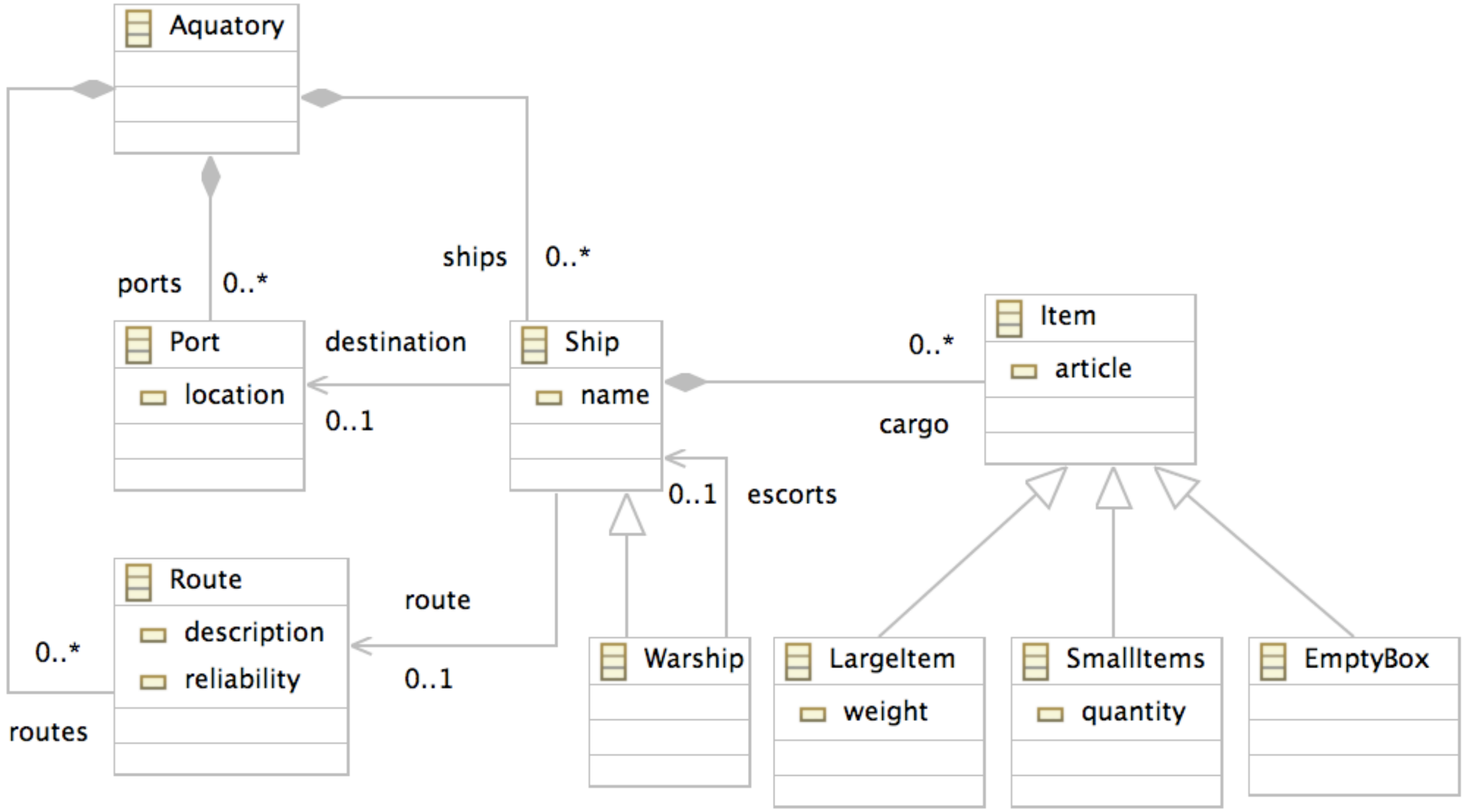
Building a DSL: Modularisation with Reuseware

- Reuseware approach
  - Define a composition system with modularisation concepts (or reuse a predefined one)
    - E.g., Modules, Packages, Aspects, etc.
  - Optional: Extend DSL syntax with concepts for variation points
    - Variation points allow definition of templates
  - Define a reuse extension for your DSL
    - Binds the composition system to your DSL
    - E.g., what are the specifics of a module in your DSL, what identifies and aspect, etc.

  - Reuseware can now handle modularization in your DSL

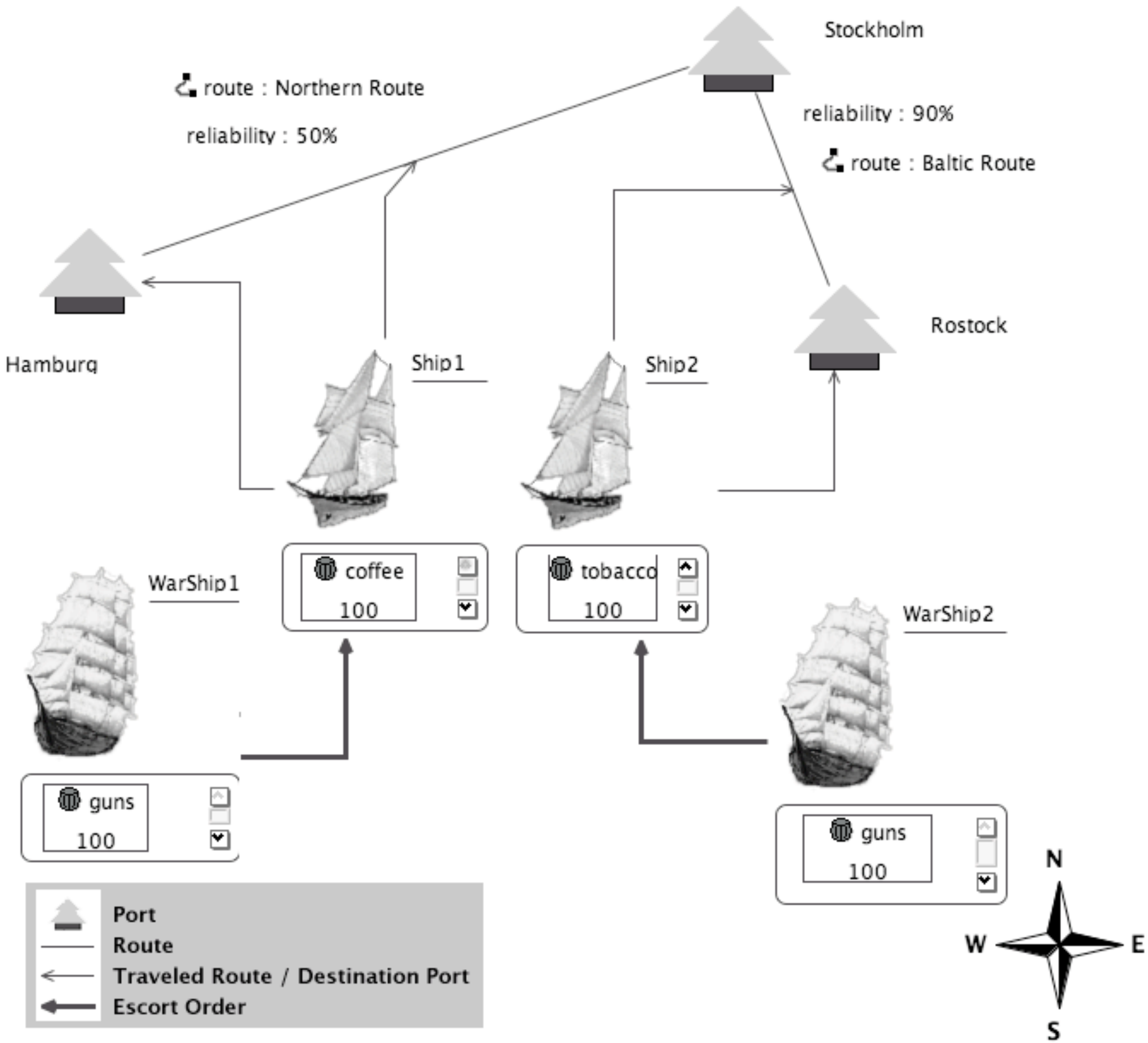
Building a DSL: Modularisation – Example

- Taipan DSL[3] (Metamodel excerpt)

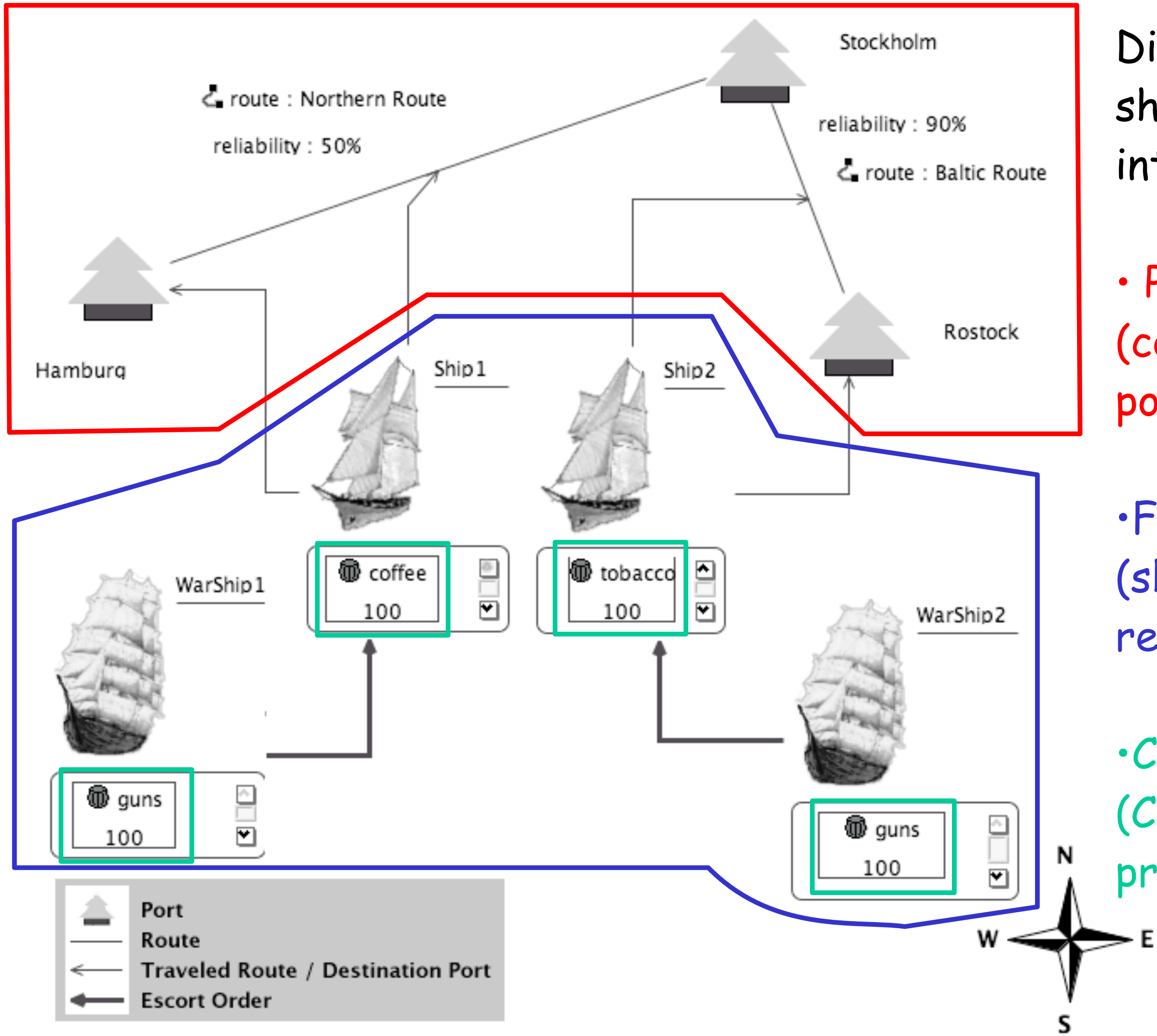


[3] http://wiki.eclipse.org/index.php/GMF Tutorial#Quick Start

Building a DSL: Modularisation – Example



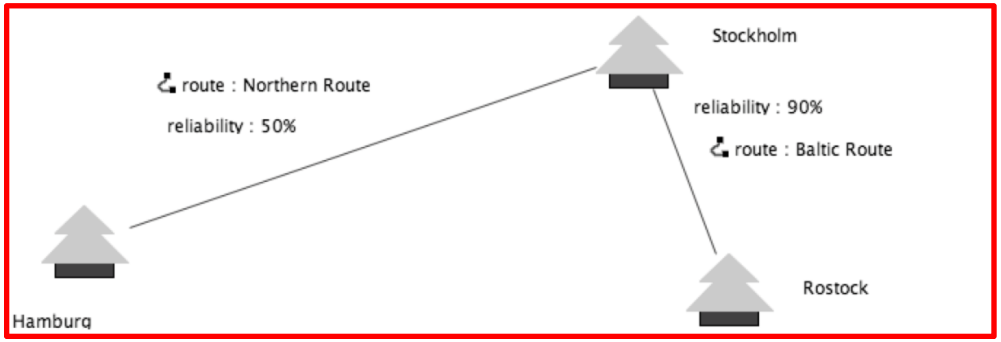
Building a DSL: Modularisation – Example

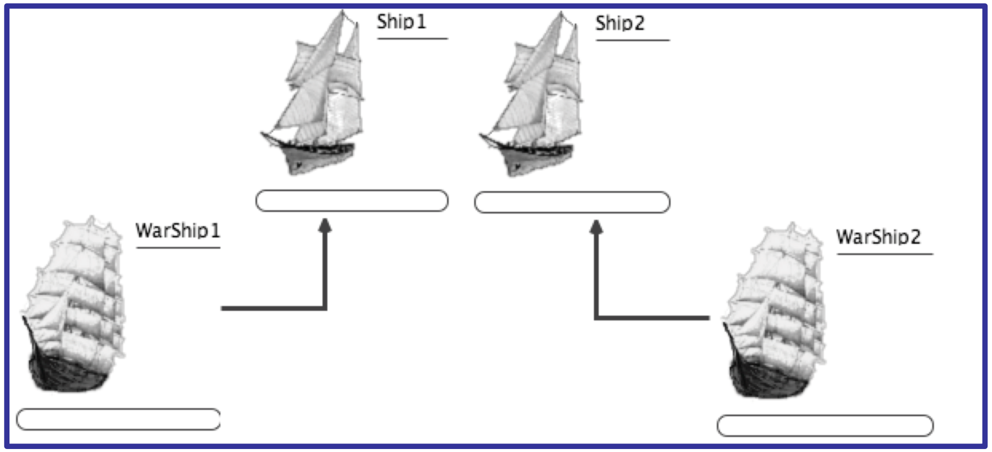


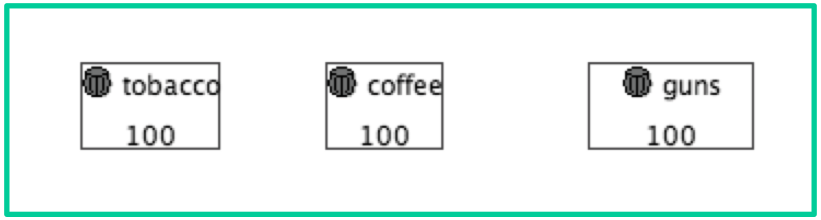
Different concerns should be separated into model fragments

- Port mode (configuration of ports and routes)
- Flotilla model (ships and their relations)
- Cargo model (Cargo and its properties)

Building a DSL: Modularisation – Example







Different concerns should be separated into model fragments

- Port mode (configuration of ports and routes)
- Flotilla model (ships and their relations)
- Cargo model (Cargo and its properties)

Building a DSL: Reuseware - Overview

- Model Fragments
  - (Partial) models that may contain variation points
  - Offer a **Composition Interface**
  - **Composition Interface** consists of **Ports**
  - **Ports** point at elements of the model fragment that can be accessed for composition
- Composition Programs
  - Define **composition links** between Ports
  - Can be executed to produce a composed model where model fragments are merged at the elements pointed out by the linked Ports

Building a DSL: Reuseware - Overview

- Composition Systems
  - Define modularisation concepts (e.g., Modules, Packages, Aspects)
  - Define relations between modularisation concepts (e.g, an aspect relates to a core)
- Reuse extensions (for DSLs)
  - Define how modularization concepts defined in a composition system are realized in a concrete DSL
  - Define which ports are related to which model elements of a model fragment

Building a DSL: Reuseware - Composition Systems

- A composition system defines
  - Fragment roles
    - Role a model fragment plays in the modularisation (e.g., aspect or core)
    - Fragment roles collaborate through associations between ports
  - Static ports
    - Defined for one fragment role
    - Each fragment playing the role has to offer the port
  - Dynamic ports
    - Defined for one fragment role
    - Each fragment playing the role can offer several of these ports
  - Contribution Associations
    - Defines that two ports are related
    - Executing a composition link between the two ports will trigger the copying of model elements
  - Configuration Associations
    - Defines that two ports are related
    - Executing a composition link between the two ports will NOT trigger the copying of model elements

Building a DSL: ReuseTaipan - a Composition System

```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

Building a DSL: ReuseTaipan - a Composition System

```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

A TravelSpace offers a place where vehicles can be placed (VehicleContainer) and a number of Routes and Places

Building a DSL: ReuseTaipan - a Composition System

```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

A Flotilla offers a set of Vehicles and has a number of placeloders for routes (RouteSlots) and places (PlaceSlots)

Building a DSL: ReuseTaipan - a Composition System

```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

A Flotilla contributes Vehicles to a TravelSpace's VehicleContainer; a RouteSlots can be configured with a Route; a PlaceSlots can be configured with a Place

Building a DSL: ReuseTaipan - a Composition System

```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

An ItemHolder offers different ItemSpaces

Building a DSL: ReuseTaipan - a Composition System

```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

An ItemContainer contains and offers Items

Building a DSL: ReuseTaipan - a Composition System

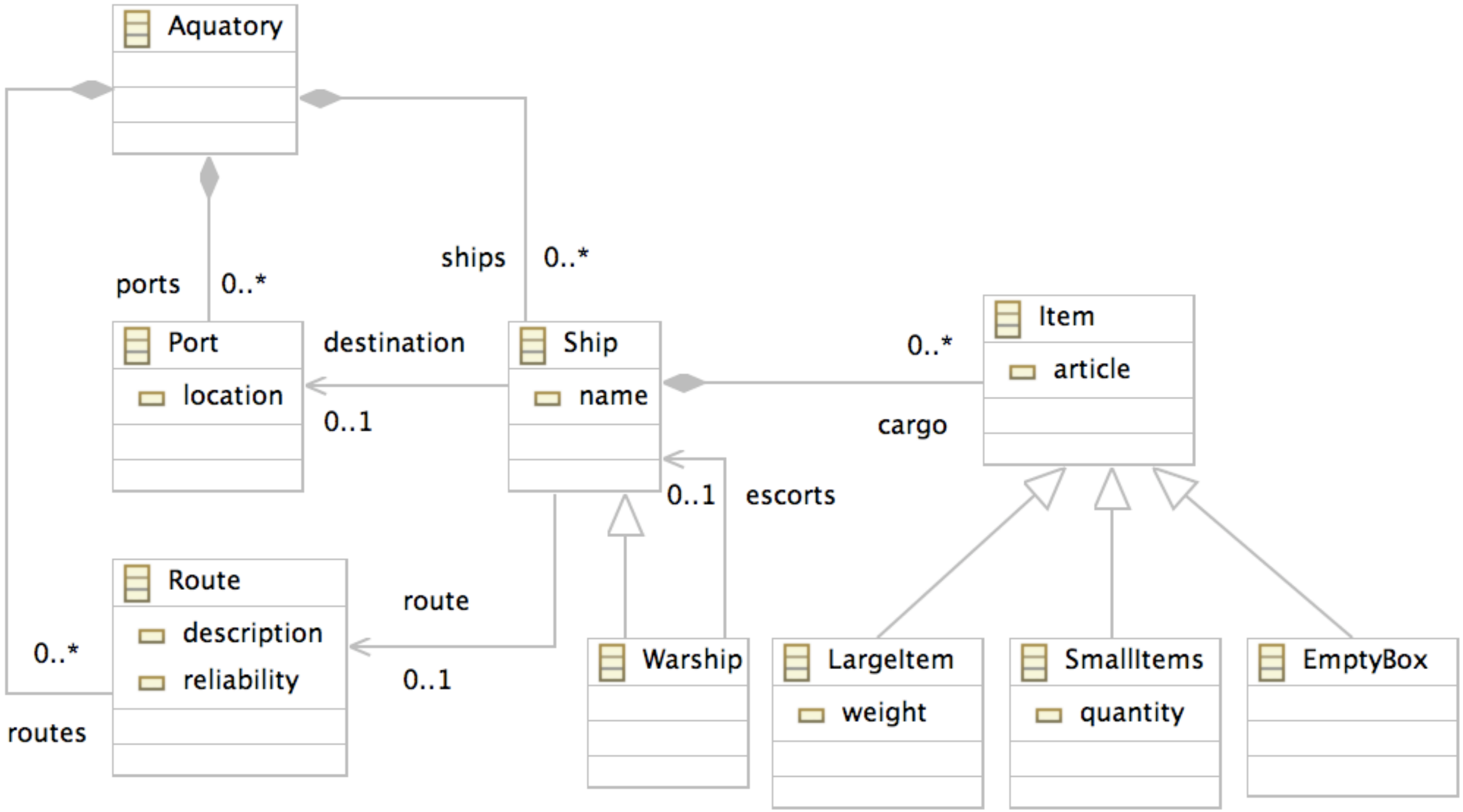
```
compositionsystem reuseTaipan {

  fragment role TravelSpace {
    static port VehicleContainer;
    dynamic port Routes;
    dynamic port Places;
  }

  fragment role Flotilla {
    static port Vehicles;
    dynamic port RouteSlots;
    dynamic port PlaceSlots;
  }

  contribution Flotilla.Vehicles --> TravelSpace.VehicleContainer;
  configuration Flotilla.RouteSlots --> TravelSpace.Routes;
  configuration Flotilla.PlaceSlots --> TravelSpace.Places;


  fragment role ItemHolder {
    dynamic port ItemSpaces;
  }

  fragment role ItemContainer {
    dynamic port Items;
  }

  contribution ItemContainer.Items --> ItemHolder.ItemSpaces;
}
```

Items can be individually assigned to ItemSpaces

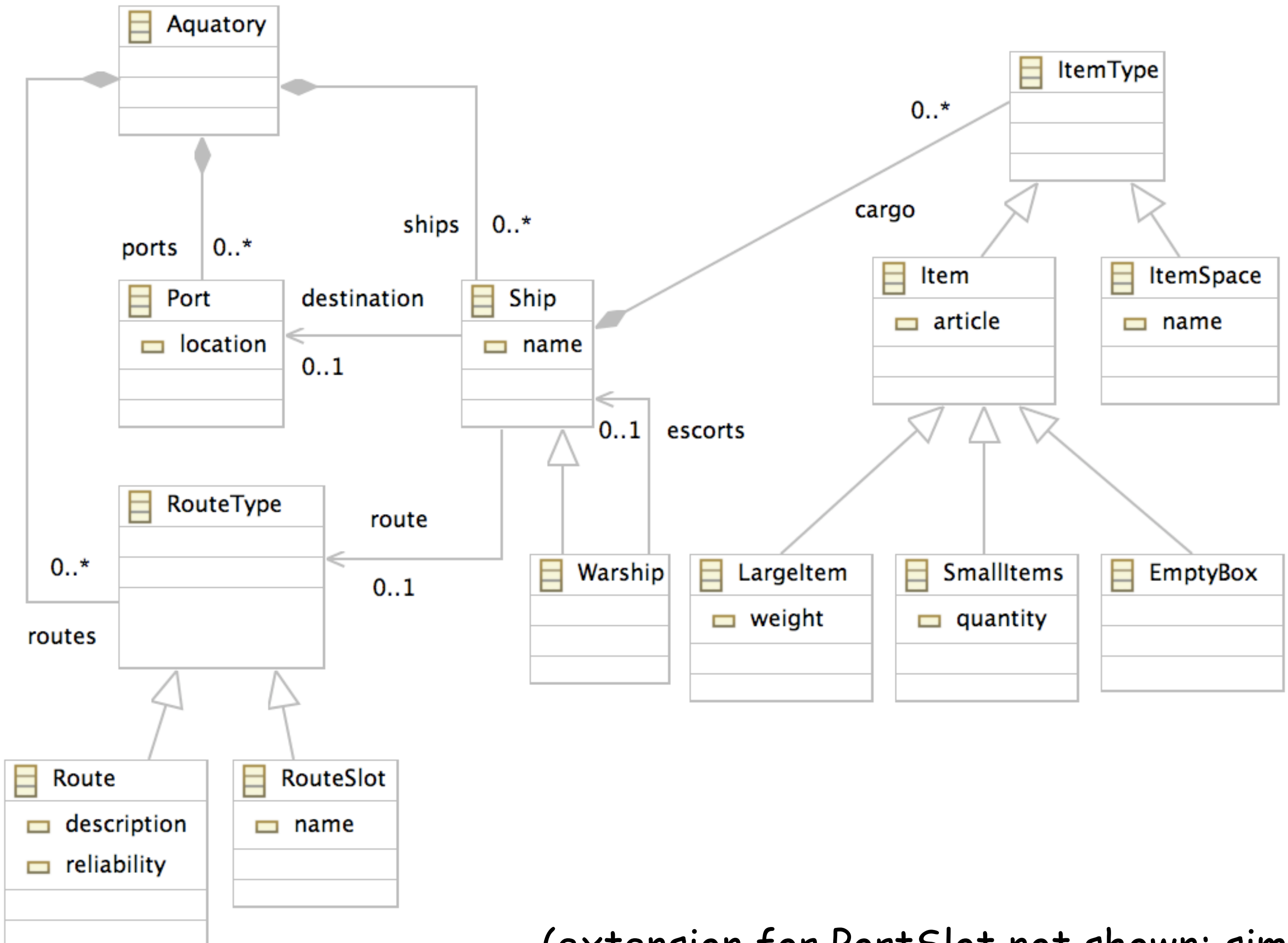
Building a DSL: Extending a Metamodel for Variation

- Three kinds of variation points required
  - RouteSlot
  - PortSlot
  - ItemSpace
- For each kind of variation point we...
  - Introduce a superclass for the metaclass that defines the elements which may replace the variation point (e.g., we introduce **RouteType** as a superclass of **Route** in the case of RouteSlot)
  - We redirect all references to the metaclass to the new superclass (e.g., all references to **Route** are redirected to **RouteType**)
  - We introduce a new subclass for the just introduced superclass that represents the variation point. This class needs properties from which a name can be derived. (e.g., we introduce **RouteSlot** as a subclass of **RoutType**)

Building a DSL: Extending a Metamodel for Variation



Building a DSL: Extending a Metamodel for Variation



(extension for PortSlot not shown; similar to R

Building a DSL: Reuseware - Reuse Extensions

- A Reuse Extension defines
  - How a composition interface define by a fragment role (which is defined in a composition system) is linked to the content of a model fragment
  - Each port links to a set of model elements treated as:
    - **Prototype**: Element that can be copied with its contained elements
    - **Anchor**: Element that can be referenced by other elements
    - **Hook**: Variation point where Prototypes can be put
    - **Slot**: Variation point where Anchors can be put

Building a DSL: Binding ReuseTaipan to Taipan DSL

```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
        port expr = $self.name$
```

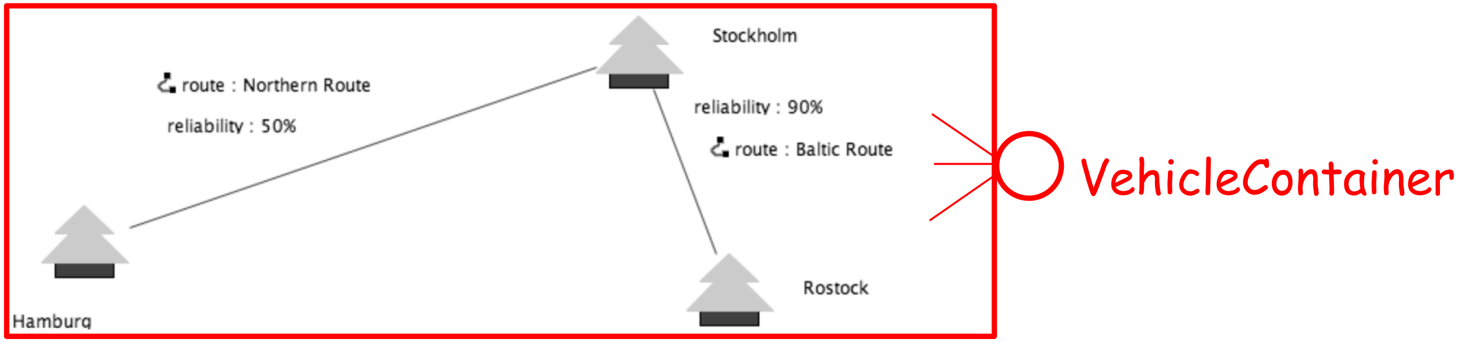
The ReuseTaipan composition system is bound to the Taipan DSL (referred to by the URI of its metamodel)

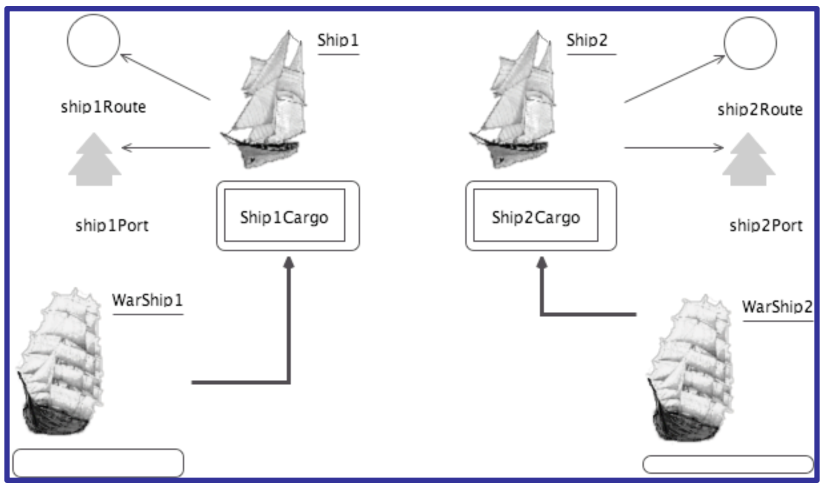
Building a DSL: Binding ReuseTaipan to Taipan DSL

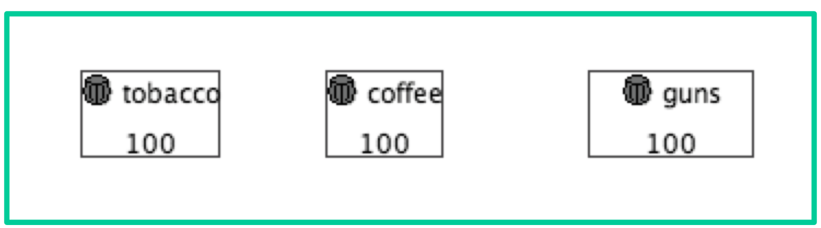
```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
```

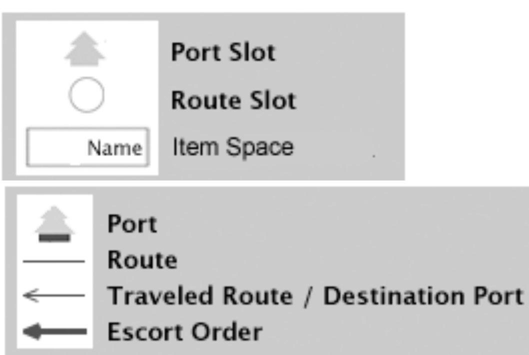
The references ships, ports and routes of the metaclass Aquatory all act as hooks accessible through the VehicleContainer port

Building a DSL: Binding ReuseTaipan to Taipan DSL







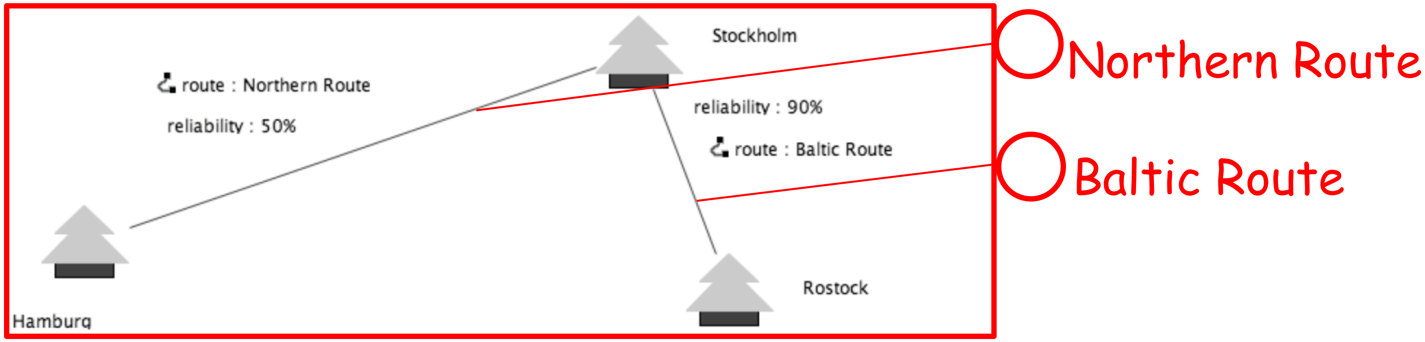


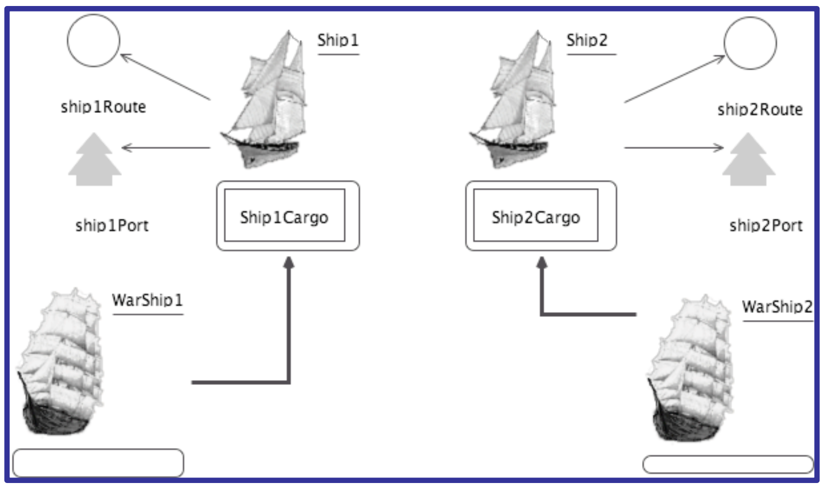
Building a DSL: Binding ReuseTaipan to Taipan DSL

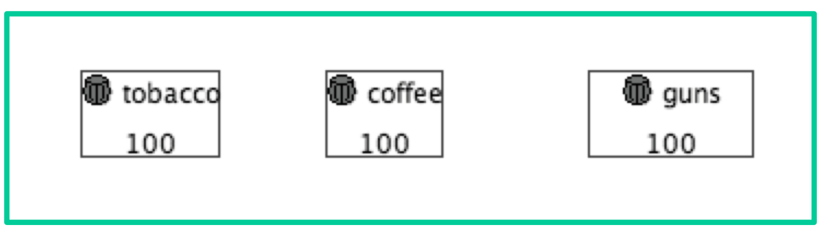
```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
        port expr = $self.name$
```

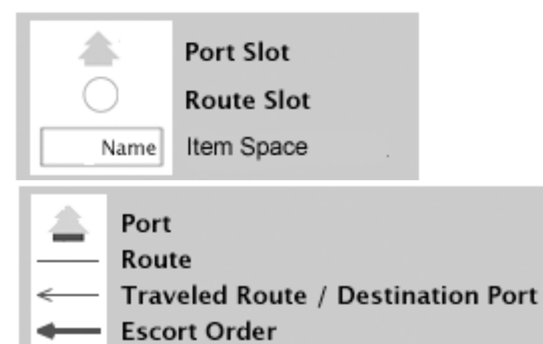
Each Route is an anchor accessible through individual ports; the ports are named using the description attribute of the Route metaclass (OCL Expression: self.description)

Building a DSL: Binding ReuseTaipan to Taipan DSL







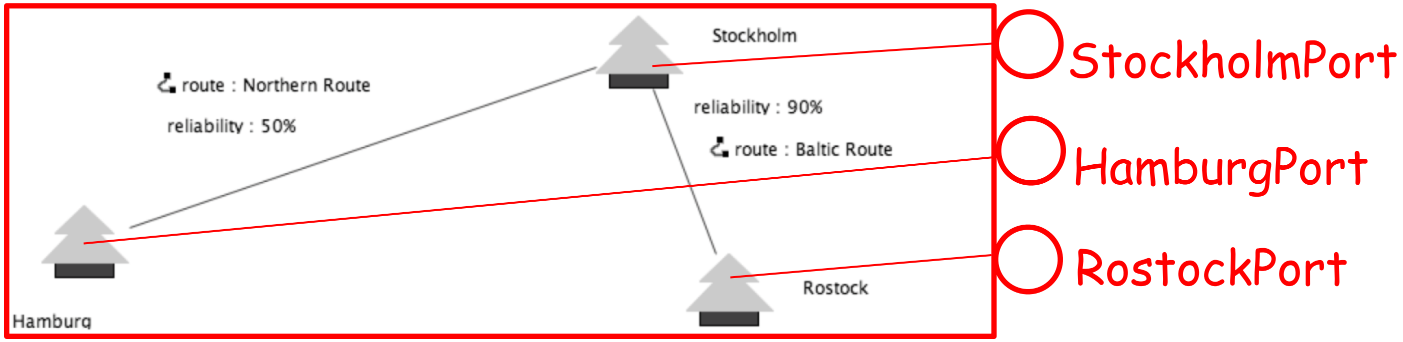


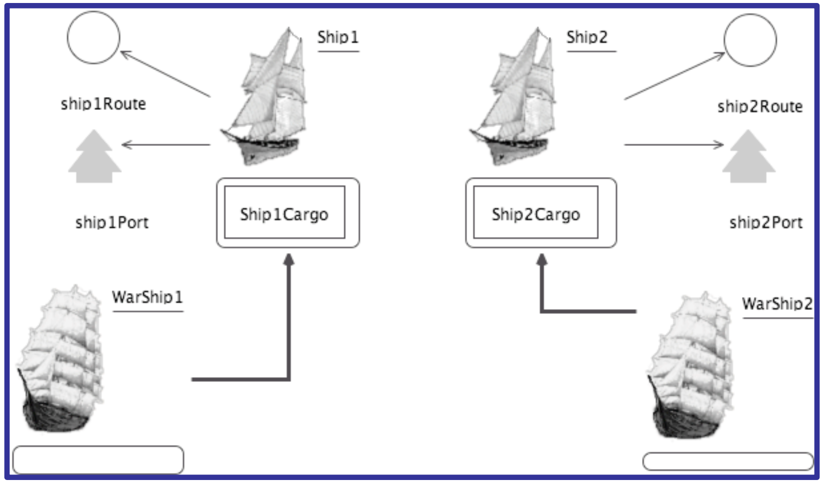
Building a DSL: Binding ReuseTaipan to Taipan DSL

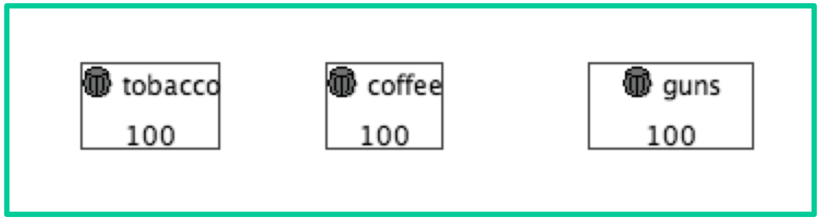
```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
        port expr = $self.name$
```

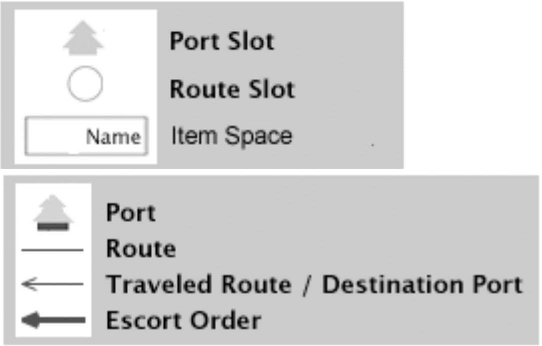
Each Port is an anchor accessible through individual ports; the ports are named using the location attribute of the Port metaclass

Building a DSL: Binding ReuseTaipan to Taipan DSL







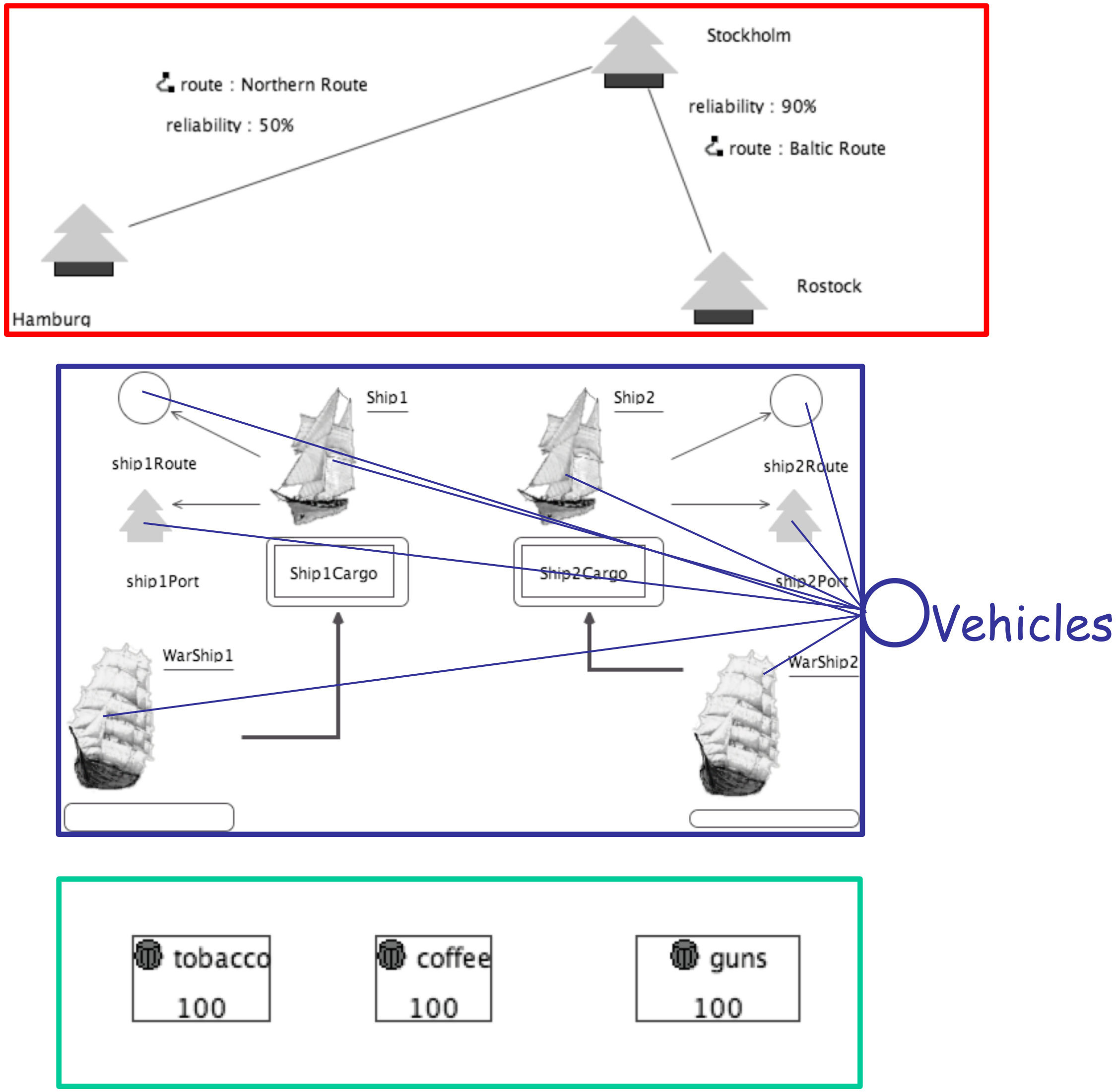


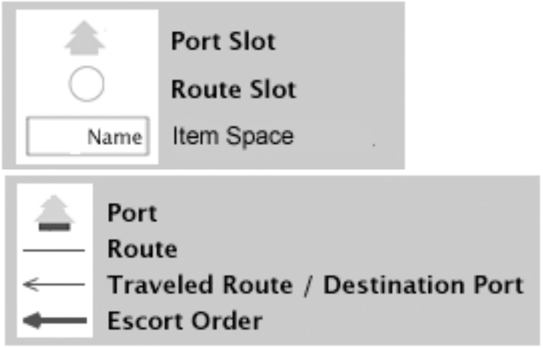
Building a DSL: Binding ReuseTaipan to Taipan DSL

```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
```

All elements of the references ships, ports and routes of the metaclass Aquatory act as prototypes accessible through the Vehicles port

Building a DSL: Binding ReuseTaipan to Taipan DSL



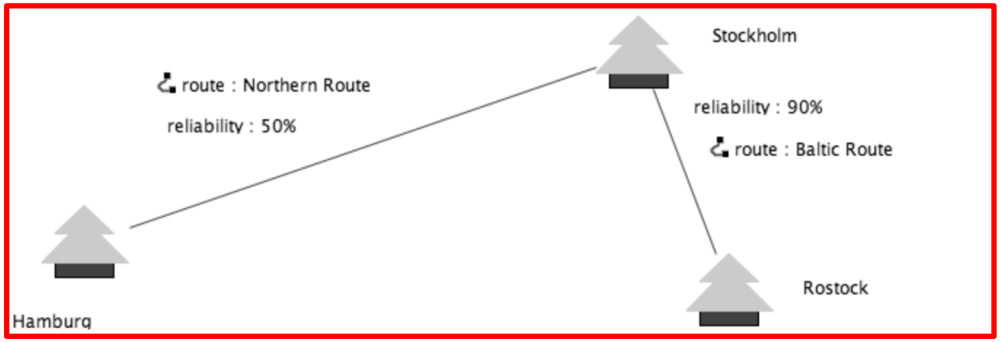


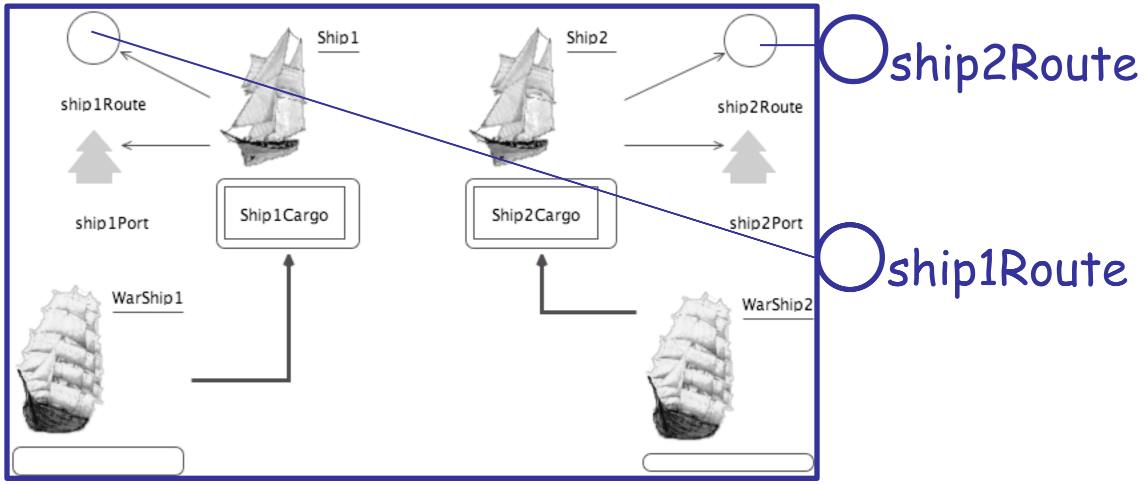
Building a DSL: Binding ReuseTaipan to Taipan DSL

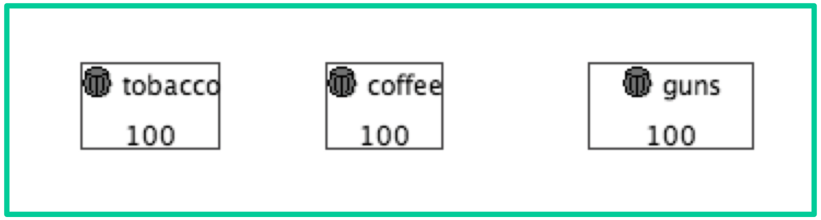
```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
        port expr = $self.name$
```

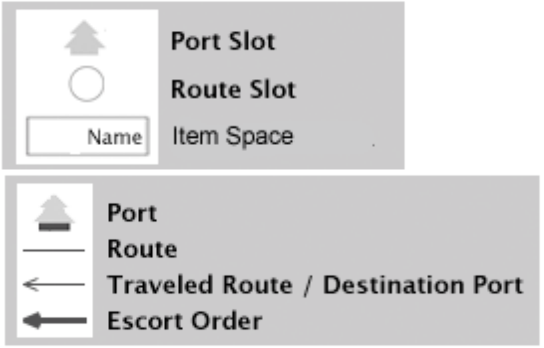
Each RouteSlot is a slot accessible through individual ports; the ports are named using the name attribute of the RouteSlot metaclass

Building a DSL: Binding ReuseTaipan to Taipan DSL







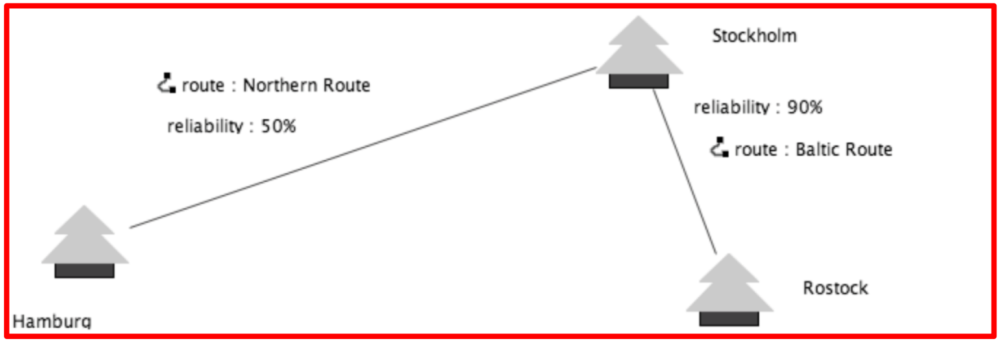


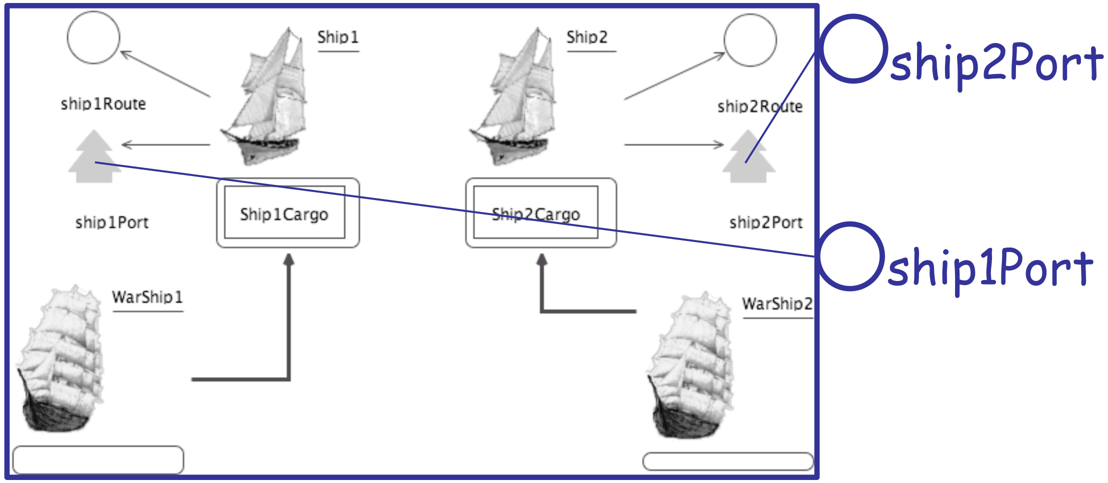
Building a DSL: Binding ReuseTaipan to Taipan DSL

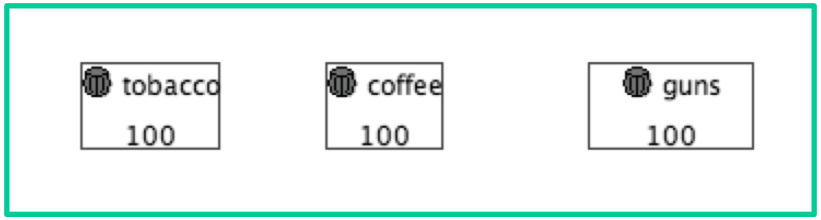
```
reuseextension reuseTaipan implements reuseTaipan
epackages <http://www.eclipse.org/examples/gmf/taipan>
Rootclass TravelSpace {
  fragment role TravelSpace {
    port VehicleContainer {
      Aquatory.ships is hook {}
      Aquatory.ports is hook {}
      Aquatory.routes is hook {}
    }
    port Routes {
      Route is anchor {
        port expr = $self.description$
      }
    }
    port Places {
      Port is anchor {
        port expr = $self.location.concat('Port')$
      }
    }
  }

  fragment role Flotilla {
    port Vehicles {
      Aquatory.ships is prototype {}
      Aquatory.ports is prototype {}
      Aquatory.routes is prototype {}
    }
    port RouteSlots {
      RouteSlot is slot {
        port expr = $self.name$
      }
    }
    port PlaceSlots {
      PortSlot is slot {
        port expr = $self.name$
```

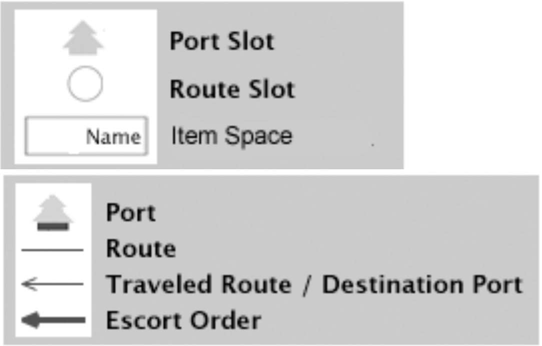
Each PortSlot is a slot accessible through individual ports; the ports are named using the name attribute of

Building a DSL: Binding ReuseTaipan to Taipan DSL







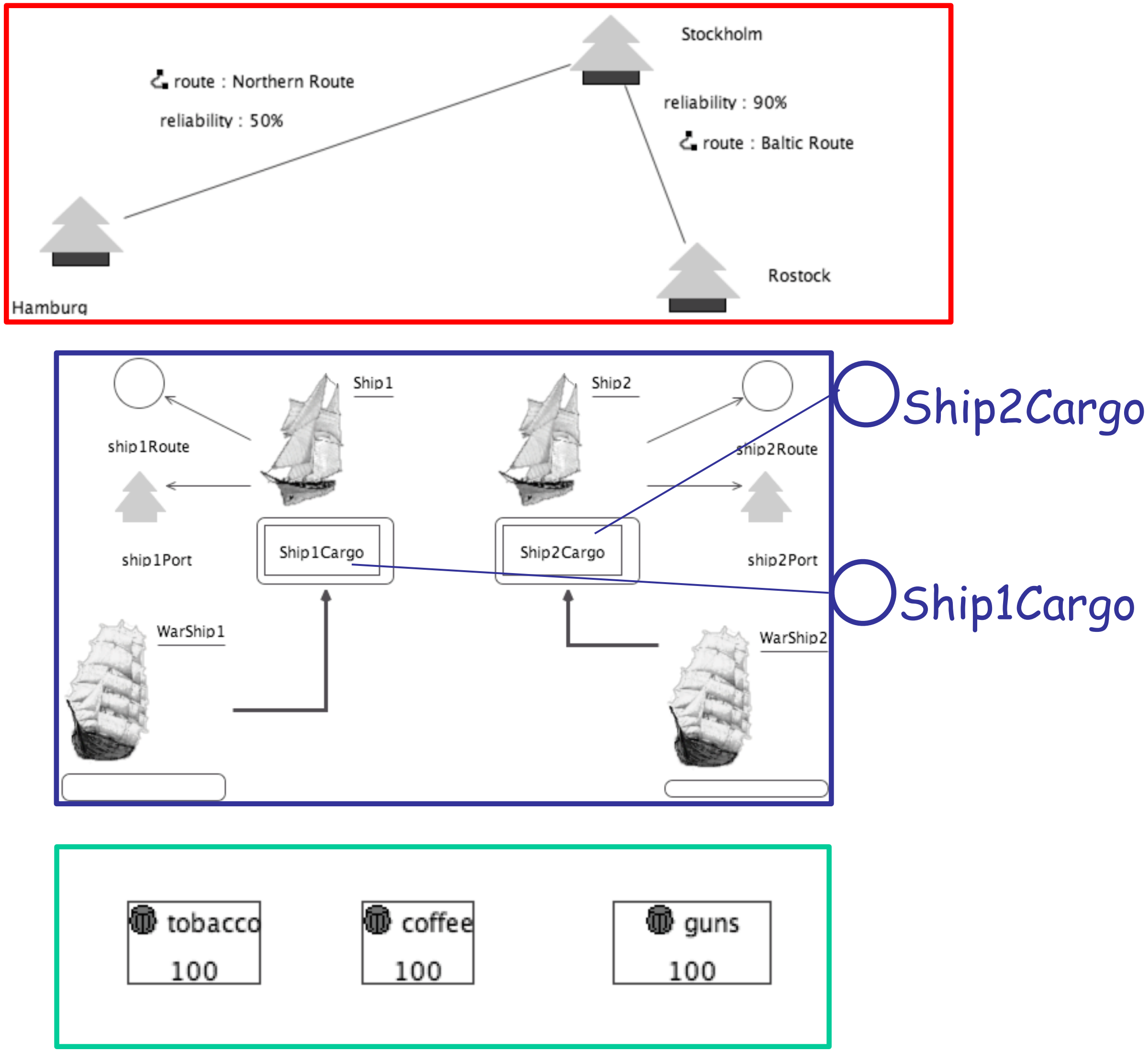


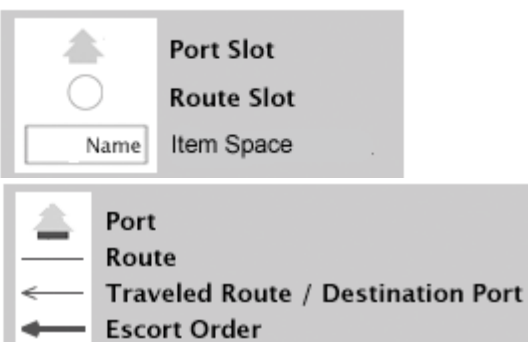
Building a DSL: Binding ReuseTaipan to Taipan DSL

```
  ...

  binding ItemHolder {
    binding ItemSpaces {
      ItemSpace is hook {
        port expr = $self.name$
      }
    }
  }

  binding ItemContainer {
    binding Items {
      Item is prototype {
        port expr = $self.article$
      }
    }
  }
}
```

Each ItemSpace is a hook accessible through individual ports; the ports are named using the name attribute of the ItemSpace metaclass

Building a DSL: Binding ReuseTaipan to Taipan DSL



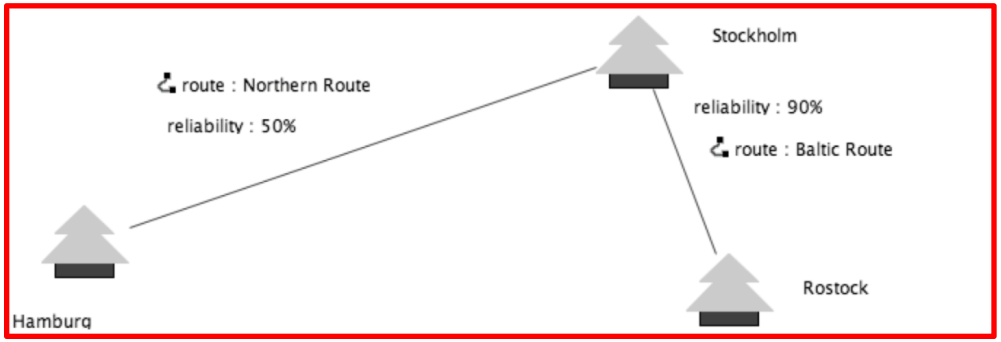


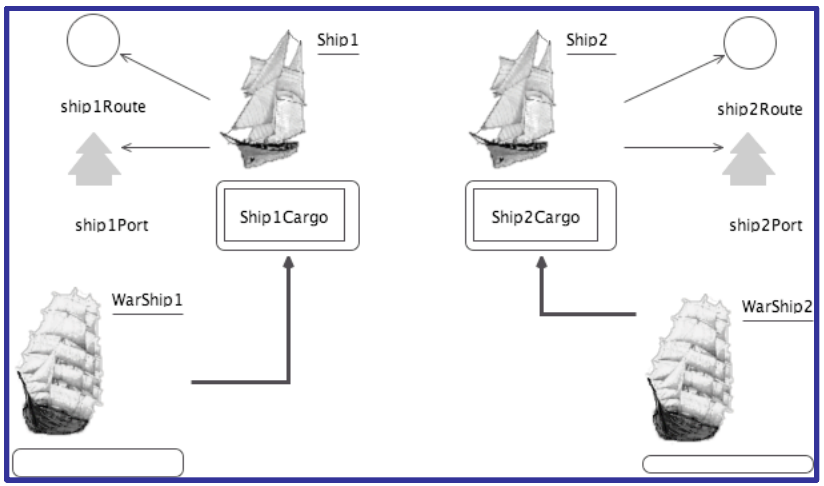
Building a DSL: Binding ReuseTaipan to Taipan DSL

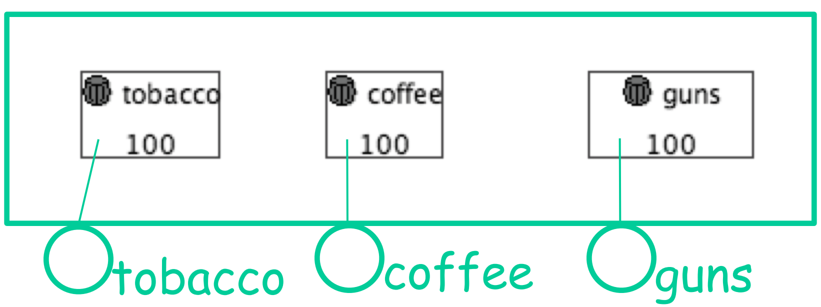
```
  ...

  fragment role ItemHolder {
    port ItemSpaces {
      ItemSpace is hook {
        port expr = $self.name$
      }
    }
  }

  fragment role ItemContainer {
    port Items {
      Item is prototype {
        port expr = $self.article$
      }
    }
  }
}
```

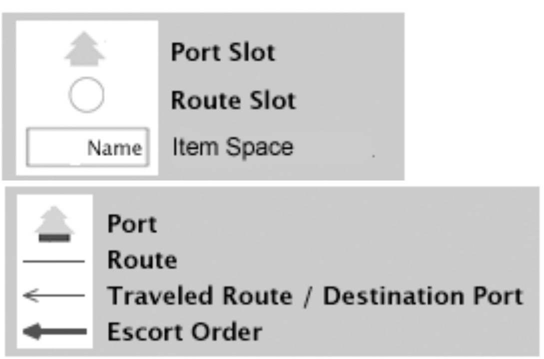
Each Item is a prototype accessible through individual ports; the ports are named using the article attribute of the Items metaclass

Building a DSL: Binding ReuseTaipan to Taipan DSL





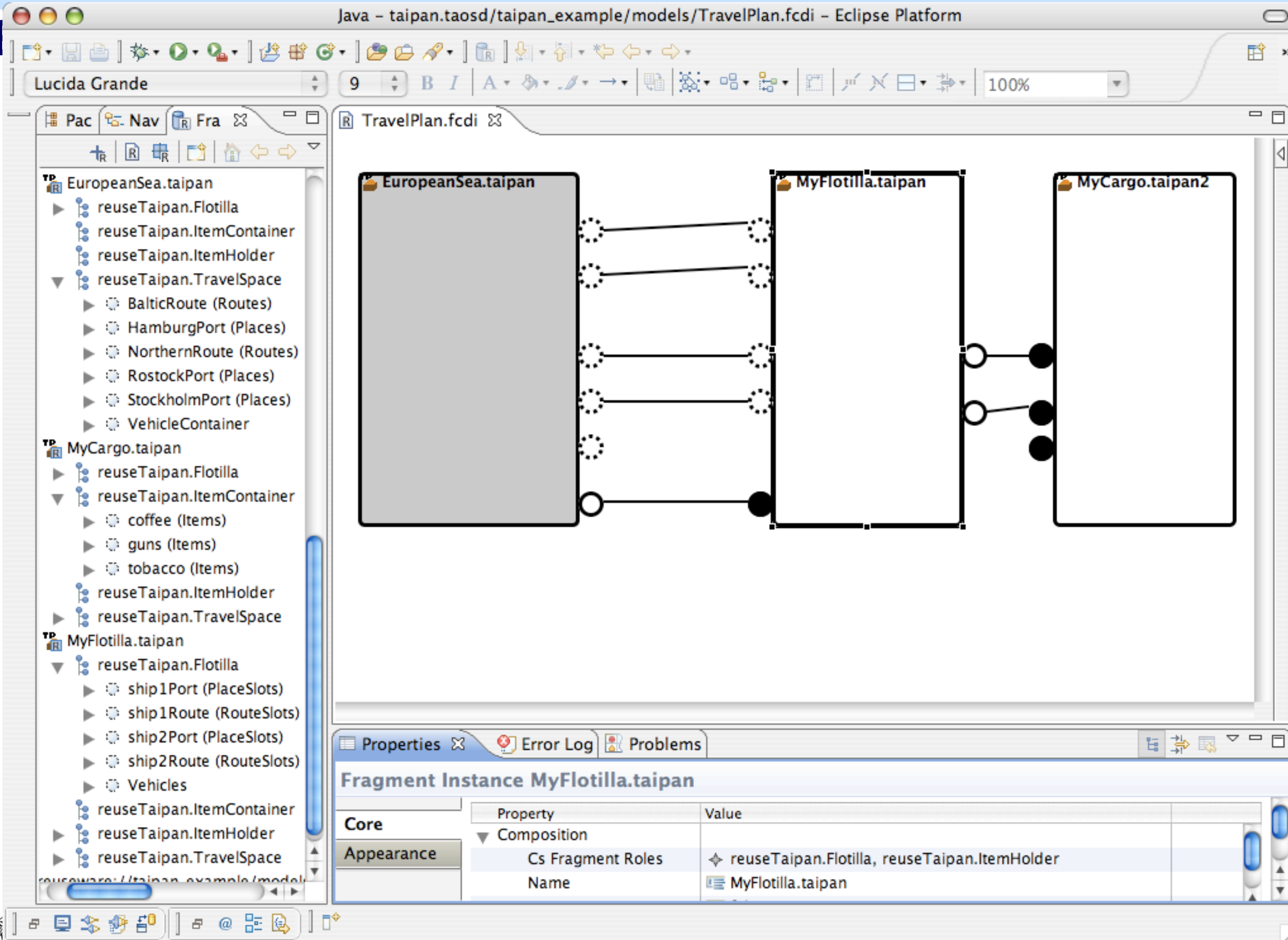


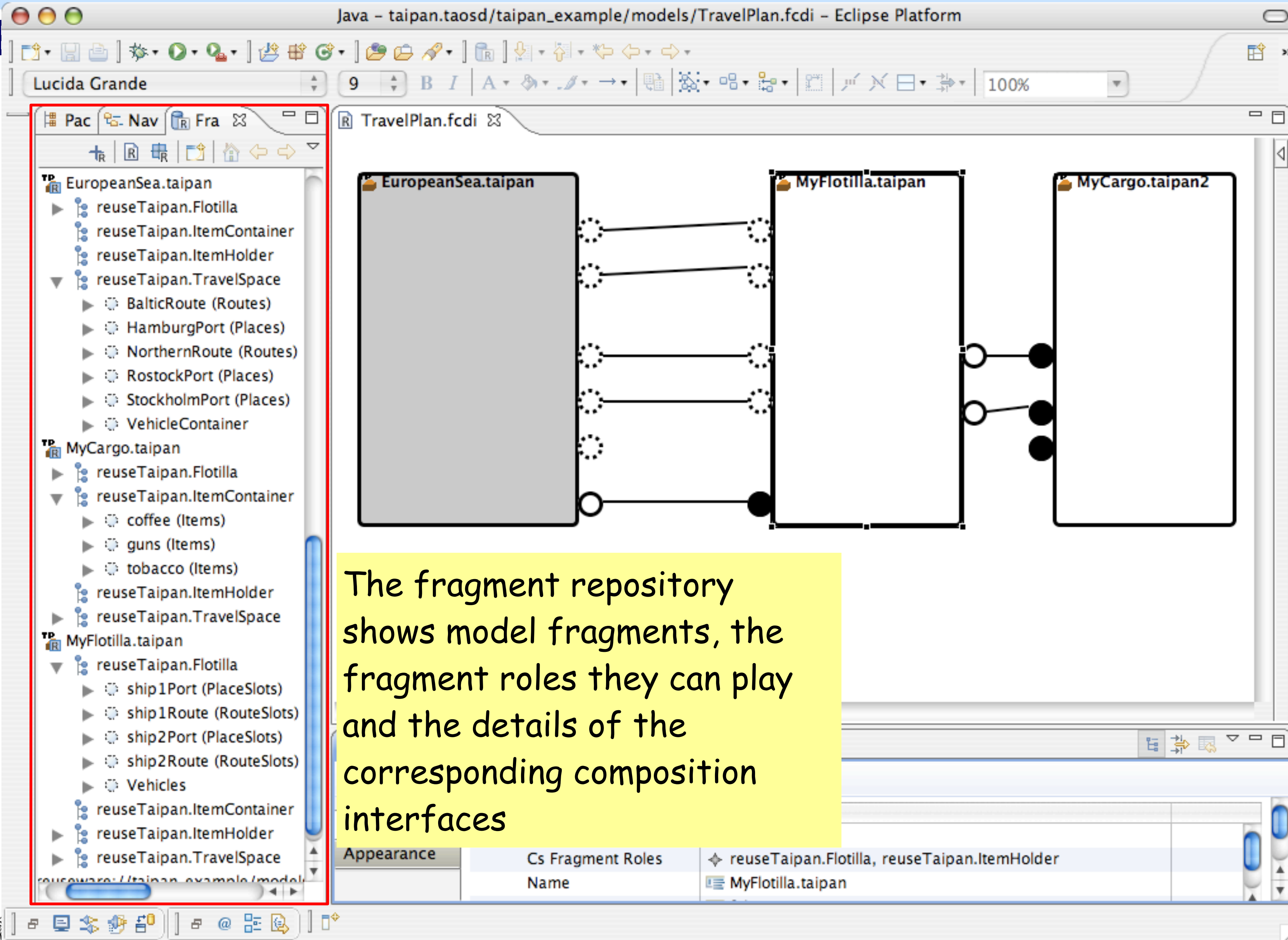


Building a DSL: Using Reuseware Tooling with a DSL

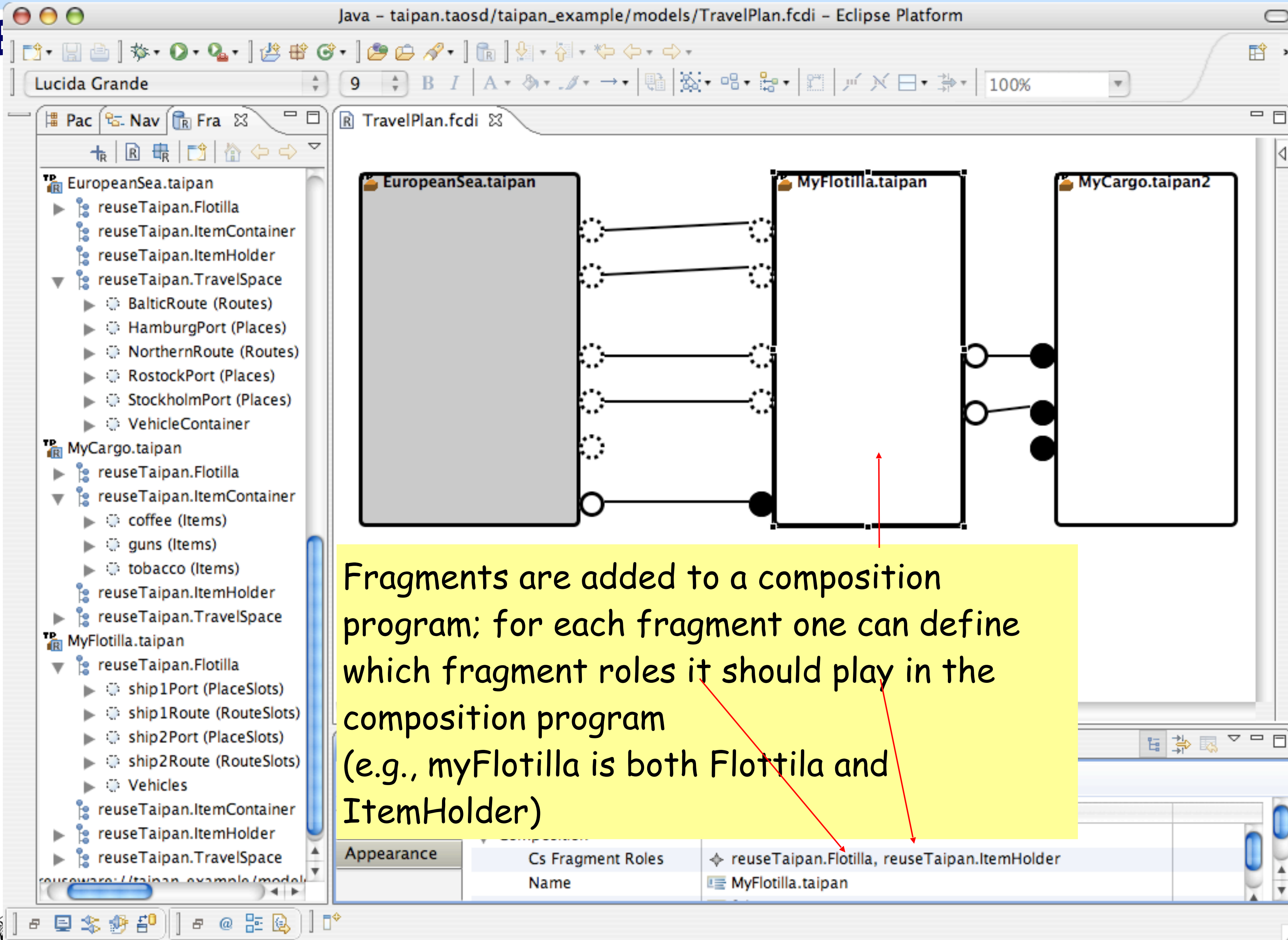
- Fragment Repository
  - Light-weight repository to manage and find reusable model fragments
  - Can instantly be used to build libraries of model fragments designed in a DSL

- Composition Program Editor
  - Independent of composition systems and reuse extensions
  - Can instantly be used to define compositions for the DSL
  - Layout can be customized if desired

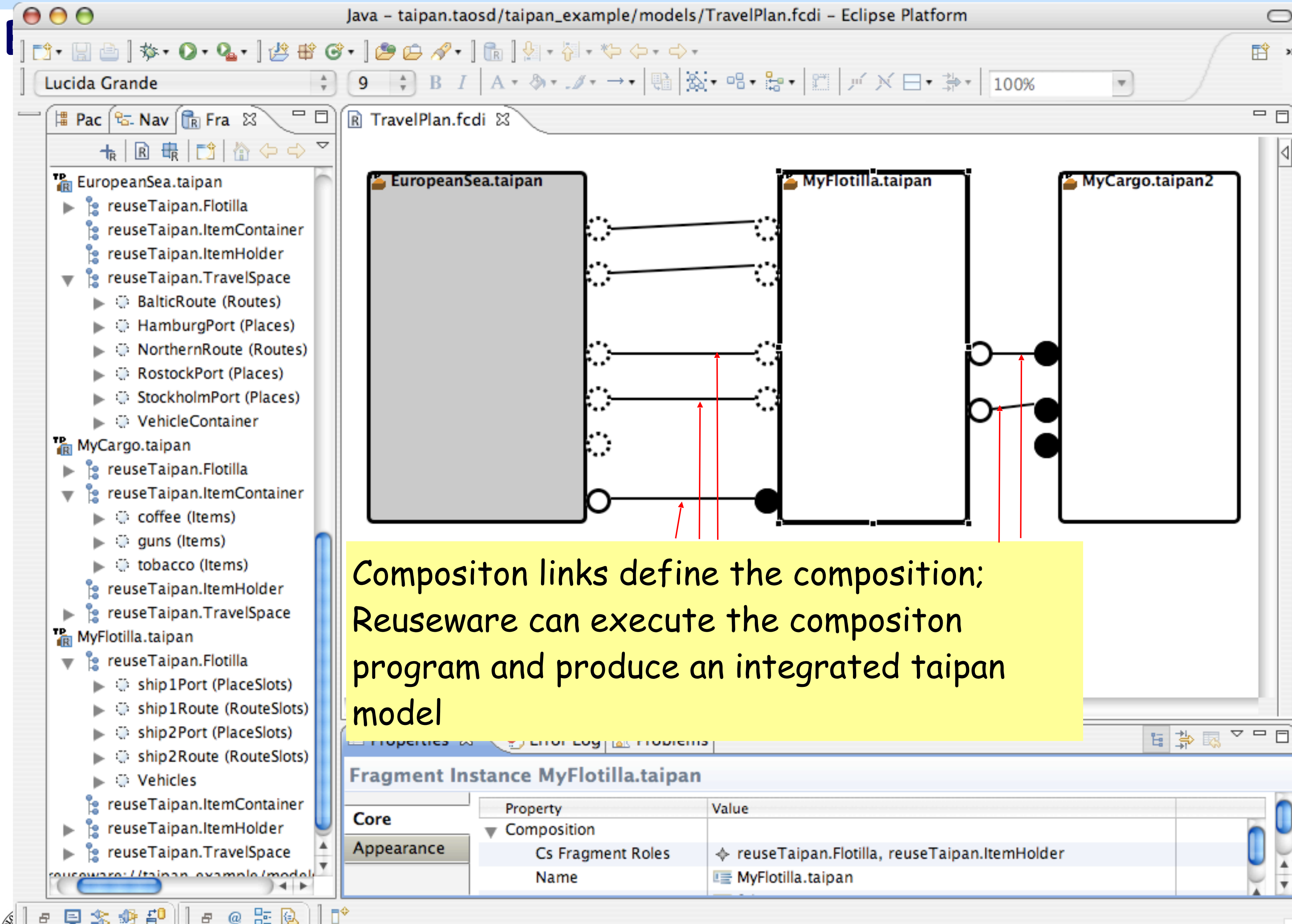

Java – taipan.taosd/taipan_example/models/TravelPlan.fcdi – Eclipse Platform
Lucida Grande
9
100%
Pac
Nav
Fra
TravelPlan.fcdi
EuropeanSea.taipan
reuseTaipan.Flotilla
reuseTaipan.ItemContainer
reuseTaipan.ItemHolder
reuseTaipan.TravelSpace
BalticRoute (Routes)
HamburgPort (Places)
NorthernRoute (Routes)
RostockPort (Places)
StockholmPort (Places)
VehicleContainer
MyCargo.taipan
reuseTaipan.Flotilla
reuseTaipan.ItemContainer
coffee (Items)
guns (Items)
tobacco (Items)
reuseTaipan.ItemHolder
reuseTaipan.TravelSpace
MyFlotilla.taipan
reuseTaipan.Flotilla
ship1Port (PlaceSlots)
ship1Route (RouteSlots)
ship2Port (PlaceSlots)
ship2Route (RouteSlots)
Vehicles
reuseTaipan.ItemContainer
reuseTaipan.ItemHolder
reuseTaipan.TravelSpace
EuropeanSea.taipan
MyFlotilla.taipan
MyCargo.taipan2
Properties
Error Log
Problems
Fragment Instance MyFlotilla.taipan
Core
Appearance
Property
Value
Composition
Cs Fragment Roles
reuseTaipan.Flotilla, reuseTaipan.ItemHolder
Name
MyFlotilla.taipan



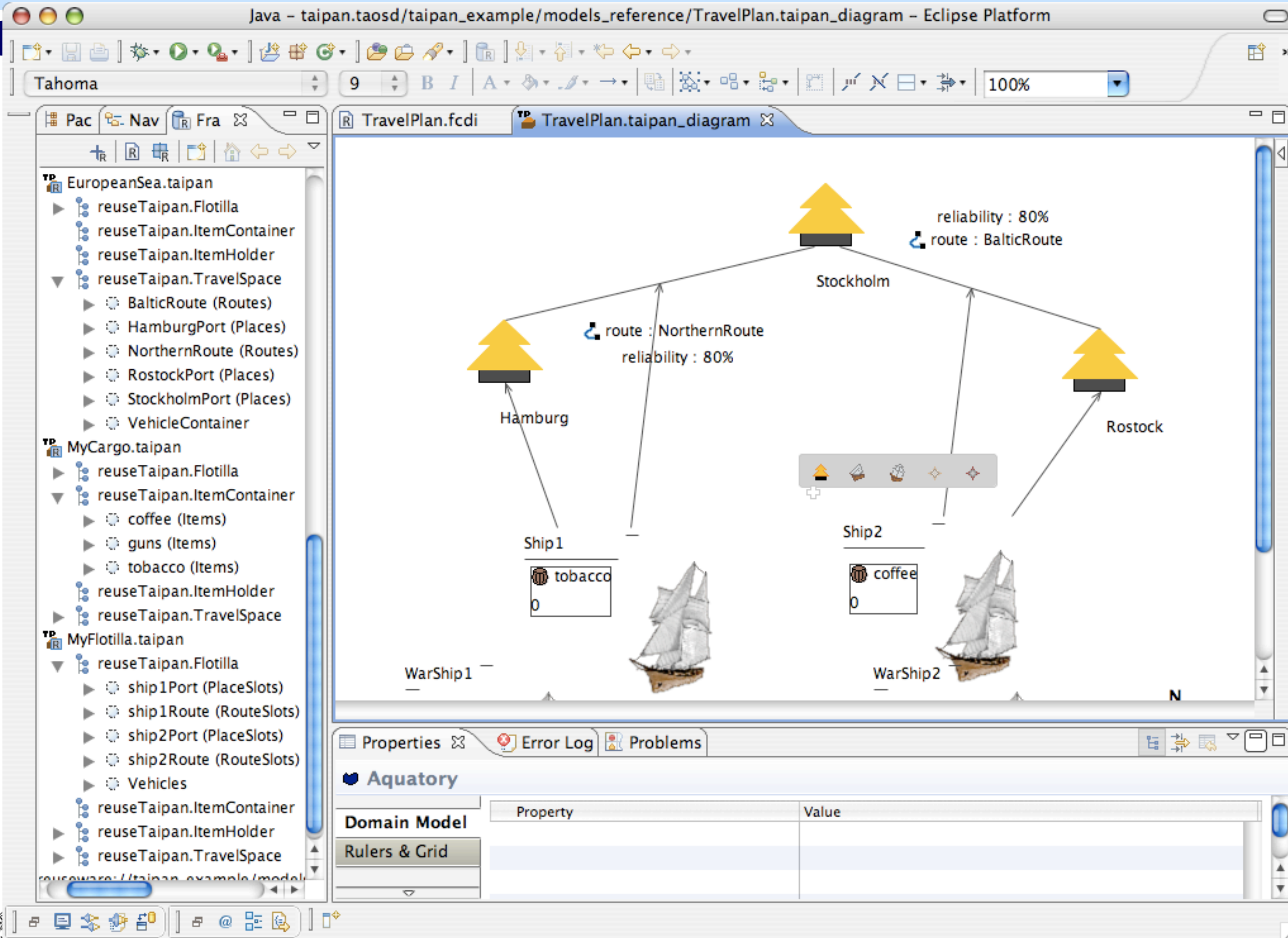
The fragment repository shows model fragments, the fragment roles they can play and the details of the corresponding composition interfaces



Fragments are added to a composition program; for each fragment one can define which fragment roles it should play in the composition program
(e.g., myFlotilla is both Flottila and ItemHolder)



Compositon links define the composition; Reuseware can execute the compositon program and produce an integrated taipan model


Java – taipan.taosd/taipan_example/models_reference/TravelPlan.taipan_diagram – Eclipse Platform
Tahoma
9
100%
Pac
Nav
Fra
TravelPlan.fcdi
TravelPlan.taipan_diagram
EuropeanSea.taipan
reuseTaipan.Flotilla
reuseTaipan.ItemContainer
reuseTaipan.ItemHolder
reuseTaipan.TravelSpace
BalticRoute (Routes)
HamburgPort (Places)
NorthernRoute (Routes)
RostockPort (Places)
StockholmPort (Places)
VehicleContainer
MyCargo.taipan
reuseTaipan.Flotilla
reuseTaipan.ItemContainer
coffee (Items)
guns (Items)
tobacco (Items)
reuseTaipan.ItemHolder
reuseTaipan.TravelSpace
MyFlotilla.taipan
reuseTaipan.Flotilla
ship1Port (PlaceSlots)
ship1Route (RouteSlots)
ship2Port (PlaceSlots)
ship2Route (RouteSlots)
Vehicles
reuseTaipan.ItemContainer
reuseTaipan.ItemHolder
reuseTaipan.TravelSpace
reliability : 80%
route : BalticRoute
Stockholm
route : NorthernRoute
reliability : 80%
Hamburg
Rostock
Ship1
tobacco
0
Ship2
coffee
0
WarShip1
WarShip2
N
Properties
Error Log
Problems
Aquatory
Domain Model
Rulers & Grid
Property
Value