



The Value of UML Models

Dr. Marko Boger
Gentleware

Gentleware



- ▶ **Founded in 2000
by M. Boger**
- ▶ **Offspring from
University research**
- ▶ **Team of 10 Employees**
- ▶ **Open Source-based**
 - ▼ ArgoUML
 - ▼ NSUML, Netbeans, Velocity, xDoclets, DocBook,
Ant, Xalan, Xerxes, OCL, CrazyBeans,
- ▶ **Profitable**



Poseidon for UML



▶ **Launch November 2001**

- ▼ Free Community Edition
- ▼ Over 125.000 downloads
- ▼ 2 Commercial Editions
 - Standard Edition
 - Professional Edition



▶ **Focus on Usability**

▶ **Standard-compliant**

▶ **Flexible Code Generation**

Argo and Poseidon



gentleware
developing differently



▶ **Jason and the**

Screenshot



The screenshot displays the Poseidon for UML Professional Edition software interface. The main window shows a UML class diagram for an ordering system. The diagram includes the following elements:

- Order** (Entity): A red box with the stereotype `<<entity>>`, methods `+getTotalSum() : int` and `-getVAT()`. It has a note "defines an order" pointing to it.
- Bill** (Entity): A red box with the stereotype `<<entity>>` and attribute `-amount : int`. It is associated with **Order** via a "payment" relationship with multiplicity `0..1`.
- Invoice** (Entity): A red box with the stereotype `<<entity>>` and method `+Invoice(order:Order)`. It inherits from **Bill**.
- Deduction** (Entity): A red box with the stereotype `<<entity>>` and method `+Deduction(order:Order)`. It inherits from **Bill**.
- UKInformation** (Entity): A yellow box that inherits from **Order**.
- USInformation** (Entity): A yellow box that inherits from **Order**.

The interface also features several panes:

- Navigation Pane**: Located on the left, showing a package hierarchy with "ordering" and "void" packages.
- Editing Pane**: A large orange oval highlighting the main diagram area.
- Details Pane**: Located at the bottom, showing the properties of the selected **Order** class, including Name, Stereotype, Namespace, and Operations.
- Overview Pane**: Located at the bottom left, showing a small overview of the entire diagram.

The software title bar reads "Poseidon for UML Professional Edition - Softsale". The menu bar includes "File", "Edit", "View", "Create Diagram", "Arrange", "Generation", "Critique", "Plugins", and "Help". The toolbar contains various icons for file operations and diagram editing. The status bar at the bottom right indicates "Class Diagram: ordering".

The Value of UML Models



► What to do with UML Models?

- ▼ Send to a colleague for implementation
- ▼ Send to a partner for coordination
- ▼ Send to a standardization committee

- ▼ Generate documentation
- ▼ Generate code
- ▼ Generate XML schema

The Value of Interchange



▶ **Model Exchange**

- ▼ Communicate from one Organization to another

▶ **Investment security**

- ▼ Migrate from one Vendor to another

▶ **Value-adding Chains of Tools**

- ▼ Whiteboard tool
- ▼ Modeling tool
- ▼ Code Generation tools
- ▼ Validation, Testing and Verification tools

XMI

▶ XML Metadata Interchange (XMI)

- ▼ Based on XML
- ▼ Standardized by OMG
- ▼ Used for highly referential Data
- ▼ E.g. Metadata
- ▼ Productions rules for MOF

▶ Supported by many Vendors

▶ Saving Format of Poseidon

Problems with XMI

▶ Problems with XMI

- ▼ Different Versions of XMI (1.0 – 1.2)
- ▼ Different Versions of MOF (1.0 – 1.4)
- ▼ Different Versions of UML (1.1, 1.3, 1.4)
- ▼ Different Interpretations
- ▼ Faulty Implementations

▶ No Diagram Information

- ▼ Only the Model Information is included

▶ RFP for Diagram Interchange by OMG

RFP Diagram Interchange



▶ RFP for UML 2.0

▶ Three Initial Submissions

- ▼ Rational

- ▼ Sun, Compuware

- ▼ Gentleware, Telelogic, Adaptive, DaimlerChrysler

▶ Joint Revised Submission

- ▼ First Revised Submission Sept. 9, 2002

- ▼ Final Revised Submission Jan. 6, 2002

- ▼ Supported by TogetherSoft, I-Logix, Softeam, KC



Tool Chain Example

Whiteboard tool,
Modeling tool,
Code Generator

The screenshot displays a multi-windowed software application. The top window is titled "Main - Class Diagram in 'Untitled' - 'Untitled' - Pointer mode - Ideogramic UML v2.2.2" and shows a menu bar with "File", "Edit", "Diagrams", "Options", "Tools", and "Help". Below it is a "Model View" toolbar. The middle window is titled "Poseidon for UML Professional Edition - Softsale" and has a menu bar with "File", "Edit", "View", "Create Diagram", "Arrange", "Generation", "Critique", "Plugins", and "Help". The bottom window is titled "XMLmodeling - hyperModel Application" and features a menu bar with "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help".

The "XMLmodeling" window is the primary focus, showing a project tree on the left with "UML Models" expanded to "CatML", which contains "CatalogItem". The main area displays an XML Schema Definition (XSD) for "CatalogItemType" in "Temp.xsd":

```
<!-- Class: CatalogItemType -->
<!-- Class: CatalogItemType -->
<!-- Class: CatalogItemType -->
<xs:element name="CatalogItem" type="CatalogItemType"/>
<xs:complexType name="CatalogItemType">
  <xs:sequence>
    <xs:element name="name" type="String"/>
    <xs:element name="description" type="String"/>
    <xs:element name="listPrice">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="Money"/>
        </xs:sequence>
      </xs:complexType>
    </xs:sequence>
  </xs:complexType>
</xs:sequence>
```

Below the XSD is a UML Class Diagram showing the relationships between classes: CatalogItem, FeatureValue, Resource, Category, Feature, and Party. CatalogItem is the superclass, with FeatureValue, Resource, and Category as subclasses. FeatureValue has a "value [0..1] : String" attribute. Resource has a "type 1" association with Feature. Category has a "parent 0..*" self-association and a "subcategory 0..*" association with itself. Feature has a "feature 0..*" association with CatalogItem. Party has a "supplier 1" association with CatalogItem. The diagram also shows a "supplier 1" association between CatalogItem and Party.

On the right side of the "XMLmodeling" window, there is a "Diagram: ordering" panel and an "Operations" panel with the following methods listed:

```
getTotalSum
getVAT
```

The bottom status bar of the "XMLmodeling" window shows "UML Diagram | XML Diagram | Web Browser".

Tool Chain Example



Powerpoint,
Internet Browser,
Word

Figure 7.1: Diagram Interchange Metamodel

Extended Graph Model

The basic concept of this metamodel extension is based on the idea of modeling the contents of the UML diagrams as graphs. The core classes are GraphNode and GraphEdge. Every visible model element is represented either by a GraphNode or by a GraphEdge. The base class of the graph elements is GraphElement. Graph elements are linked via a class called Connector. It is used to link GraphEdges with a GraphNode or another GraphEdge. The latter



Responding to Different Needs

▶ Model-oriented

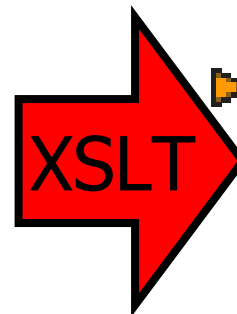
- ▼ Modeling tools, Code Generator
- ▼ Need Layout information
- ▼ Semantic understanding
- ▼ Differing representations
- ▼ Model editing

▶ Graphic-oriented

- ▼ Word Processing, Presentations, Publishing
- ▼ Need precise Graphic
- ▼ No Semantic understanding
- ▼ Exact representation
- ▼ Only graphical editing

▶ XMI

- ▼ Diagram Metamodel
- ▼ Graph-based



▶ SVG

- ▼ Lines and Boxes
- ▼ Graphic-based

Key Ideas

- ▶ **Graph-oriented Metamodel for UML and other Diagrams**
- ▶ **Unchanged UML Metamodel**
- ▶ **Extension of XMI**
 - ▼ XMI[UML]
 - ▼ XMI[DI]
 - ▼ XMI[UML+DI]
- ▶ **XMI for Model-oriented tools**
- ▶ **SVG for Graphic-oriented tools**

Demo

- ▶ **View UML model**
- ▶ **Save as XMI [UML+DI]**
- ▶ **Transforme to SVG using XSLT**
- ▶ **View SVG in Browser**

The Value of Code Generation



Generate to different Platforms

- ▼ Java / General Application
- ▼ EJB / Business Application
- ▼ C / Embedded Systems

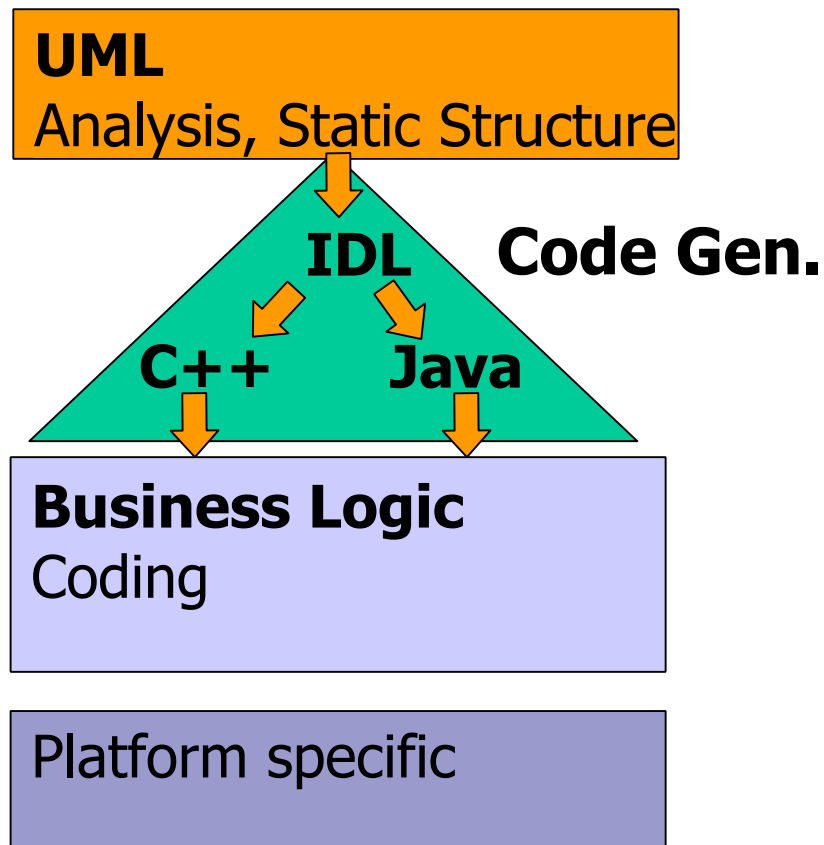
▶ Generate documentation

- ▼ HTML
- ▼ PDF /DocBook
- ▼ Management information / Metrics

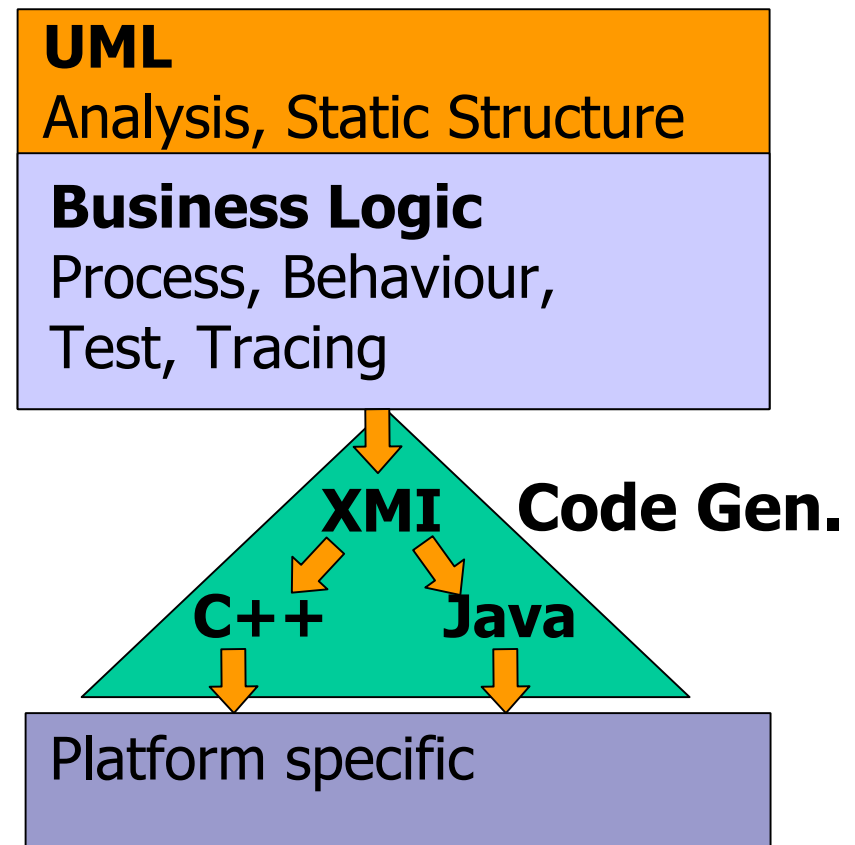
▶ The Foundation for MDA

Increased Importance of Code Generation

► CORBA



► MDA



Code Generation



▶ **Template-based**

- ▼ Velocity Project of Apache
- ▼ Template-Language
 - #if
 - #foreach
 - Variables: \$var
 - Java Classes: \$myClass
- ▼ Design an example file
 - Java, C++, HTML, XML, XML-Schema
- ▼ Replace variable parts with model info.

Example for HTML

```
<HTML>
<BODY>
Hello $customer.Name!
<table>
  #foreach( $product in $catalog.productlist )
    #if ( $customer.hasPurchased($product) )
      <tr>
        <td>
          $product.getTitle(
        </td>
      </tr>
    #end
  #end
</table>
```

```
<HTML>
<BODY>
  Hello Marko Boger!
  <table>
    <tr>
      <td>
        Poseidon for UML
      </td>
    </tr>
    <tr>
      <td>
        UMLdoc Plugin
      </td>
    </tr>
  </table>
```

How Velocity works



► Establish Context, use in Templates

Context

<code>clazz</code>	<code>MClassifier clazz = new MClassifierImpl("Car")</code>
<code>attributes</code>	<code>Vector attributes = { kfz , color }</code>
	<code>...</code>

Template

```
public class $clazz.getName() {  
  #foreach( $attribut in $attributes)  
    private $attribut;  
  #end  
}
```

Java

```
public class Car {  
  private kfz;  
  private color;  
}
```

Changing Templates



► Add simple metric to each class

▼ Count number of methods.

▼ If too many, generate a warning in header

```
/** Java class "${className}.java" generated from Poseidon for UML.  
 * Poseidon for UML is developed by Gentleware.  
 * Generated with velocity template engine.  
 */  
#if (preparedOperations.size())>10  
    // This class has too many operations. Consider a refactoring.  
#end
```

```
#if ($package != "")  
package $package;  
#end
```

```
#renderImports($imports)
```

```
#renderClassSignature($preparedClassifier)
```

Dr. Marko Boger

Changing Templates



▶ Add log information to Method

- ▼ Call Logmanager before every method call

```
#macro (renderOneOperation $currentOp $editable)
  $currentOp.getDocs()
  ${visibility}${static}${final}${returnType} ${name}($inOutPars)$thrownClause {
    LogManager.log.info("calling method ${name}()");
    #renderMethodBody($currentOp.getMethodBody() $currentOp.hasReturnType())
  } // end ${name}
#end
```

Example: UMLdoc



HTML

- generated
- including JavaScript
- Simple templates

Model softsale - Microsoft Internet Explorer

Adresse <C:\temp\softsale\doc\index.html>

Google Web-Suche Site-Suche PageRank Seiten-Info Aufwärts Hervorheben

softsale

- Deployment
- UseCases
- boni
- clients
 - Account
 - Address
 - Client
 - ClientController
 - CreditCard
 - E-Mail
- ordering
 - Bill
 - CountryInformation
 - Deduction
 - Invoice
 - Order
 - OrderController
 - UKInformation
 - USInformation
- products
 - Book
 - DigitalProduct

All Classifiers

- Account
- Address
- Administrator
- Bill
- Book
- Burning
- Client

Overview Package Classifier Tree Index Help

PREV CLASSIFIER NEXT CLASSIFIER FRAMES NO FRAMES

SUMMARY: INNER | ATTR | ASSOC | CONSTR | METHOD DETAIL: INNER | ATTR | ASSOC | CONSTR | METHOD DIAGRAMS: COLLAB | SEQ | STATE | ACTIV

ordering

Class Order

public class **Order**

Order is the central class for the order process. It records what status an order is in and provided various informations about the content of the order.

State diagrams of class Order

```
stateDiagram-v2
    [*] --> initialized
    initialized --> in_progress : order(product:Product)
    in_progress --> canceled : cancel
    in_progress --> completed : complete
    in_progress --> in_progress : entry / addProduct(product)
```

Example: Java



UML Class Diagram illustrating the structure of an ordering system in Java.

The diagram shows the following classes and relationships:

- Order** (Entity):
 - Operations: `+getTotalSum(): int`, `-getVAT()`
 - Relationships:
 - Aggregates **Bill** (indicated by a dashed line with an open diamond at the Order end).
 - Aggregates **CountryInformation** (indicated by a solid line with an open diamond at the Order end).
- Bill** (Entity):
 - Attribute: `-amount: int`
 - Relationships:
 - Generalized by **Invoice** and **Deduction** (indicated by solid lines with hollow triangle heads pointing to Bill).
- CountryInformation**:
 - Attributes: `-currencyName: String`, `-VATValue: Integer`
 - Relationships:
 - Generalized by **UKInformation** and **USInformation** (indicated by solid lines with hollow triangle heads pointing to CountryInformation).
- Invoice** (Entity):
 - Operation: `+Invoice(order: Order)`
- Deduction** (Entity):
 - Operation: `+Deduction(order: Order)`

Additional details from the diagram:

- A yellow note attached to the Order class states "defines an order".
- A relationship labeled "payment" exists between Order and Bill with a multiplicity of "0..1" at the Bill end.
- A relationship labeled "holds" exists between Order and CountryInformation with a multiplicity of "+country" at the CountryInformation end.

The diagram is displayed in the Poseidon for UML Professional Edition - Softsale environment. The Package-centric view on the left shows the package structure, and the Properties window at the bottom displays the Java source code for the `getTotalSum()` method.

```
9  */
10 public int getTotalSum() {          /** lock-end */
11     int sum=0;
12     Iterator it = product.iterator();
13     while (it.hasNext()) {
14         Product prod = (Product) it.next();
15         sum+=prod.getPrice();
16     }
17     return sum;
18 } // end getTotalSum          /** lock-begin */
```


Example: XML Schema



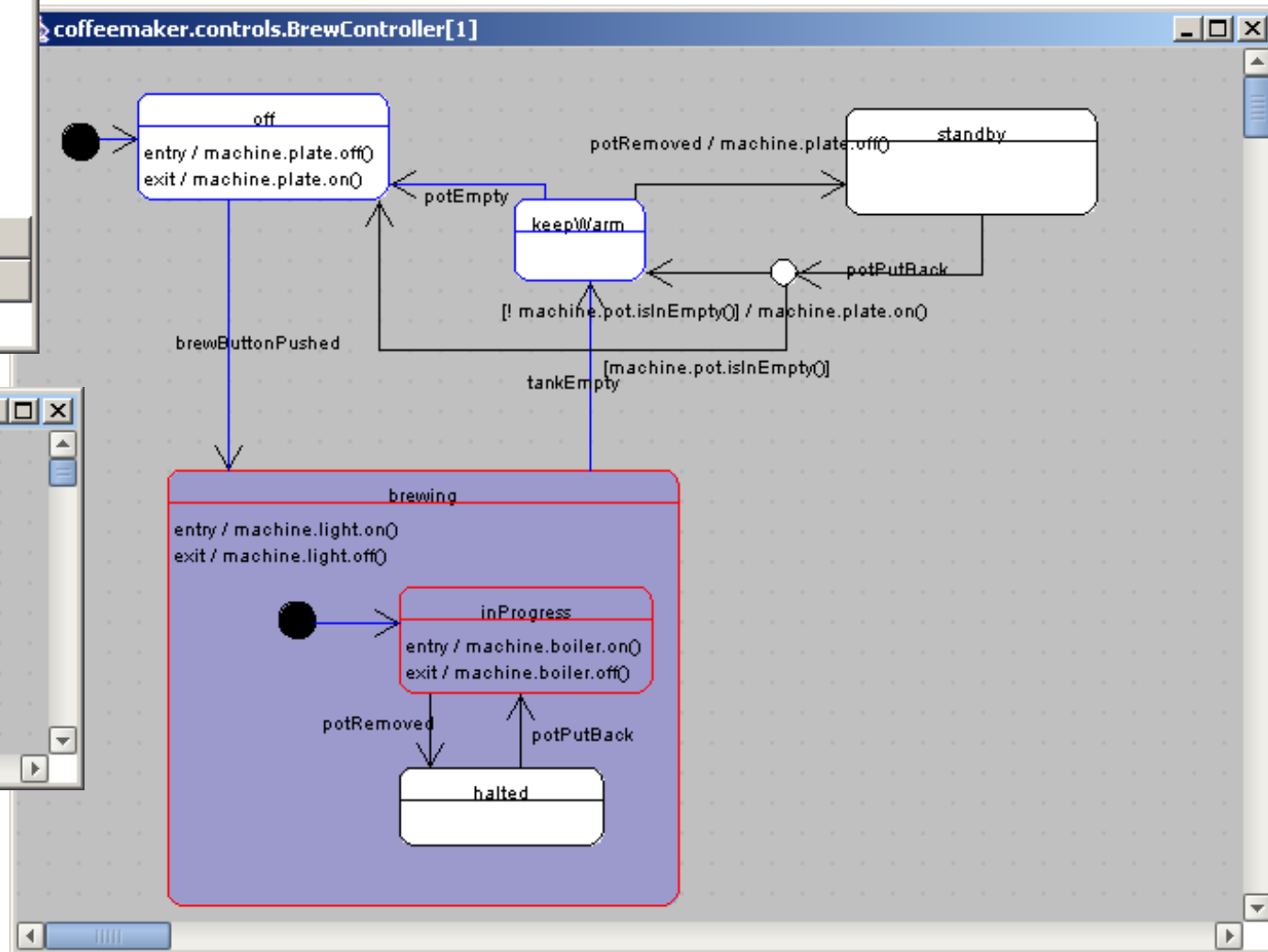
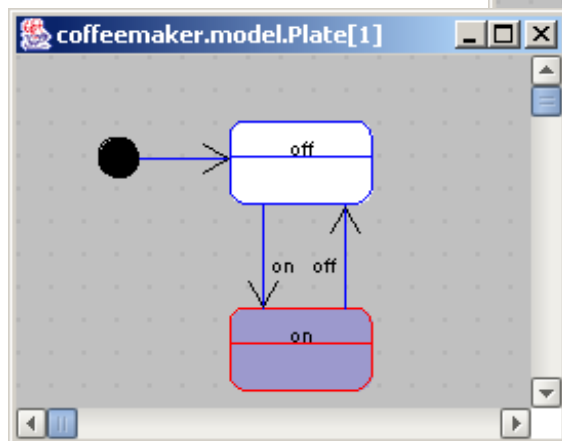
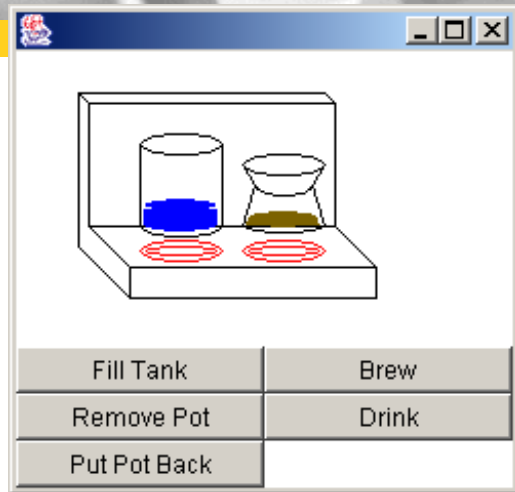
The screenshot displays the Poseidon for UML Professional Edition interface. The main window shows a UML Class Diagram titled "Class Diagram: PO diagram". The diagram includes the following classes and relationships:

- Item**: Attributes include `-partNum : SKU`, `-productName : string`, `-quantity : QuantityType`, `-USPrice : decimal`, `-comment[0..1] : string`, and `-shipDate[0..1] : date`. It has a collection relationship `0..*` with `+items`.
- Address**: Attributes include `-name : string`, `-street : string`, and `-city : string`.
- PurchaseOrder**: A `<<XSDcomplexType>>` class with attributes `-orderDate[0..1] : date` and `-comment[0..1] : string`. It has associations `+shipTo` and `+billTo` pointing to the **Address** class.
- SKU**: A `<<XSDsimpleType>>` class with the pattern `-pattern = \d{3}-[A-Z]{2}`.
- QuantityType**: A `<<XSDsimpleType>>` class with `-maxExclusive = 100`.

The XML Schema view at the bottom shows the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ipo="http://www.example.com/IPO" elementFormDefault="qualified" targetNamespace="http://www.example.com/IPO">
  <xsd:include schemaLocation="Address.xsd"/>
  <!-- ===== -->
  <!-- Package: <<XSDschema>> PurchaseOrder -->
  <!-- ===== -->
  <xsd:annotation>
    <xsd:documentation>International Purchase Order schema. Copyright 2000 Example.com. All rights reserved.</xsd:documentation>
  </xsd:annotation>
  <!-- ===== -->
  <!-- Class: ItemType -->
  <!-- ===== -->
  <xsd:element name="Item" type="ipo:ItemType"/>
  <xsd:complexType name="ItemType">
    <xsd:sequence>
      <xsd:element name="productName" type="xsd:string"/>
      <xsd:element name="quantity" type="ipo:QuantityType"/>
      <xsd:element name="USPrice" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="SKU" base="xsd:string" pattern="\d{3}-[A-Z]{2}"/>
  <xsd:simpleType name="QuantityType" base="xsd:integer" maxExclusive="100"/>
</xsd:schema>
```

Example: State Chart



Poseidon CD

▶ CD

- ▼ All submissions
 - Infrastructure
 - Superstructure
 - OCL
 - Diagram Interchange
- ▼ Poseidon for UML
 - Community Edition
 - Professional Edition

▶ Website

- ▼ www.gentleware.com/projects/diagraminterchange/

