From Declarative Business Models to J2EE Applications: An MDA Based Approach

Vladimir Bacvanski
vladimir@inferdata.com

Petter Graff
petter@inferdata.com

Topics of the Talk

• Business models
• UML extensions
• Services
• Platform independent extensions
• Platform dependent models
• Transformation
**Overview**

- Development of J2EE applications includes number of steps that can be automated
- We use:
  - Declarative business models,
  - Analysis models,
  - Decorated with additional requirements information
to generate design models resulting in J2EE applications
- Depending on the input parameters for generation, different components are generated

---

**Transformation of Models**

- UML analysis models, decorated with requirements information and design rules are used to generate design models resulting in J2EE applications
Platform Independent UML Models

- Specification of structure and required behavior

Use of Type Models

- Example: Tracking customer’s interest in services
  - No localized operations
  - Defined semantics for aggregation
- Type models are used to express what the system must persist
**Atomic System Level Operations**

- Pre and postconditions formally specified using OCL using the system type model as the formal vocabulary
- We are avoiding the “use case” term for its variant informal interpretation
  - We use system operations instead
  - Inspired by Fusion

**System Operation Example**

- Informally defined operation

```
addService(c:Customer, s: Service)
  pre:
    -- customer c does not already have interest in service s
  post:
    -- there is a new interest instance i
    -- i is linked with customer c
    -- i is linked with service s
```

- We would add OCL specification
- System operations are grouped in services
**Services**

- A service is a set of system operations of interest for a particular task
- Services come in two flavors:
  - Stateful services
  - Stateless services
- More than interfaces, also specifies
  - Pre and postconditions
  - Transactional specifications
  - Service level security
  - Performance requirements

---

**Services: Example**

- In addition to UML models, services use XML specifications of interesting properties
- System context model is not strictly required but it can be helpful to identify services
**Platform Independent Extensions**

- Information usage patterns
  - Readonly, read/write
- System level security requirements
  - Access rights
- Information volatility specification
  - How often does information change (and do we care)?
- Design criteria
  - Design goals with priorities
  - E.g.: reusable components vs. highest performance

---

**Extensions: XML Documents**

- Information not present in UML models is supplied as XML documents

```xml
<xs:simpleType name="TransactionDemarcation">
  <xs:annotation>
    <xs:documentation>
      The required or assumed transaction requirement
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="TransactionRequired"/>
    <xs:enumeration value="TransactionRequiresNew"/>
    <xs:enumeration value="TransactionSupports"/>
    <xs:enumeration value="TransactionMandatory"/>
  </xs:restriction>
</xs:simpleType>
```
**Persistence Platform Dependent Model**

**Query DAO**
- findCustomerByName(name : String) : Set
- findCustomerByEmail(email : String) : Set
- findCustomerByService(s : Service) : Set
- findCustomerByPrimaryKey(customer_id : String) : CustomerVO
- findInterestByCustomer(customer_id : String) : Set
- findInterestByCustomer(service_id : String) : Set
- findInterestByPrimaryKey(id : String) : Interest_VO

**Update DAO**
- addCustomer(name : String, email : String)
- removeCustomer(customer_id : String)
- changeCustomer(cvo : CustomerVO)
- addInterest(customer_id : String, service_id : String)
- cancelInterest(interest : id : String)
- addService(name : String, description : String)

**J2EE Generated Design: Browser**

**IBrowser**
- findServiceByName(name : String) : Set
- findCustomerByName(email : String) : Set
- findCustomerByEmail(email : String) : Set
- findCustomerByService(s : Service) : Set

**Browser_SBHome**
- create() : Browser_SB

**Browser_SB**
- findServiceByName(name : String) : Set
- findCustomerByName(email : String) : Set
- findCustomerByEmail(email : String) : Set
- findCustomerByService(s : Service) : Set

**Browser_BBEJax**
- findServiceByName(name : String) : Set
- findCustomerByName(email : String) : Set
- findCustomerByEmail(email : String) : Set
- findServiceByCustomer(c : Customer) : Set
- findCustomerByService(s : Service) : Set

**Browser_Implmentation**
- findServiceByName(name : String) : Set
- findCustomerByName(email : String) : Set
- findCustomerByEmail(email : String) : Set
- findServiceByCustomer(c : Customer) : Set
- findCustomerByService(s : Service) : Set

**ServiceFactory**
- getBrowser() : IBrowser
- getCustomerMaintenance() : ICustomerMaintenance
- getRegistration() : IRegistration
- getServiceMaintenance() : IServiceMaintenance

Copyright © InferData Corporation 2001
Transformation

- Transformation of models is based on XMI and XML information
- XMI models exported from CASE tools
- XSLT based transformer
  - Problem: lacking support for abstractions
- JDOM based Java transformer aware of MOF
**Conclusion**

- Business and Analysis UML models drive the generation of J2EE components and persistence
- Services organize the system operations
- Requirements extensions (as XML documents) add parameters for the J2EE component generation
- Parameters are used to determine the kind of J2EE platform specific components
- Integrated with CASE tools through XMI
- Tool implemented as XML transformation system

**References**

- OMG MDA site: www.omg.org/mda
- InferData Course: *Object-Oriented Analysis and Design with UML*, 2001: www.inferdata.com
- XSL Transformations (XSLT): www.w3.org/TR/xslt
- JDOM: jdom.org

- For the latest version of this presentation, please visit www.inferdata.com/resources/