A Plugin-Based Framework for Domain Models and Persistence in Environmental Management Information Systems

Thorsten Busse[1], Nicolas Denz[1], Bernd Page[2]

[1]ifu Hamburg GmbH
[2]University of Hamburg, Department of Informatics

Open Source in the Field of Environmental Informatics
EnviroInfo 2008 in Lüneburg

09-12-2008

1 PLUGIN.NET
2 Domain Models and Persistence
3 DSL for Material Flow Analysis
4 Discussion

Outline

1. Objectives of the PLUGIN.NET Project
2. Domain Models and Persistence in Dynamic Architectures
3. Domain Specific Language for Material Flow Analysis
4. Discussion, Conclusions, and Outlook

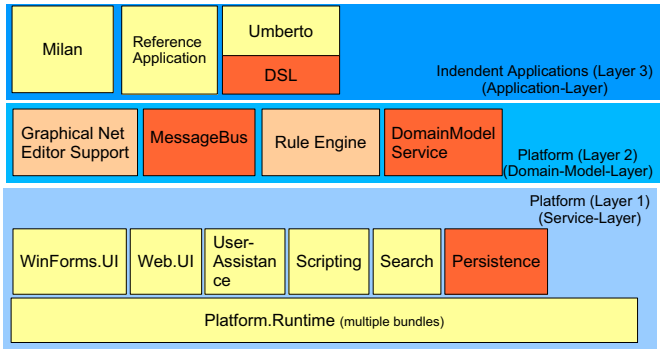
1 PLUGIN.NET
2 Domain Models and Persistence
3 DSL for Material Flow Analysis
4 Discussion

Project PLUGIN.NET

- Cooperative industry project
  - funded by Innovationsstiftung / RIS Hamburg
  - applied research with marketable results
  - project partners
    - ifu Hamburg GmbH
    - Department of Informatics UHH
    - related to EMPORER project at FHTW Berlin
- Project objectives
  - develop parts of .NET plugin framework *Empinia*
  - distribution under open source license (Apache)
  - application to ifu EMIS software Umberto, e!Sankey

1 PLUGIN.NET
2 Domain Models and Persistence
3 DSL for Material Flow Analysis
4 Discussion

Focus of PLUGIN.NET



Based on Schnackenbeck, Panic, Wohlgemuth (2007)

Domain Models

„An object model of the domain that incorporates both behaviour and data“ (Fowler 2003, p. 116)

- Data (platform level)
- Behaviour
  - actions (domain object level)
  - rules (domain object level and platform level)
- One DSL to cover all aspects of the domain model
  - and more: graphical presentation in a GUI etc.

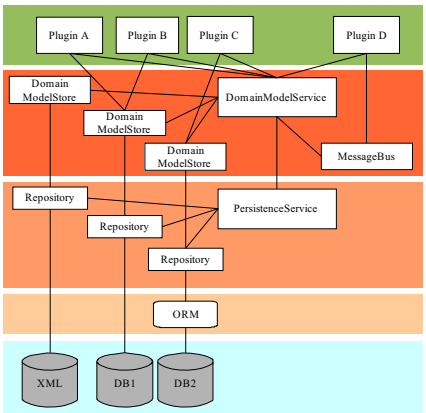
Dynamic Domain Models

- Challenges of domain models in dynamic architectures
  - modularization
  - dynamic assembly
  - extensions and changes
  - observation of domain objects
- 'Scattered' domain models (distinct bundles)
  - break dependencies between domain model elements
  - layered approach to improve re-usability (Fowler 2003, p. 17)

Persistence in Dynamic Architectures

- Requirements
  - ease of use
  - easy availability for other plugins
  - provision and management of persistence for other plugins
  - abstraction from specific persistence backends
  - encapsulation of existing ORMs
  - establish database relations at load time or runtime
  - one or more data repositories
  - support for different persistence strategies
  - extensibility and changeability of the persistence service itself

Architecture Overview



Architecture Details

- Observation of domain objects
  - *Message Bus* following modified Observer pattern (GoF)
  - filters to limit the number of notifications (compare to Enterprise Service Bus architectures)
  - check domain rules; support undo/redo (work in progress)
- Extensions of domain models
  - dynamically extend any domain object with *Extending Properties* (similar to GoF *Decorator* pattern)
  - managed by central *Extending Property Service*
  - realization of layered architecture for modularization and re-use of domain models

Domain Object Definitions

```
<!-- Definition of 'Material' bundle -->
<bundle id="platform.material" description="components related to materials" version="0.1">
 <extension id="platform.material.persistenceconsumer"
   point="platform.persistence.repositoryconsumerpoint">
  <repositoryConsumer><requiredMapping>Platform.Material.dll</requiredMapping></repositoryConsumer>
 </extension> ...
</bundle>
```

```
<!-- Domain object definition of class 'Material' -->
<businessObject>
 <classDefinition classNamespace="Platform.Material" name="Material">
  <property name="Name" domain="MaterialName" translatable="true"/>
  <property name="BasicUnit" domain="Interface" customType="IUnit" notNull="true">
   <interfaceAttribute name="TypeConverter" parameters="typeof(StringConverter)"/>
  </property> ...
 </classDefinition>
</businessObject>
```

```
<!-- Domain object definition for extension of material by project attribute -->
<businessObject> ...
 <classDefinition classNamespace="Platform.Project" name="MaterialToProject">
  <property name="Project" domain="Interface" customType="IProject" />
  <property name="Material" domain="InterfaceCollection" customItemType="IMaterial"
    nameSpace="Platform.Material" columnName="MATERIALID" />
 </classDefinition>
</businessObject>
```


Related Work

- Domain modelling and persistence in .NET and Java
  - *Spring.NET* (*NHibernate* integration, transaction concept)
  - *Eclipse EMF* in combination with Java Persistence API implementations (*Hibernate*, *EclipseLink*, *Teneo*)
  - limited support for the specific challenges of domain models in dynamic architectures!
- *CAP.NET* (Wolfinger et al. 2005)
  - eclipse-like core framework and workbench UI for .NET
  - focus on slot / extension mechanism with .NET attributes
  - no focus on domain modelling and persistence
  - apparently no open source project but commercial use

Conclusions and Outlook

- Domain models and persistence in dynamic architectures
  - abstract approach for domain models in plugin environments instead of 'yet another ORM'
  - handle multiple aspects of domain models in one DSL
  - part of OS plugin framework *Empinia* for the .NET platform
- Future work
  - improve transaction support (in progress)
  - improve and further implement domain rule concepts
  - more expressive DSL (rules, presentation)
  - continue evaluation (free and commercial EMIS software)

Thank you for your attention!

Questions...?

Sources

- Project Application at 'Innovationsstiftung Hamburg' (2006)
- P. Jahr, S. Simroth: Entwicklung von Teilbereichen eines Software-Frameworks im Einsatzkontext betrieblicher Umweltinformationssysteme. Bac. Thesis, FHTW Berlin (2007).
- T. Schnackenbeck, D. Panic, V. Wohlgemuth: Eine offene Anwendungsarchitektur als Fundament eines Methodenbaukastens für BUIS. Workshop „Simulation in den Umwelt- und Geowissenschaften, Medizin und Biologie", Berlin (2007).
- Martin Fowler, Patterns of Enterprise Architecture. Addison Wesley, Boston (2003).
- E. Gamma et al.: Design Patterns. Addison-Wesley, Reading (1999).
- A. Möller: Grundlagen stoffstrombasierter betrieblicher Umweltinformationssysteme. Projekt-Verlag, Bochum (2000).
- Spring.NET - http://www.springframework.net
- Eclipse - http://www.eclipse.org
- Hibernate - http://www.hibernate.org
- EclipseLink - http://planeteclipse.net/eclipselink
- Teneo - http://www.eclipse.org/modeling/emf/?project=teneo
- CAP.NET: R. Wolfinger, D. Ghungana, H. Prähofer, H.P. Mössenböck: A Component Plug-In Architecture for the .NET Platform. In: Proceedings of the Joint Modular Languages Conference (2006).