UML Profiles versus Metamodel extensions: An ongoing debate

Philippe Desfray

www.objecteering.com
www.umlopenedition.com
www.softeam.fr
UML 1.4 profiles modeling capacities

- Structuring the extensions (Profile = Packages)
- Defining new meta-classes (Stereotypes)
- Defining new meta-attributes (tagged values)
- Defining new meta-associations (tagged values, referencing to other model elements)
- Defining new constraints
- Modeling graphically profiles

This is almost all we need for defining metamodels
UML Profiles: Adapting UML to each domain

PROFILES STRUCTURE UML EXTENSIONS

- <<profile>> Scheduling, perf & time
- <<profile>> EDOC
- <<profile>> CORBA
- <<profile>> EJB
- <<profile>> S.P.E.
- <<profile>> EAI
- <<profile>> QOS & Fault Tolerance

(Software Process Engineering Management)
Persistence

Model example (1)

- **Constraint**: A persistent class, or one of its parent classes must have at least one **identifier** attribute.

- **storageMode**: KindOfStorage
UML Profiles : Model examples (2)
Constraint
{Classes having "class attributes" cannot be mutable.}

UML Profiles : Model examples (3)
MOF : Model interoperability
A major goal, hard to combine with flexibility

- Troubles with different versions of UML, becoming even harder when combined with MOF/XMI versions
- Tool implementer testimony: moving from one metamodel to another is a real heavy task, hard for tool implementers, heavy for end users
MOF architecture (implicit) postulates for interoperability

• Metamodels are stable (standardized). They do not evolve, or do change only after a long stable period
• Metamodels are formal: their semantics are completely defined, in a precise and unambiguous way

The reality is:
• We (end users) wish a stable root standard but we never have (yet)
• The extensions that we define are incomplete, informal, and may even be contradictory
• We need flexibility, ability to change fast, to combine different views
A complementary view of MOF and profiles

All at level 2 regarding the MOF architecture

MOF based metamodels
STABLE, FORMAL, FROZEN CORE

UML profiles based extensions
FLEXIBLE, MIXABLE, MUTABLE
UML Profiles Flexibility

- Supporting profile combination: several profiles can be applied to the same model
  - Ex: A class can be reactive (real time profile), and persistent (RDB profile) at the same time
  - Even inconsistent profiles can be combined (ex: Java and C++)
- Supporting model exchange between different profiles
- Supporting the dynamic change of applied profiles to a model, in order to change perspective during the development lifecycle

UML profiles is a mechanism for defining flexible projections of a stable predefined core metamodel.

UML model elements have an immutable part (their core UML definitions) and mutable combinable extensions
Inherent properties of profiles

• A profile defines a projection of a reference metamodel

• Profiles provide a mechanism to define facets that can be applied to model elements and combined

• All elements defined in a profile are mutable. Mutability rules are driven by the reference metamodel
Rational for choosing the right metamodeling technique

- Your domain is well defined, and has a unique well accepted main set of concepts
- A model realized under your domain is not subject to be transferred into other domains
- There is no need to combine your domain with other domains
  ➔ Choose a MOF based technique

- Your domain is not subject to consensus, many variations and point of view exist
- Many changes and evolutions may occur
- Your domain may be combined with other domains, in an unpredictable way
- Models defined under your domain may be interchanged with other domains
  ➔ Choose a UML Profile based technique
Advanced profile usage

• Structuring case tool customizations using the UML profile mechanism
• Adding procedural features structured by UML Profile, thus providing
  – Inheritance between tool customizations
  – Model transformation rules
  – Model presentation rules
  – Model consistency checks rules
Combining profile for driving software development

UML MODEL

Analysis Model

Design Model

Code

PROFILES

UML Analysis

UML Design

RDB Modeling

UML/ C++

UML/ ORACLE

PHASE

Analysis

Design

Realization
Building Profile: a new kind of expertise in software development

UML Modeler

UML Profile Builder

Software Process
Components
Real Time
XML
C++
Java
EJB

UML

(packaged profiles)

Designer

Use a customized Case tool adapted to your domain

Domain Expert

Design and implement UML expertise for any kind of domain
Questions to be solved (UML2.0)

• Can the profile mechanism be merged with the MOF mechanism?
• Is it desirable to do so?
• If so there should be specific concepts for
  – specifying the mutability, and view point aspects inherent to the profile technique,
  – providing an absolute guarantee of strong conformance to the reference (MOF based) metamodel.
MOF/Profiles
A possible Approach for UML 2.0

Isomorphism

MOF Based
extension mechanisms

UML Profiles
extension mechanisms

MOF Based implementation
(backward compatibility)

Annotation based implementation
(backward compatibility)

Semantics for:
Metamodel projection,
Mutability,
Facets management