Werkzeugunterstützung für UML Profiles

Verteidigung des Großen Belegs

Andreas Pleuß

Aufgabenstellung

- Sammlung der Anforderungen an UML Profiles
- Untersuchung bestehender UML-CASE-Tool Unterstützung
- Untersuchung der Erweiterungsmöglichkeiten eines ausgesuchten UML-CASE-Tools
- Ausarbeitung von Empfehlungen für die Erweiterung des ausgewählten CASE-Tools
- Nachweis der Machbarkeit der Empfehlungen mittels eines einfachen Prototyps

Gliederung

- Einführung in UML Profiles
- Bestehende Werkzeugunterstützung am Beispiel „Objecteering“
- Untersuchung des Werkzeugs „Together“ auf Erweiterbarkeit
- Empfehlungen zur Erweiterung von „Together“
- Prototypische Umsetzung
- Zusammenfassung

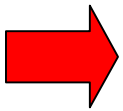
UML Profiles

- Motivation: Anpassung bzw. Erweiterung von UML für spezielle Anwendungsbereiche (z.B. CORBA)
- UML Metamodell: Beschreibt UML
- UML Profiles spezialisieren das Standard-UML-Metamodell, ohne dieses zu verletzten
- Verwenden dazu in UML eingebaute Erweiterungsmechanismen:
  - Stereotypes
  - Constraints
  - Tagged Values

Profiles in der UML Spezifikation

- UML 1.3: Keine Festlegungen zu UML Profiles, nur Beispiele
- UML 1.4: Profiles als Package mit Stereotype <<profile>>
- Nicht vollständig erfüllte Anforderungen:
  - Komposition von Profiles
  - Austauschformat: Graphische Repräsentation des Stereotypes
  - Graphische Notation für Definition von Stereotypes: Nicht mit Metamodell vereinbar
  - Generelle Spezialisierung einer Metaklasse im Profile

Gliederung



- Einführung in UML Profiles
- Bestehende Werkzeugunterstützung am Beispiel „Objecteering“
- Untersuchung des Werkzeugs „Together“ auf Erweiterbarkeit
- Empfehlungen zur Erweiterung von „Together“
- Prototypische Umsetzung
- Zusammenfassung

Objecteering

- Hersteller: Softeam (Frankreich)
- Version: 5.1.0
- Objecteering UML Profile Builder:

  Erstellung von UML Profiles
- Objecteering UML:

  Erstellung von Modellen und Generierung von Dokumentation und Code unter Verwendung von UML Profiles
- Wichtigstes CASE-Tool bezüglich Profile-Unterstützung

UML Profile in Objecteering: Standardbestandteile

- Stereotypes, Constraints und Tagged Values werden unterstützt
- Grundlage UML 1.3
- Nur wenige Einschränkungen bezüglich der Umsetzung der UML Spezifikation
- Dadurch jedoch z.B. „UML Profile for CORBA“ nicht direkt realisierbar

UML Profile in Objecteering: Proprietäre Bestandteile

- „Arbeitselemente“:
  - Commands: Repräsentieren Befehle
  - Work Products: Repräsentieren Dateien
  - Generation/Documentation Templates: Repräsentieren Struktur textbasierter Dokumente
  - Parameter: Zur Konfiguration von UML Profiles
- Sprache J: Einfache, Java-ähnliche Sprache zum Zugriff auf Modellelemente und Elemente des Werkzeugs.

UML Profile in Objecteering: Proprietäre Bestandteile

- „Semantische Elemente“:
  - J Methods: Methoden mit J-Code
  - J Attributes
- Semantik sowohl für proprietäre als auch für Standardbestandteile festlegbar
- Besonders weitreichende Unterstützung bezüglich der Anpassung der Generierung von Dokumentation und Code durch ein UML Profile

Gliederung

- Einführung in UML Profiles
- Bestehende Werkzeugunterstützung am Beispiel „Objecteering“
- Untersuchung des Werkzeugs „Together“ auf Erweiterbarkeit
- Empfehlungen zur Erweiterung von „Together“
- Prototypische Umsetzung
- Zusammenfassung

Together

- Hersteller: Togethersoft (USA)
- Untersuchte Version: 5.02
- Aktuelle Version: 5.5

Standardmäßige Profile-Unterstützung

- Zuweisung vorgegebener oder beliebiger eigener Stereotypenamen zu Modellelementen
- Anzeige im Diagramm in üblicher Notation
- Einige der vorgegebenen Stereotypes werden graphisch repräsentiert
- Keine semantische Auswertung
- Einbeziehung in Code als Javadoc-Kommentar
- Einbeziehung in XMI
- Tagged Values nur implizit als Eigenschaften enthalten (z.B. „synchronized“)

Mechanismen zur Erweiterung

- Open API: Java API zur Erweiterung von Together
- Modules: Zur Einbindung von Code zur Erweiterung in Together
- Config-Dateien: Dateien zur Konfiguration von Together

Grundaufbau der Open API

- Insgesamt 40 Packages, eingeteilt in drei Gruppen:
  - IDE (Interactive Development Environment): Lesen und Repräsentieren von Informationen
  - RWI (Read-Write-Interface): Lesen und Schreiben von Eigenschaften des Modells
  - SCI (Source Code Interface): Bearbeiten generierten Quellcodes

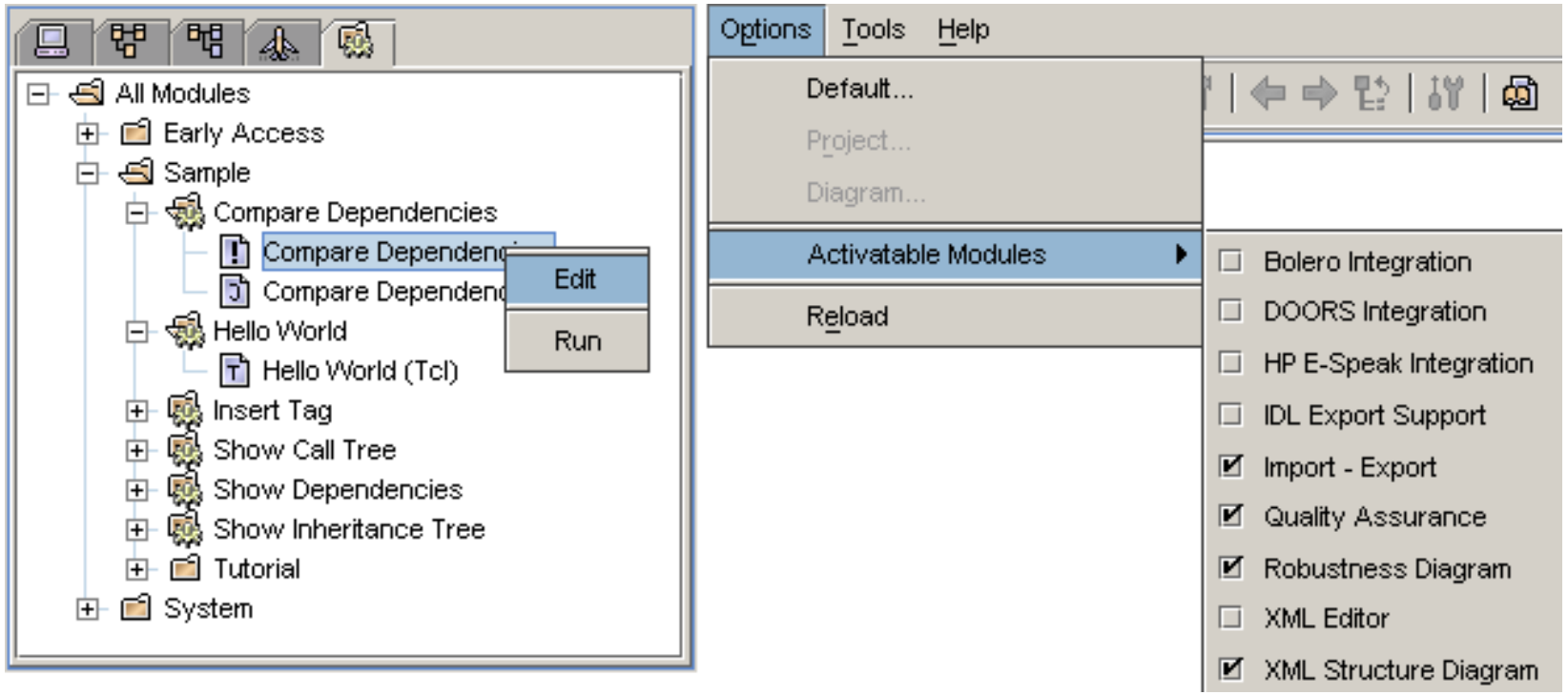
Zusammenfassung Open API

- Wenig einheitliche Mechanismen
- Einige Bereiche gut beeinflussbar, z.B.:
  - Aufruf eigener Dialoge
  - Eigene Dateizugriffe
  - Hinzufügen eigener Steuerelemente, z.B. Befehle
  - Eigenschaftsdialog zu Modellelementen
- Starke Einschränkungen bei Modifikation vorhandener Elemente, z.B Diagrammelemente
- An vielen Stellen Config-Dateien mächtiger

Modules

- Enthalten Code zur Erweiterung von Together (Java oder TCL/JPython)
- Typen von Modules:
  - „User“: Werden durch Icon oder Kontextmenü in Together ausgeführt und direkt danach beendet.
  - „Startup“: Werden stets zum Start von Together ausgeführt und bleiben dauerhaft aktiv
  - „Activatable“: Können in Together aktiviert werden und bleiben (auch über mehrere Sitzungen hinweg) solange aktiv, bis sie deaktiviert werden.
- Manifest-Datei enthält Informationen für Together zu einem Module

Modules: Anwendung in Together



Config-Dateien

- Konfigurieren Together durch Zuweisungen *Eigenschaft = Wert*
- Können auch Kontrollstrukturen (z.B. *if*) und Methodenaufrufe enthalten
- Erlauben weitreichende Erweiterungen und Anpassungen von Together, z.B. bei:
  - Eigenschaftsdialogen
  - Konfigurationsdialogen
  - Diagrammen
- Sind kaum dokumentiert

Gliederung

- Einführung in UML Profiles
- Bestehende Werkzeugunterstützung am Beispiel „Objecteering“
- Untersuchung des Werkzeugs „Together“ auf Erweiterbarkeit
- Empfehlungen zur Erweiterung von „Together“
- Prototypische Umsetzung
- Zusammenfassung

Empfehlungen zur Erweiterung (1)

- Stereotypes, Constraints und Tagged Values als Eigenschaften vom Typ String
  - Eingabe und Verwaltung durch beliebige eigene Dialoge
  - Speicherung in eigenen Dateien
  - Anzeige im Eigenschaftsdialog
- Visualisierung von Tagged Values in Diagrammen
  - Diagramme nicht durch Open API modifizierbar
  - Realisierung mittels Config-Datei konnte nicht gefunden werden

Empfehlungen zur Erweiterung (2)

- Graphische Repräsentation von Stereotypes:
  - Problem analog 2.
  - Config-Dateien ermöglichen Modifikation der graphischen Repräsentation eines Modellelementes (z.B. eine Klasse als Oval statt als Rechteck).
  - Jedoch offenbar nur vordefinierte Vektorgraphik-Elemente möglich, keine Icons

Empfehlungen zur Erweiterung (3)

- Graphische Definition von Stereotypes:
  - Möglich durch Definition eines eigenen Diagrammtyps mittels Config-Dateien
- Semantische Auswertung:
  - Bisher keine Mechanismen in Together vorgesehen
  - Together nicht am UML Metamodell orientiert

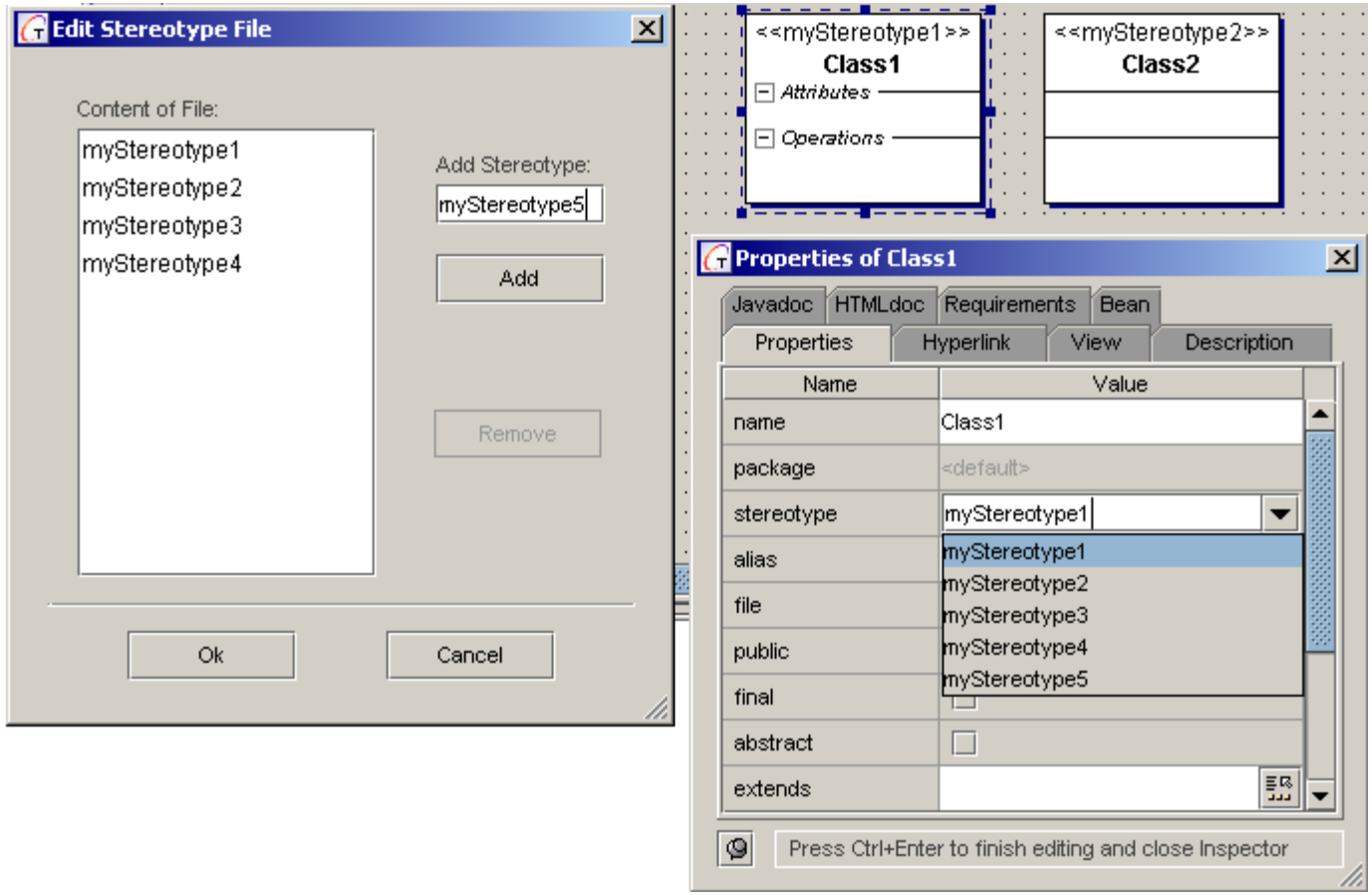
Gliederung

- Einführung in UML Profiles
- Bestehende Werkzeugunterstützung am Beispiel „Objecteering“
- Untersuchung des Werkzeugs „Together“ auf Erweiterbarkeit
- Empfehlungen zur Erweiterung von „Together“
- Prototypische Umsetzung
- Zusammenfassung

Prototypische Umsetzung

- Umsetzung der Empfehlung 1) für Stereotypes und Klassen
- Funktionalität:
  - Module vom Typ „Activatable“
  - Eingabe eigener Stereotypes und speichern in eigener Stereotype-Datei (Textformat)
  - Bearbeiten einer Stereotype-Datei
  - Auswahl einer Stereotype-Datei für das aktuelle Projekt bzw. als Standardwert für alle Projekte
  - Stereotypes aus Datei im Eigenschaftsdialog verfügbar
  - Behandlung der Stereotypes analog vordefinierter Stereotypes

Prototyp: Eigene Stereotypes



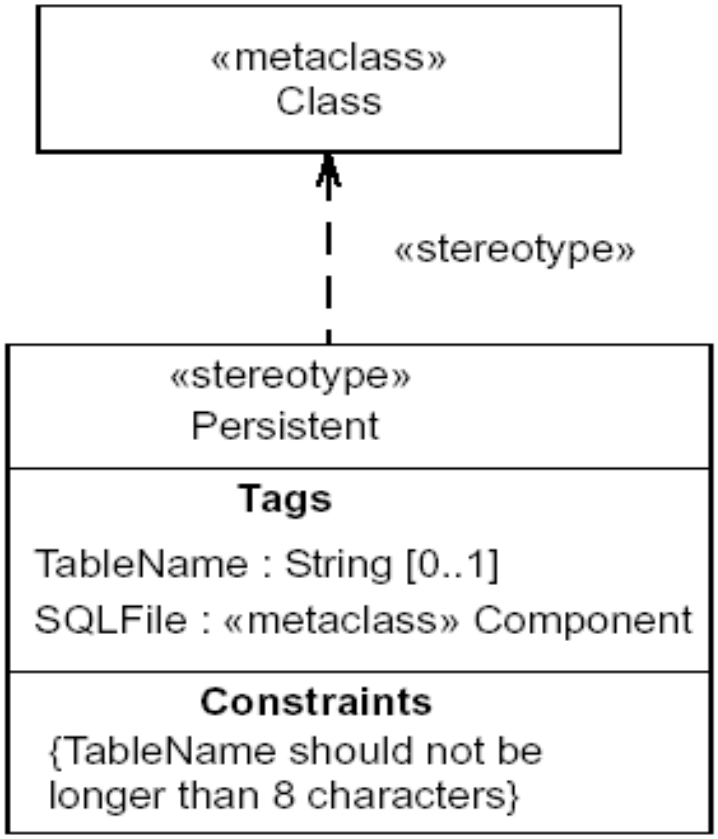
Gliederung

- Einführung in UML Profiles
- Bestehende Werkzeugunterstützung am Beispiel „Objecteering“
- Untersuchung des Werkzeugs „Together“ auf Erweiterbarkeit
- Empfehlungen zur Erweiterung von „Together“
- Prototypische Umsetzung
- Zusammenfassung

Zusammenfassung

- Anforderungen an UML Profiles durch UML 1.4 im Wesentlichen erfüllt
- Objecteering:
  - UML Profiles nach UML 1.3 unterstützt; dabei (wenige) Einschränkungen
  - Proprietäre Mechanismen ermöglichen sehr weitreichend und wohlstrukturiert Zuweisung von Semantik an ein UML Profile
- Together:
  - Erweiterbarkeit mittels Open API eingeschränkt
  - Config-Dateien kaum dokumentiert
  - Erweiterungen teilweise realisierbar

Graphische Definition eines Stereotypes



IDE: Beispiele

- Anpassung des Property-Inspectors (Eigenschaftsdialog)
- Hinzufügen eigene Befehle
- Aufruf vorgegebener oder beliebiger eigener Dialoge
- Anpassung und Nutzung der Konfigurationsdialoge