UNIKASSEL VERSITÄT

SOFTWARE ENGINEERING

Software Engineering II

Software Engineering II – Übung 4

Wintersemester 12/13

Fachgebiet Software Engineering

Andreas Scharf

Wiederholung

- **Bisher im Laufe des Semesters umgesetzt:**
  - Modellierung eines Meta-Modells für Task Flow Diagramme mit Fujaba
  - Generierung von EMF Quelltext aus dem Fujaba Klassendiagramm
  - Generieren eines Editor- und Edit Eclipse Plugins für das EMF Modell (baumartiger Editor)
  - Hinzufügen von Annotationen im Klassendiagramm und automatische Generierung des grafischen Editors aus diesen Annotationen mit Hilfe von Eugenia
  - Implementierung von Refactoring Operationen

Nächstes Ziel

- **Validierung des gezeichneten Diagramms**
  - Diagramme können Inkonsistenzen oder unerlaubte Zustände enthalten.
  - Prüfung der Diagramme um Fehler zu finden automatisieren
  - Regeln (Constraints) für den Aufbau der Diagramme definieren, die über die im Meta-Modell festgelegten Eigenschaften hinaus gehen.
    - Pro Lane nur 1 Startknoten
    - Alle Knoten vom Startknoten aus erreichbar
    - Tasks müssen als Namen einen gültigen Java Identifier haben
    - Etc.

Constraints mit Fujaba bauen

- **Neue Constraint Klasse (z.B. ValidateTaskNames)**
  - **Erbt von** `AbstractModelConstraint` **aus** `EclipseClasses.ctr`
- `validate(context:IValidationContext):Istatus` **als Storydiagramm implementieren**
- **Evtl. neues Validation package für die Validierungsklassen.**
  - Codestyle ist java bzw. inherited, NICHT emf
- **Code generieren**

Validierung in plugin.xml einbauen I

1. **Runtime Tab: Validation package mit exportieren (Runtime Tab)**
2. **Extensions Tab: o.e.e.v.constraintProviders Extension Point hinzufügen**
3. **Auf o.e.e.v.constraintProviders Extension Point:**
   - Rechtsklick -> New –> category
     - Sinnvollen Namen für category vergeben
   - Rechtsklick -> New -> constraintProvider
4. **Auf constraintProvider:**
   - Rechtsklick -> New -> package: Namespace eures Modells
   - Rechtsklick -> New -> constraints

Validierung in plugin.xml einbauen II

5. **Auf constraints:**
   - *categories:* Category eintragen (siehe 3.)
   - *cache:* true
   - Rechtsklick -> New -> constraint
6. **Auf constraint:**
   - Rechtsklick -> New -> message
     - Nachricht bei Validierungsfehlern angzeigt. Unterstützt „Platzhalter“ der Form {0}
   - Rechtsklick -> New -> description
     - Allgemeine Beschreibung des Constraints
   - Rechtsklick -> New -> target
     - Simplename der Modellklasse, auf der der Constraint laufen soll

Validierung in plugin.xml einbauen III

7. Constraint konfigurieren

- *id:* am besten gleich der Klasse
- *name:* beliebig vergeben
- *lang:* Java
- *statusCode:* beliebig (für logging Zwecke)
- *severity:* ERROR
- *class:* Klasse die ihr gerade implementiert
- *mode:* Batch
- *isEnabledByDefault:* true

Validierung in plugin.xml einbauen IV

8. **o.e.e.v.constraintBindings Extension Point hinzufügen**
9. **Auf o.e.e.v.constraintBindings Extension Point**
   - Rechtsklick -> New -> clientContext
10. **Auf clientContext:**
    - *id:* sinnvoll vergeben
    - *default:* true

Validierung in plugin.xml einbauen V

11. **ClientContext enablement konfigurieren**
    - Rechtsklick -> New -> enablement -> and
    - Auf and:
        - Rechtsklick -> New -> instanceof
            - *value:* org.eclipse.emf.ecore.EObject
        - Rechtsklick -> New -> test
            - *property:* Namespace eures Modells + .ePackage
            - *value:* Namespace eures Modells

Validierung in plugin.xml einbauen VI

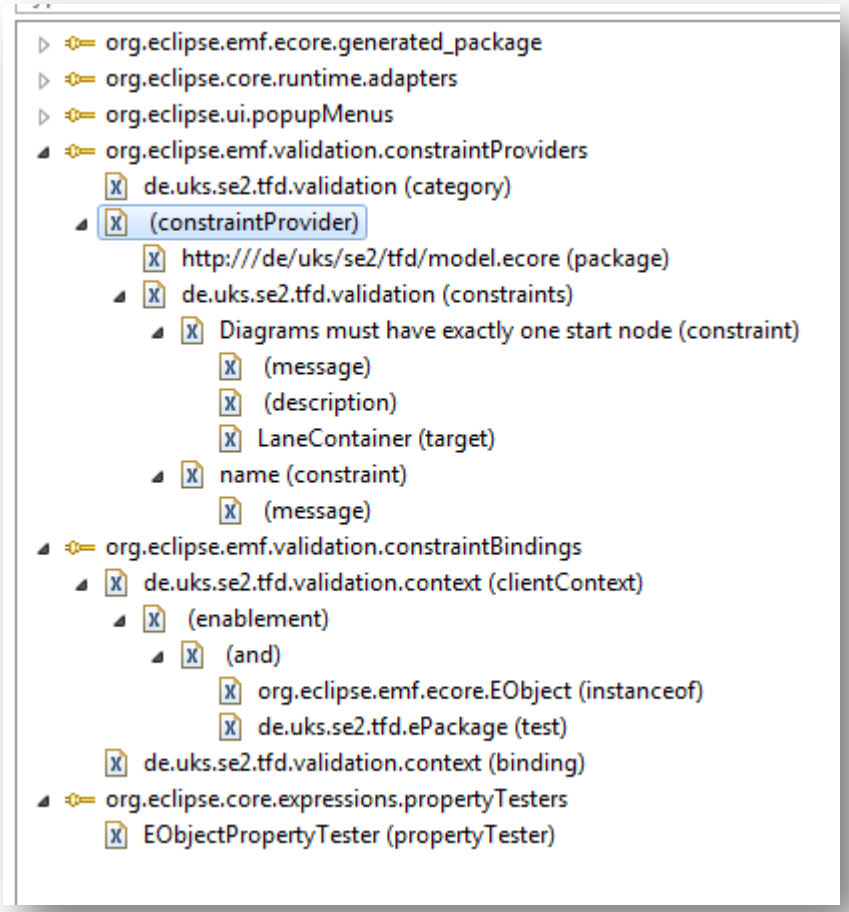
12. **Auf o.e.e.v.constraintBindings Extension Point**
    - Rechtsklick -> New -> binding
13. **Auf binding:**
    - *context:* id des clientContext (siehe 10.)
    - *category:* id der category (siehe 3.)

Validierung in plugin.xml einbauen VII

14. **o.e.c.expressions.propertyTesters Extension Point hinzufügen**
15. **Auf o.e.c.expressions.propertyTesters Extension Point**
    - Rechtsklick -> New –> propertyTester
      - *id:* sinnvoll vergeben
      - *type:* org.eclipse.emf.ecore.EObject
      - *namespace:* Namespace eures Modells
      - *properties:* ePackage
      - class: de.uni_kassel.eclipse.properties.EObjectPropertyTester

Validierung in plugin.xml einbauen VIII

- **Fazit: Viel Arbeit in der plugin.xml!**



Und jetzt: Laufen lassen!

- **Im Runtime Eclipse:**
  - Diagramm mit gewünschtem Fehler malen
  - Edit -> Validate
    - Fehlermeldung im Problems View
    - Fehlericon am entsprechenden Diagrammteil

Hausaufgabe 4 - Validierung

- Nur 1 Startknoten LaneContainer (Diagramm)
- Alle Knoten sind vom Startknoten aus erreichbar
- Tasks müssen als Namen einen gültigen Java Identifier haben

Abgabe bis zum 09.12.2012