Modellezési szabályok ellenőrzése és reverse engineering az ArgoUML eszközben

Hartung István

Tartalom

- Fogalmi háttér
- ArgoUML bemutatása
- Modellezési szabályok ellenőrzése
- ArgoUML kiterjesztések

Fogalmak

- Modell
  - A valóság egy absztrakciója, bizonyos aspektusokat kiemelve, másokat elrejtve
  - Cél: különböző szakterületekre modellek építése, hibák korai detektálása, költségcsökkentés
- Refactoring
  - Olyan átalakítás a kódon vagy modellen, ami nem jár annak szemantikai változásával
  - Cél: kódminőség javítása

Fogalmak/2

- Kódgenerálás
  - M2T transzformáció (futó kód, konfiguráció, tesztek)
  - Módosítható/nem módosítható, kód DOM/sablon alapú
- Reverse Engineering
  - Kódgenerálás ellentéte, létező rendszerhez modell készítése
  - Probléma: túl sok információ → absztrakció végzése
- Round trip engineering
  - Nehézség: modell és kódszinkronizáció

ArgoUML

- Ingyenes, nyílt forráskódú modellező eszköz

  http://argouml.tigris.org/
- 2003: Jason E. Robbins, ma már 150 fejlesztő, 19 ezer regisztrált fejlesztő
- Külön kezeli a modellt és a diagramokat
- Modell:
  - Saját API: UML 1.4 (Netbeans MDR) és 2.2 (Eclipse)
  - Minden eleme egy metamodellbeli osztály példánya
  - OCL kifejezésekkel bővíthető
- Diagramok elemei:
  - Referencia modellelemre
  - Grafikus tulajdonságok

ArgoUML/2

- Képességek:
  - „Felhasználóbarát”, több nézet a modellezéshez
  - Folyamattámogatás: „To do” lista segítségével
  - Modellezés támogatása: „critics” és „checklists”
  - Reverse engineering, refactoring és kódgenerálás támogatása

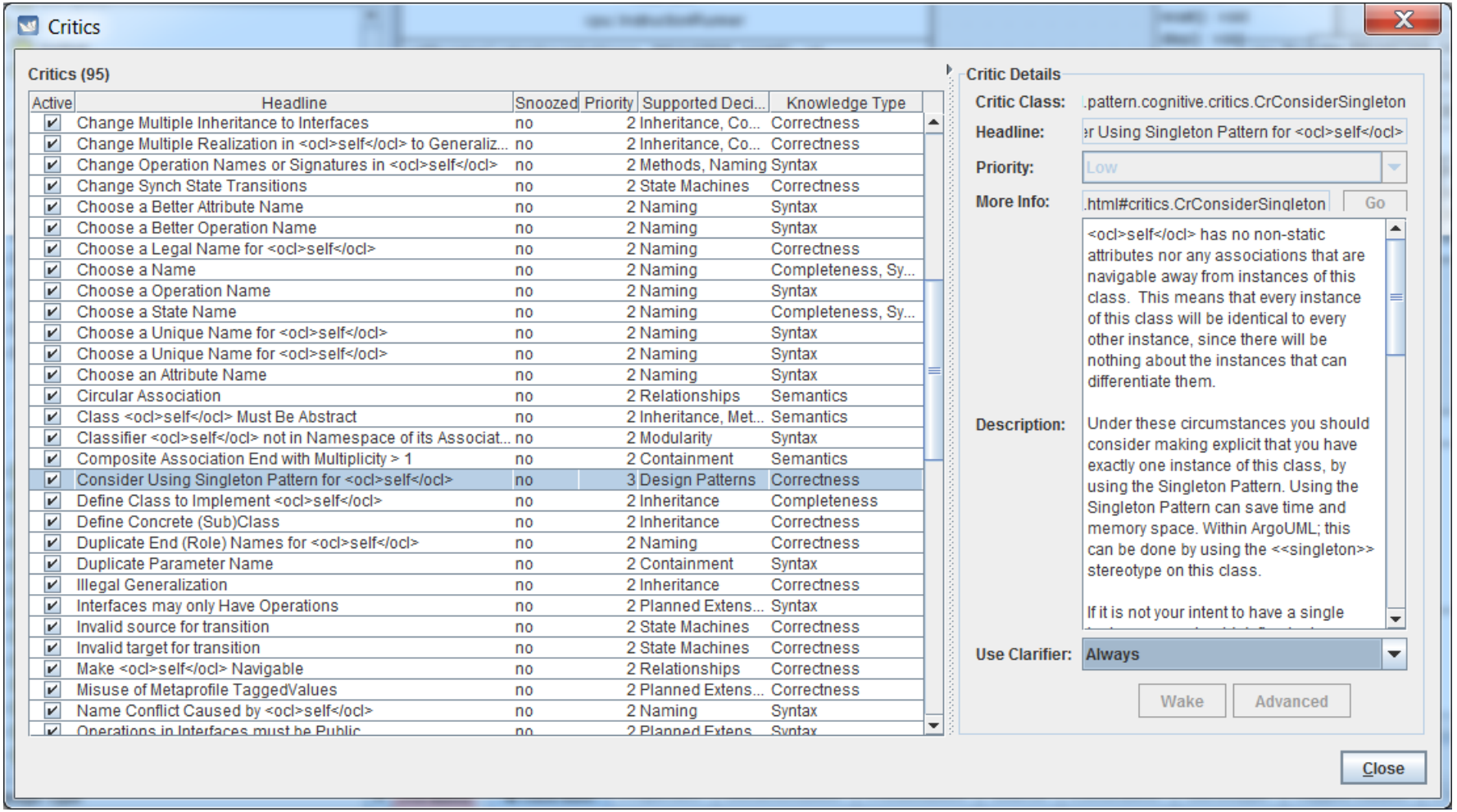
Modell

- Előállítása:
  - XML-ből importálva (XMI formátum)
  - Kódból reverse engineering (Java, C#, C++)
  - UML diagramok segítségével
- Használat:
  - Tervezés, UML diagramok előállítása
  - Modellellenőrzés
  - Kódgenerálás (Java, C#, C++)

Ellenőrzés

- Kikapcsolható szabályok, prioritás és típus szerint kategorizálva, részletes leírással
- Ezek ellenőrzése a felépített modellen történik, ami tartalmazza az objektumokat és a köztük levő kapcsolatokat is.
- A modell ellenőrzése által mind a statikus, mind a dinamikus viselkedési diagramok ellenőrzése megtörténik, valamint azok szintaktikája is ellenőrzésre kerül.
- Javításhoz varázsló támogatás

Szabályok

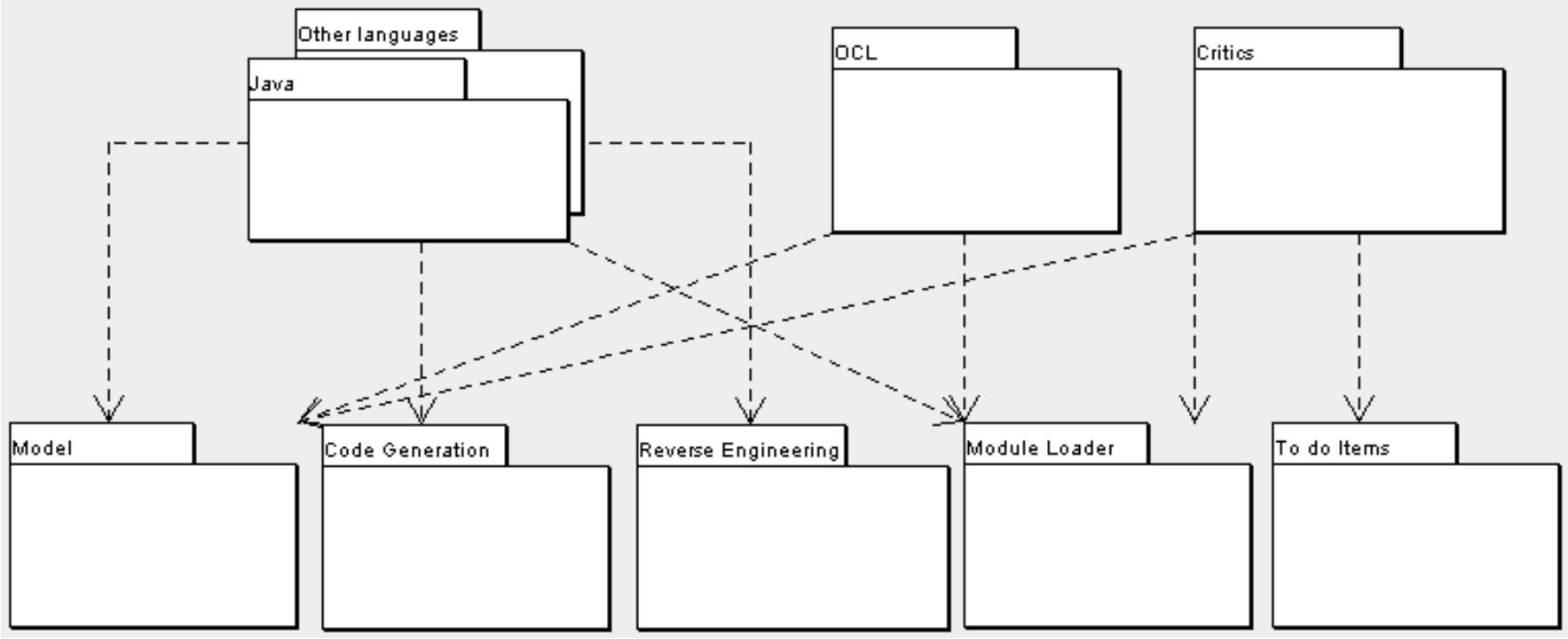

Critics
Critics (95)
Active
Headline
Snoozed
Priority
Supported Deci...
Knowledge Type
Change Multiple Inheritance to Interfaces no 2 Inheritance, Co... Correctness
Change Multiple Realization in <ocl>self</ocl> to Generaliz... no 2 Inheritance, Co... Correctness
Change Operation Names or Signatures in <ocl>self</ocl> no 2 Methods, Naming Syntax
Change Synch State Transitions no 2 State Machines Correctness
Choose a Better Attribute Name no 2 Naming Syntax
Choose a Better Operation Name no 2 Naming Syntax
Choose a Legal Name for <ocl>self</ocl> no 2 Naming Correctness
Choose a Name no 2 Naming Completeness, Sy...
Choose a Operation Name no 2 Naming Syntax
Choose a State Name no 2 Naming Completeness, Sy...
Choose a Unique Name for <ocl>self</ocl> no 2 Naming Syntax
Choose a Unique Name for <ocl>self</ocl> no 2 Naming Syntax
Choose an Attribute Name no 2 Naming Syntax
Circular Association no 2 Relationships Semantics
Class <ocl>self</ocl> Must Be Abstract no 2 Inheritance, Met... Semantics
Classifier <ocl>self</ocl> not in Namespace of its Associat... no 2 Modularity Syntax
Composite Association End with Multiplicity > 1 no 2 Containment Semantics
Consider Using Singleton Pattern for <ocl>self</ocl> no 3 Design Patterns Correctness
Define Class to Implement <ocl>self</ocl> no 2 Inheritance Completeness
Define Concrete (Sub)Class no 2 Inheritance Correctness
Duplicate End (Role) Names for <ocl>self</ocl> no 2 Naming Correctness
Duplicate Parameter Name no 2 Containment Syntax
Illegal Generalization no 2 Inheritance Correctness
Interfaces may only Have Operations no 2 Planned Extens... Syntax
Invalid source for transition no 2 State Machines Correctness
Invalid target for transition no 2 State Machines Correctness
Make <ocl>self</ocl> Navigable no 2 Relationships Correctness
Misuse of Metaprofile TaggedValues no 2 Planned Extens... Correctness
Name Conflict Caused by <ocl>self</ocl> no 2 Naming Syntax
Operations in Interfaces must be Public no 2 Planned Extens... Syntax
Critic Details
Critic Class: .pattern.cognitive.critics.CrConsiderSingleton
Headline: er Using Singleton Pattern for <ocl>self</ocl>
Priority: Low
More Info: .html#critics.CrConsiderSingleton
Go
Description:
<ocl>self</ocl> has no non-static attributes nor any associations that are navigable away from instances of this class. This means that every instance of this class will be identical to every other instance, since there will be nothing about the instances that can differentiate them.
Under these circumstances you should consider making explicit that you have exactly one instance of this class, by using the Singleton Pattern. Using the Singleton Pattern can save time and memory space. Within ArgoUML; this can be done by using the <<singleton>> stereotype on this class.
If it is not your intent to have a single
Use Clarifier: Always
Wake
Advanced
Close

Szabályok/2

Java-ban implementálva, org.argouml.cognitive.Critic ősosztály, ennek implementációi, jelenleg ~100 különböző

- Típusok:
  - Helyesség, pl.: állapotdiagramon kezdőállapot megléte
  - Teljesség, pl.: használatlan interfészek implementálása
  - Szintaxis, pl.: foglalt név feloldása
  - Szemantika, pl.: átalakítás absztrakt osztállyá
  - Prezentáció, pl.: felesleges öröklés eltörlése

Felépítés – opcionális csomagok



ArgoUML - kiterjesztések

- Több publikáció is foglalkozik az ArgoUML kiterjesztésével: http://argouml.tigris.org/docs/
- Statikus ellenőrzés szempontjából:
  - Mark Micallef – 2001: An Automated Software Qualtiy Measurement Tool
  - Jason E. Robbins: Software Architecture Critics in the Argo Design Environment

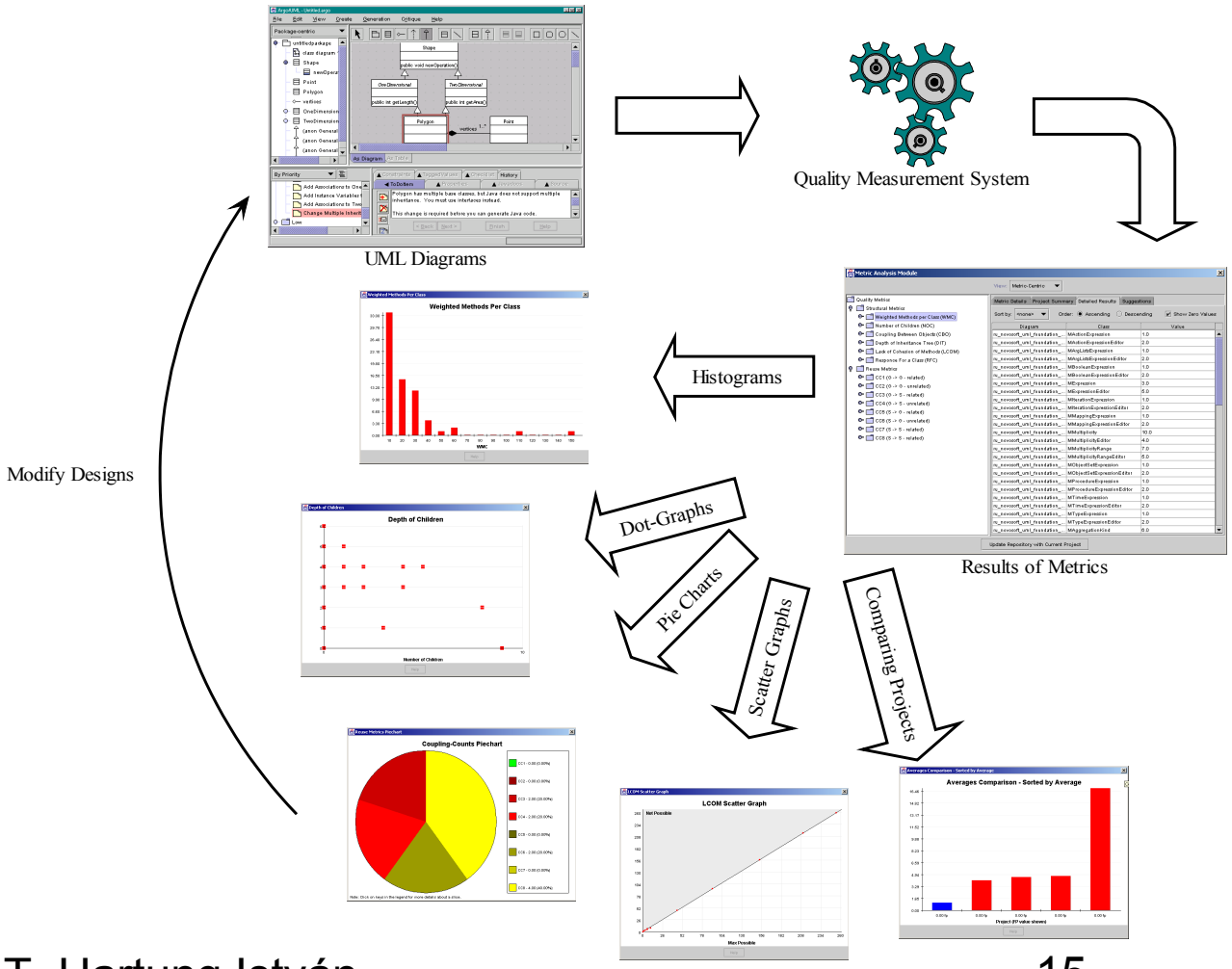
Software Qualtiy Measurement kiterjesztés

- Cél:
  - Terv minőségének mérése
  - Hibák, hibalehetőségek korai detektálása
- OO metrikák mérése:
  - Strukturális metrikák: WMC (Weighted Methods per Class), DIT (Depth of Inheritance Tree), NOC (Number of Children), CBO (Coupling Between Objects), LCOM (Lack of Cohesion of Methods)
  - Újrahasználhatósági metrikák: csatolás alapján

Software Qualtiy Measurement kiterjesztés - működés

- Adatok gyűjtése a modellből és a diagramokból ArgoUML API-n keresztül
- Osztálydiagramok, Activity diagramok, Szekvencia diagramok, Kollaborációs diagramok felhasználása
- Grafikonok, kimutatások generálása a kész adatokból
- Ezek elemzéséből következtetni lehet hibapontokra, újratervezendő pontokra a modellben

Software Qualtiy Measurement kiterjesztés – működés/2



Összefoglalás

- Hibák minél korábbi detektálásával költségcsökkentés
- Modellezés kulcsfontosságú
  - Modellek analízise
  - Hibamentes kód generálása
  - Meglévő kódból modell előállítása
- Lehetséges eszköz: ArgoUML