(Pro-)Seminar Softwaresystemtechnik (SST)

“Refactorings”

(SS 15, Proseminar 2 CP | Seminar 4 CP)

TECHNISCHE UNIVERSITÄT DARMSTADT

Einführungsveranstaltung

Johannes Bürdek

johannes.buerdek@es.tu-darmstadt.de
Tel.+49 6151 16 76089

ES Real-Time Systems Lab
Prof. Dr. rer. nat. Andy Schürr
Dept. of Electrical Engineering and Information Technology
Dept. of Computer Science (adjunct Professor)

www.es.tu-darmstadt.de

14.04.2016

Herzlich Willkommen!

TECHNISCHE UNIVERSITÄT DARMSTADT

Als Proseminar (mit reduzierten Ansprüchen, 2 CP):

- B.Sc. ETiT (5. Sem.)

Als Seminar (4 CP):

- B.Sc. (5. Sem.) und M.Sc. Informatik (2. Sem.)
- Dipl. Informatik
- B.Sc. Informationssystemtechnik (5. Sem.)
- Dipl. ETiT (DT, Hauptstudium)
- entsprechende Wirtschaftsstudiengänge
- Sonstige passende Fachrichtung

Was wir von den Teilnehmern erwarten…

TECHNISCHE UNIVERSITÄT DARMSTADT

- Interesse am Thema + Motivation
- Wille zur Zusammenarbeit mit
  - Betreuer
  - Kommilitonen bei einer Gruppenarbeit
- Wissenschaftliches Vorgehen (unter Anleitung)
- Fristgerechte Abgabe der geforderten Arbeiten
- Teilnahme an *allen* Pflichtveranstaltung

ACHTUNG: Das Seminar ist inhaltlich und vom Umfang her anspruchsvoll!
→Wir geben uns Mühe bei der Betreuung, und erwarten im Gegenzug von allen Teilnehmern ebenfalls **vollen Einsatz**!!!

Ziele

TECHNISCHE UNIVERSITÄT DARMSTADT

- Grundfertigkeiten zur Erstellung einer **wissenschaftlichen Arbeit**
  - Selbständiges Erarbeiten eines Themengebietes (unter Anleitung)
  - Literaturrecherche
    → finden, lesen, verstehen, bewerten
  - Wissenschaftliches Schreiben
    → Gliedern, Zitieren, Formulieren

- Mitwirken am **Reviewprozess**
  - Verwendbares Feedback zu fremden Arbeiten geben
    → Gegenseitige Unterstützung, Schwachstellen identifizieren

- **Präsentation**
  - Aufbereiten, bewerten der Ergebnisse
  - Vorstellen der Ergebnisse
    → Techniken, Stil, Zeiteinteilung, Reden vor der Gruppe

Organisatorisches

TECHNISCHE UNIVERSITÄT DARMSTADT

- Heute
  - Themenvorstellung
  - Themenvergabe durch uns

- Während des Semesters
  - Erstellen einer Ausarbeitung
    → (Auf-)Schreiben von (Zwischen-) Ergebnissen
  - Vortrag vorbereiten
  - Regelmäßige Absprachen mit Betreuer!
    → Individuelle Absprachen
    → Fortschritt, Fragen, Feedback, Tipps

- Am Ende des Semesters **(4. + 5. Juli)**
  - Vortrag im Blockseminar → Präsentation + Ausarbeitung liegen bereits vor

Plagiatshinweis – „Abschreiben“ verboten!

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wir messen der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei.

- Mit der Abgabe einer Lösung (Hausaufgabe, Programmierprojekt, Diplomarbeit, etc.) bestätigen Sie, dass (Sie/Ihre Gruppe) **(der alleinige Autor/die alleinigen Autoren)** des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen **Quellen deutlich zitiert haben**.

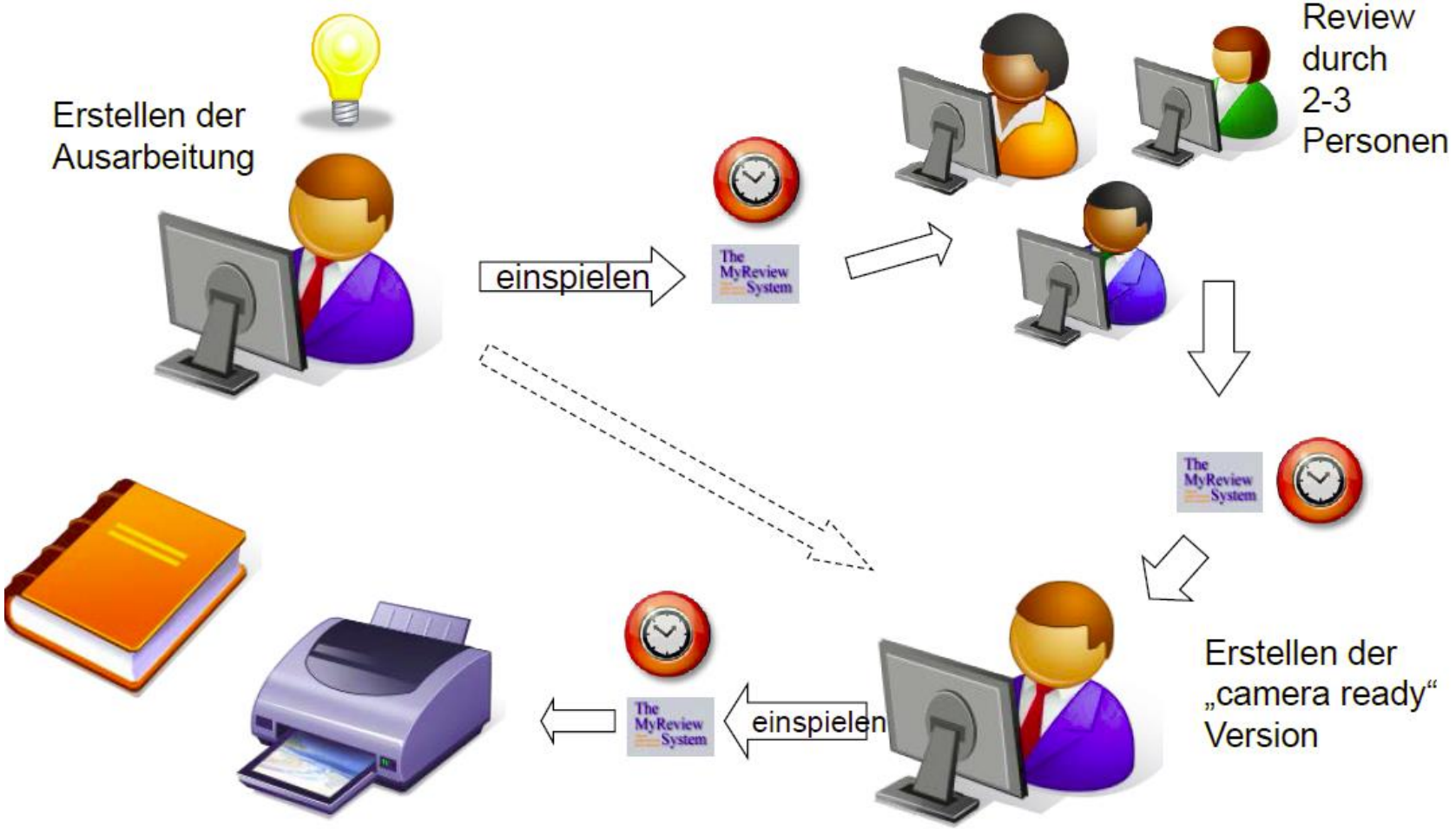
- Weiterführende Informationen unter http://www.es.tu-darmstadt.de/lehre/plagiat/

Wichtige Links zur Veranstaltung

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Seminarrichtlinien**
  - http://www.es.tu-darmstadt.de/fileadmin/download/lehre/Seminarrichtlinien.pdf
- **Plagiatshinweise**
  - http://www.es.tu-darmstadt.de/lehre/plagiat/
- **Schreibkurse**
  - http://www.owl.tu-darmstadt.de/owl_ueber_uns/ueber_uns_1.de.jsp
  - http://www.hda.tu-darmstadt.de/angebote_fuer_studierende_zentral/trainings_fuer_schluesselkompetenzen_margin/startseite_stud_sk/startseite_stud_sk.de.jsp

Übersicht Review-Prozess



Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

Refactorings

- Wichtiger Teil der Software-Entwicklung, -Wartung und -Evolution

Definition: Änderung der Struktur von Software ohne Verhaltensänderung

Ziel:

- Verbesserung von Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit
- Reduzierung des Aufwands für Fehleranalysen und funktionale Erweiterungen

Zeitplan

TECHNISCHE UNIVERSITÄT DARMSTADT

Datum	Topic	Abgabe
14.04.	Auftaktveranstaltung	-
09.05.	Gliederung der Ausarbeitung	Betreuer
06.06.	Erste Fassung der Ausarbeitung	moodle
13.06.	Abgabe der Reviews	moodle
27.06.	Endfassung der Ausarbeitung	moodle
30.06.	Endfassung der Vortragsfolien	Betreuer
04. + 05.07.	Abschlussvortrag	-

Themenübersicht

PS = Proseminar-geeignet

Thema	PS
Reasoning about product-line evolution using complex feature model differences	X
Automated Testing of Refactoring Engines	
Integration of Smells and Refactorings	X
Propagating Model Refactorings to Graph Transformation Rules	
Refactoring von Software-Produktlinien	X
Variant-Preserving Refactoring in Feature-Oriented Software Product Lines	
Evaluation of Model Transformation Approaches for Model Refactoring	
Eliminating Design Defects or Refactoring to Patterns?	X
Refactoring Communication Systems	
Die tatsächliche Anwendung von Refactoring	X
Differential Precondition Checking	
Characterizing Meta-Model Changes	X
Specifying Integrated Refactoring with Distributed Graph Transformations	

Themenvorstellungen

TECHNISCHE UNIVERSITÄT DARMSTADT

Automated Testing of Refactoring Engines

SST Seminar – SS 16

(Proseminar geeignet)

Johannes Bürdek

Automated Testing of Refactoring Engines (Proseminar geeignet)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kompilierungsfehler und Verhaltensänderungen werden üblicherweise durch **Pre-Conditions für Refactorings** ausgeschlossen

- **Pre-Conditions sind jedoch nicht formalisiert**, was zu fehlerhaften Refactoring-Definitionen in Refactoring-Engines führen kann

- Deswegen müssen Refactoring-Engines getestet werden

Aufgabe: Beschreiben von Ansätzen zum automatisierten Testen von Refactoring-Engines

G. Soares, R. Gheyi and T. Massoni: Automated Behavioral Testing of Refactoring Engines, in *Software Engineering*, vol. 39, no. 2, pp. 147-162, Feb. 2013.
Brett Daniel, Danny Dig, Kely Garcia, and Darko Marinov: Automated testing of refactoring engines. In *FSE*. ACM, 185-194, 2007.

TECHNISCHE UNIVERSITÄT DARMSTADT

Reasoning about product-line evolution using complex feature model differences

SST Seminar – SS 16

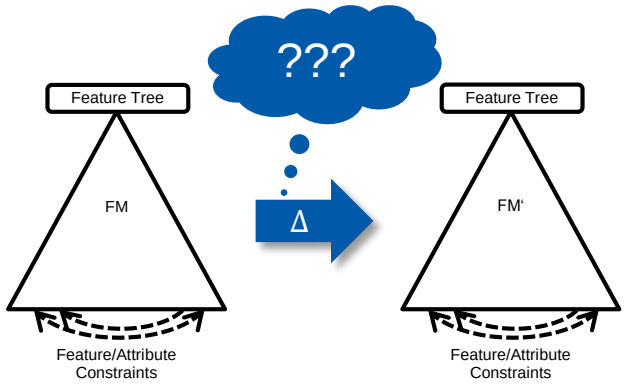
(Seminar)

Johannes Bürdek

Reasoning about product-line evolution using complex feature model differences (Seminar)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Variabilität von Software-Produktlinien (SPL) wird durch Feature-Modelle beschrieben



- SPL evolvieren mit der Zeit
  - Hinzufügen, Löschen, Verändern von Features, …

- Probleme:
  - Änderungen werden nicht sauber dokumentiert
  - Semantische Auswirkungen sind oft unklar

Aufgabe: Beschreiben von Ansätzen zur automatisieren Dokumentation von Änderungen und deren semantischen Auswirkungen

Bürdek, J.; Kehrer, T.; Lochau, M.; Reuling, D.; Kelter, U. & Schürr, A.: Reasoning about product-line evolution using complex feature model differences *Automated Software Engineering,* **2015**, 1-47

TECHNISCHE UNIVERSITÄT DARMSTADT

Integration of Smells and Refactorings

SST Seminar – SS 16

(Proseminar geeignet)

Erhan Leblebici

Integration of Smells and Refactorings

TECHNISCHE UNIVERSITÄT DARMSTADT

1-2 Studenten

als Seminar oder als Proseminar

Umfang wird abhängig von Studentenanzahl und Studiengang bestimmt.

Defining and Checking Model Smells: A Quality Assurance Task for Models based on the Eclipse Modeling Framework

(T. Arendt, M. Burhenne, G. Taentzer)

EMF Refactor: Specification and Application of Model Refactorings within the Eclipse Modeling Framework

(T. Arendt, F. Mantz, G. Taentzer)

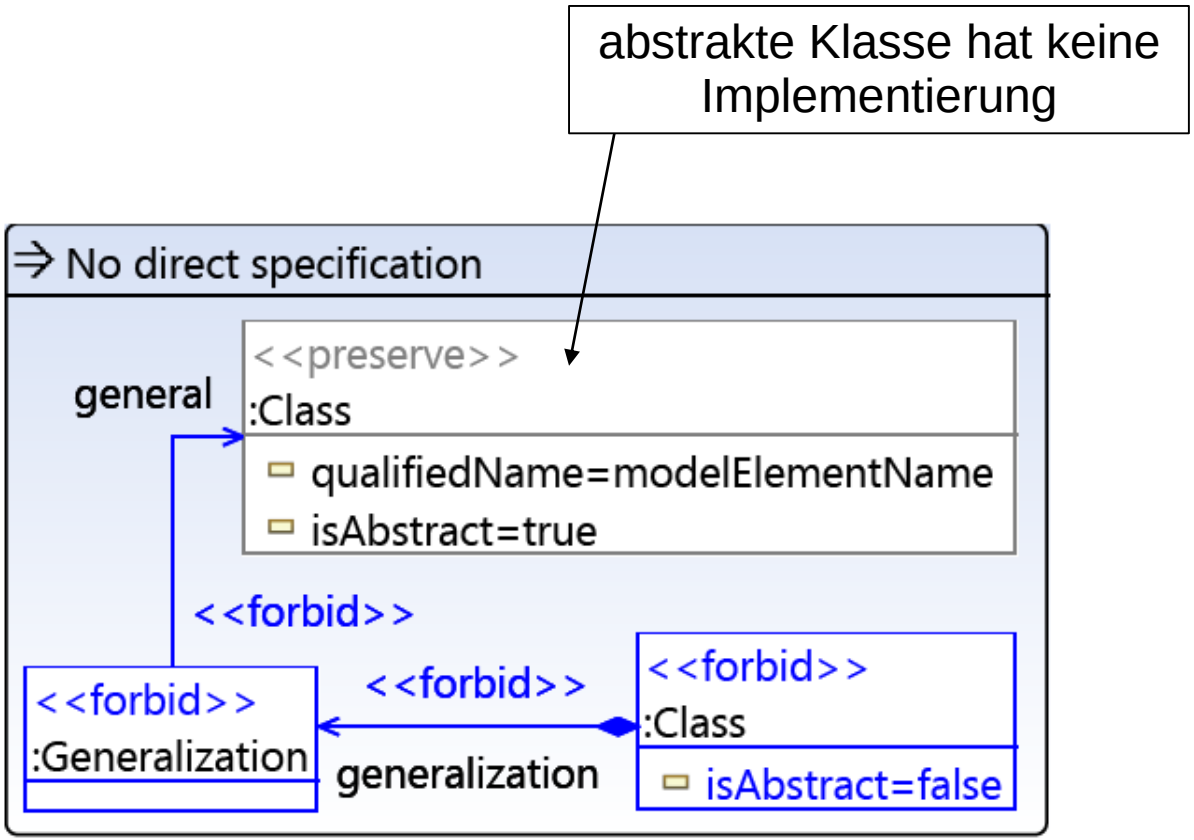
Integration of Smells and Refactorings within the Eclipse Modeling Framework

(T. Arendt, G. Taentzer)

Integration of Smells and Refactorings

TECHNISCHE UNIVERSITÄT DARMSTADT

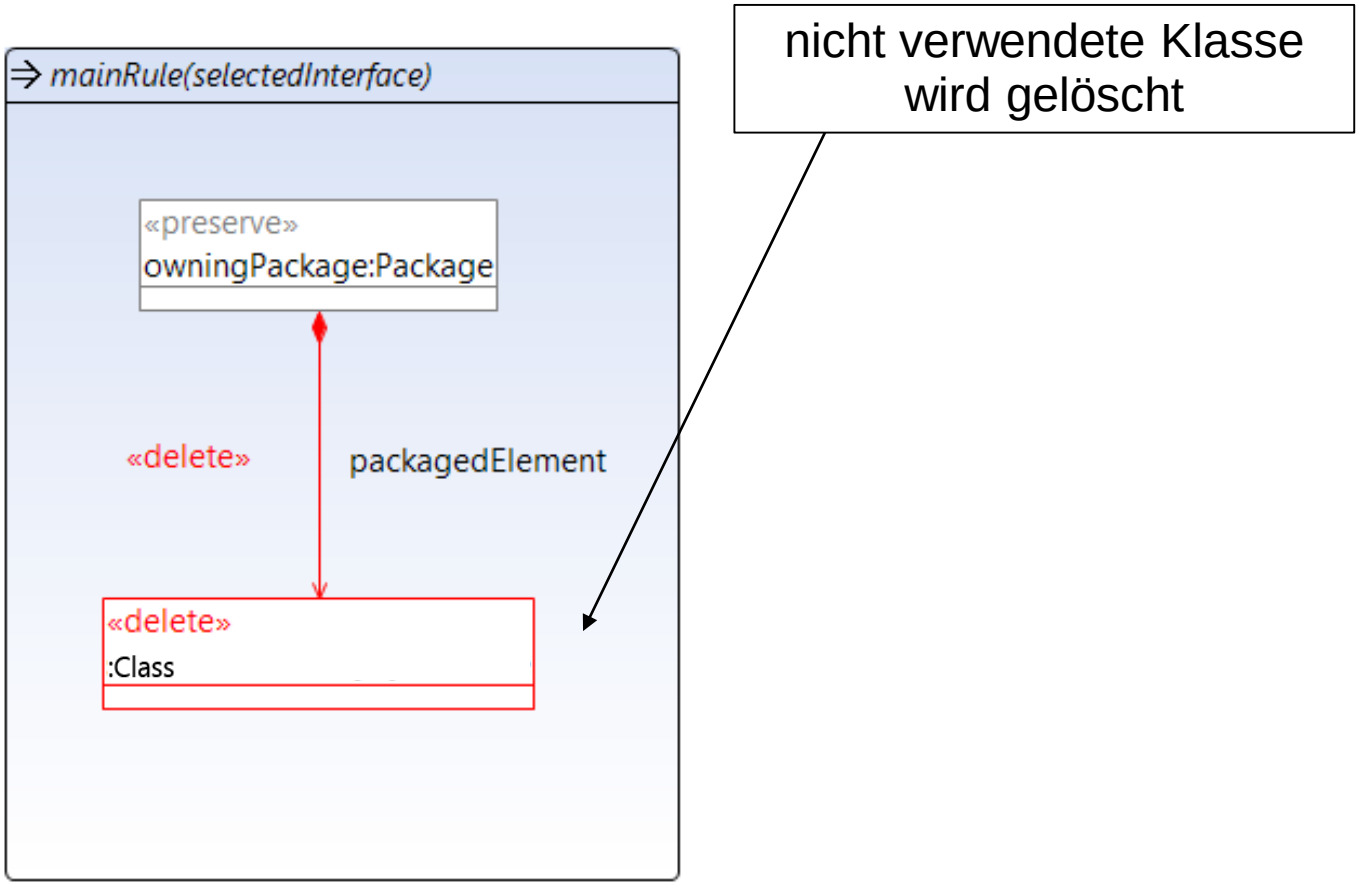
Smell



Integration of Smells and Refactorings

TECHNISCHE UNIVERSITÄT DARMSTADT

Refactoring



Integration of Smells and Refactorings

TECHNISCHE UNIVERSITÄT DARMSTADT

Kombination von Smell- und Refactoring-Regeln

Welches Refactoring **löst** welches Smell?
Welches Refactoring **verursacht** welches Smell?

Integration of Smells and Refactorings

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgaben / Bestandteile der Ausarbeitung:

- Einarbeitung in beide Ansätze und deren Kombination
- **Sehr Wichtig:** Vorbereitung eines demonstrativen Beispiels zusammen mit dem Betreuer (auf Papier)
- (Kritische) Diskussion über die Grenzen und Fähigkeiten des Ansatzes

Voraussetzungen:

- UML-Kenntnisse (Klassendiagramme)
- Interesse an Eclipse Modeling Framework
- Interesse an Literatur-Recherche (3 Papers als Ausgangspunkt)

TECHNISCHE UNIVERSITÄT DARMSTADT

Propagating Model Refactorings to Graph Transformation Rules

SST Seminar – SS 16

(Proseminar geeignet)

Erhan Leblebici

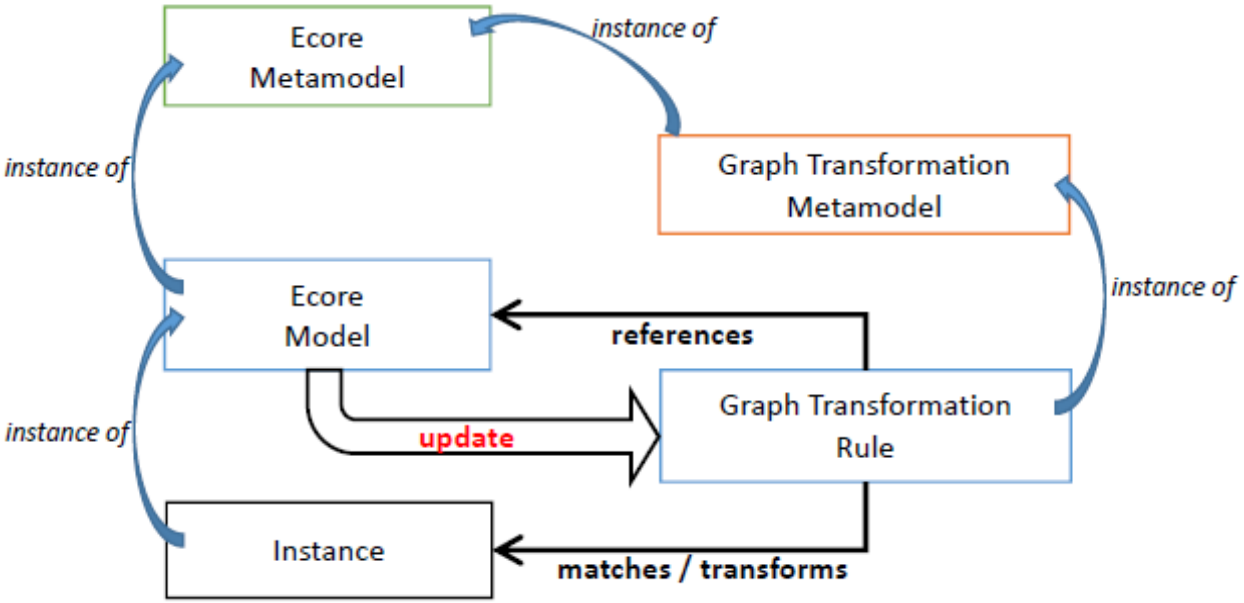
Propagating Model Refactorings to Graph Transformation Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

1 Student – nur Seminar

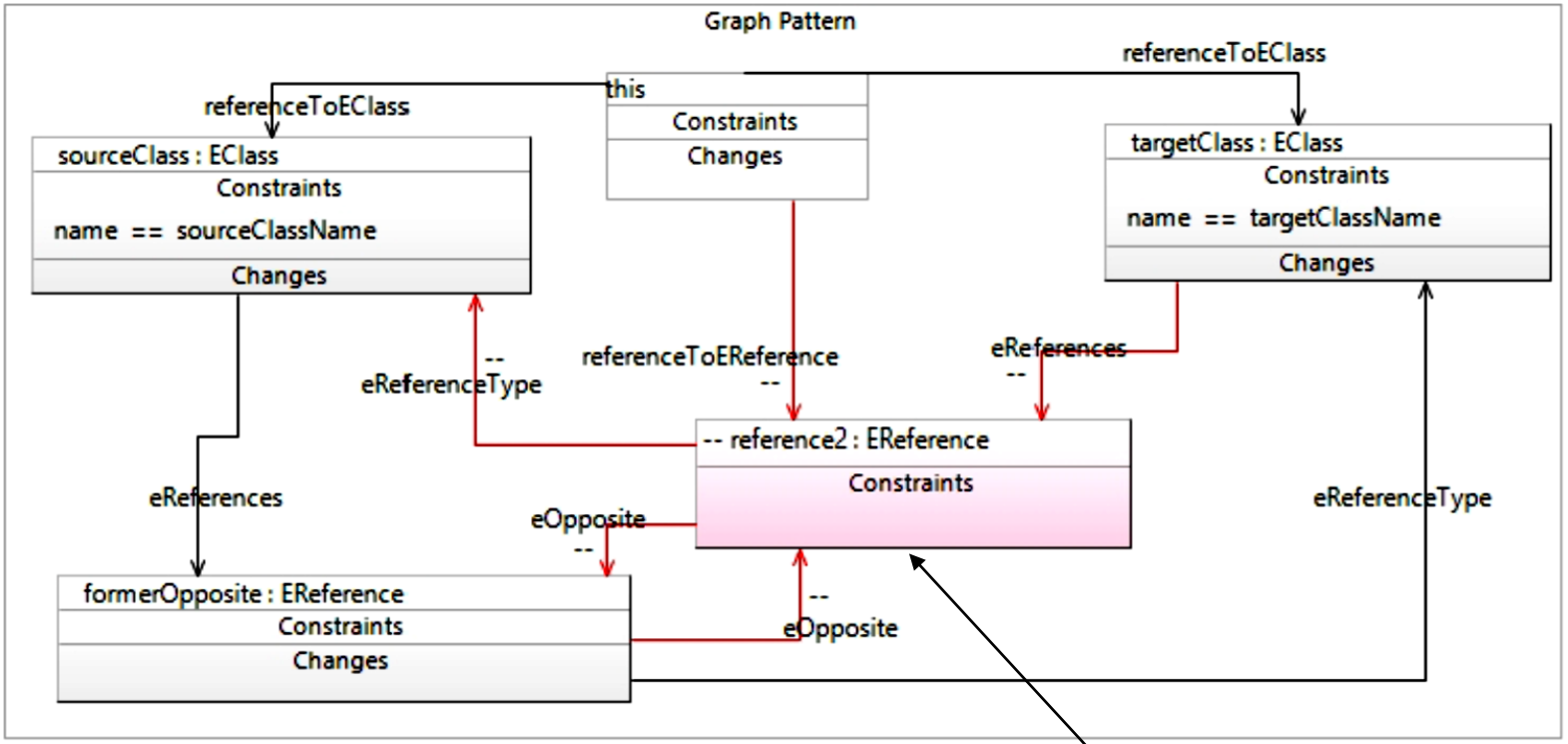
Propagating Model Refactorings to Graph Transformation Rules

S. Winetzhammer, B. Westfechtel



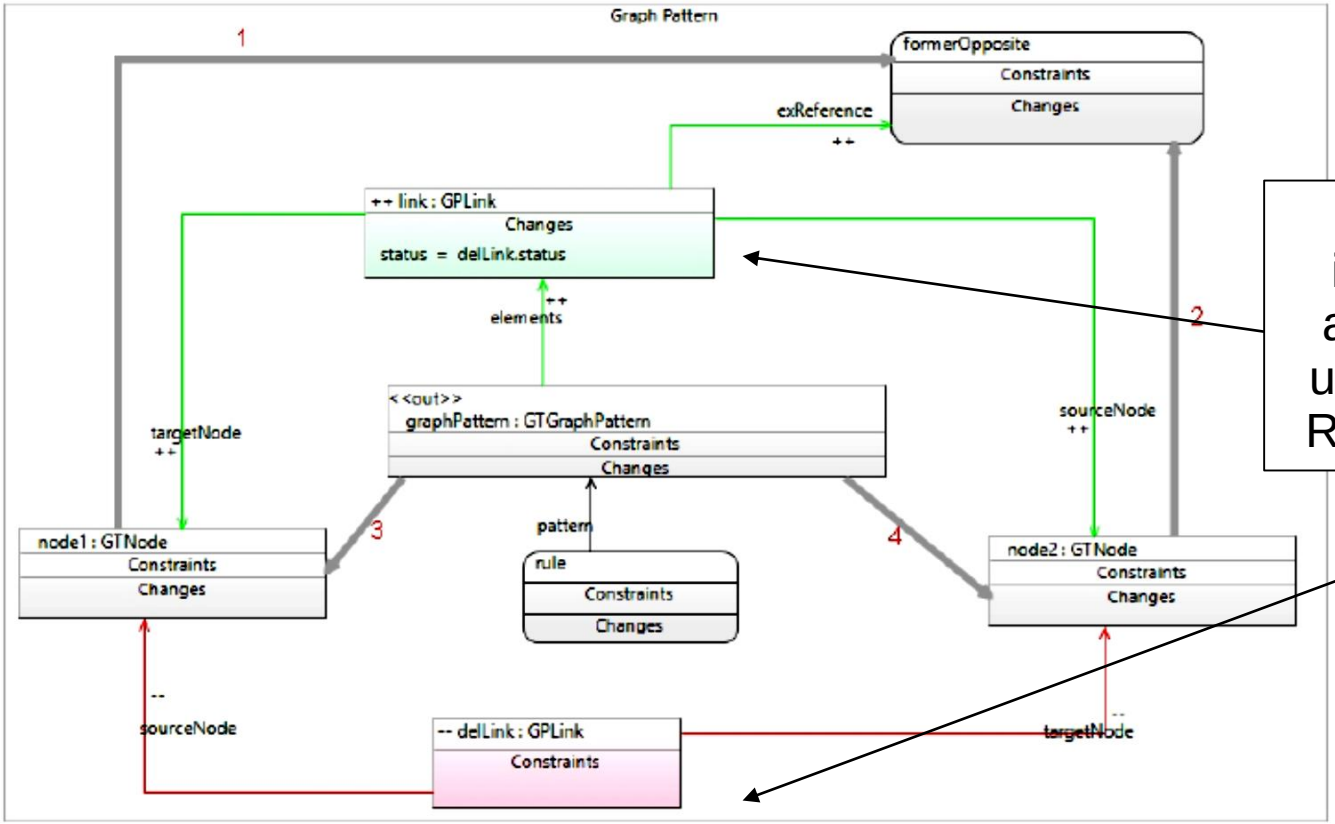
Propagating Model Refactorings to Graph Transformation Rules

TECHNISCHE UNIVERSITÄT DARMSTADT



Eine bidirektionale Referenz wird zu einer unidirektionalen Referenz

Propagating Model Refactorings to Graph Transformation Rules



Regel wird angepasst, indem die Instanze der alten Referenz gelöscht und Instanzen der neuen Referenz erzeugt werden

Propagating Model Refactorings to Graph Transformation Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

Katalog von unterstützten Refactorings

Affected rule element	Change in Ecore model	Change modeled in meta-rule
Whole rule	*Class* rename delete make abstract *Operation* rename delete	 adapt references to class adapt references to class, if not successful: delete adapt references to class adapt references to operation delete
Nodes	*Class* rename delete make abstract	 retype delete retype or delete
Attributes	*Attribute in Class* rename move retype delete	 adapt name try to move, if not successful: delete retype and assign new value delete
Links	*Reference* rename retarget delete	 adapt name try to retarget, if not successful: delete delete
Textual conditions, paths and fields	any change	parse condition and propagate re-factoring, if refactored element is found

Propagating Model Refactorings to Graph Transformation Rules

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgaben / Bestandteile der Ausarbeitung:

- Einarbeitung in Graph Transformation
- Demonstration des angegebenen Refactoring-Katalogs
- **Sehr Wichtig:** (Kritische) Diskussion über die Grenzen und Fähigkeiten des Ansatzes

Voraussetzungen:

- UML-Kenntnisse (Klassendiagramme)
- Interesse an Literatur-Recherche (alternative Ansätze)

TECHNISCHE UNIVERSITÄT DARMSTADT

Refactoring von Software-Produktlinien

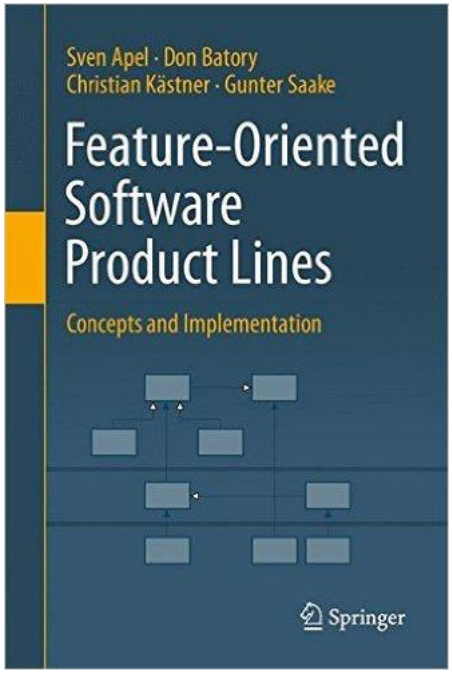
SST Seminar – SS 16

(Proseminar geeignet)

Markus Weckesser

Refactoring von Software-Produktlinien

(Pro-)Seminar



Ablauf

1. Buchkapitel lesen (Umfang kann nach Absprache individuell angepasst werden)
2. Interessante Fragestellungen aus Buchkapitel selektieren
3. Dokumentation der Ergebnisse
4. Review-Prozess

TECHNISCHE UNIVERSITÄT DARMSTADT

Variant-Preserving Refactoring in Feature-Oriented Software Product Lines

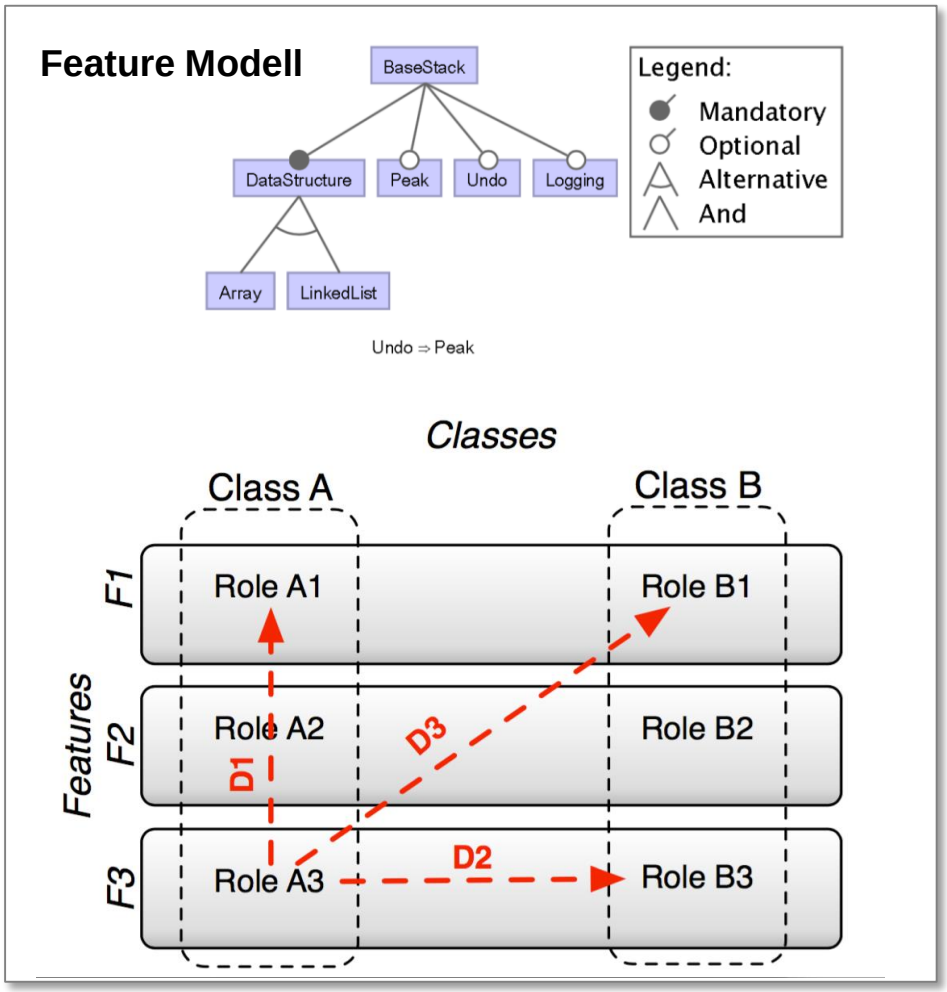
SST Seminar – SS 16

(Seminar)

Markus Weckesser

Variant-Preserving Refactoring in Feature-Oriented Software Product Lines

Seminar



Schulze, Sandro, et al. "Variant-preserving refactoring in feature-oriented software product lines." *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 2012.

Ablauf

1. Paper lesen (Umfang kann nach Absprache individuell angepasst werden)
2. Paper diskutieren und offene Fragen klären
3. Interessante Fragestellungen aus Paper ggf. zusätzlich recherchieren
4. Dokumentation der Ergebnisse
5. Review-Prozess

TECHNISCHE UNIVERSITÄT DARMSTADT

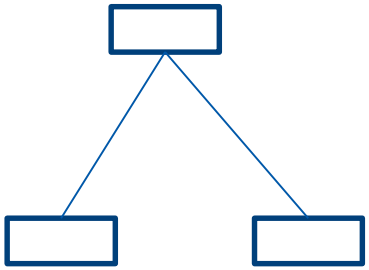
Evaluation of Model Transformation Approaches for Model Refactoring

SST Seminar – SS 16

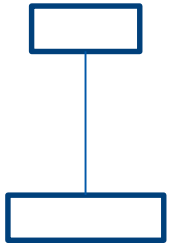
(Seminar)

Géza Kulcsár

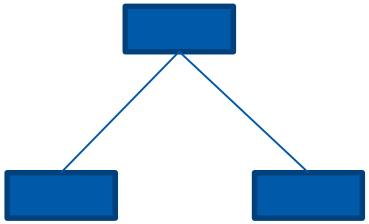
What is model transformation?



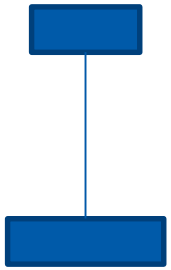
Source meta-model



Target meta-model

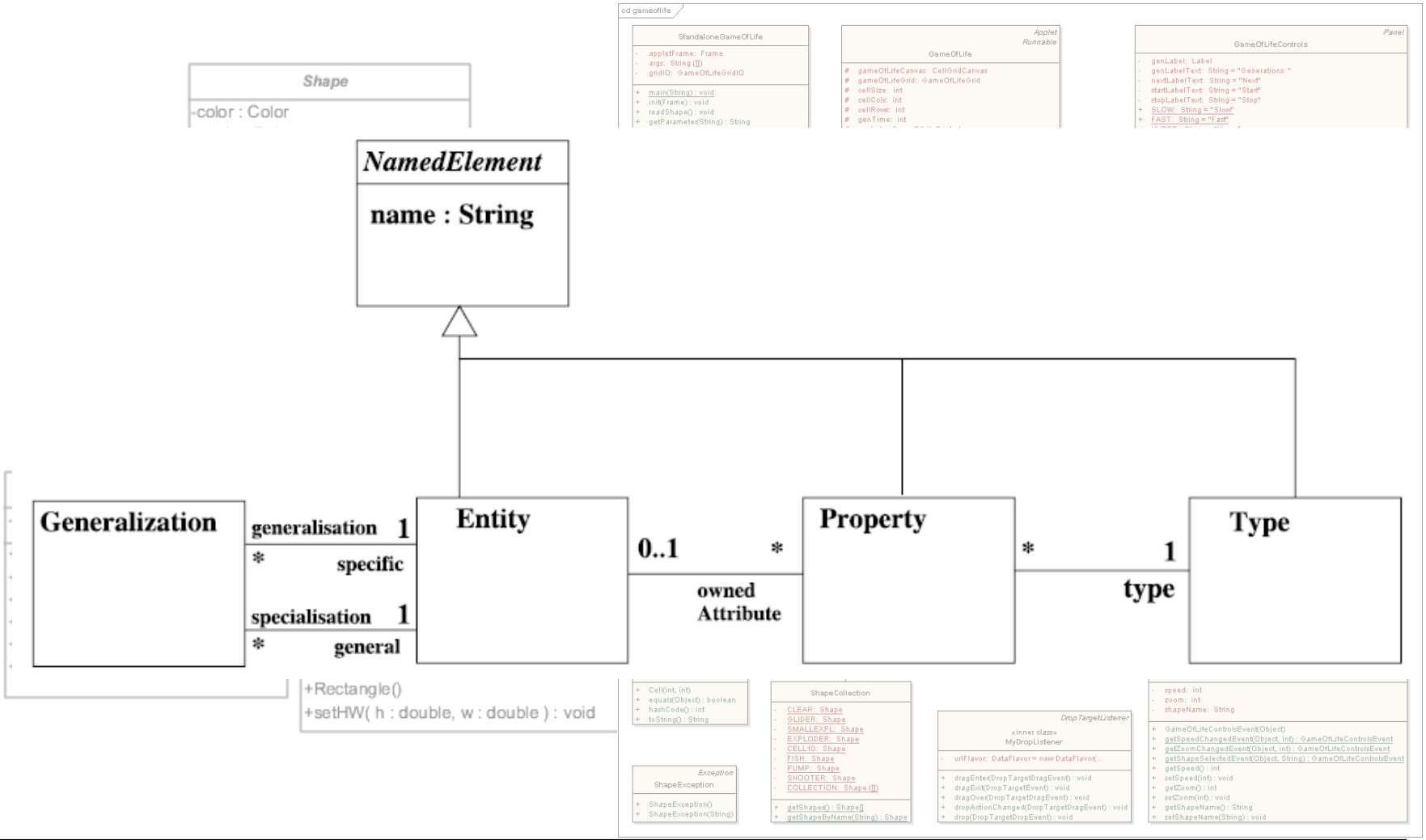


Source model



Target model

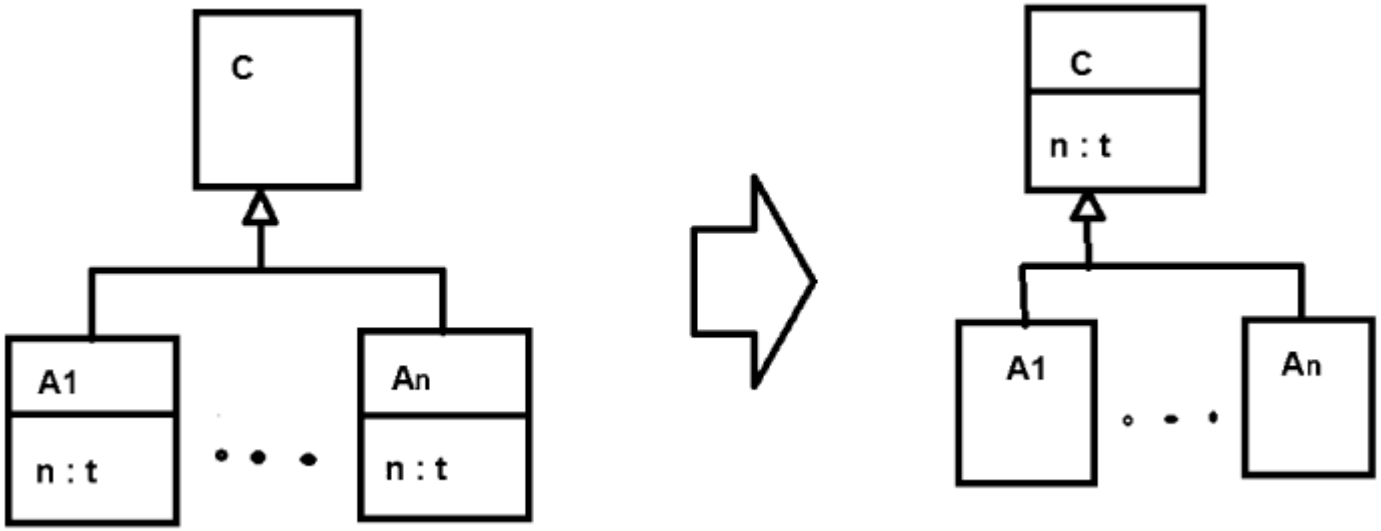
UML Class Diagrams



Class Diagram Refactoring

TECHNISCHE UNIVERSITÄT DARMSTADT

- Refactoring: improving structure while retaining meaning



Model Refactoring Approaches

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kolahdouz-Rahimi et al.: Evaluation of Model Transformation Approaches for Model Refactoring (2014)
  - Kermeta: imperative
  - **QVT-R: declarative (rule-based)**
  - **ATL: hybrid (imperative + declarative)**
  - UML-RSDS: general purpose MDE
  - **GrGen.NET: graph transformation**
  - **eMoflon SDM: programmed graph transformation (?)**

Goals of the Seminar

TECHNISCHE UNIVERSITÄT DARMSTADT

- Getting familiar with model transformation approaches based on the survey
- Reproducing some of the implemented solutions with focus on rule-based approaches
- Devising an own evaluation framework to analyze usability and performance
- (Optional) Investigate the applicability of eMoflon for the given case study

TECHNISCHE UNIVERSITÄT DARMSTADT

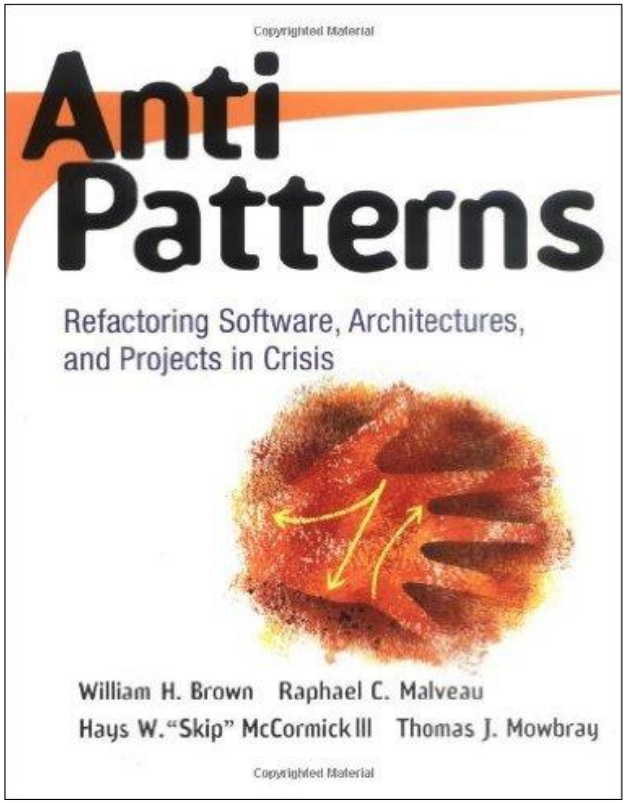
Eliminating Design Defects or Refactoring to Patterns?

SST Seminar – SS 16

(Proseminar geeignet)

Géza Kulcsár

Anti-Patterns

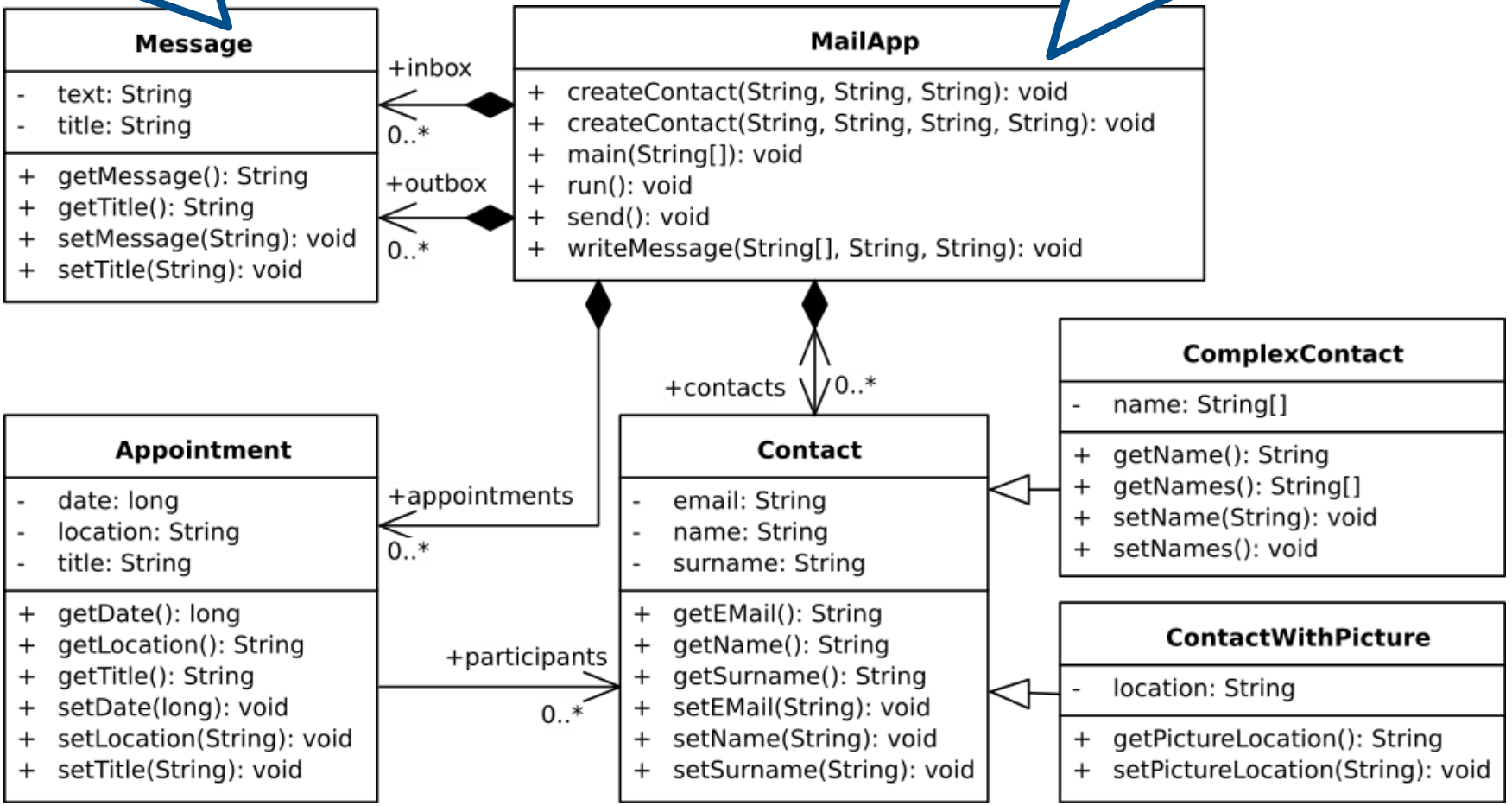


"An Anti-Pattern is a literary form that describes a commonly occurring solution to a problem that generates decidedly negative consequences."

- William Brown, 1998

Data Classes: storage (only getters/setters)

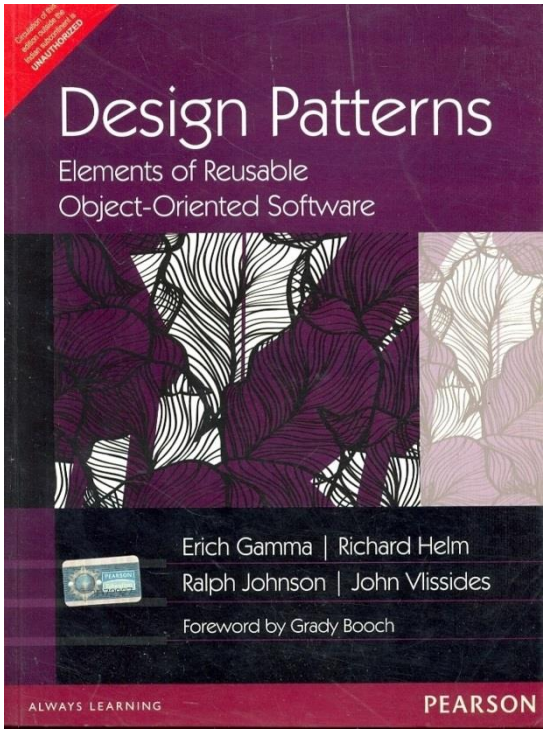
Controller Class: monopolizes processing



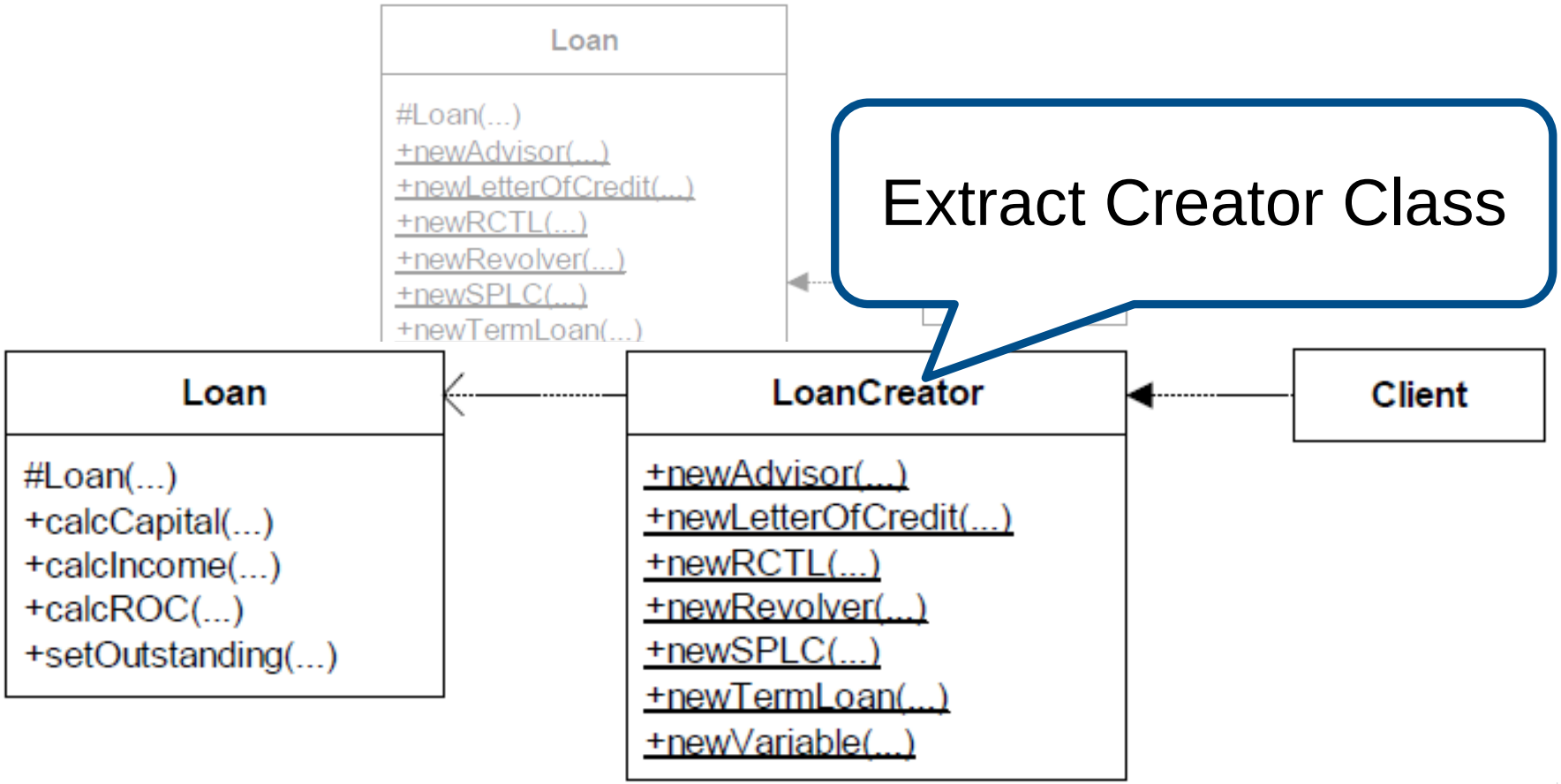
Design Patterns

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gamma et al.: Design Patterns. Elements of Reusable Object-Oriented Software (1994)
  - also known as **Gang of Four**



Introduce Patterns into Legacy Software

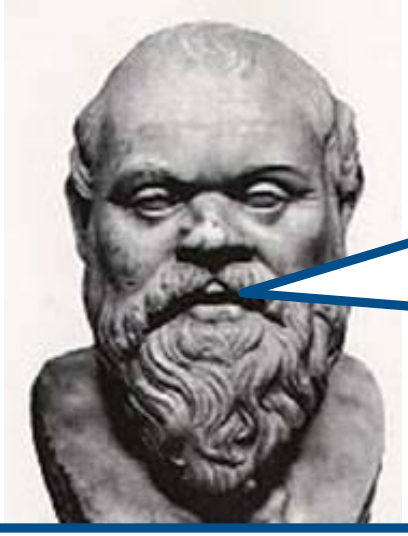


Refactoring as Maintenance

TECHNISCHE UNIVERSITÄT DARMSTADT

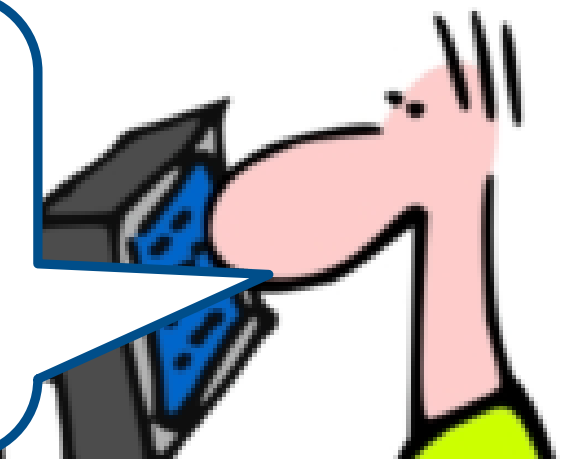
- Maintenance consumes about 60% of overall development costs, 60% of which is spent with **refactoring**
- Refactoring: improving structure while retaining behavior
- That is, refactoring does not aim at functional corrections (i.e., eliminating bugs) but at enhanced **maintainability and extensibility**

But, how to refactor?



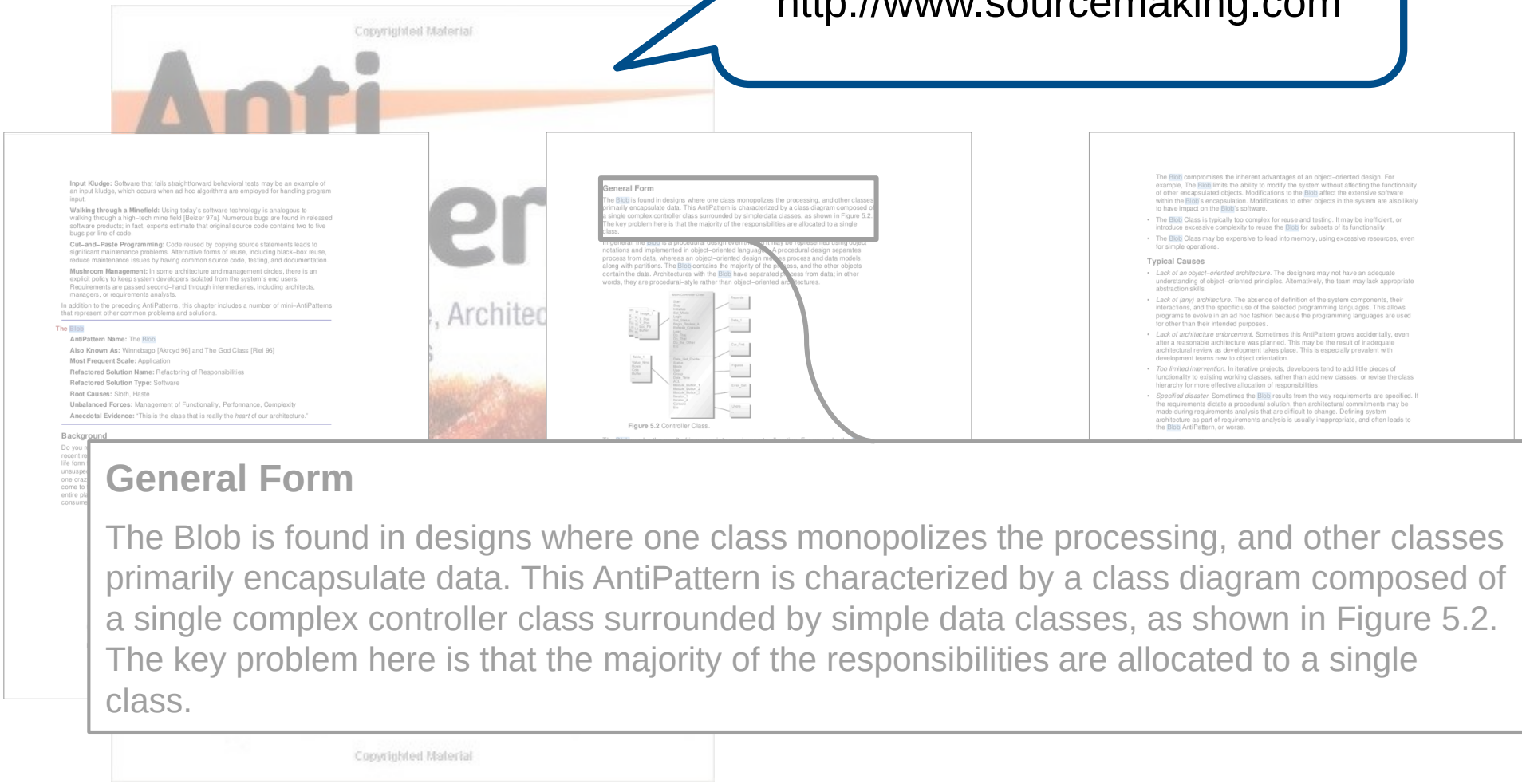
What is better: to ged rid of ugliness or to strive for beauty?

Let us analyze established object-oriented patterns and anti-patterns! How to achieve them, how to eliminate them?



Catalog of Anti-Patterns

Also available at (in a nicer form):
http://www.sourcemaking.com



General Form

The Blob is found in designs where one class monopolizes the processing, and other classes primarily encapsulate data. This AntiPattern is characterized by a class diagram composed of a single complex controller class surrounded by simple data classes, as shown in Figure 5.2. The key problem here is that the majority of the responsibilities are allocated to a single class.

Catalog of Refactorings to Patterns

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation

This refactoring is essentially *Extract Class* [F], only it's done on a class's Creation Methods. There's nothing wrong with a few Creation Methods on a class, but as the number of them grows, a class's own primary responsibilities – it's main purpose in life – may begin to feel obscured or overshadowed by creational logic. When that happens, it's better to restore the class's identity by moving its Creation Methods to a Creation Class.

Creation Classes and Abstract Factories [GoF] are similar in that they create families of objects, but they are quite different, as the following table illustrates:

Mechanics

1. Identify a class (which we'll call "A") that is overrun with Creation Methods.

2. Create a class that will become your Creation Class. Name it after it's purpose in life, which will be to create various objects from a set of related classes.

Example

```
public class Loan {
   private double notional;
   private double outstanding;
   private int rating;
```

Goals of the Seminar

TECHNISCHE UNIVERSITÄT DARMSTADT

- Analyze a selection of anti-pattern countermeasures as well as pattern-oriented refactorings
- Extract a quantitative formalization of patterns/anti-patterns from the natural-language descriptions
- Characterize (formally and/or informally) the situations where they are useful in terms of software architecture

TECHNISCHE UNIVERSITÄT DARMSTADT

Refactoring Communication Systems

SST Seminar – SS 16

(Seminar)

Stefan Tomaszek

Refactoring eines Heimnetzwerkes
Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

Warum ein Refactoring im Heimnetzwerk durchführen?

- Hohe Komplexität für die Nutzer
- Unterstützung vieler Geräte, Komponenten, Applikationen, Protokolle,…

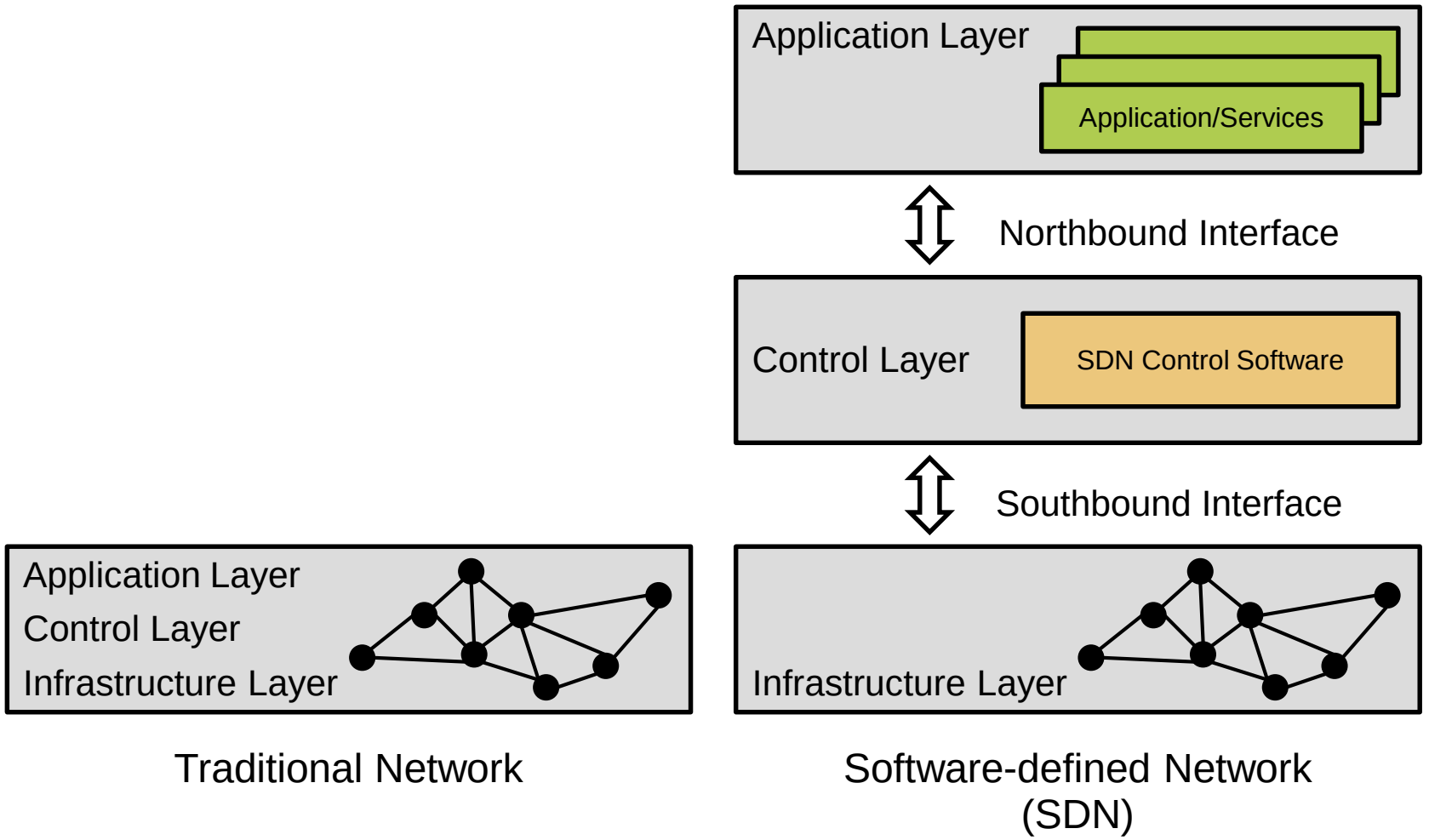
Was können Ziele des Refactorings sein?

- Einfache Verwaltung durch die Nutzer
- Sichtbarkeit der Konfiguration durch den Nutzer einschränken oder erweitern

Welche Technology könnte dafür eingesetzt werden?

Software-defined Networks (SDN)

Einführung in Software-defined Networks



TECHNISCHE UNIVERSITÄT DARMSTADT

Die tatsächliche Anwendung von Refactoring

SST Seminar – SS 16

(Proseminar geeignet)

Lars Luthmann

Die tatsächliche Anwendung von Refactoring (Proseminar)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Tools vereinfachen Nutzung von Refactorings stark
- Viel Forschung und viele Tools zum Thema Refactoring
- Probleme:
  - Nutzung von Tools wird kaum evaluiert
  - Viele Annahmen haben kaum empirische Unterstützung
- Aufgabe:
  - Annahmen von Forschern kritisch überprüfen
  - Gründe für das Verhalten von Entwicklern finden

TECHNISCHE UNIVERSITÄT DARMSTADT

Differential Precondition Checking

SST Seminar – SS 16

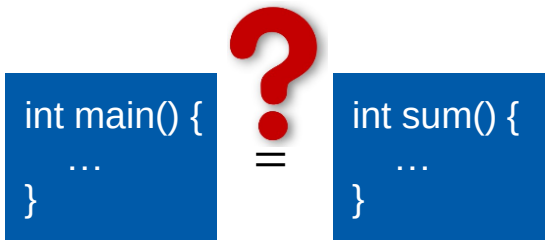
(Seminar)

Lars Luthmann

Differential Precondition Checking

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verhalten muss erhalten bleiben
- Überprüfung durch Preconditions
- Problem: Preconditions hängen oft von der gewählten Sprache ab
- Aufgabe: Beschreiben eines Ansatzes für sprachunabhängige Precondition-Überprüfung



Refactoring eines Heimnetzwerkes durch SDN Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

Seminar für 1 Studierenden

Ziele:

- Kennenlernen von SDN
- Ideen zur Refactoring eines Heimnetzwerkes sammeln
- Diskussion über die Nutzung von SDN für das Refactoring eines Heimnetzwerkes

Literatur:

- Liyanage, Gurtov, Ylianttila: Software Defined Mobile Networks (SDMN): Beyond LTE Network Structure. Wiley Series in Communications Networking & Distributed Systems, 2015
- Chetty, Feamster: "Refactoring Network Infrastructure to Improve Manageability", SIGCOMM Computer Communication Review, Volume 42 Issue 3, 2012

Roland Kluge

SST Seminar – SS 16

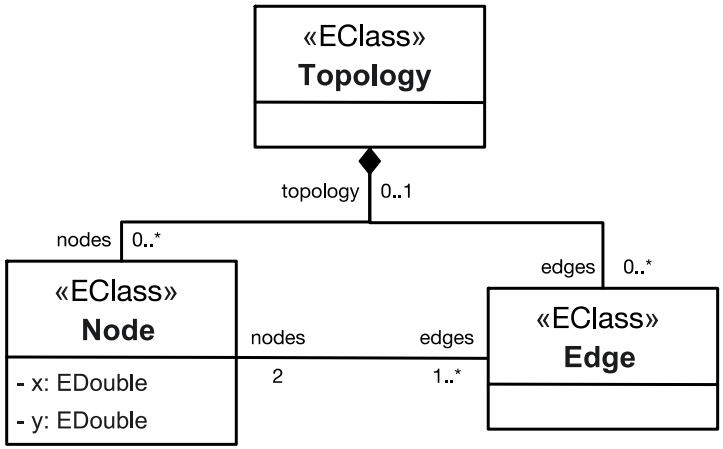
(Proseminar geeignet)

Roland Kluge

Refactoring meta-models requires co-evolving related artefacts

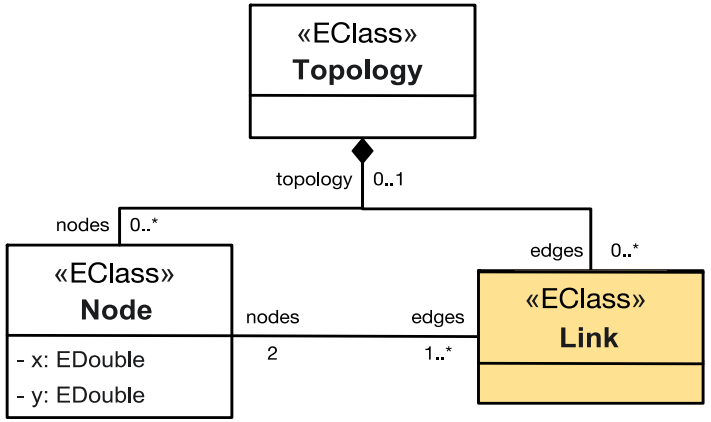
TECHNISCHE UNIVERSITÄT DARMSTADT

Meta-model Version 1.0



Refactoring

Meta-model Version 1.1



Consistent

Consistent

Co-Evolution

Dependent artefacts Version 1.0

- Additional constraints (e.g., OCL)
- Sequence diagrams, state charts
- Generated source code
- Concrete models

Dependent artefacts Version 1.1

- Additional constraints (e.g., OCL)
- Sequence diagrams
- Generated source code
- Concrete models

TECHNISCHE UNIVERSITÄT DARMSTADT

Characterizing Meta-Model Changes

SST Seminar – SS 16

(Proseminar geeignet)

Roland Kluge

Characterizing Meta-Model Changes
Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

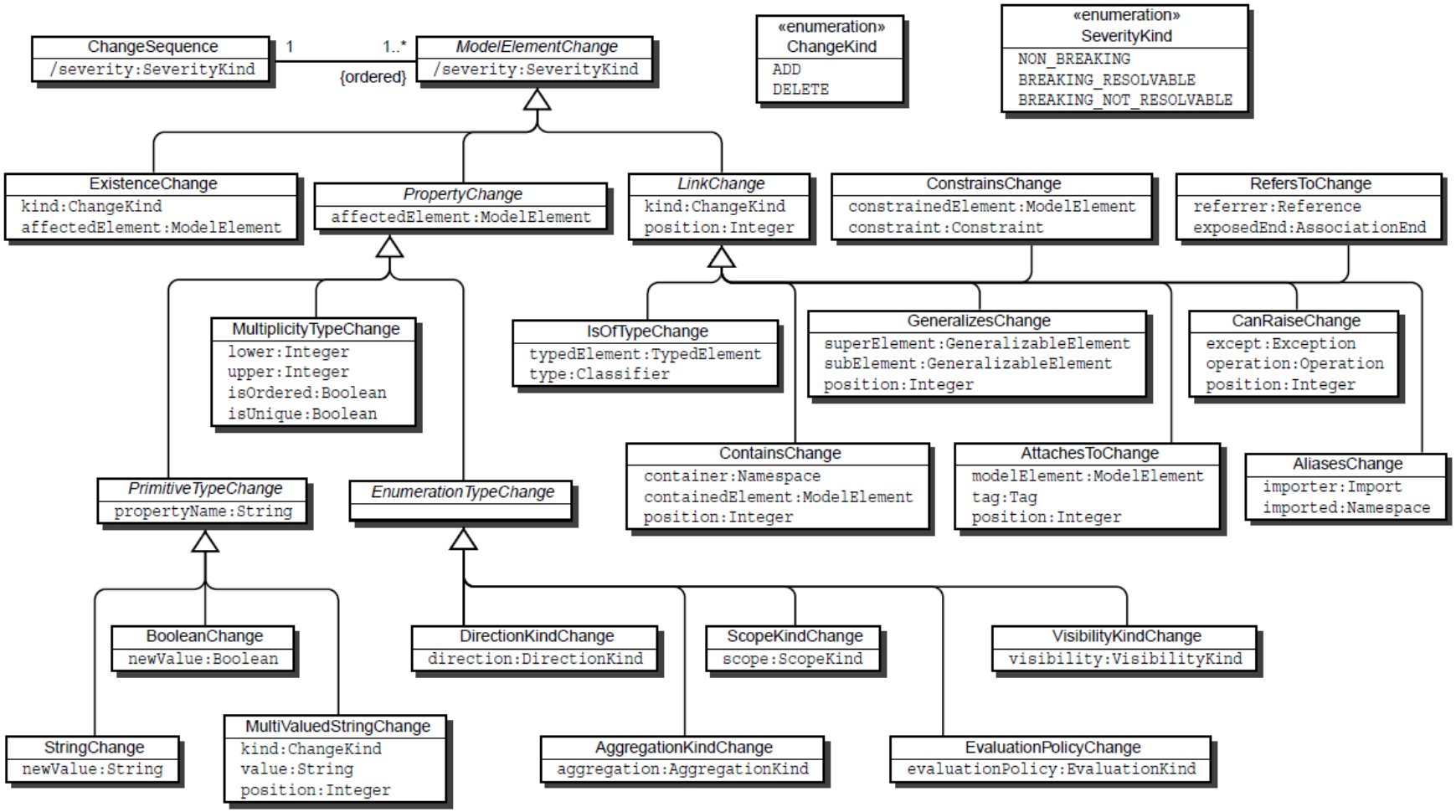
To fully support meta-model evolution, we have to know …

1. about all **possible types of modifications** (wrt. to the selected modeling standard (EMF,MOF, UML) and …

2. about the **impact** of each type of modification
   - e.g., inserting a new type into a meta-model will probably not break anything
   - e.g., removing a type X will break all models that contain instances of X

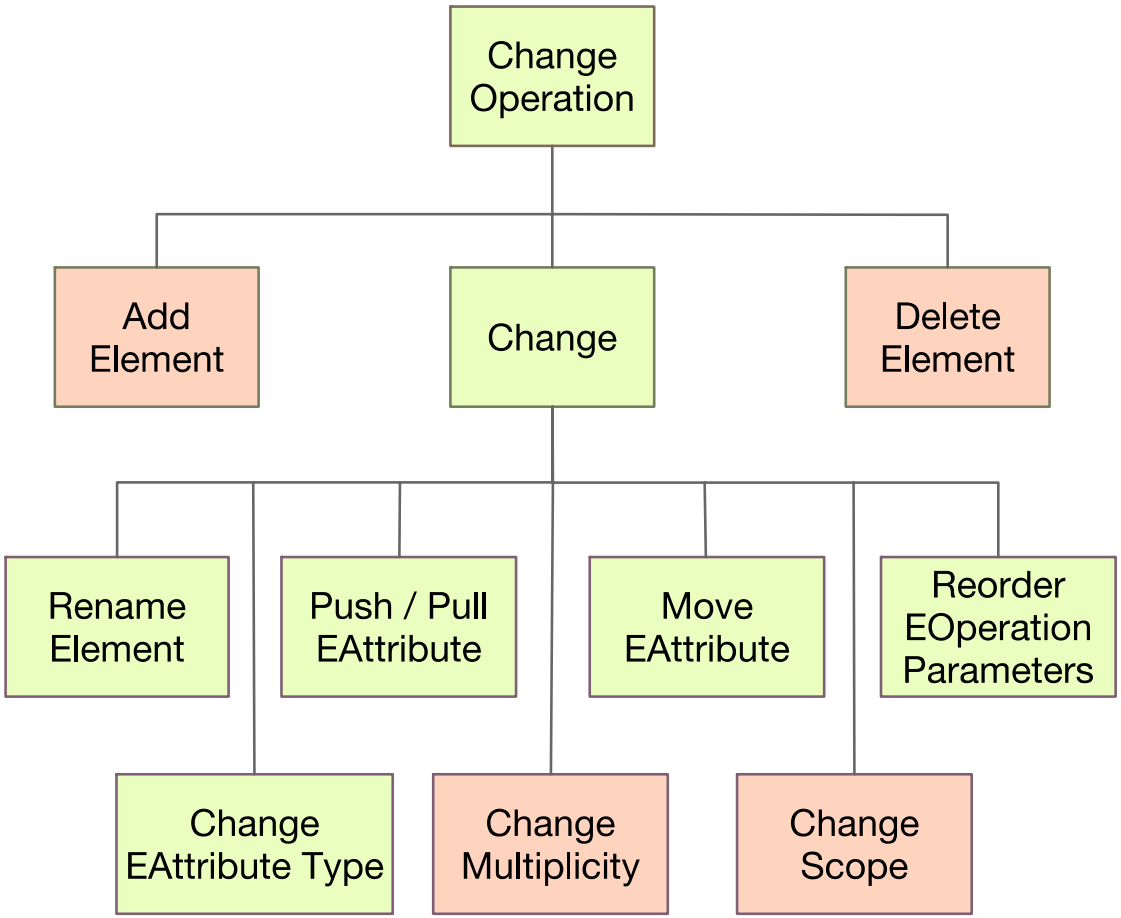
Characterizing Meta-Model Changes

Types of MM changes



Characterizing Meta-Model Changes

Classification of MM changes



Klassifikation nach: B. Gruschko

automatisiert behandelbar

nicht automatisiert behandelbar

Characterizing Meta-Model Changes
Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- **1 Student** – „Seminar“ **AND** Proseminar – supervised by Roland Kluge
- You should be interested in…
  - … **refactoring** / software evolution, of course,
  - … **intense** literature research & **(really) critical** discussion of approachs
  - … **model-driven software engineering** (e.g., UML/Ecore)

- Starting points:
  - Becker, Goldschmidt, Gruschko, and Koziolek: “A Process Model and Classification Scheme for Semi-Automatic Meta-Model Evolution”, 2007
    https://www.researchgate.net/publication/220032076_A_Process_Model_and_Classification_Scheme_for_Semi-Automatic_Meta-Model_Evolution
  - Gruschko, Kolovos, Paige: “Towards Synchronizing Models with Evolving Metamodels,” 2007
    http://www.sciences.univ-nantes.fr/MoDSE2007/p15.pdf
  - Burger, Gruschko: “A change metamodel for the evolution of MOF-based metamodels,” 2010
    https://sdqweb.ipd.kit.edu/publications/pdfs/burger2010a.pdf

TECHNISCHE UNIVERSITÄT DARMSTADT

Specifying Integrated Refactoring with Distributed Graph Transformations

SST Seminar – SS 16

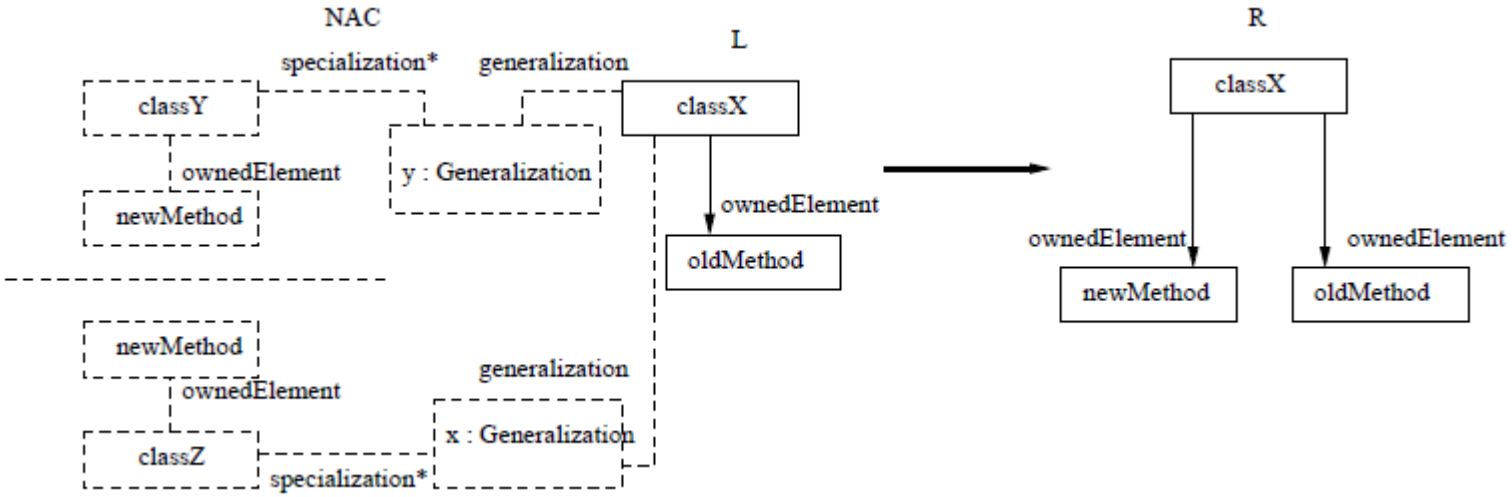
(Seminar)

Roland Kluge

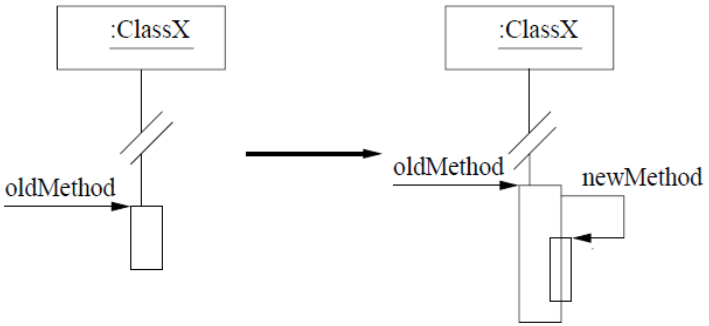
Integrated „Extract Method“ Refactoring in Code, State Charts and Seq. Diagrams

TECHNISCHE UNIVERSITÄT DARMSTADT

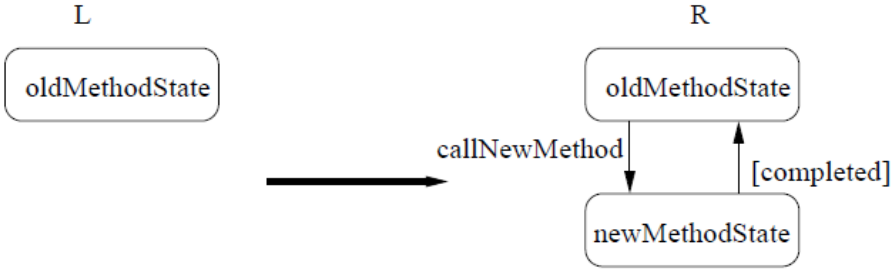
“Extract Method” in Flowgraph (~Code)



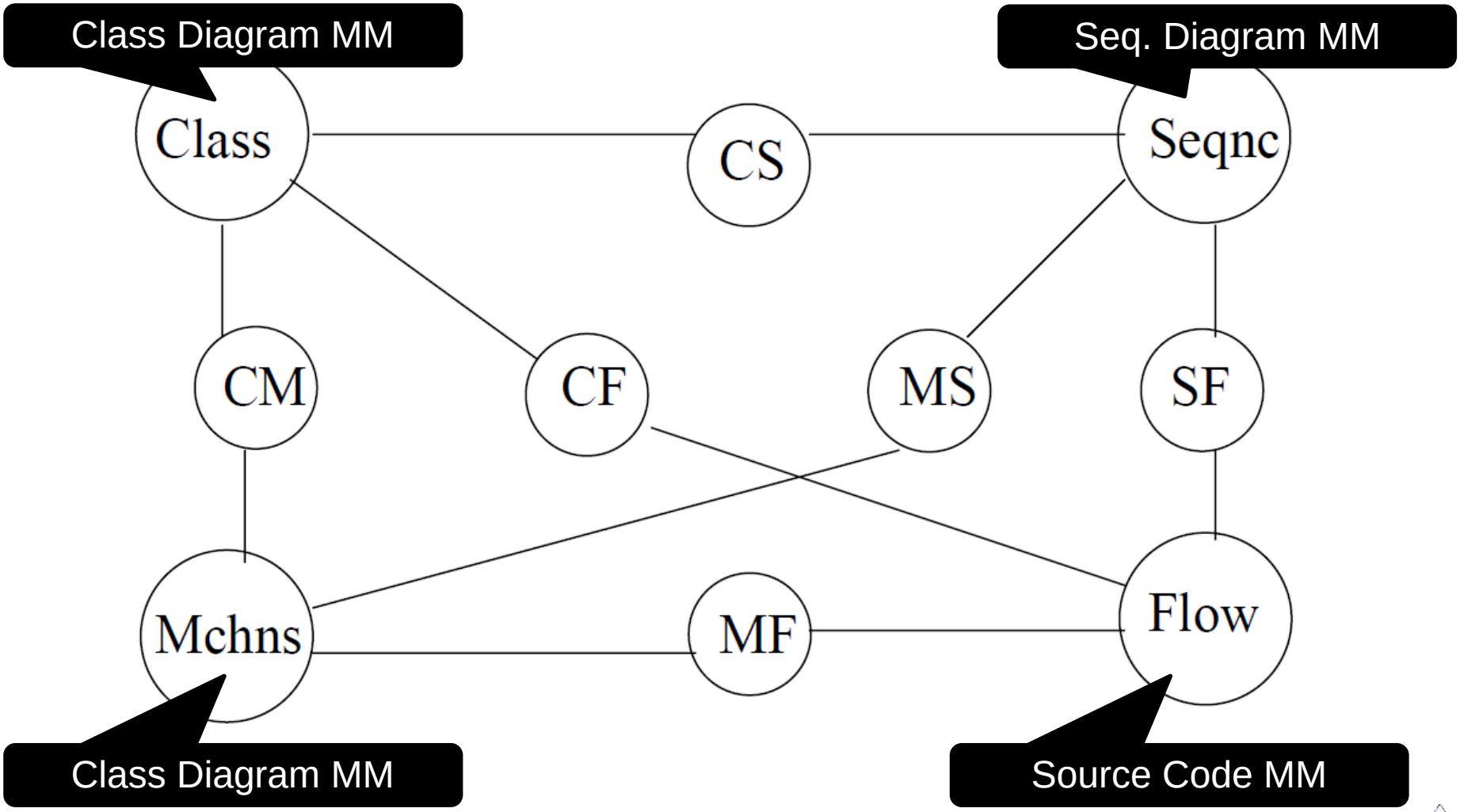
“Extract Method” in Sequence Diagram



“Extract Method” in State Chart



High-Level Connection between Models



Summary

- **1 Student** – suitable **ONLY** for „Seminar“ (too much to understand for a Proseminar) – supervised by Roland Kluge

- You should be interested in…
  - … **refactoring** / software evolution, of course,
  - … **formal methods** (→ distributed graph transformation)
  - … **model-driven software engineering** (e.g., UML/Ecore)
- Starting points:
  - Bottoni, Presicce, Taentzer: “Specifying Integrated Refactoring with Distributed Graph Transformations,” 2004 http://link.springer.com/chapter/10.1007/978-3-540-25959-6_16
  - Bottoni, Presicce, Taentzer: “Coordinated Distributed Diagram Transformation for Software Evolution,” 2003 http://www.sciencedirect.com/science/article/pii/S1571066104806261
  - Bottoni, Parisi-Presicce, Pulcini, Taentzer: “Maintaining Coherence Between Models With Distributed Rules: From Theory to Eclipse,” 2008 http://www.sciencedirect.com/science/article/pii/S1571066108002478

Themenauswahl

(jetzt geht's los)

Wie geht es weiter…!?

TECHNISCHE UNIVERSITÄT DARMSTADT

E-Mail an johannes.buerdek@es.tu-darmstadt.de mit

- eurem Erst-, Zweit- und Drittwunsch und
- Solo oder Partner (bei Bearbeitung zu zweit)
- Studiengang + Fachsemester

Deadline: Heute Abend.

Unsere Aufgaben

- Wir verteilen die Themen schnellstmöglich auf die Interessenten
- Geben das Ergebnis bekannt (→ moodle, E-Mail)
- Bereiten alles vor, damit anschließend die Bearbeitung unmittelbar starten kann

Eure Aufgaben

- Warten auf Ergebnisse der Zuteilung
- Anschließend meldet ihr euch bitte unmittelbar beim Betreuer → Termin für ein erstes, persönliches Treffen
- Anschließend: Start Einarbeitung → Zeitplan beachten

Fragen?