cooperate

Schulung des Cooperate-Werkzeuges

Projektpartner:

Karlsruher Institut für Technologie

FZI

Gefördert durch:

Bundesministerium für Arbeit und Soziales

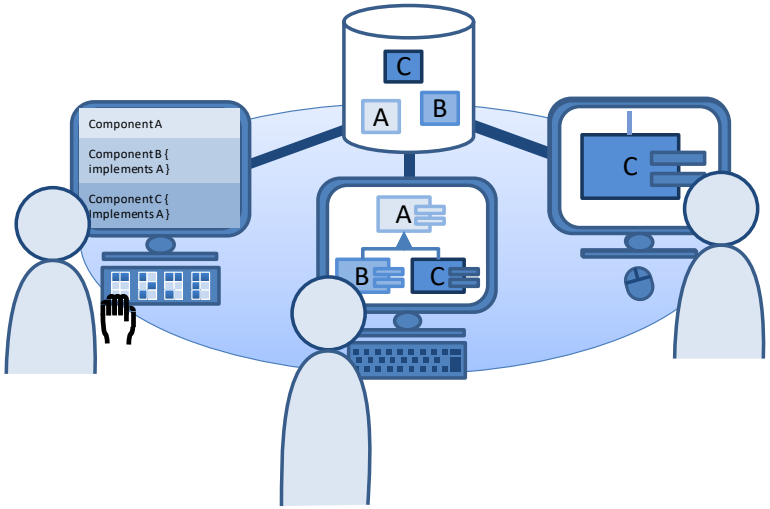
Förderkennzeichen: 01KM141108

Projektdaten

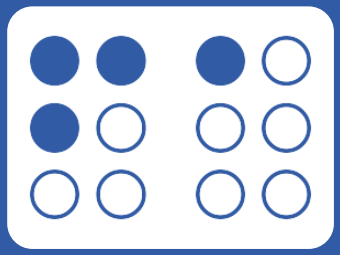
- Laufzeit: 01.01.2015 – 30.06.2018
- Förderer: Bundesministerium für Arbeit und Soziales (BMAS) aus dem Mitteln des Ausgleichsfonds
- Projektpartner:
  - Studienzentrum für Sehgeschädigte (SZS): Koordination und Entwicklung von Schulungsmaterial
  - Forschungszentrum für Informatik (FZI): Entwicklung des Kooperationswerkzeugs + Evaluierung
  - Unterauftrag: msg DAVID GmbH

Ziel des Cooperate-Projektes

1. Verbesserung des Zugangs zu UML
   a) Entwicklung eines Kooperationswerkzeugs
   b) Evaluierung in einem Diversity Team (Teams bestehend aus Menschen mit und ohne Sehschädigung)

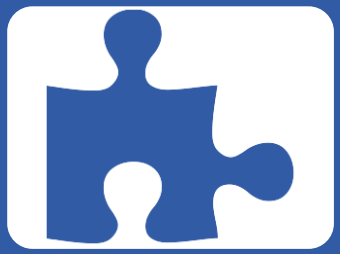


Anforderungen



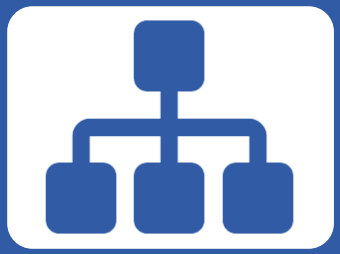
Barrierefreie Nutzbarkeit

- Bedienbarkeit mit Screen-Reader
- Bedienbarkeit ohne Assistenz



Nutzung üblicher Eclipse-Mechanismen

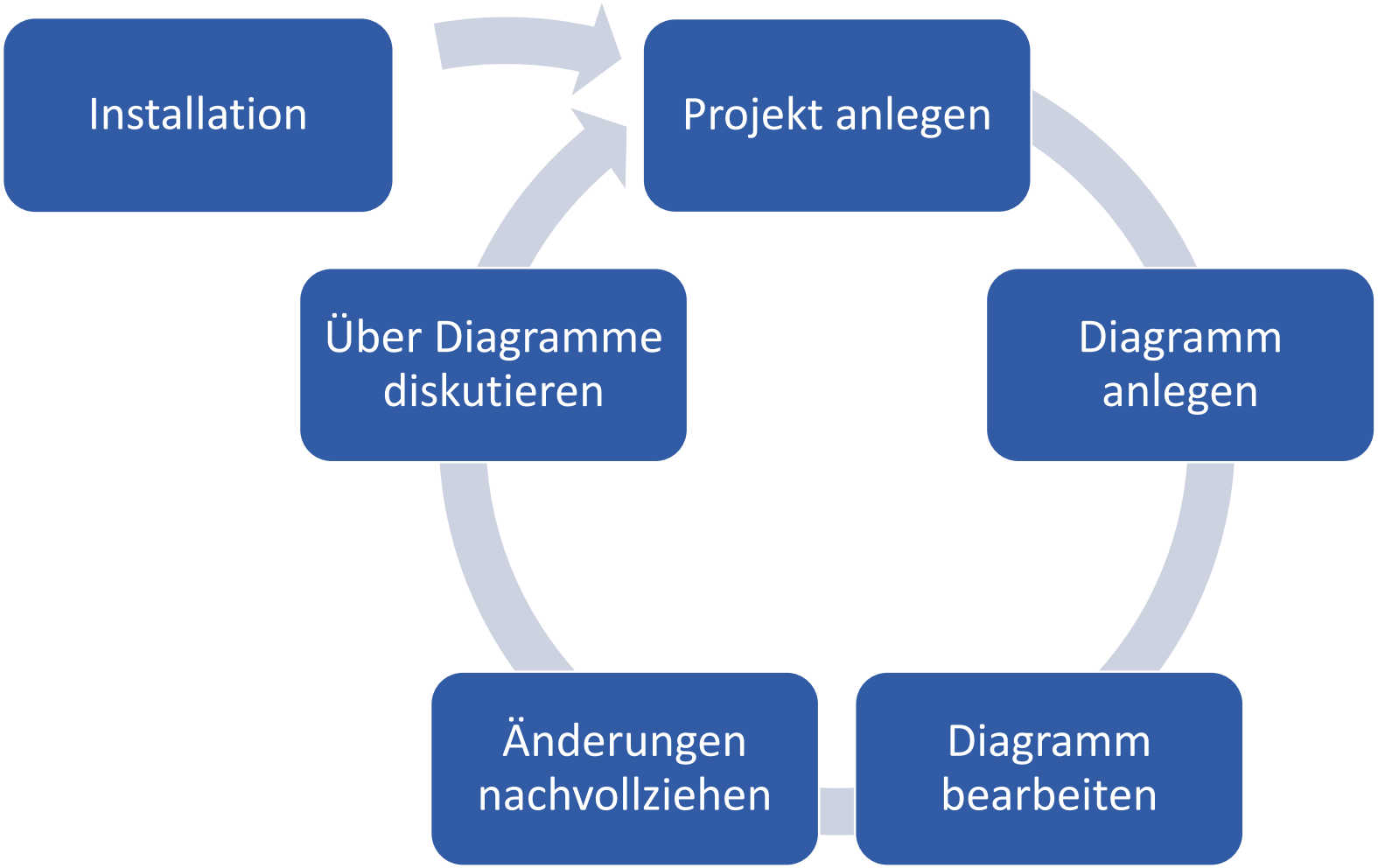
- Intuitive Nutzbarkeit für Eclipse-Kenner
- Alternative Bedienkonzepte nur wo notwendig



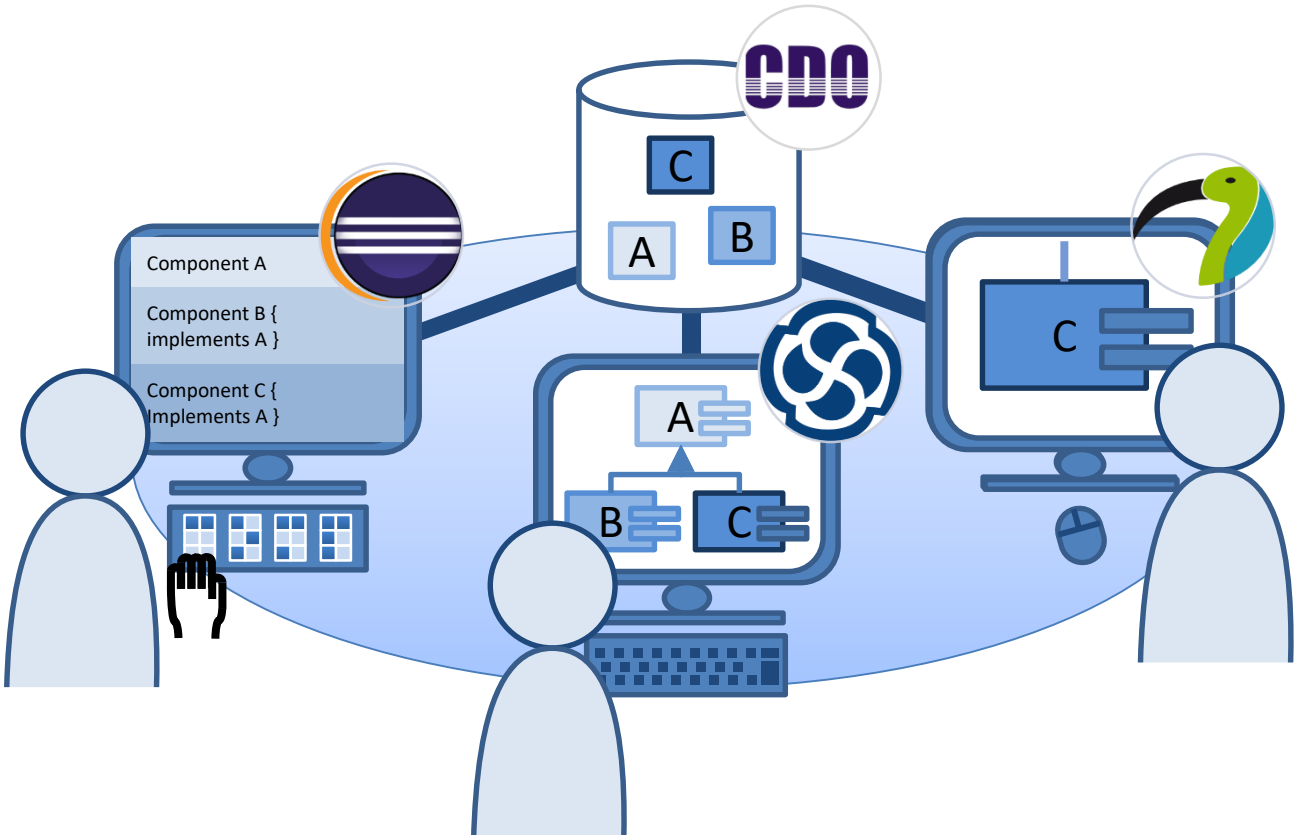
Komplexität UML-Modellierung reduzieren

- Modellierung ohne detaillierte UML-Kenntnisse
- Konkrete Syntax ist primärer Modellierungsgegenstand

Grundlegende Funktionen



Grobarchitektur

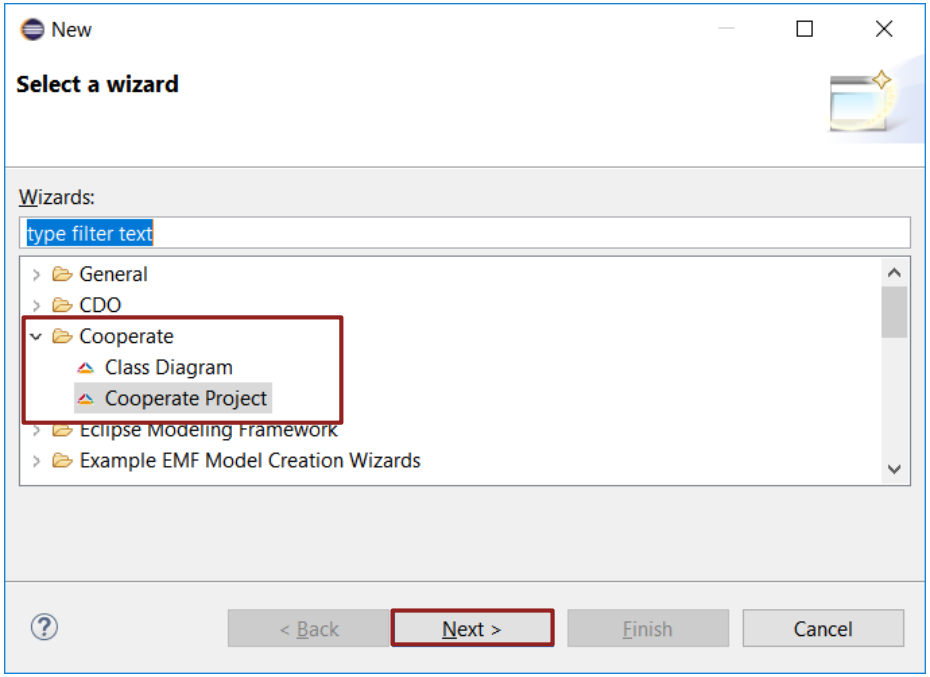


Grundlegende Funktionen

Praktischer Teil, Hands-on session

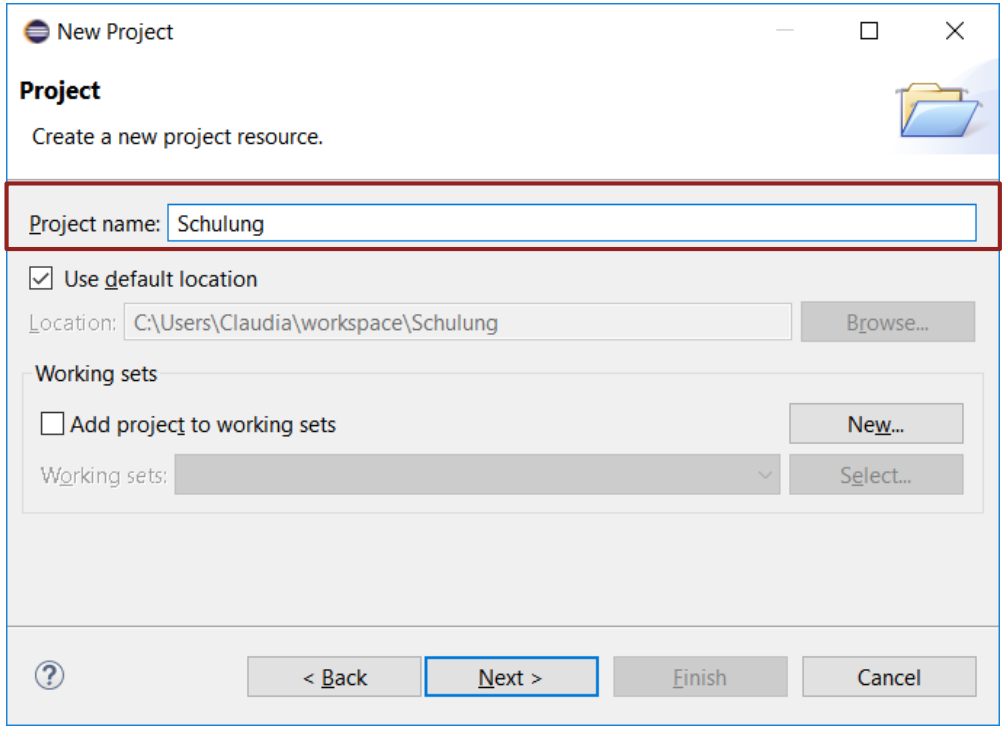
Anlegen eines Projekts (Schritt für Schritt)

1. File - New (Strg + N)
2. In der Cooperate Kategorie, wählen Sie “Cooperate Project”



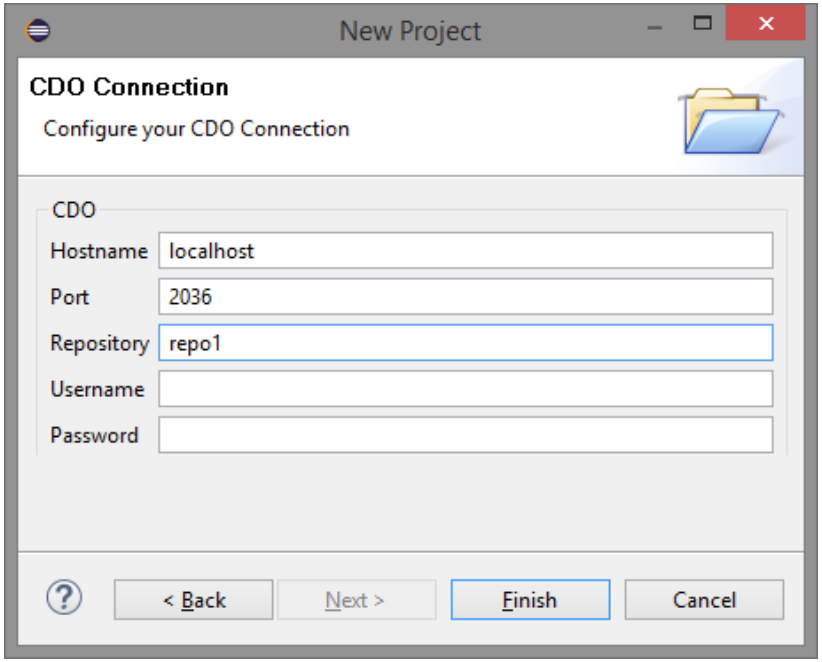
Anlegen eines Projekts (Schritt für Schritt)

3. Projektnamen angeben und weiter mit Next.



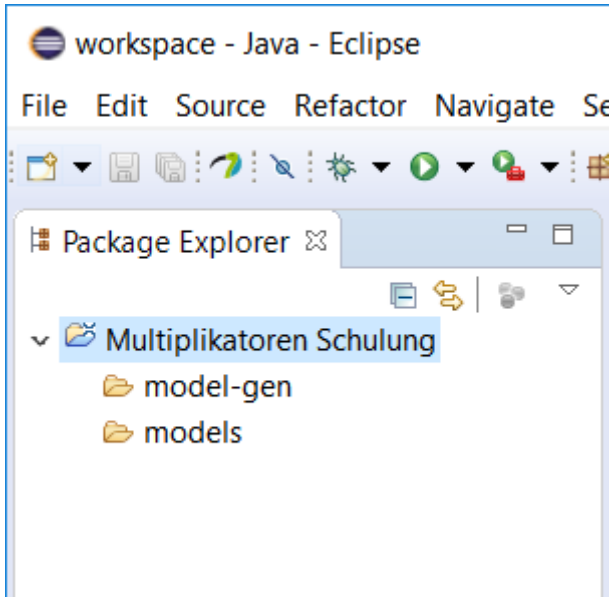
Anlegen eines Projekts (Schritt für Schritt)

4. Prüfen der Verbindungsdaten zum CDO Server. Bestätigen mit Finish.



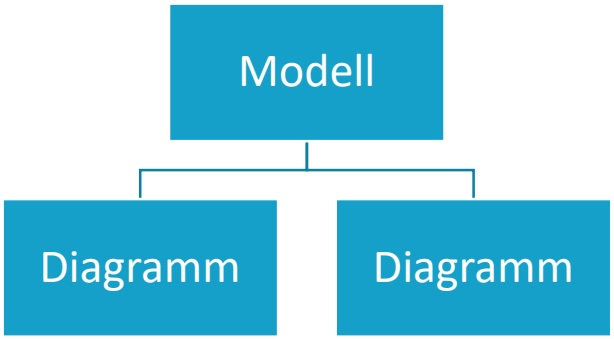
Anlegen eines Projekts (Schritt für Schritt)

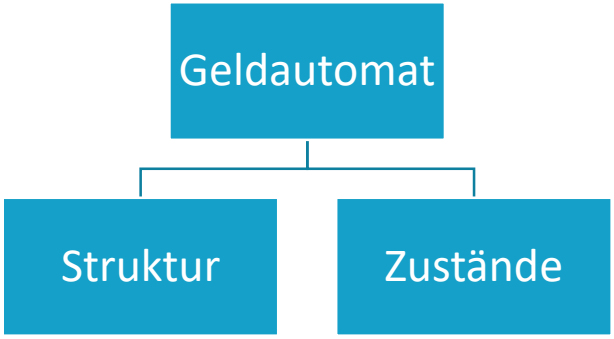
5. Ein neues Projekt erscheint nun im Package Explorer, welches folgenden Aufbau hat:
   - Der **model-gen** Ordner enthält alle Inhalte des Projektes, welche auf dem CDO Server gespeichert sind.
   - Der **models** Ordner enthält die eigentlichen Diagramme.



Zusammenhang Modell und Diagramm

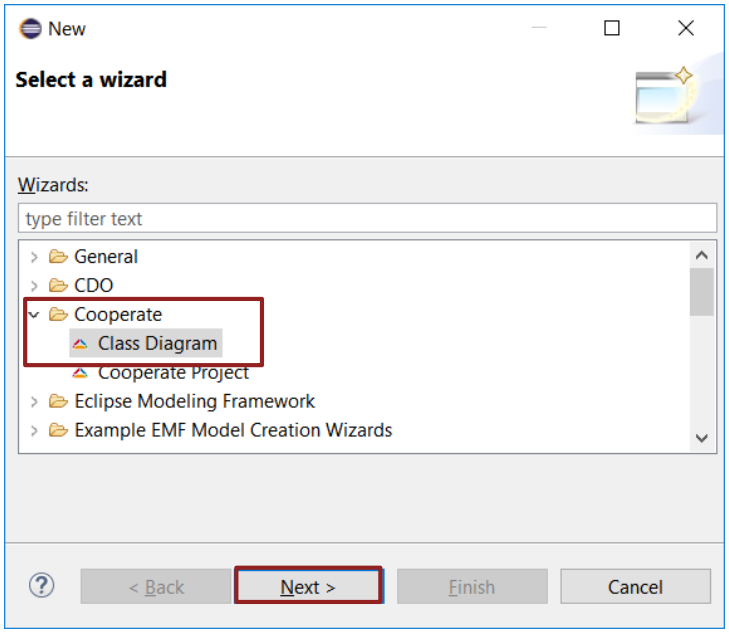
- Diagramm ist eine Untermenge des gesamten UML Models
- Ein Model kann mehrere Diagramme haben:





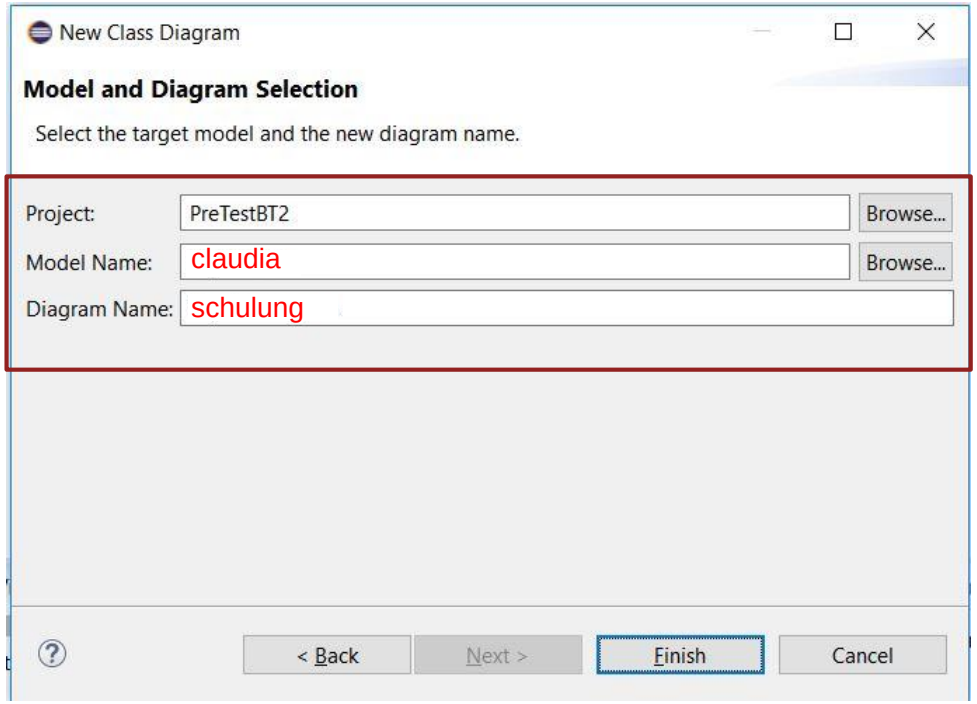
Anlegen eines Diagramms

- Erstellen eines neuen Diagrammes:
  - File - New (Strg + N)
  - Wählen Sie in der Kategorie Cooperate Class Diagramm



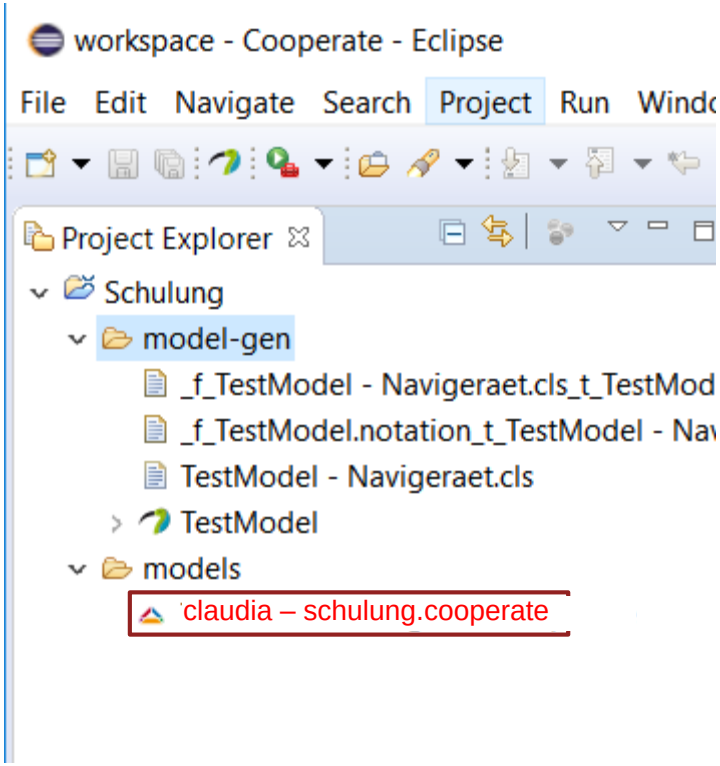
Anlegen eines Diagramms

1. Projekte sowie Modelle auswählen bzw. angeben. Diagrammname angeben.
2. Finish



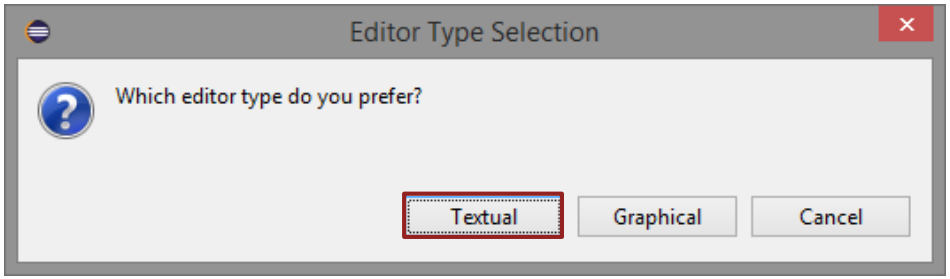
Anlegen eines Diagramms

- Das neue Diagramm und das Modell ist nun auf dem Server gespeichert.
- Ein neuer Eintrag im models Ordner erscheint und wir können beginnen ein Diagramm zu erstellen.



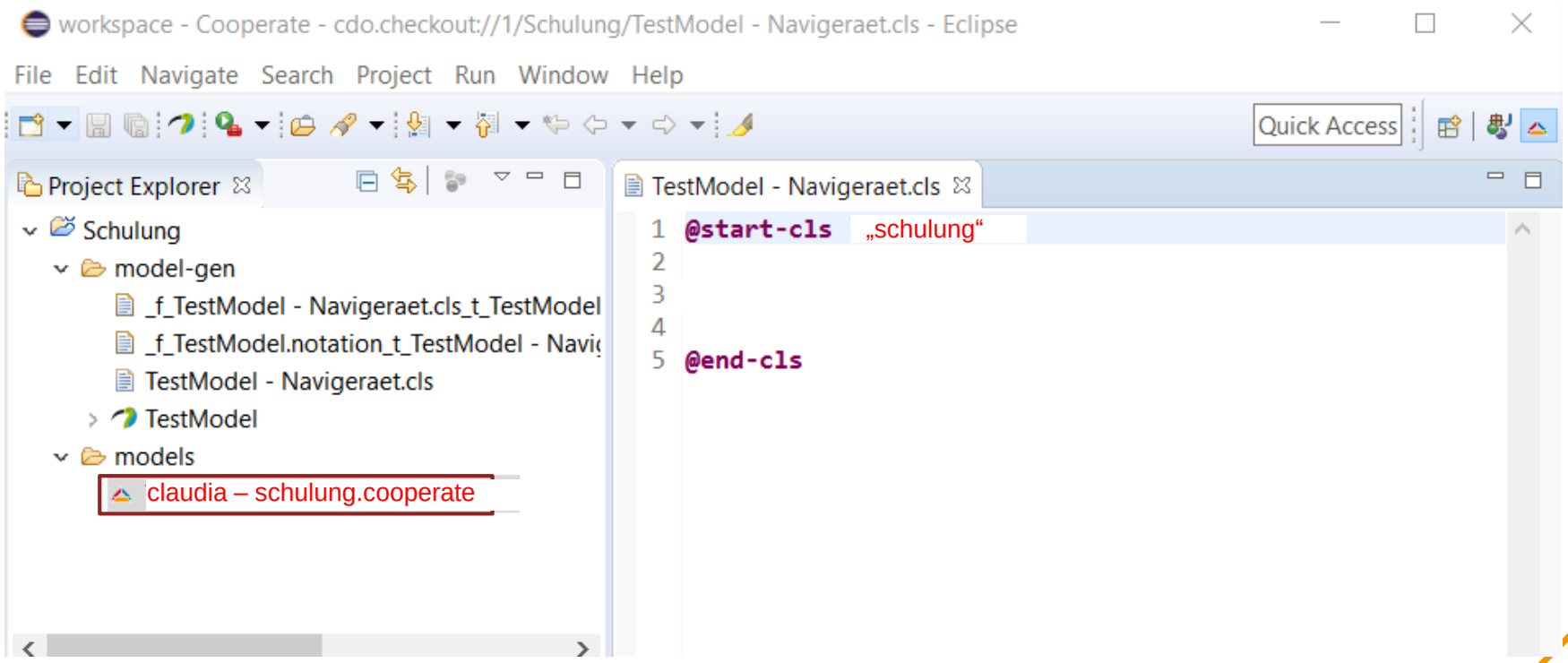
Bearbeiten eines Diagramms

- Doppelklicken Sie auf das neue Diagramm.
- Ein Dialog erscheint zur Auswahl der Darstellungsansicht**: textual oder grafisch**.
- Klicken Sie zunächst auf Textual.



Bearbeiten eines Diagramms

- Texteditor öffnet sich und das Minimalgerüst eines Klassendiagrammes ist auch schon da



Textueller Editor

UML4All-Syntax am Beispiel von Klassendiagrammen

Motivation für die UML4ALL-Syntax

- Berücksichtigung der Arbeitsweise von Menschen mit Blindheit
  - Kompakte Darstellung auf der Braillezeile
  - Verständliche Ausgabe über den Screenreader
- Verbesserung der Navigierbarkeit
  - Eigner textueller Editor mit Zusatzunterstützung
- Unterstützung der Zusammenarbeit in Entwicklungsprozessen

Grundgerüst

- Benennung des Diagrammtyps zur besseren Auffindbarkeit

```
@start-ID
@end-ID
```

Diagrammtyp	Identifier (ID)
Klassendiagramms (class diagram)	**clsd**
Anwendungsfalldiagramm (use case diagram)	uscd
Aktivitätsdiagramm (activity diagram)	actd
Zustandsdiagramm (state machine diagram)	stmd
Sequenzdiagram, (sequence diagram)	seqd

Klassendefinition

- **Schlüsselwort: class**
- **Eindeutiger Bezeichner (eindeutige ID)**
  - Bezeichner ohne Leerzeichen (camelCase)
  - Beispiel: `class zoo`
  - Langer Bezeichner über Alias as angeben
    - `class zKeeper as „Zookeeper“`
- Nützliches Feature: **Quick Fix**
  - Cursor in Zeile, dann mit Strg + 1 Quick Fix öffnen (über die Pfeiltasten Navigation)
  - Werden beim Speichern direkt angewendet.

Sichtbarkeiten und spezielle Klassen

- Sichtbarkeiten (public, private, protected) vor den Operand *class*
  - `private class giraffe`
  - `- class giraffe`
  - Wird automatisch in Kurzform (+, -, #) übersetzt
- Abstrakte Klassen
  - `private abstract class MeineKlasse`

Attribute und Methoden

- Attribute oder Methoden: geschweifte Klammern als Block hinter der Klassendefinition
- Attribute: Name, gefolgt von einem Doppelpunkt und dem Typ, z.B.:

```
class animal{
+ name:string
}
```

- Unterstützte Typen: `string, int, double, boolean, char, byte, short, long, float`
- Sichtbarkeiten bzw. `static` oder `final` analog der Darstellung bei Klassen

Attribute und Methoden

- Methoden analog zu Attributen nur mit ():
  - `methodenname():rückgabetyp`
  - `methodenname(name:typ,param:string)`
  - Sichtbarkeiten analog der Darstellung bei Klassen und Attributen, z.B.: `public static main(argument: String)`
- Beispiel:
  - `class zkeeper as „Zookeeper"{`
    - `- feed (animal : animal)`

  `}`

Relationen

- Operator Bezeichner (Operand1, Operand2)
  1. **Operatoren**: Typ eines Elementes oder einer Relation (über Schlüsselwörter).
  2. **Bezeichner**: verpflichtend und muss eindeutig im Diagramm sein
  3. **Operanden:** sind die beteiligten Elemente an einer Operation
  4. Weitere Merkmale wie **Rollen** oder **Kardinalitäten**
- Beispiel:
  - `asc employed (zKeeper, zoo)`
- Nützliches Feature: **Code Completion**
  - Strg + Space

Einfache Assoziation - Leserichtung

- Keine explizite Angabe der Leserichtung, z.B.:
  - `asc hat (Buchhandlung, Kunde)`
  - `asc stoebert (Kunde, Buchhandlung)`
- Bidirektionale Verbindungen
  - Schlüsselwort `bi`
  - Beispiel: **`bi`** `asc name (A,B)`

Teil-eines-Ganzen Verbindungen

- Aggregation ()
  - Schlüsselwort: `agg`
- Komposition
  - Schlüsselwort: `com`
- Die Klasse mit dem Kompositions-, oder Aggregationsende wird zuerst genannt.
- Beispiel:
  - `agg besitzt (Buchhandlung,Raum)`
  - `com bestehtAus (Buchhandlung,Raum)`

Vererbung

1. Schlüsselwort: `isa`
2. Beteiligte Operanden in runden Klammern, mit Komma voneinander getrennt.

- Ausnahme hier: kein Bezeichner.
- Die erbende Klasse wird zuerst genannt.
- Beispiel:
  - `isa(giraffe,animal)`

Rollen

1. Schlüsselwort: `role`
2. Attribute in eckigen Klammern mit **Komma** voneinander getrennt.

- Reihenfolge der Werte in eckigen Klammern bestimmt die Zuordnung zu den Operanden!
- Position: nach Operanden.
- Beispiel:
  - `asc has (zoo,animal) role[_: inhabitant]`

Kardinalitäten

1. Schlüsselwörter: `card`
2. Attribute in eckigen Klammern mit **Doppelpunkt** voneinander getrennt

- Reihenfolge der Werte in eckigen Klammern bestimmt die Zuordnung zu den Operanden!
- Position: nach Operanden
- Default: 0..1
- Beispiel:
  - `asc employed (zKeeper,zoo) card [1..* : 1..1]`

Rollen und Kardinalitäten verbinden

- Zuerst Rollen dann Kardinalitäten
- Beispiel:
  - `asc has (zoo, animal) role[_: inhabitant] card[1..*:1..1]`

N-äre Verbindungen

- Beispiel:
  - `asc Bezeichner (A,B,C) role[RolleA, RolleB, RolleC] card[KardA:KardB:KardC]`

Pakete

1. Schlüsselwort: `package`
2. Eindeutiger Bezeichner (eindeutige ID)
3. Geschweifte Klammern
4. (Optional) Referenzierung von Elementen zwischen Pakten mit `import` + ID

- Beispiel:

```
package Paket1{
    Klasse1
}
package Paket2{
    import Paket1
    class Klasse2
}
```

Verschachtelte Pakete

- Beispiel:

```
package Paket{
    class Test
    asc hat (Test,Unterpaket.Unterklasse)

    package Unterpaket {
        class Unterklasse
    }
}
```

Notizen

1. Schlüsselwort: `note`
2. Notiz in Anführungszeichen

- Anwendung bei: Klassen, Interfaces und Assoziationen.
- Beispiel:

```
class A note["Notiz an einer Klasse"]
interface A note["Notiz an einer Interface"]
class A {
   name: string note ["Notiz für die Klasse"]
}
```

Aufbau eines Klassendiagramms

- Feste Reihenfolge im Diagramm:
  1. Einbindungen (Imports) definieren
  2. Definition von Klassen
  3. Relationen
  4. Pakete
- Innerhalb von Paketen muss diese Reihenfolge auch eingehalten werden!
- Vorgegebene Struktur verbessert die Orientierung, das Suchen und das Finden von Informationen.

Weitere Funktionen zur Verbesserung der Kollaboration

Explizite Commits, Refactoring, Outline View, Problems View, Repository zurücksetzen, Versionskontrolle (Diff View), Zeigegeste

Diagramm im Repository speichern

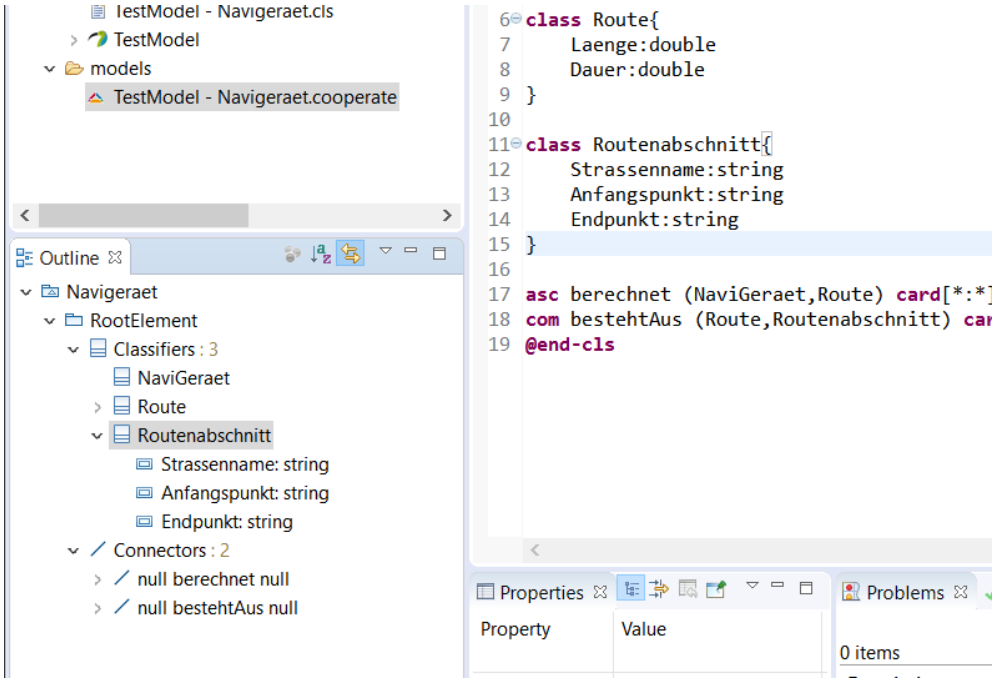
- Normales Speichern mit Strg + S hat keine Auswirkung auf das Repository.
- Speichern von Diagrammen im Repository erfordert ein Commit. 2 Möglichkeiten
  - Beim Schließen der Datei erfolgt Eingabe einer Commit-Message
  - Explizit über **Strg + Alt + C**

Refactoring

- Umbenennen von Diagrammelementen sollte über Refactoring erfolgen.
  - Kontextmenü + Rename oder
  - **Alt + Shift + R**

Die Outline View

- Hierarchischer Überblick der wichtigsten Diagrammelemente.
- Focus in der Outline View synchronisiert mit Cursor im Text Editor.
- Outline View öffnen mit **Strg + O**



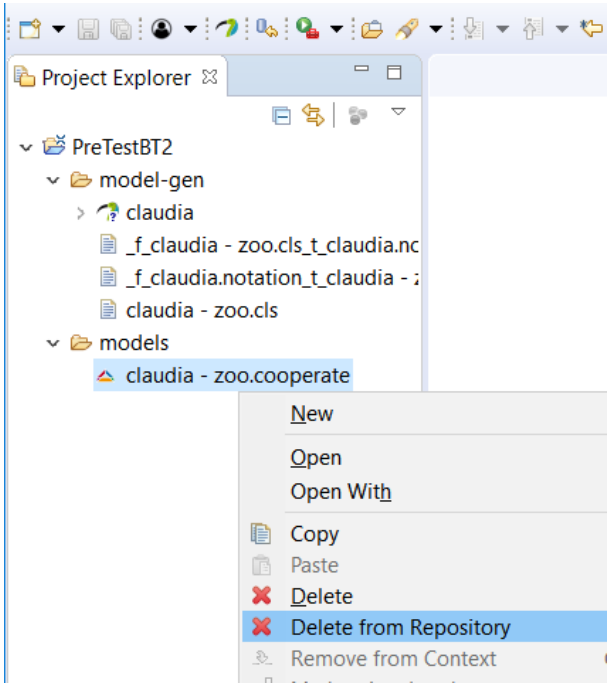
Syntaxfehler und Problems View

- Informationen über Fehler im Diagramm direkt beim Bearbeiten eines Diagrammes
- Speichern von Diagrammen mit Fehlern nicht möglich!
- Quick fixes, zum Beispiel wenn neue, noch nicht im Modell enthaltene Elemente, angelegt werden.
- Fehler Notifikation: Windows – Preference – Cooperate
  - No Audio Indicator
  - Line Audio Indicator
  - Area Audio Indicator

Repository Zurücksetzen

- Rücksetzoperation im Fehlerfall:

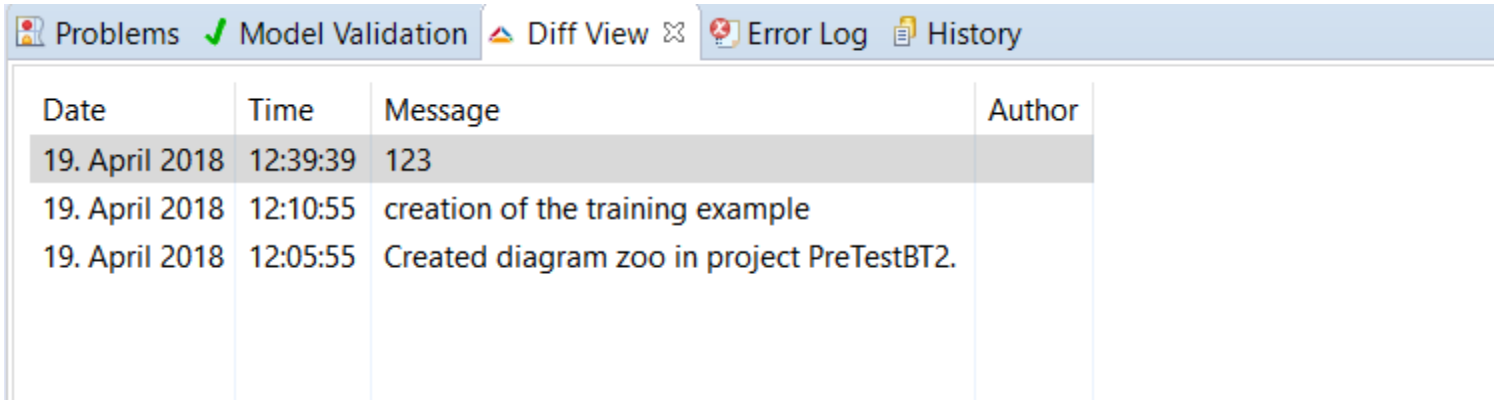
1. Fokus aus Datei Explorer
2. Diagramm auswählen
3. Kontextmenü
4. Delete from Repository



Versionskontrolle (Diff View)

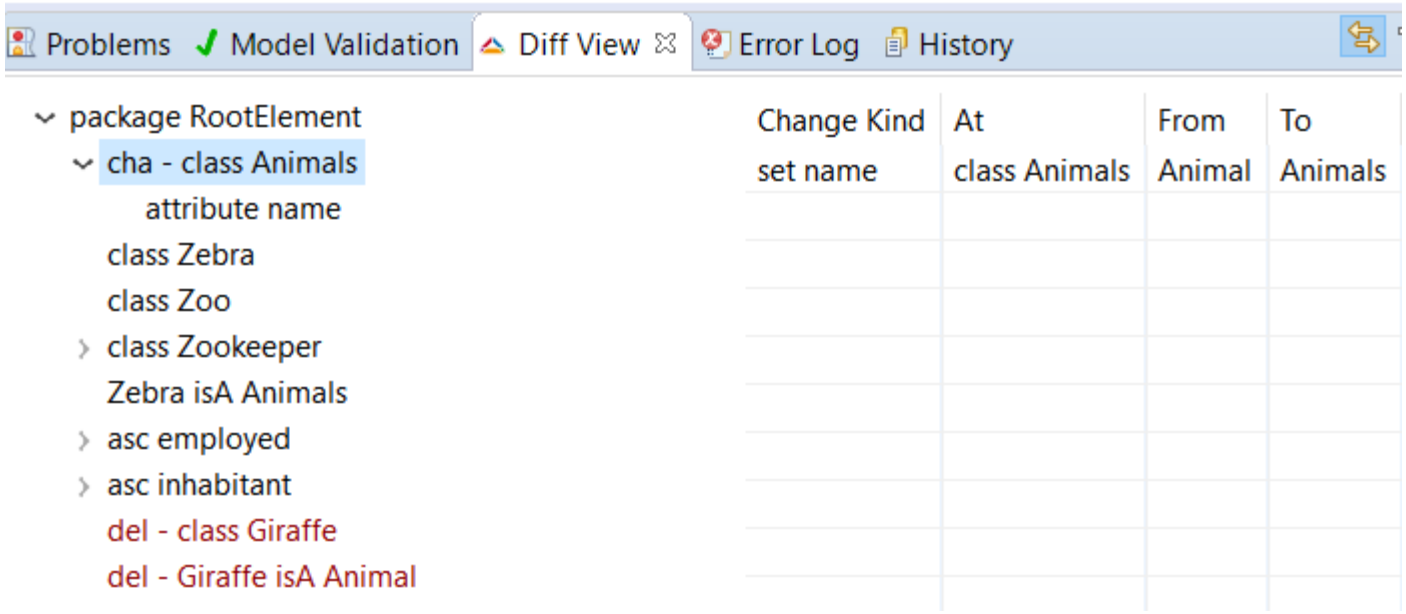
- Überprüfen von Änderungen am Diagramm:

1. Fokus aus Datei Explorer
2. Diagramm auswählen
3. Kontextmenü
4. **„Show in Diff View“ öffnet Commit Historie:**



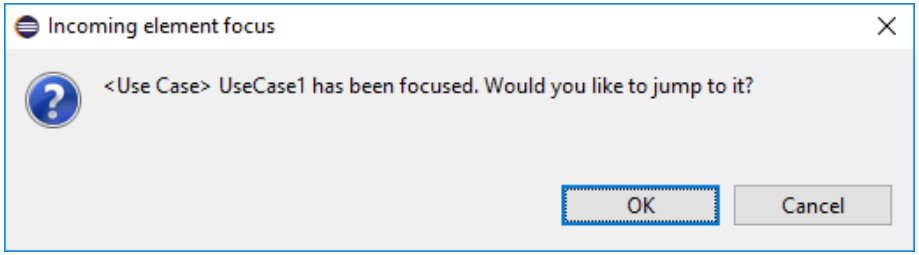
Versionskontrolle (Diff View)

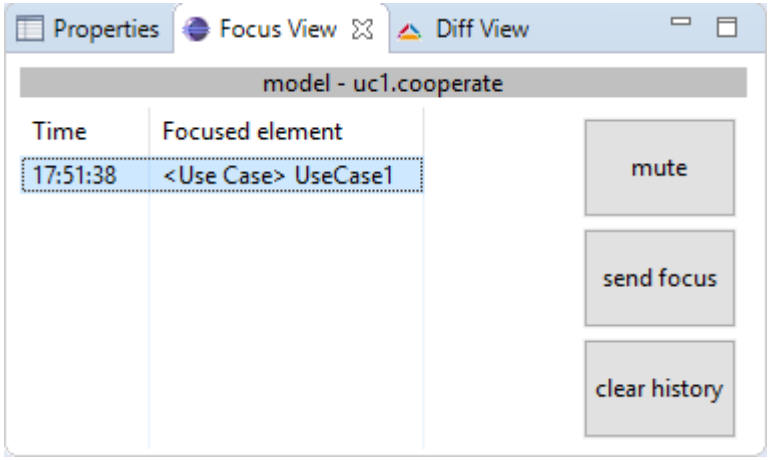
- Eintrag in der Commit Historie auswählen und mit Enter bestätigen:



Diskussion mit Zeigegeste

- Übertragung der aktuellen Selektion an Teammitglieder
- Annehmen/ablehnen möglich
- Liste mit bisherigen Anfragen
- Stummschalten möglich
- Automatisch aktiviert





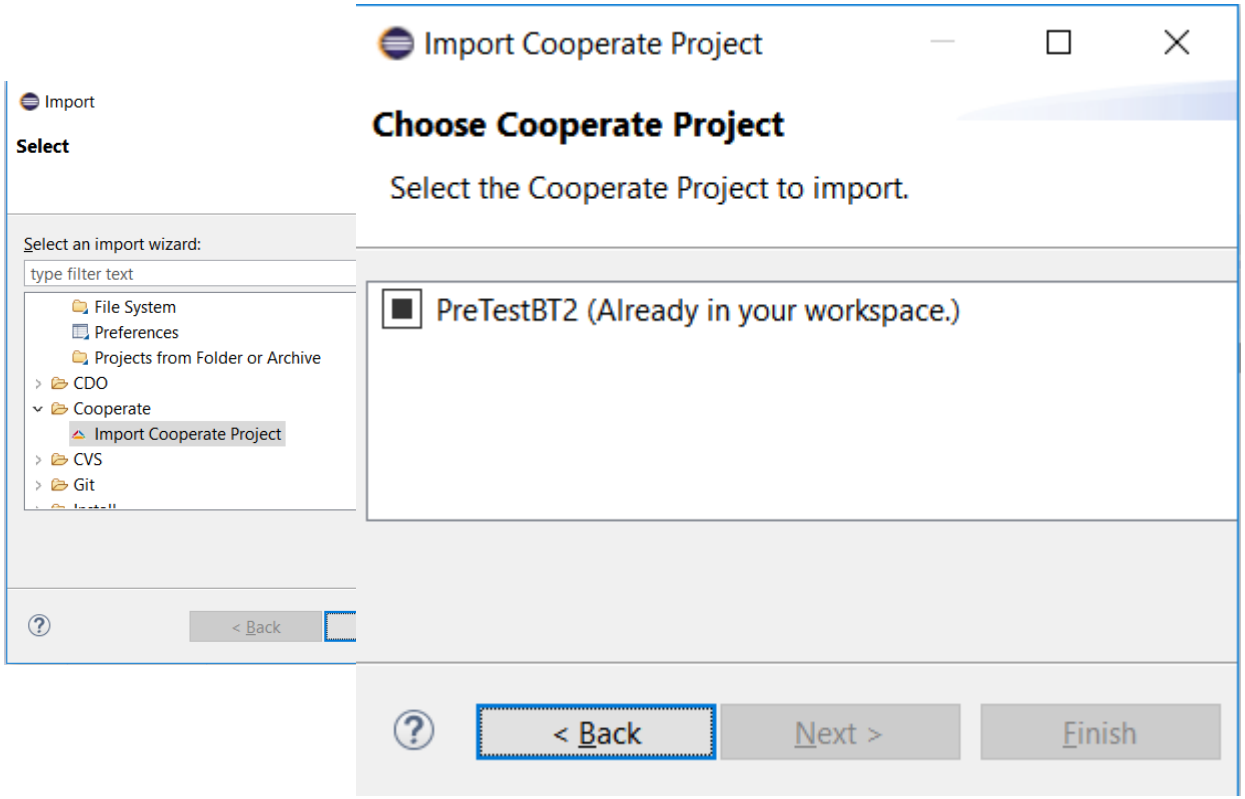
Diskussion mit der Zeigegeste

- Fokus in geöffneter Datei Teammitglieder senden.

1. **Strg + F7** öffnet die Views-List
2. Bei gedrückter Strg Taste Focus View auswählen
3. Fokus senden: **Strg + Alt + F**
4. Benachrichtigung ausschalten (Strg + Alt + M)

Import

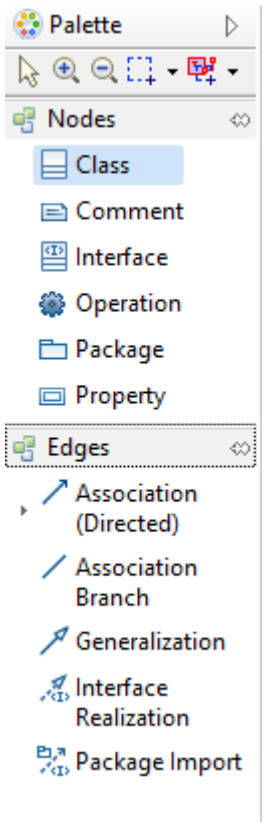
- File – Import – Import Cooperate Projekt



Grafische Editor

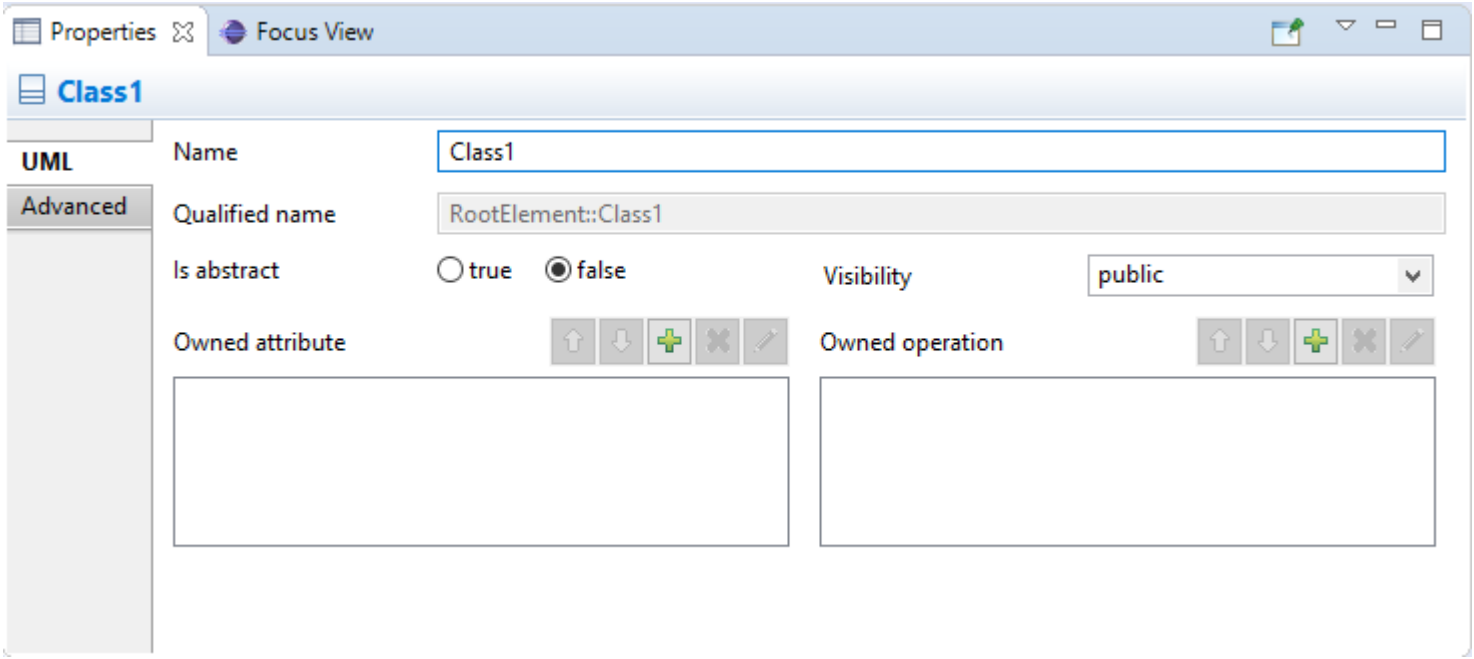
Erstellen neuer Elemente

- Palette enthält Knoten und Kanten
- Erst auswählen, dann im Diagramm verwenden
- CTRL halten zum Beibehalten der Selektion



Ändern von Eigenschaften

- Alle Eigenschaften verfügbar über Properties-View
- Erst Element selektieren, dann Änderungen vornehmen
- View reagiert oft träge (Einschränkung von Papyrus)



Validieren des Modells

- Passiert automatisch beim Speichern
- Fehler müssen zum Speichern korrigiert werden
- Zusammenfassung im View „Model Validation“