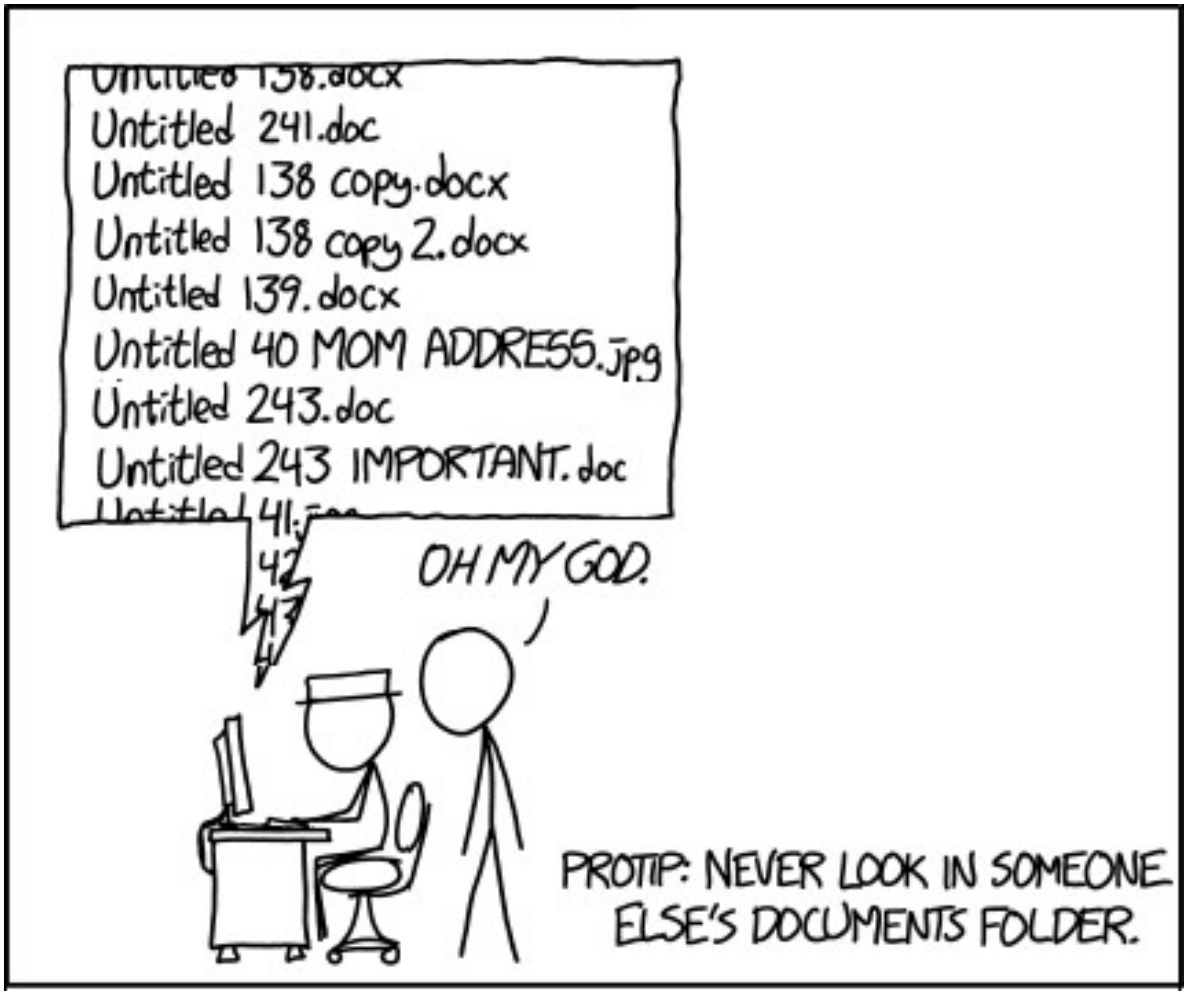
Thomas Rau
(mit Material von Peter Brichzin)

Repositories zur Unterstützung von kollaborativen Arbeiten in Softwareprojekten

In Softwareprojekten gibt es oft organisatorische Probleme, die von der inhaltlichen Arbeit ablenken

- Wie lassen sich die Teilergebnisse regelmäßig zusammenfügen?
- Wie vermeidet man, dass das Team blockiert ist, weil der Schüler mit wesentlichen Teilen des aktuellen Quelltextes krank zu Hause ist?
- Wie ermöglicht man, dass begeisterte Schülerinnen und Schüler zu Hause weiterentwickeln können?
- Wie lässt sich einfach dokumentieren, wer wann welche Teile der Software bearbeitet hat?

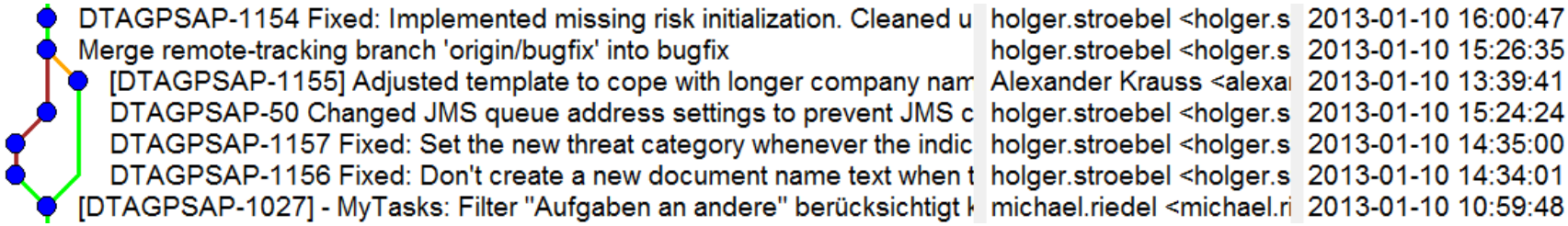
Dateien können in unterschiedlicher Qualität verwaltet werden



Quelle: https://xkcd.com/1459/

Versionskontroll-Systeme lösen die dargestellten Probleme

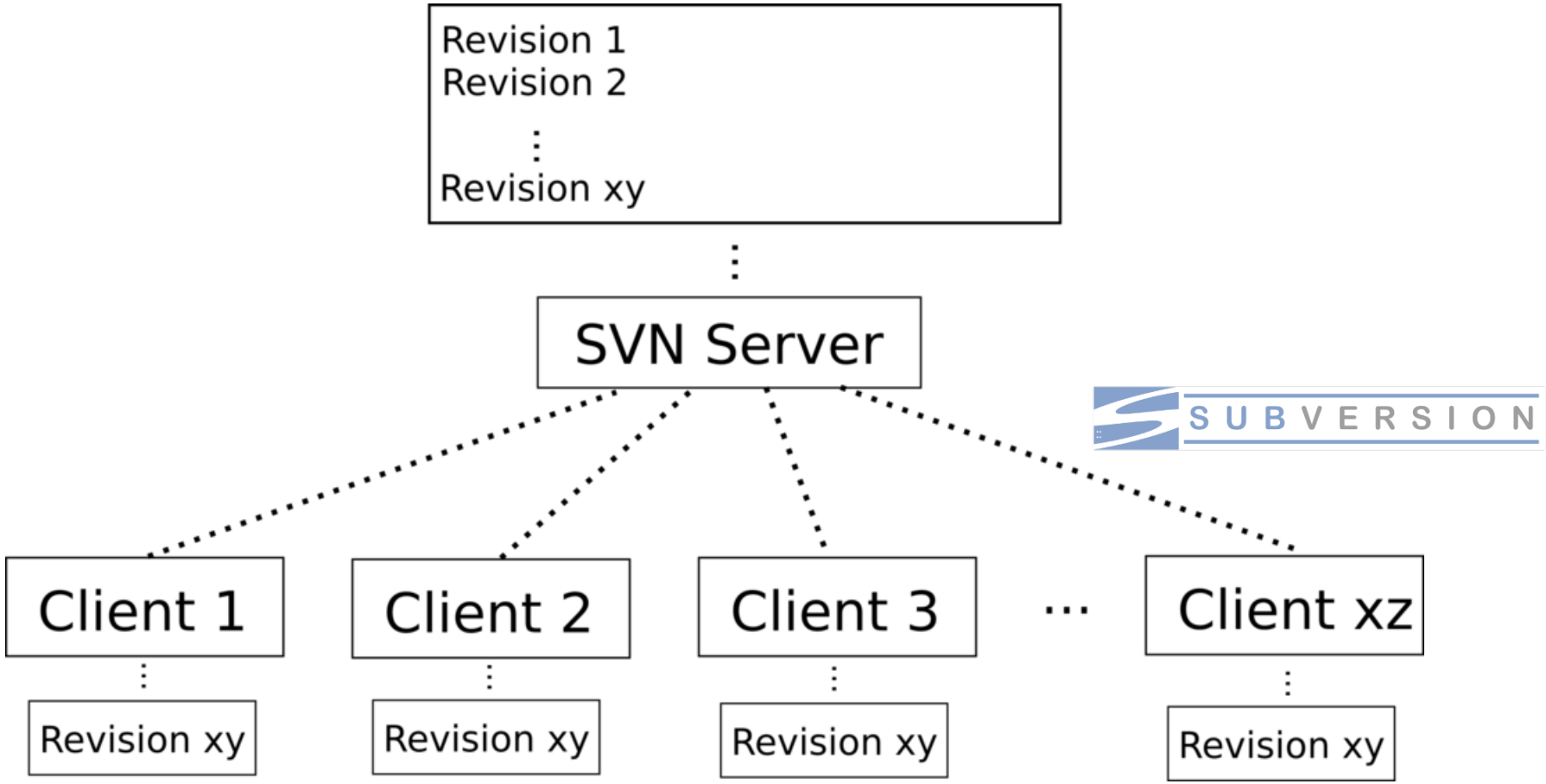
- **Verteilter Zugriff** auf Dateien als Voraussetzung für verteiltes kollaboratives Arbeiten
- **Versionierung**: Übersicht über verschiedene Versionen inklusiv der **Dokumentation** wer, wann, was geändert hat
- **Datensicherheit**, u.a. durch das Rücksetzen der Datei auf eine ältere Version
- **Automatisches Zusammenführen** (engl. merge) von Quelltexten bzw. Konflikterkennung, falls mehrere Entwickler Veränderungen im selben Bereich durchführen



Überblick

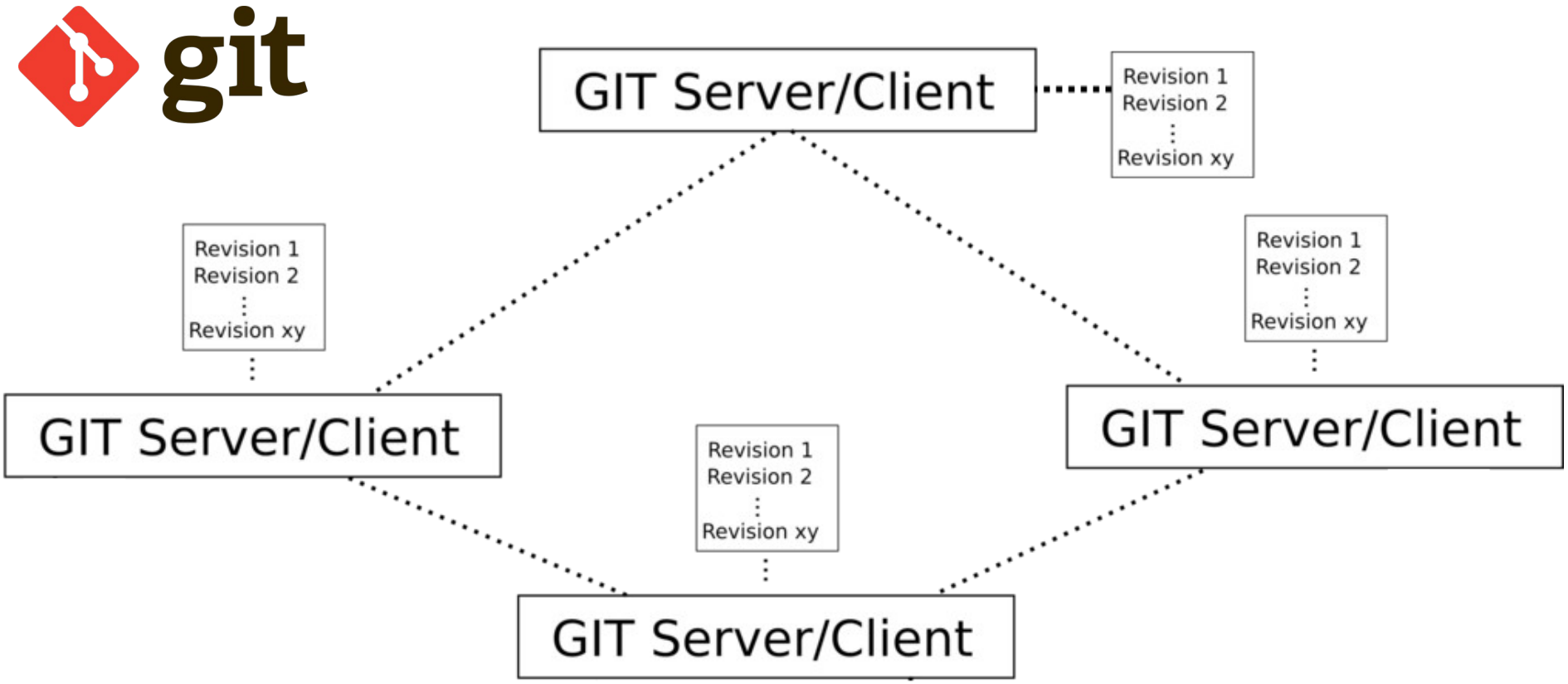
1. Grundlegendes zu Versionsverwaltung
2. Übung 1: Auschecken
3. Übung 2: Abgeben und Aktualisieren
4. Übung 3: Konflikte
5. SVN-Server
6. Übung 4: Workflow für Lehrer
7. Erfahrungen
8. Anwendungsbeispiel Entwurfsmuster MVC

Bei einer zentralen Versionsverwaltung werden bei Änderungen nur die Unterschiede übertragen



https://commons.wikimedia.org/wiki/File:SVNvsGITServer_1.png
Paul Vincent, Creative Commons Attribution 3.0 Unported

Bei einem verteilten Versionsverwaltungssytem besitzt jeder eine lokale Kopie inkl. der Historie

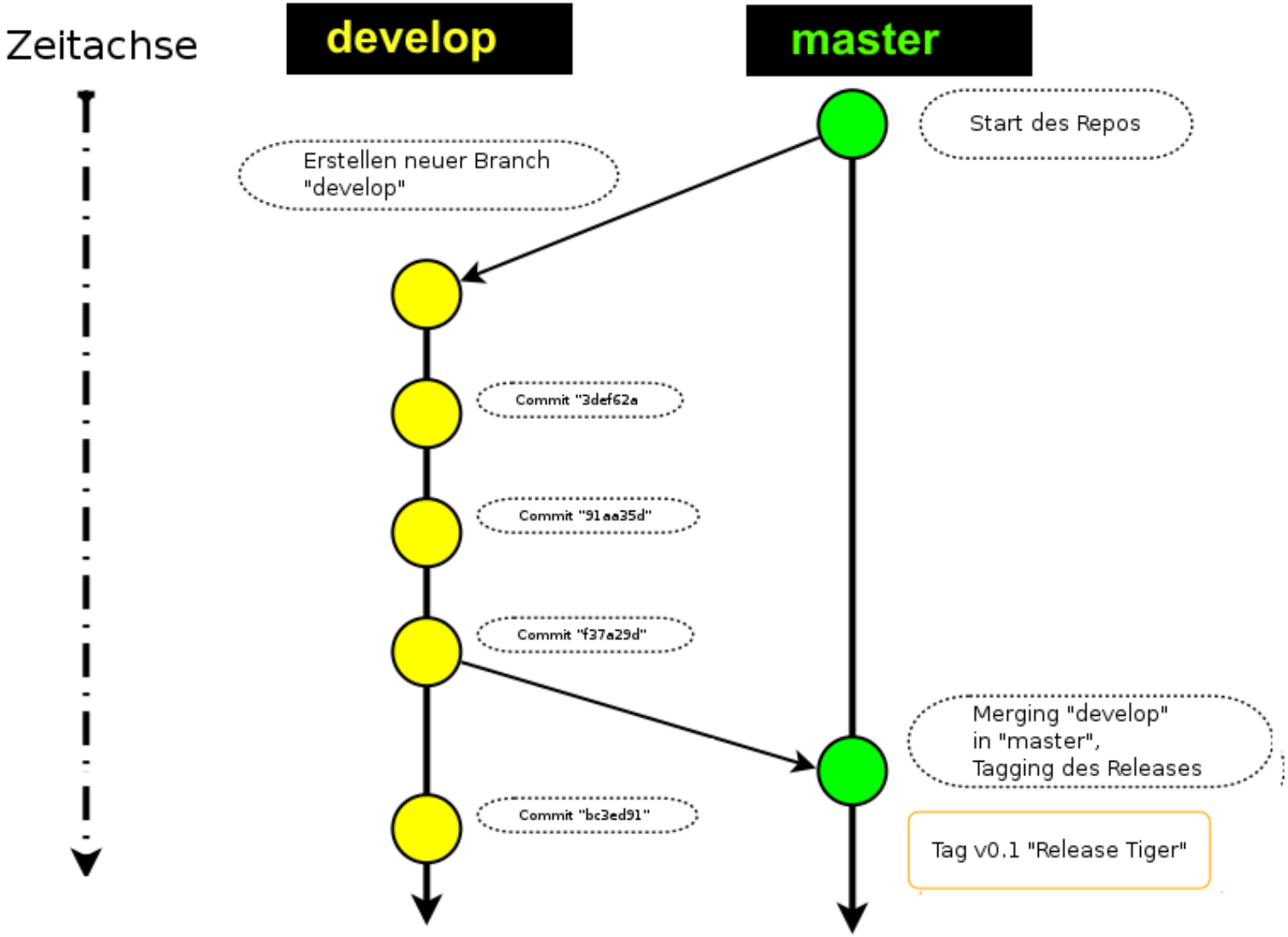


Quelle: https://commons.wikimedia.org/wiki/File:SVNvsGITServer_2.png
Paul Vincent, Creative Commons Attribution 3.0 Unported

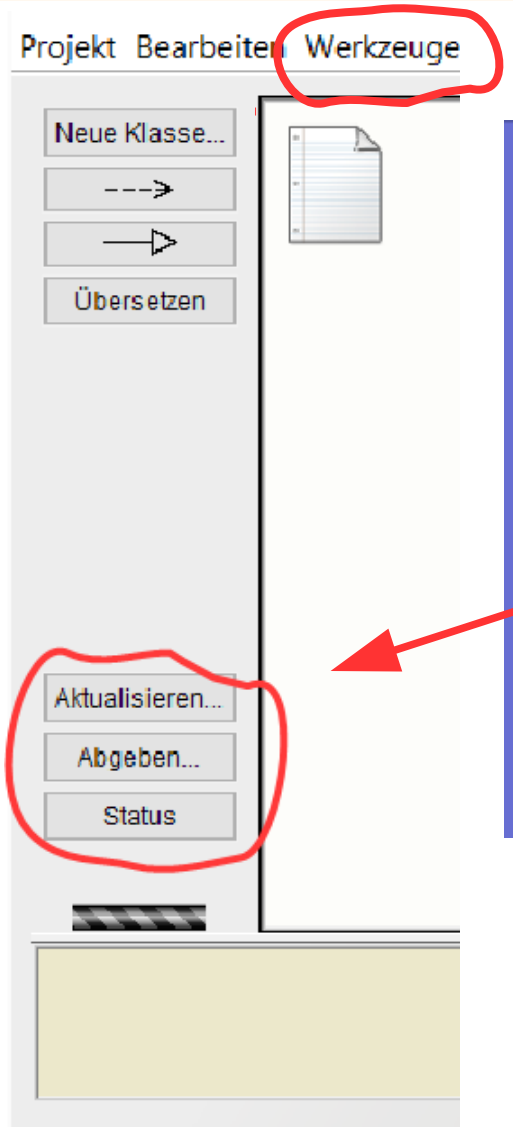
Die grundlegenden Operationen bei Repositories sind Checkout, Commit und Update

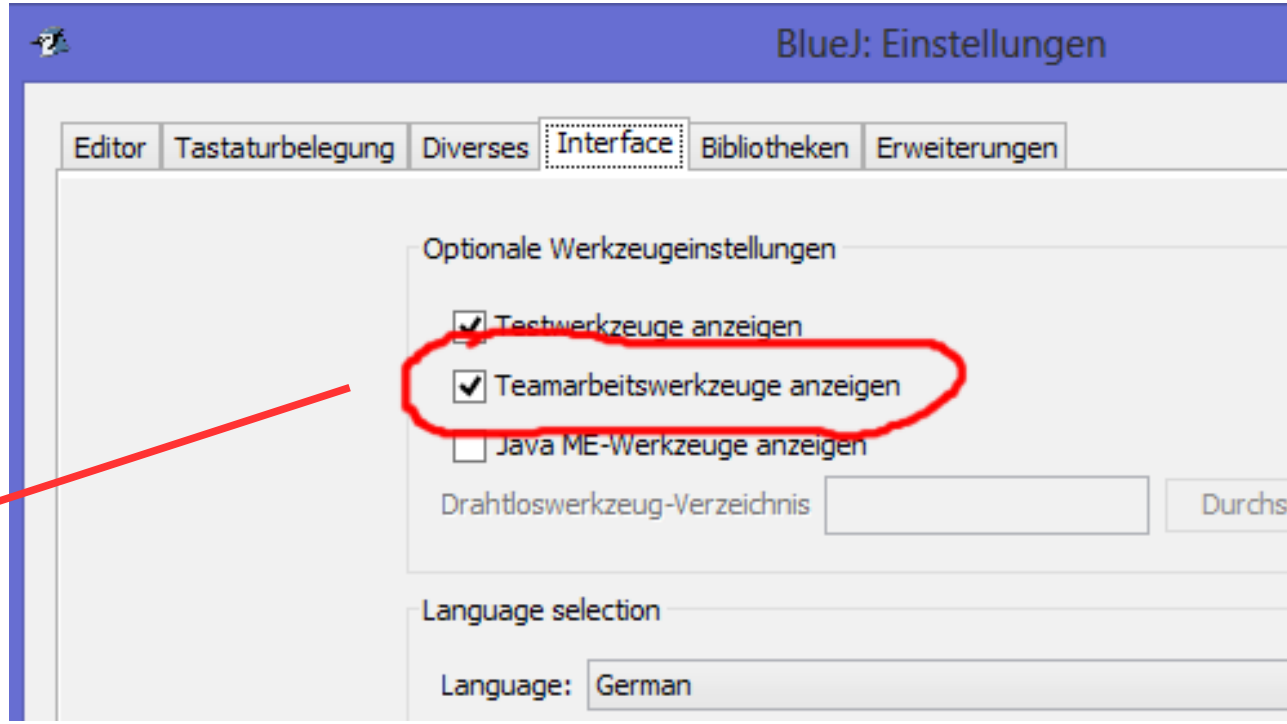
- **Checkout – Arbeitskopie erstellen:**
  Lokal alle Inhalte des Versionskontrollsystems neu laden.
- **Commit – Abgeben:** Eine Änderung am Inhalt des Versionskontrollsystems: Eine semantische Einheit ohne Übersetzungsfehler, built-fähig
- **Update – Aktualisieren**: Von anderen abgegebene Änderungen lokal laden.

Die Profis verwenden zwischen Entwicklung und Releases unterschiedliche Zweige.



In der Entwicklungsumgebung BlueJ ist ein SVN-Client integriert





Teamarbeits-Menü sichtbar machen

Überblick

1. Grundlegendes zu Versionsverwaltung
2. Übung 1: Auschecken
3. Übung 2: Abgeben und Aktualisieren
4. Übung 3: Konflikte
5. SVN-Server
6. Übung 4: Workflow für Lehrer
7. Erfahrungen
8. Anwendungsbeispiel Entwurfsmuster MVC

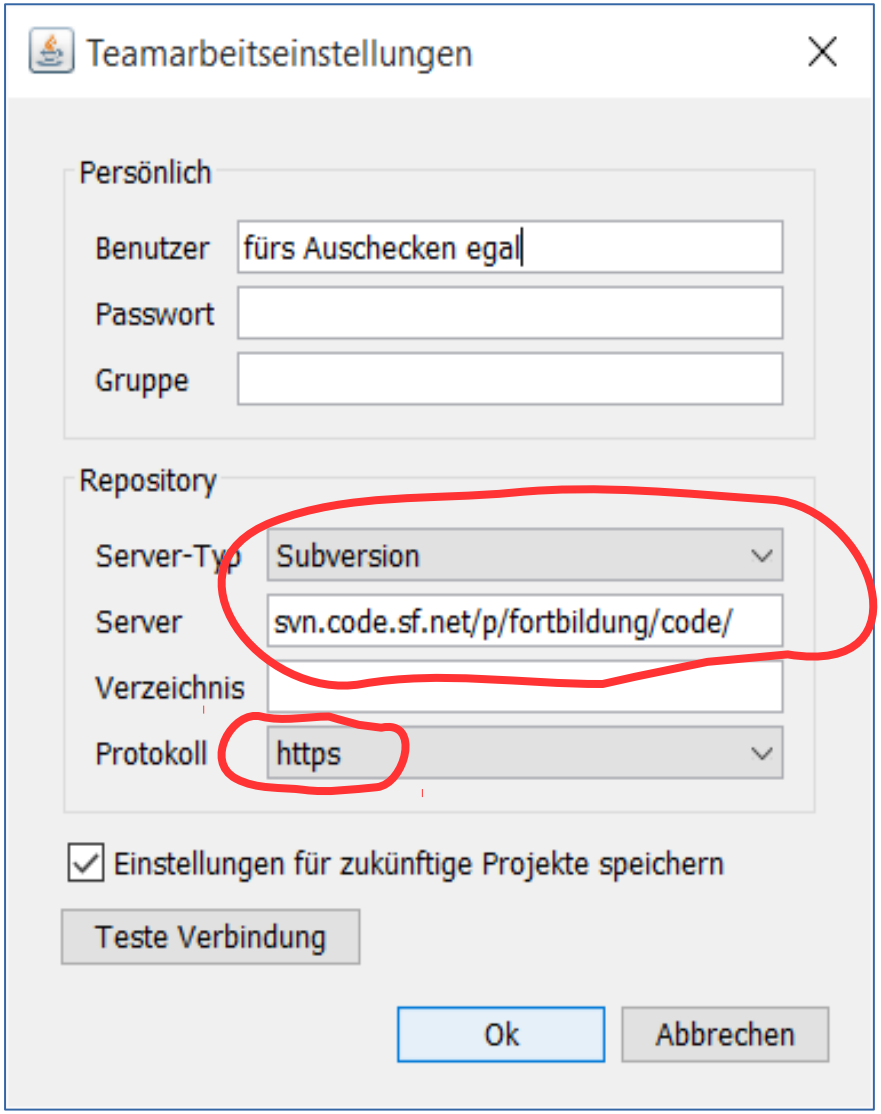
Übung 1: Erstmalig ein BlueJ-Projekt aus einem Repository auschecken

Bei einem öffentliches Repository ist für lesenden Zugriff keine Authentifizierung nötig.

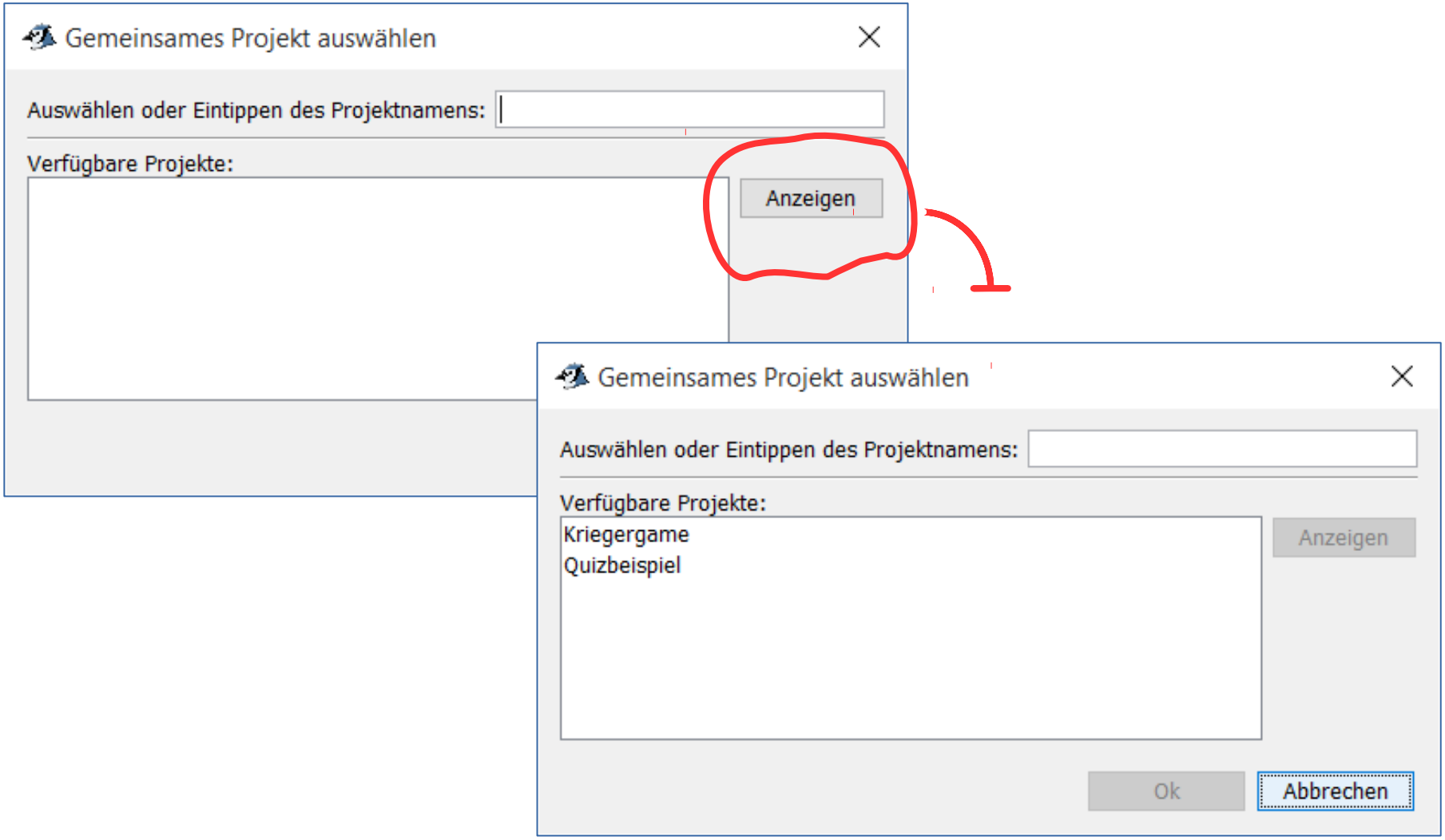
Menü Werkzeuge: Teamarbeit
→ Arbeitskopie erstellen

(Englisch:

Team
→ checkout project)



Übung 1: Erstmalig ein BlueJ-Projekt aus einem Repository auschecken



Übung 1: Erstmalig ein BlueJ-Projekt aus einem Repository auschecken

Aufgabe:
Checken Sie beliebige Projekte aus dem Server aus, aber auf jeden Fall das BlueJ-Projekt „SVN Aufgabe 1“ für den nächsten Schritt.

Benutzer: fortbildung

Passwort: **********

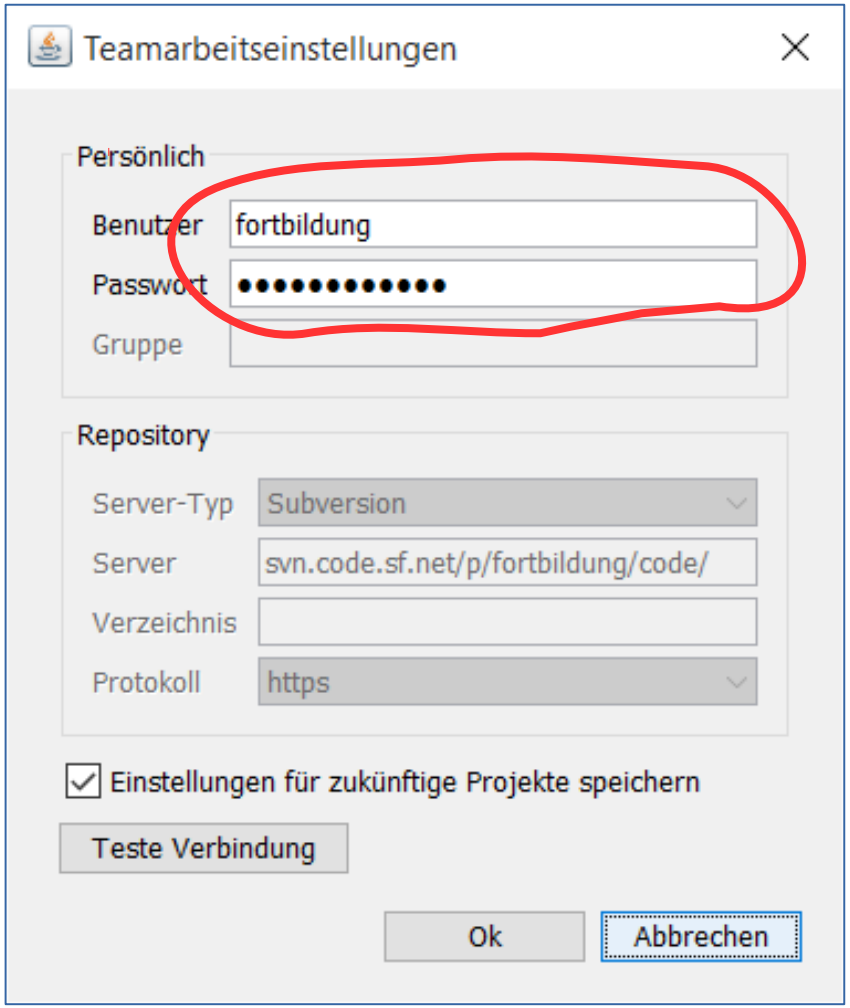
Server: svn.code.sf.net/p/fortbildung/code

Path: [leer]

Protokoll: https

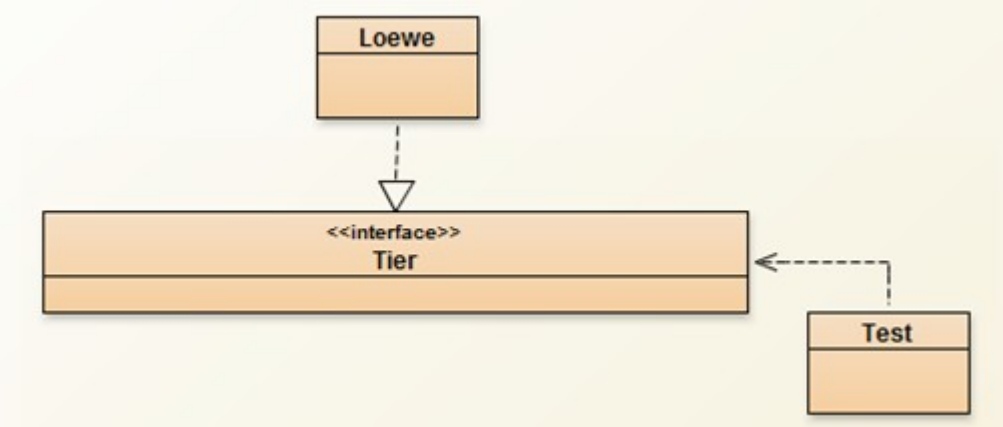
Übung 2:
Abgeben (Commit) und Aktualisieren (Update)

Authentifizierung über Benutzername und Passwort



Übung 2:
Abgeben (Commit) und Aktualisieren (Update)

Legen Sie in „SVN Aufgabe 1“ eine neue Klasse an, die das Interface „Tier“ implementiert. Die Klasse „Loewe“ kann als Beispiel dienen.

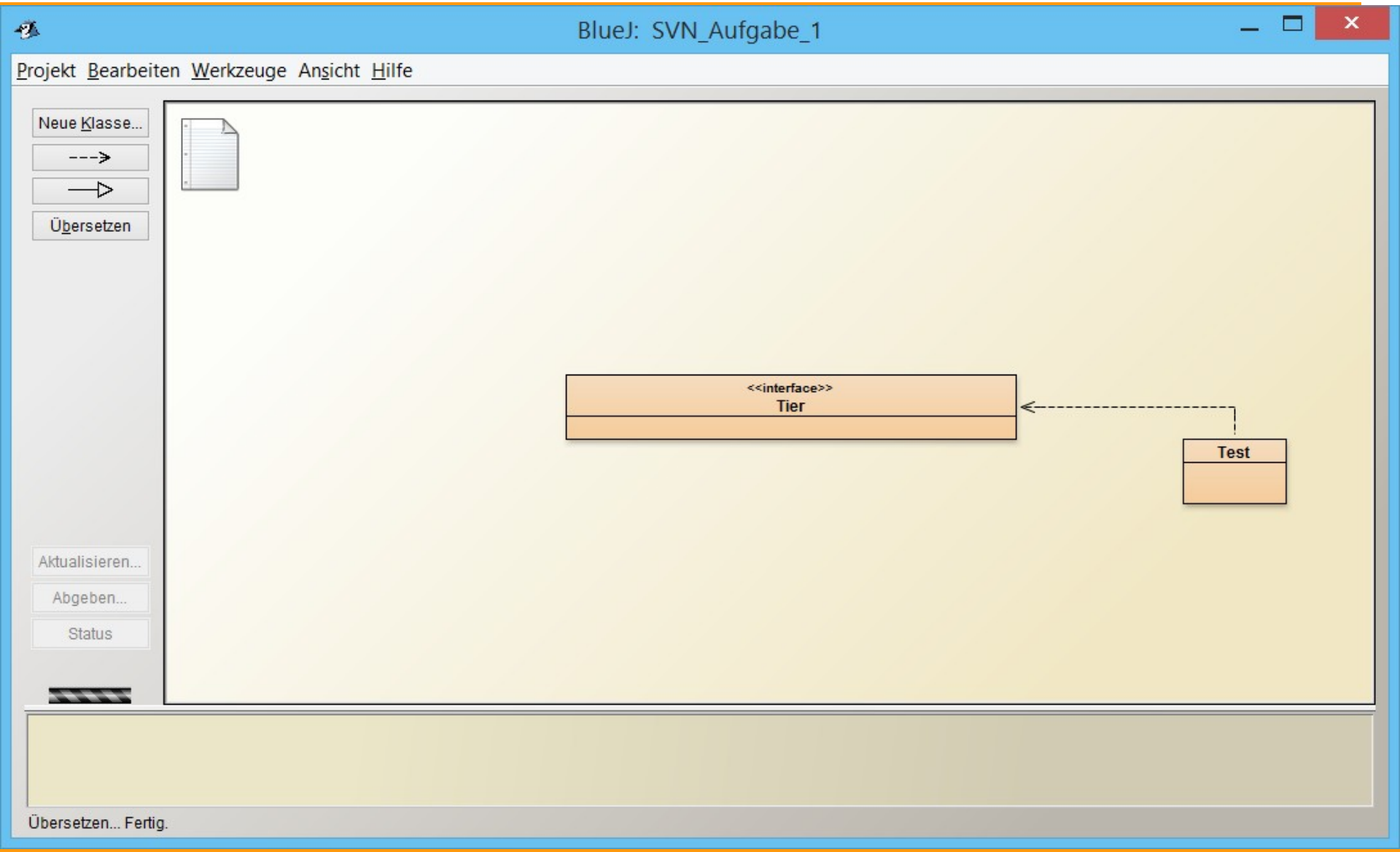


Aktualisieren Sie.

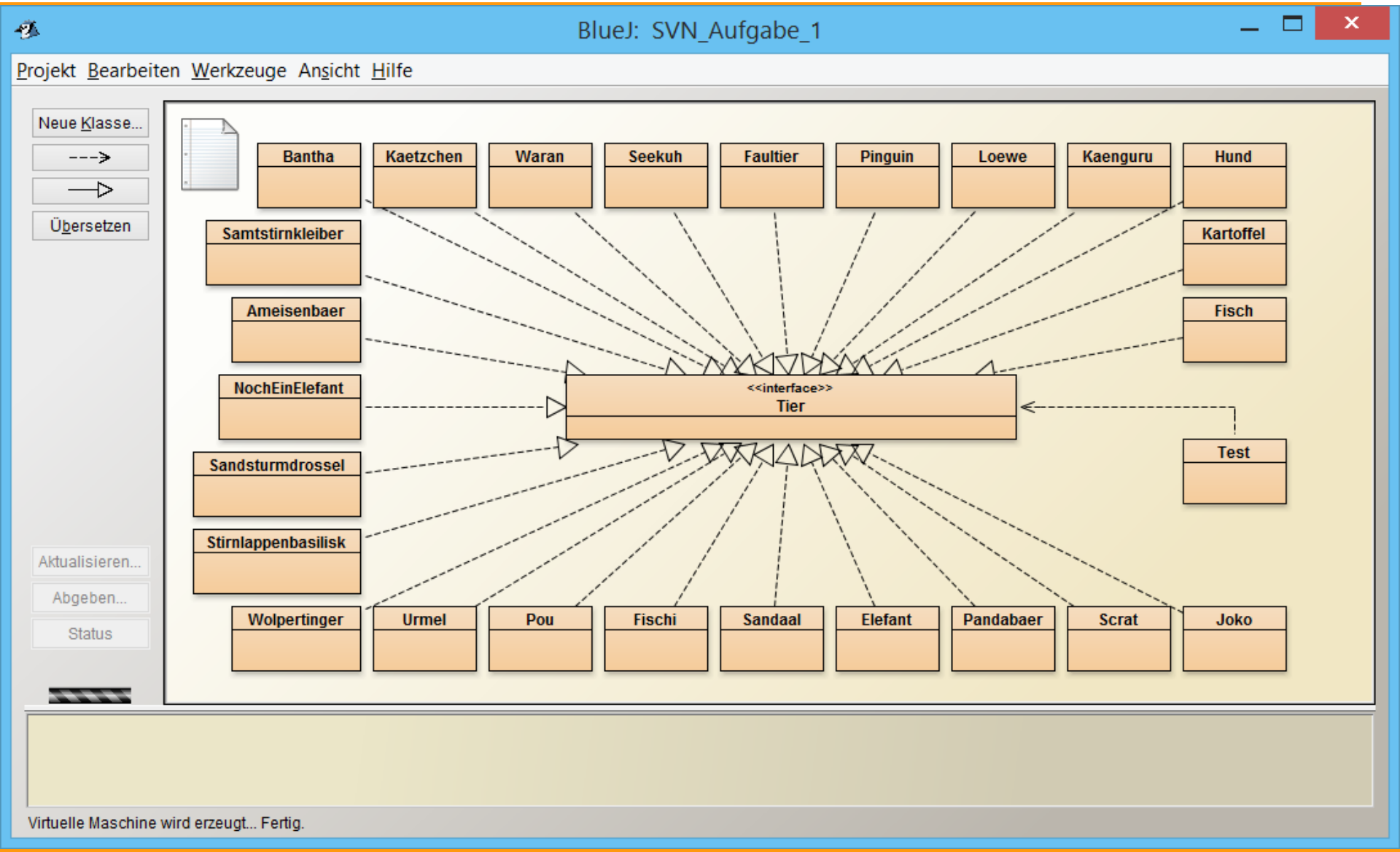
Geben Sie Ihre Änderungen mit einer aussagekräftigen Beschreibung (Kommentar) ab.

Führen Sie weitere Updates und Commits durch. Sollten irgendwelche Konflikte auftreten, so ignorieren Sie diese vorläufig.

Übung 2: Schülerergebnisse

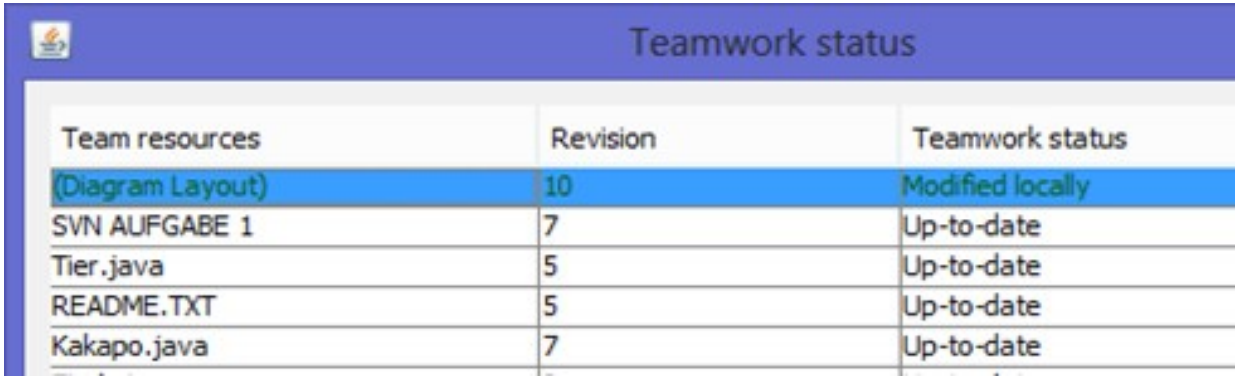


Übung 2: Schülerergebnisse



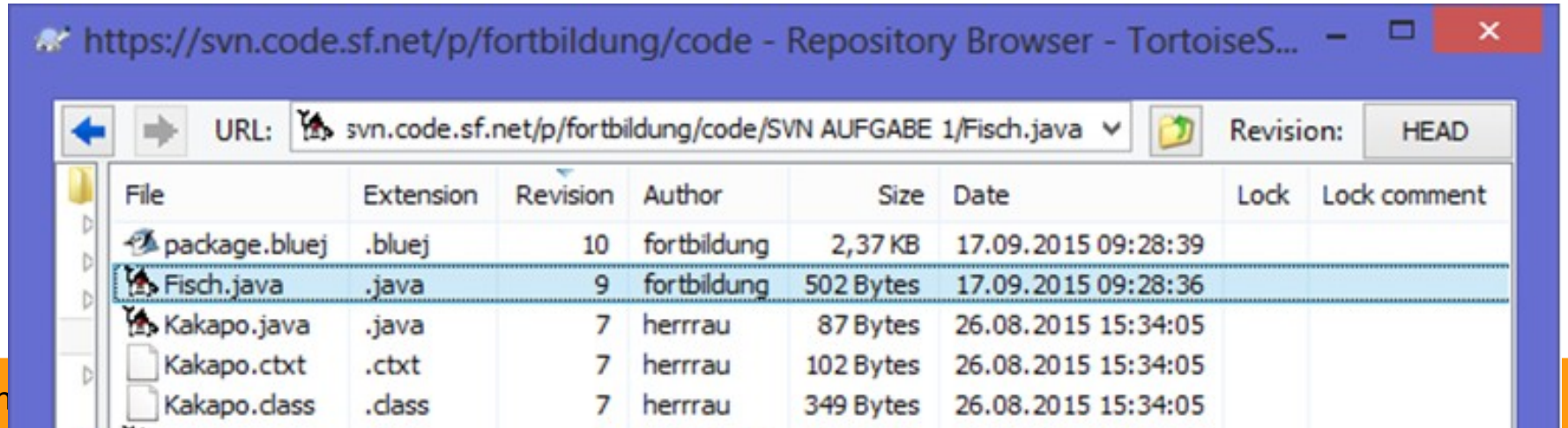
Übung 2: Das Repository speichert die Historie der Dokumentatenablage

Rufen Sie den Status ab.



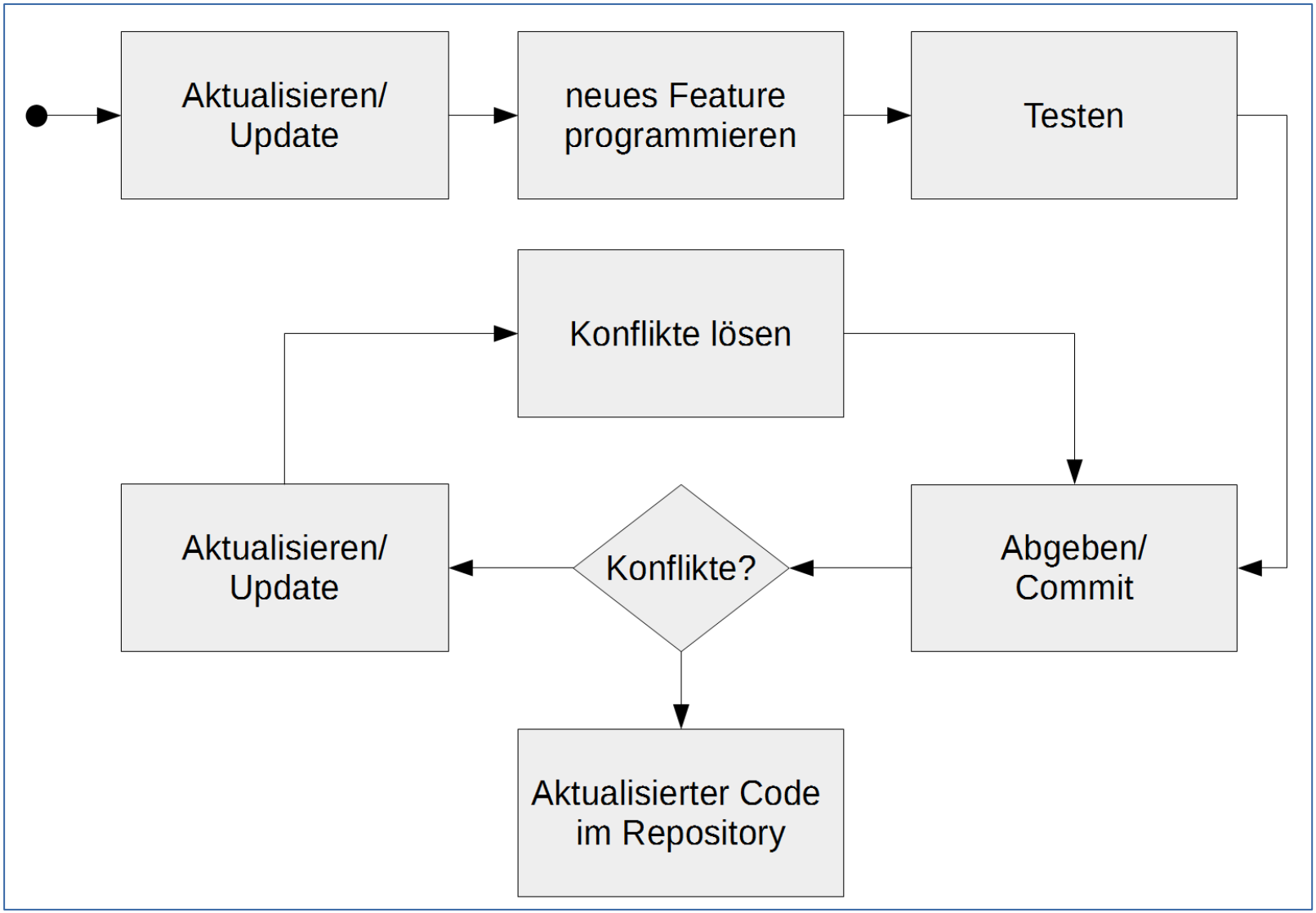
Team resources	Revision	Teamwork status
(Diagram Layout)	10	Modified locally
SVN AUFGABE 1	7	Up-to-date
Tier.java	5	Up-to-date
README.TXT	5	Up-to-date
Kakapo.java	7	Up-to-date

Nachteil an BlueJ-Client: Keine Information über Autor und Datum



File	Extension	Revision	Author	Size	Date	Lock	Lock comment
package.bluej	.bluej	10	fortbildung	2,37 KB	17.09.2015 09:28:39		
Fisch.java	.java	9	fortbildung	502 Bytes	17.09.2015 09:28:36		
Kakapo.java	.java	7	herrrau	87 Bytes	26.08.2015 15:34:05		
Kakapo.ctxt	.ctxt	7	herrrau	102 Bytes	26.08.2015 15:34:05		
Kakapo.class	.class	7	herrrau	349 Bytes	26.08.2015 15:34:05		

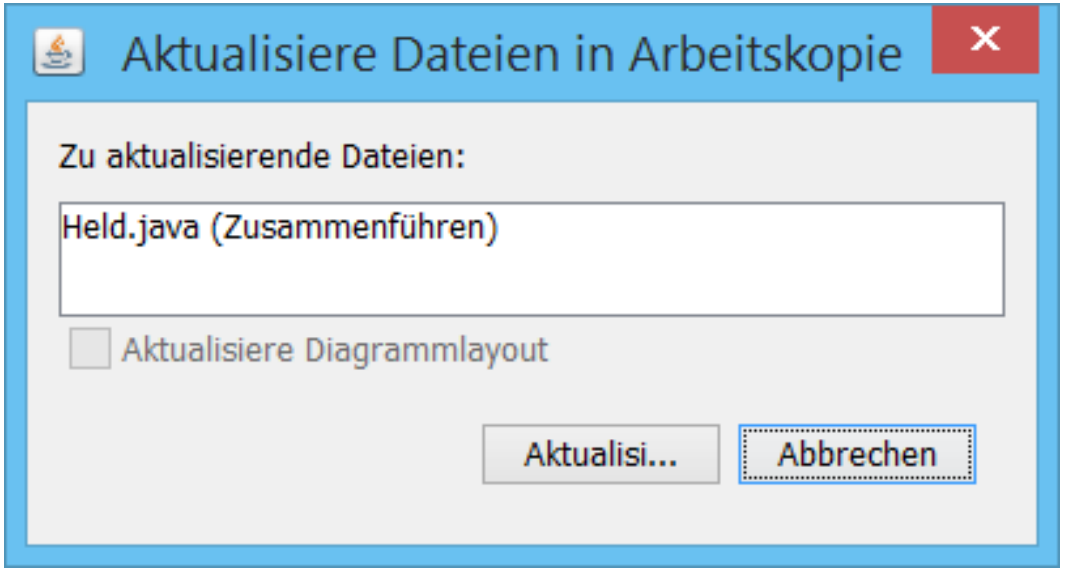
Grundsätzliche Abfolge

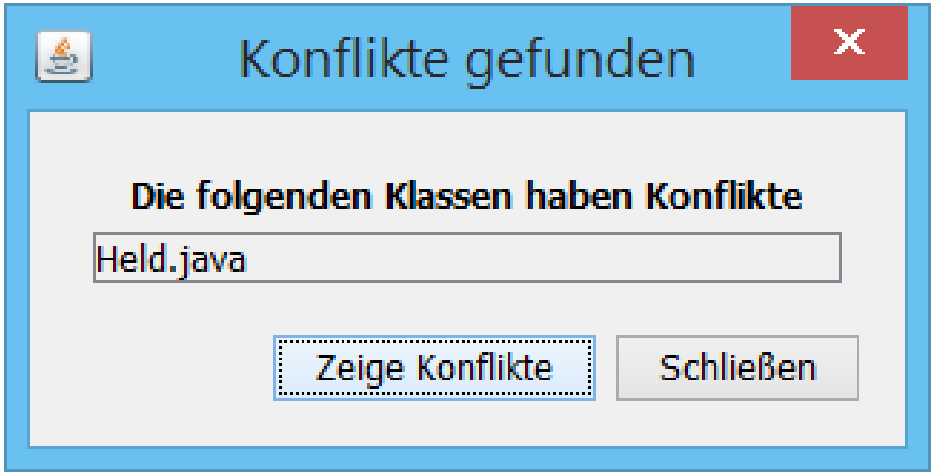


Übung 3: Konflikte

Aufgabe: Checken Sie das Projekt „SVN Aufgabe 2“ aus und ergänzen Sie eine noch nicht vorhandene getter- oder setter-Methode für ein Attribut der Klasse „Held“. Führen Sie dann einen Commit durch (bzw. vorher ein eventuell eingefordertes Update).

Übung 3: Konflikte automatisch zusammenführen





Übung 3: Konflikte manuell zusammenführen

Held - SVN AUFGABE 2

Klasse Bearbeiten Werkzeuge Optionen

Übersetzen | Rückgängig | Ausschneiden | Kopieren | Einfügen | Suchen... | Schließen | Quelltext

```
22      }
23
24  <<<<<<< .mine
25      public int geschicklichkeitGeben() {
26          return this.geschicklichkeit;
27      }
28
29  =======
30      public int geschicklichkeitGeben() {
31          return geschicklichkeit;
32      }
33
34
35  >>>>>>> .r187
36  }
37
```

Datei gespeichert

gespeichert

Überblick

1. Grundlegendes zu Versionsverwaltung
2. Übung 1: Auschecken
3. Übung 2: Abgeben und Aktualisieren
4. Übung 3: Konflikte
5. SVN-Server
6. Übung 4: Workflow für Lehrer
7. Erfahrungen
8. Anwendungsbeispiel Entwurfsmuster MVC

Der Server

Zum Beispiel: SourceForge

Vorteile: kostenlos

Nachteile: öffentlich, User müssen einzeln angelegt werden

☑ **Wiki**
Documentation is key to your project and the wiki tool helps make it easy for anyone to contribute.

☐ **Files & Stats**
Use the largest free, managed, global mirror network to distribute your files, and follow the download trends that enable you to develop better software.

☐ **Git**
Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

☑ **Tickets**
Bugs, enhancements, tasks, etc., will help you plan and manage your development.

☑ **Forums**
Collaborate with your community in your forum.

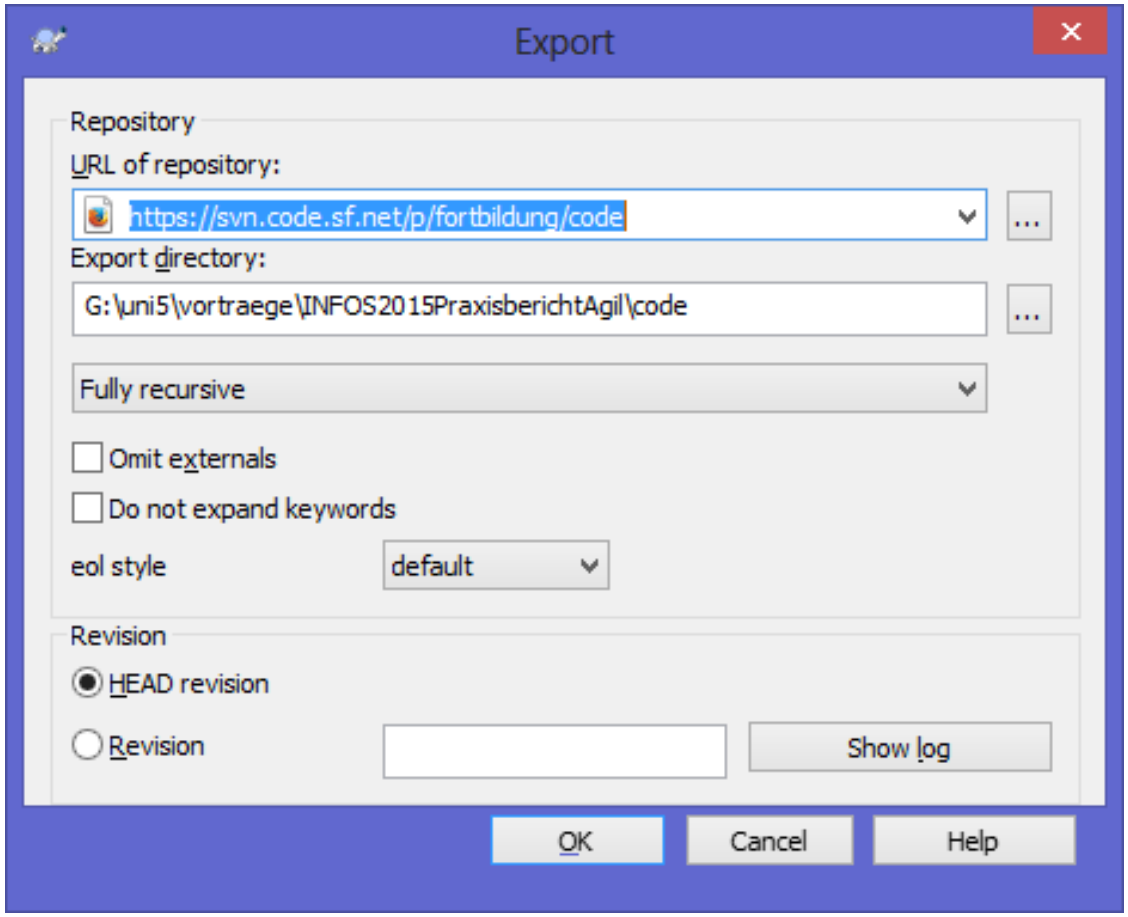
☐ **Blog**
Share exciting news and progress updates with your community.

☑ **SVN**
Enterprise-class centralized version control for the masses.

☐ **Mercurial**
Mercurial is a distributed source control management tool that efficiently handles projects of any size and offers an easy and intuitive interface.

Übung 4: Workflow für die Lehrkraft

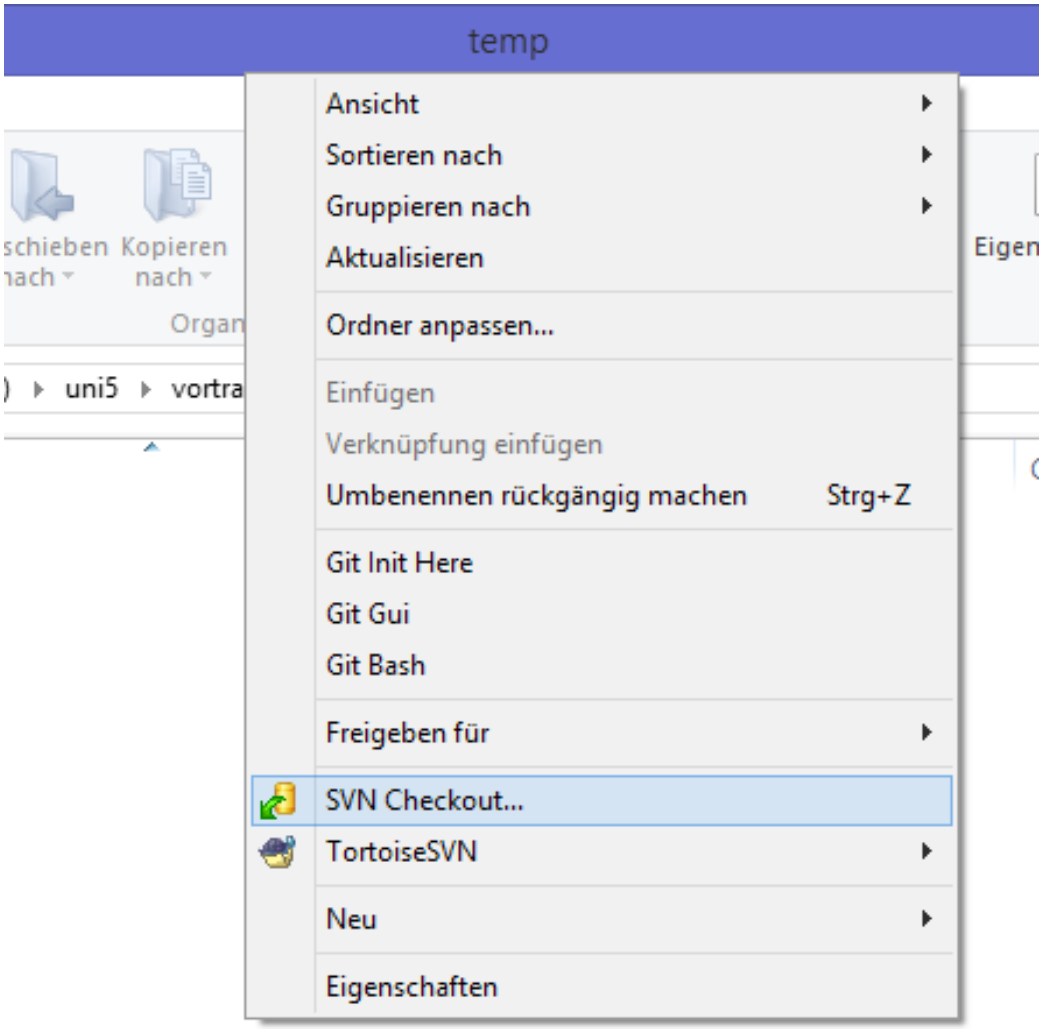
Tortoise SVN – Auswahl von zu synchronisierenden Verzeichnissen



Übung 4: Workflow für die Lehrkraft – Checkout

SVN Client:
Tortoise SVN –

Checkout über Kontextmenü in der Dateiverwaltung

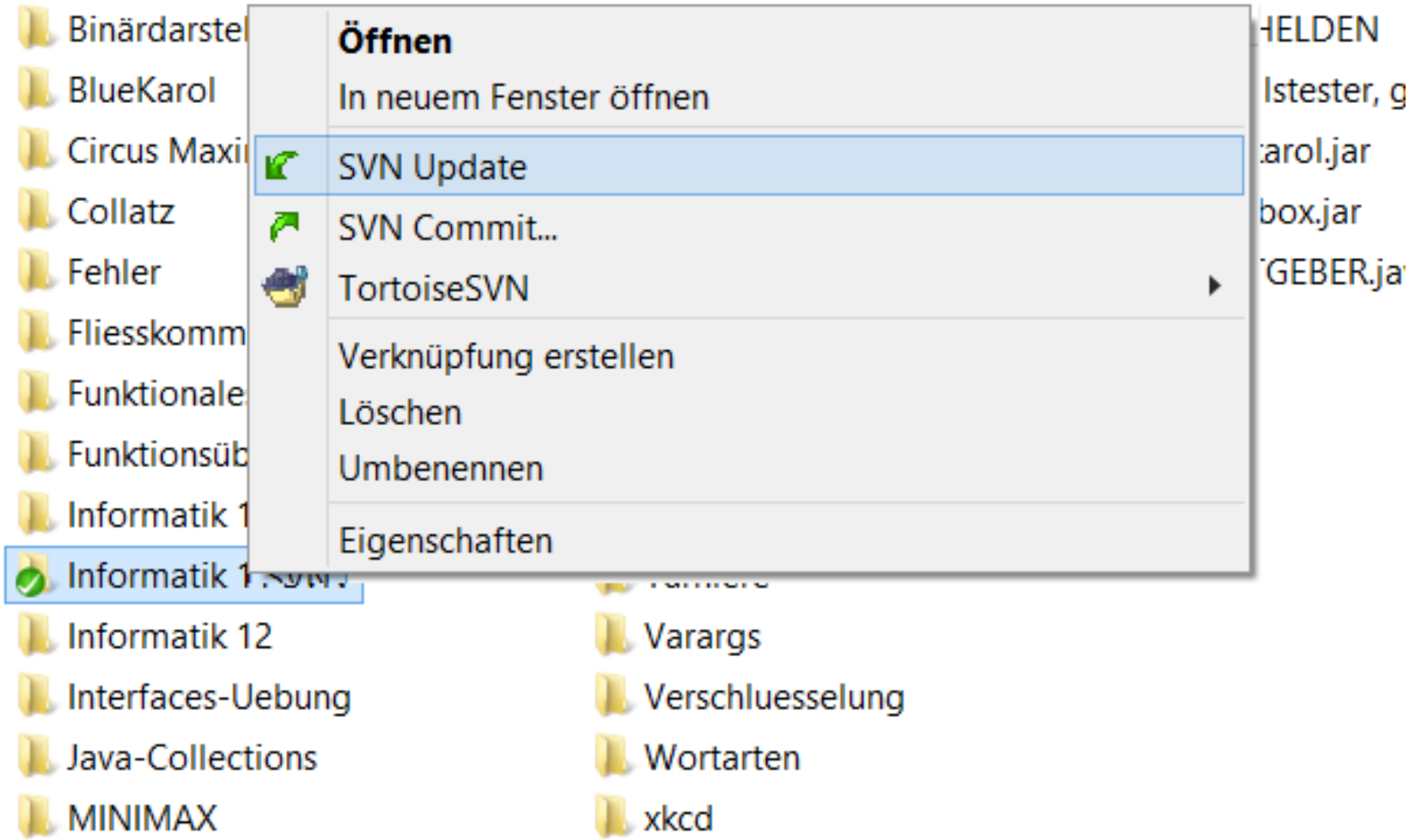


Übung 4: Workflow für die Lehrkraft

Tortoise SVN – Auswahl von zu synchronisierenden Verzeichnissen

- 9-5 Binbaum tiefeGeben
- 9-a Bäume (aus Gdl)
- 9-Aufgabe xxx (Faust sortieren & File Reader)
- 10-1 Graph mit Liste (S. 99)
- 11-0 Graph mit MVC-Fenster
- 11-1 Graph ohne Einfuegen
- 11-2 Graph mit Einfuegen
- 11-3 Graph mit Personen (Zentralitaet, Aufgabe)
- 11-4 Graph mit Personen (Zentralitaet, Loesung)
- 11-5 Autobahnen (109-4b, Schuelerloesung)
- 12-1 Graph mit Tiefensuche
- 12-2 Irrgarten 115-4

Workflow für die Lehrkraft: Update und Commit



Überblick

1. Grundlegendes zu Versionsverwaltung
2. Übung 1: Auschecken
3. Übung 2: Abgeben und Aktualisieren
4. Übung 3: Konflikte
5. SVN-Server
6. Übung 4: Workflow für Lehrer
7. Erfahrungen
8. Anwendungsbeispiel Entwurfsmuster MVC

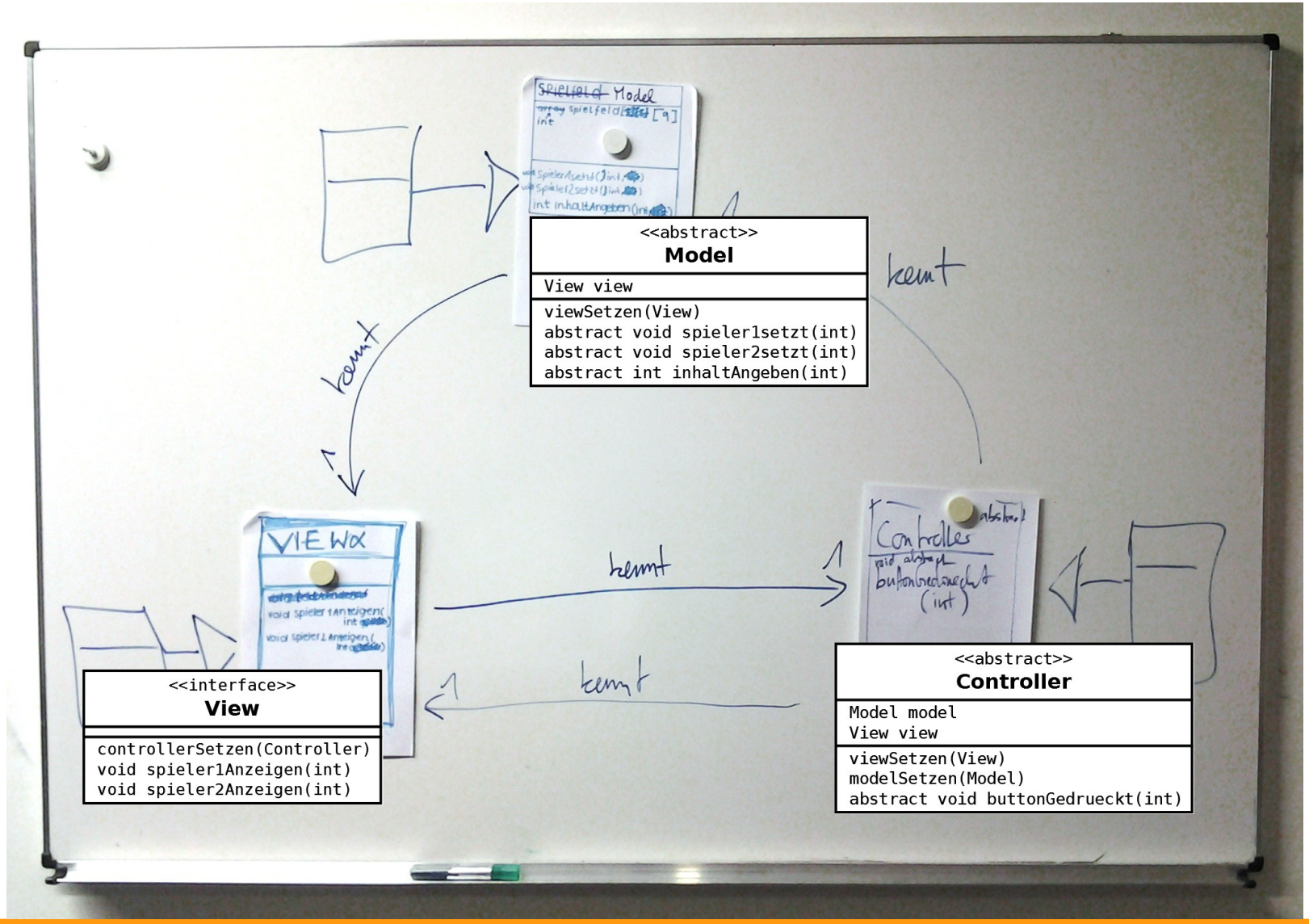
Erfahrungen

Rückmeldungen aus der Praxis

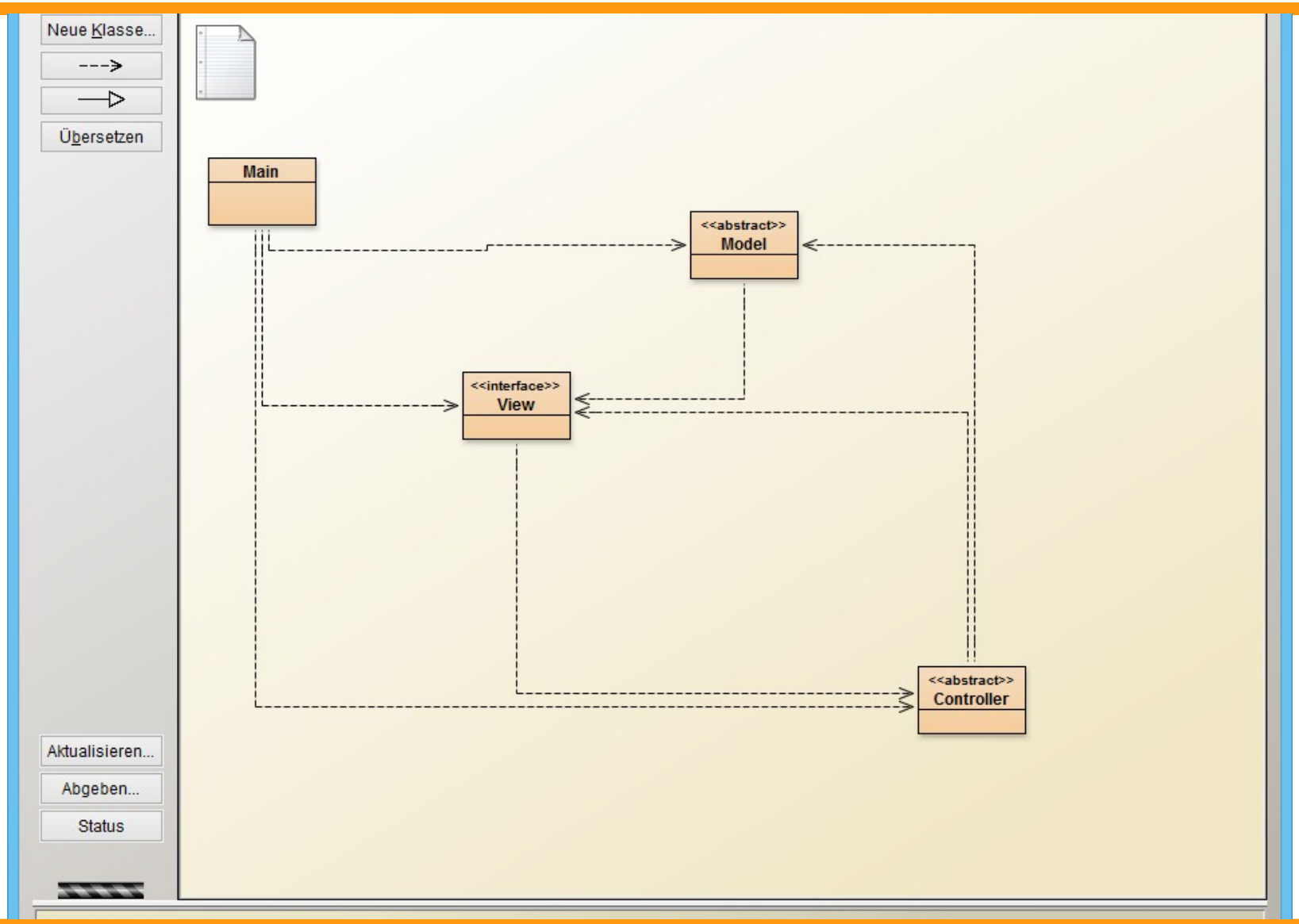
Typische Probleme

Randall Munroe/xkcd, Creative Commons Attribution-NonCommercial 2.5

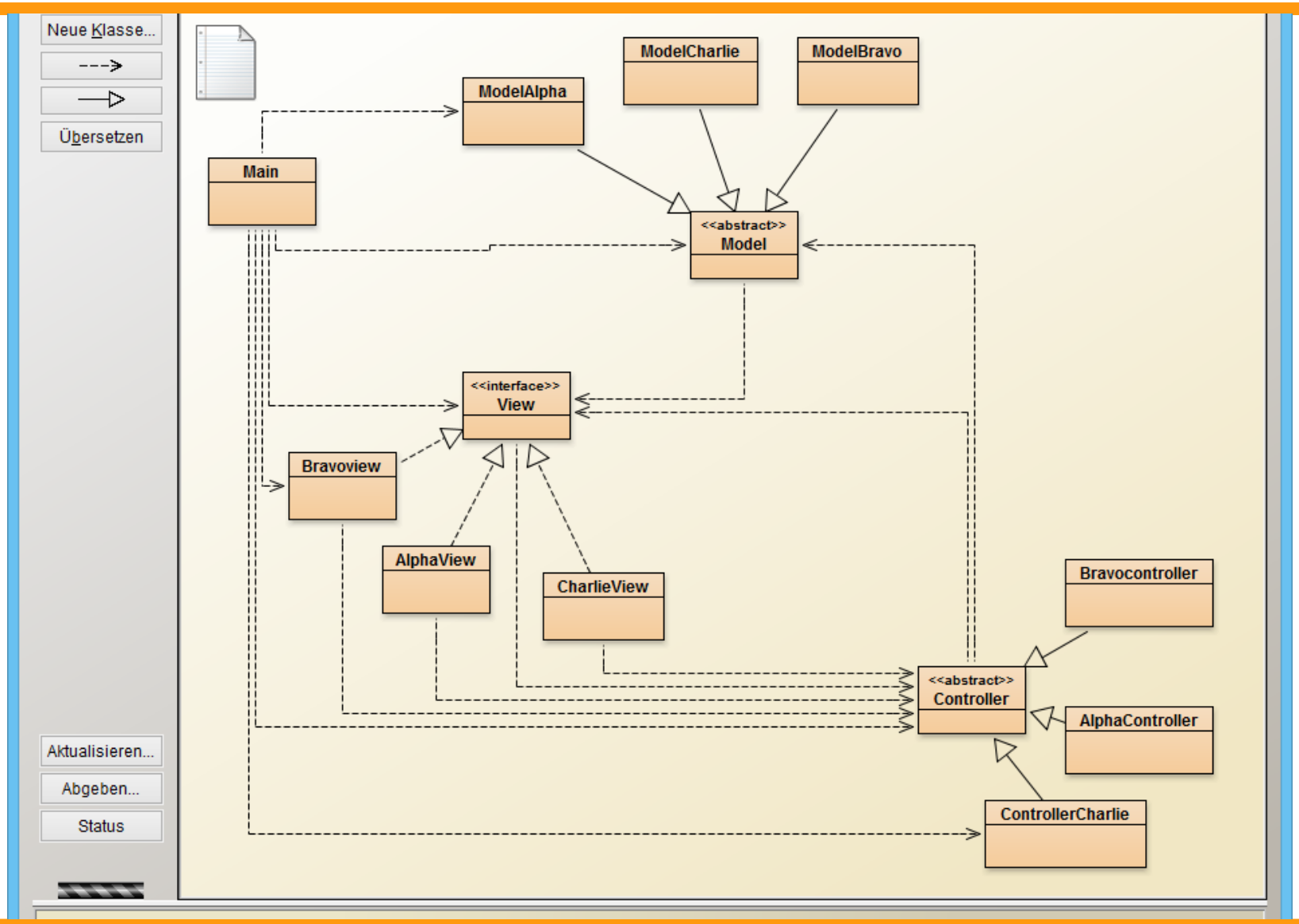
Anwendungsbeispiel Entwurfsmuster MVC



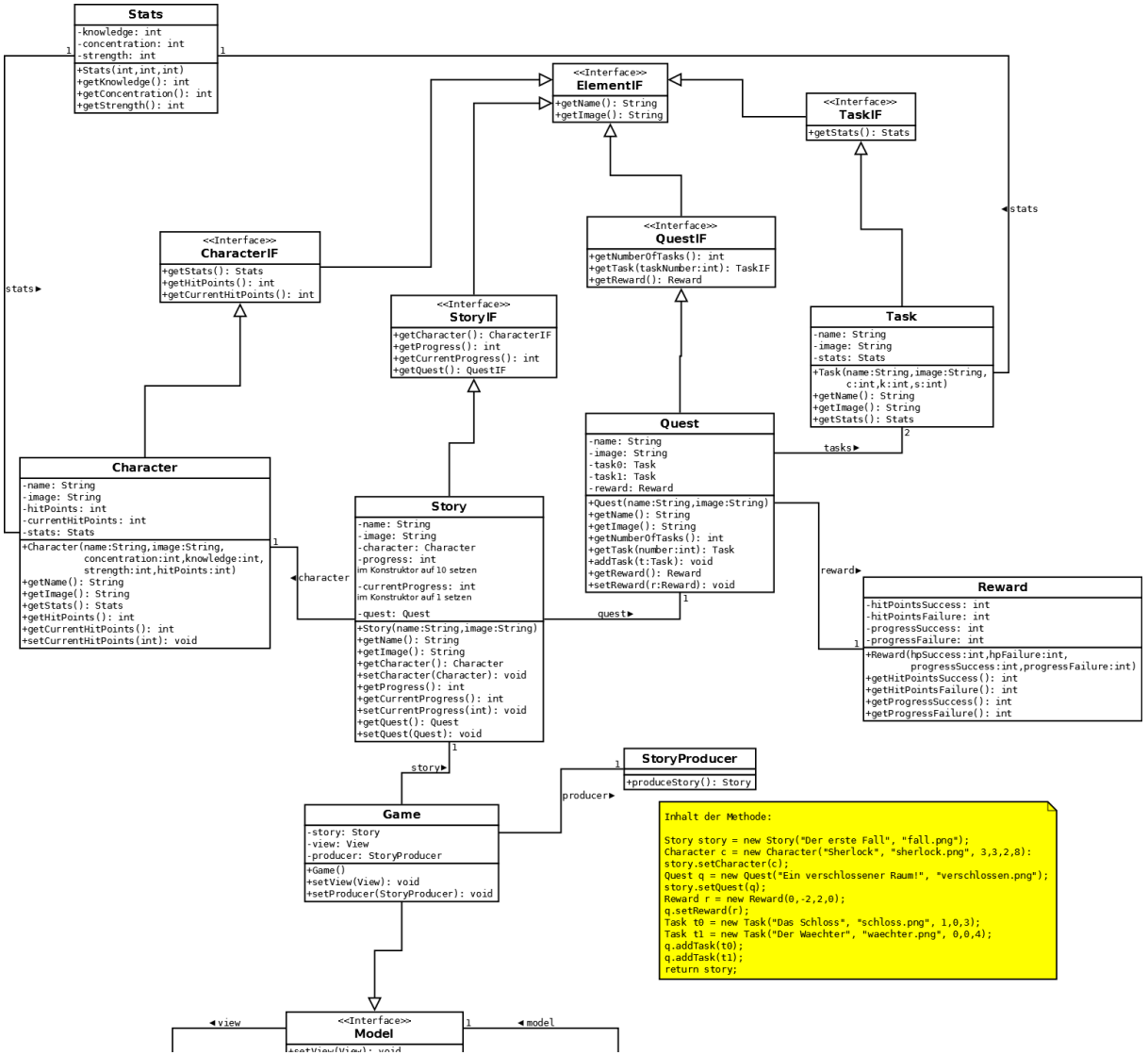
Anwendungsbeispiel Entwurfsmuster MVC



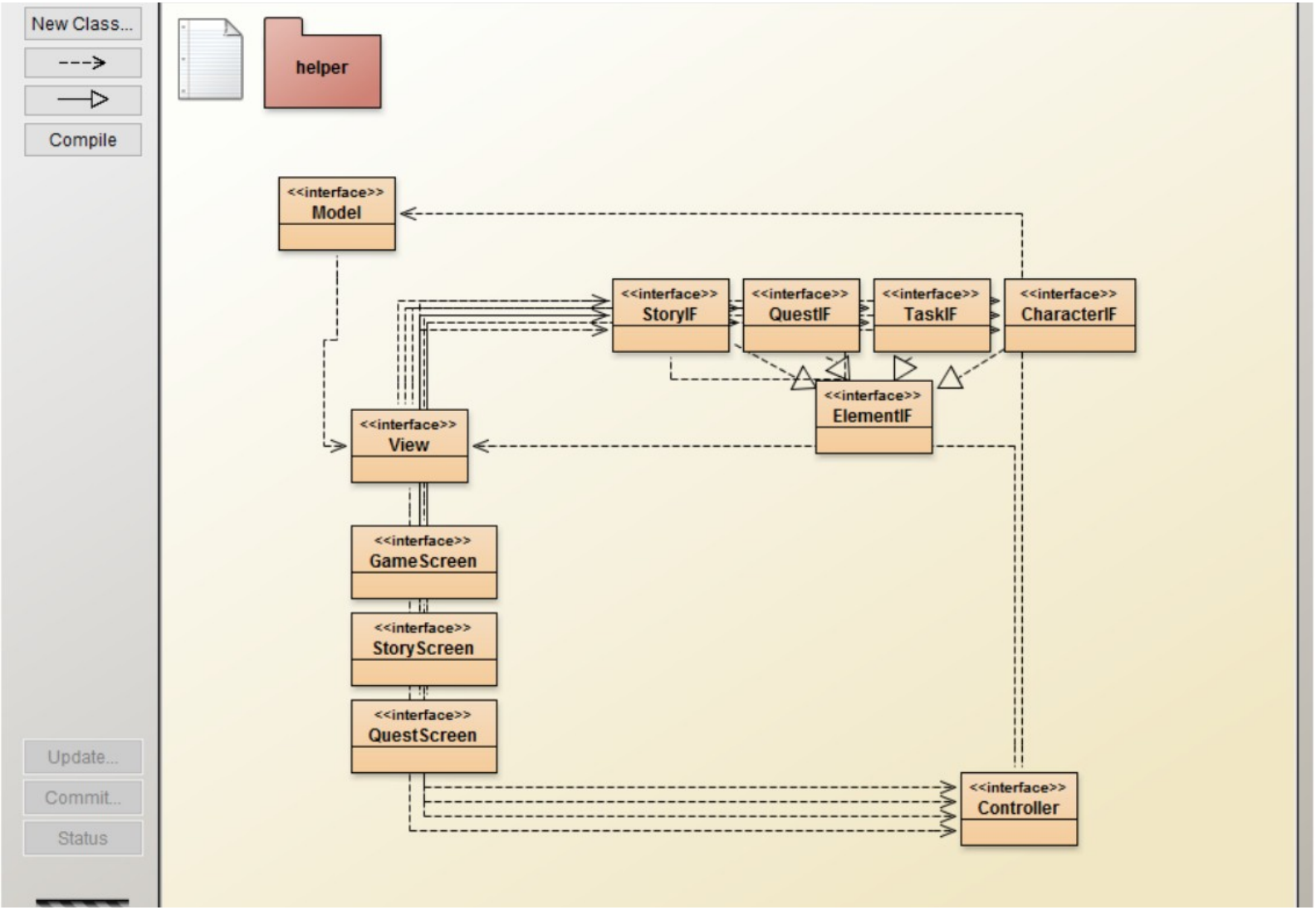
Anwendungsbeispiel Entwurfsmuster MVC



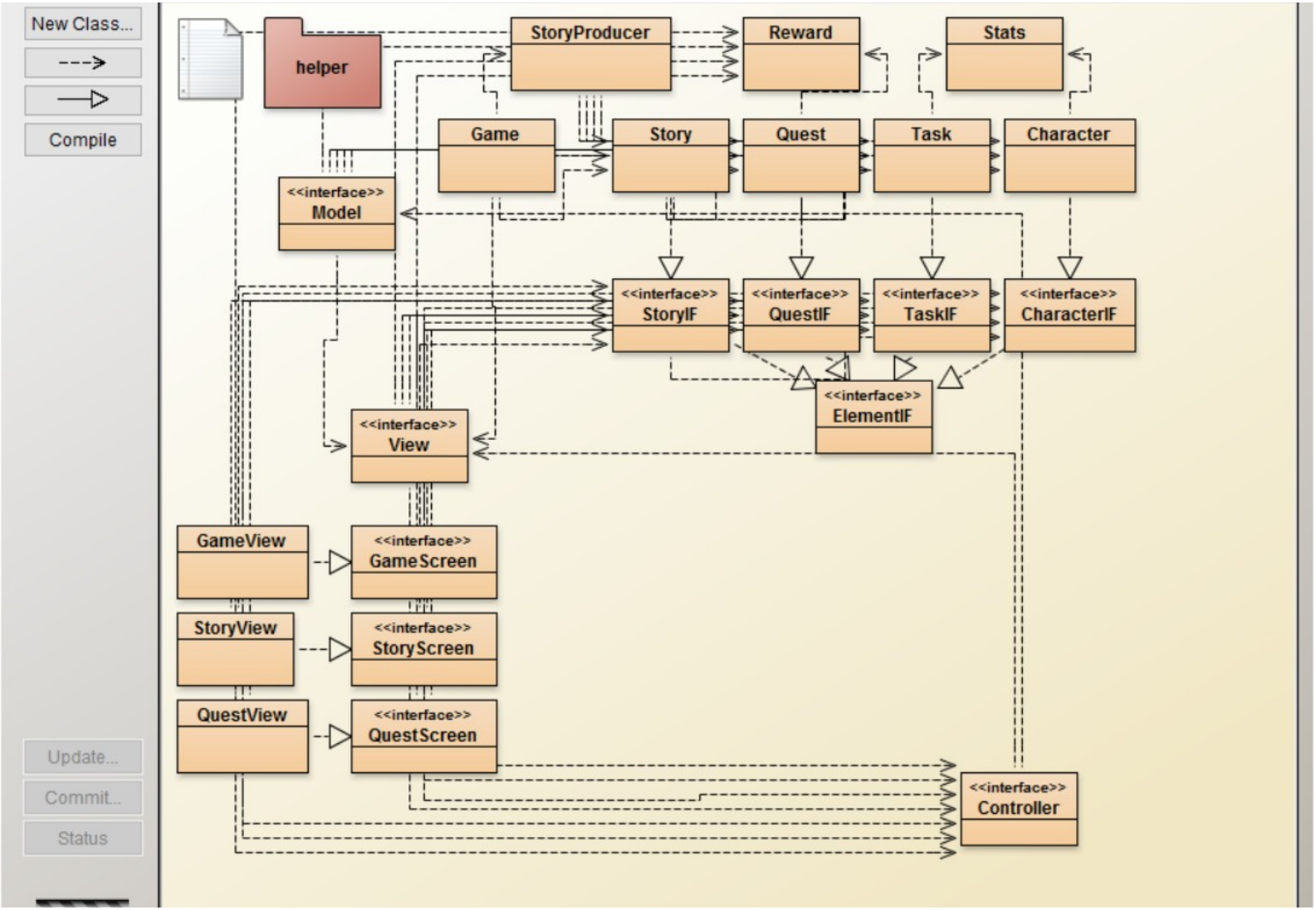
Anwendungsbeispiel Programmier-AG



Anwendungsbeispiel Programmier-AG



Anwendungsbeispiel Programmier-AG



Fazit

Repositories zur Unterstützung von kollaborativen Arbeiten in Softwareprojekten

Versionskontrollsysteme bieten bei (Software-entwicklungs-)Projekten einen großen Mehrwert

- In Softwareprojekten gibt es oft organisatorische Probleme, die der inhaltlichen Arbeit ablenken!
  Versionskontroll-Systeme bieten mit
  Verteiltem Zugriff
  Versionierung
  Datensicherheit
  Automatischem Zusammenführen
  Lösungen zu den Problemen.
- Repositories vereinfachen dem Lehrer das Verteilen von Daten.
- Mit Versionskontrollsysteme können Schülerinnen und Schüler deutlich und schnell erfahren, welchen **Gewinn kollaboratives Arbeiten** hat.

Material: ddi.ifi.lmu.de/fortbildungen