Imperative Programmierung (aka IP)

Übung 1: Entwicklungsumgebung und Versionskontrolle

Justin Kreikemeyer
Informatik, Uni Rostock

Organisatorisches

Wer bin ich?

M.Sc. Justin Kreikemeyer (Du, “Justin”)

Wissenschaftlicher Mitarbeiter und Promotionsstudent am Lehrstuhl für Modellierung und Simulation (MoSi), Institut für Visual and Analytic Computing (VAC)

- VAC/MoSi/Justin
- justin.kreikemeyer@uni-rostock.de
- Konrad-Zuse-Haus, R229
- +49 381 498-7614
- Sprechzeiten nach Vereinbarung

Wann findet diese Übung statt?

- (238843) WINF Gruppe 1: Mittwochs, 09:00 - 10:30 Uhr (PC-Pool Raum 201)
- (238843) WINF Gruppe 2: Freitags, 09:15 - 10:45 Uhr (PC-Pool Raum 201)
- (238841) INF Gruppe 3: Freitags, 15:00 - 16:30 Uhr (PC-Pool Raum 201)

Vorlesung vs. Übung vs. Praktikum

- Vorlesung: Präsentation eines Themas als Vortrag
- Übung:
  - Fragerunde
  - Vertiefung der wichtigsten Vorlesungsinhalte
  - Erlernen von fachspezifischen Fähigkeiten
  - Vorbereitung auf die Prüfung
  - *Nicht* Ersatz für die Vorlesung oder Praktikum
- Praktikum: Anwendung auf praktische Probleme, Vertiefung der Fähigkeiten

Hinweise für den Erfolg

- Besucht *alle drei* Veranstaltungen
- Übt zu Hause weiter ($\rightarrow$ Selbststudium)
- Lasst euch von anderen helfen ($\rightarrow$ Hausaufgaben-/Lerngruppe)
- Checkt regelmäßig eure Uni-Mails (1-2 Mal pro Tag)
- **Wenn ihr Fragen habt, stellt sie!**; ggf. per Mail/Stud.IP
  - zur Vorlesung: Prof. Mühl
  - zur Übung/Hausaufgaben: Übungsleiter
  - zum Praktikum: Tutoren

Hinweise zur Nutzung generativer KI

Aktuelle große Sprachmodelle wie Llama, ChatGPT, etc. . . .

- . . . eignen sich als *Werkzeug* für *erfahrene* Programmierer
- . . . werden wohl (irgend-)eine Rolle in eurem Beruf spielen
- . . . können vermutlich Programmieraufgaben im ersten Semester lösen
- . . . eignen sich **nicht** dazu, das Programmieren zu erlernen!

Denn:

- Die Güte der Antwort kann nur mit Grundlagenwissen eingeschätzt werden
- **Programmieren lernt man nur durch selber Programmieren**!

Organisation Prüfungsvorleistung aka Hausaufgaben

- Prüfungszulassung: 50% der Hausaufgabenpunkte
- Bearbeitung in Gruppen von 3-4 Studierenden
- Eintragen in Stud.IP in der Vorlesungsveranstaltung unter Teilnehmende→Gruppen (links am Rand)
- **Eintragung nur bis 23.10. möglich!**
- Abgabe der Hausaufgaben via Versionskontrollsystem SVN `https://svn.informatik.uni-rostock.de/lehre/ip2024/groups/<nr>`, wobei <nr> durch die Gruppennummer zu ersetzen ist
- Bewertet wird nur, was bis zur Deadline in SVN hochgeladen (“eingecheckt”) ist!

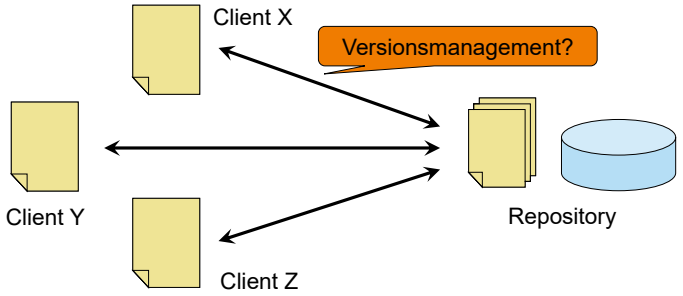
Erste HA 20.10.2024 - 27.10.2024 (23:59)

Ende Organisatorisches

Plan für heute

1. Versionskontrolle (Folien von Dr. Parzyjegla)
2. Arbeitsumgebung (Software)

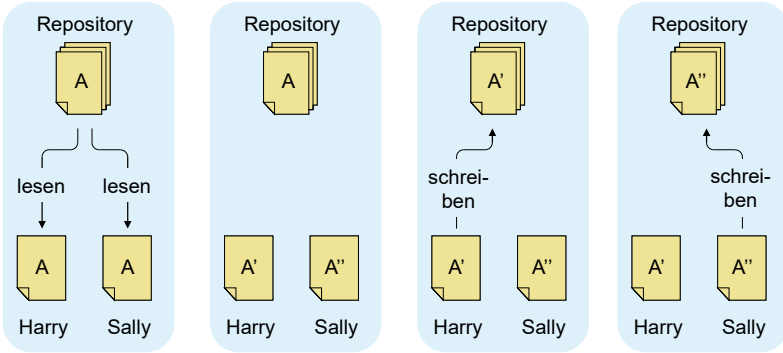
Verteilte Entwicklung



Arbeitskopie
- > Lokale Kopie zur Bearbeitung beim Client
- > Kann Teile oder gesamtes Repository umfassen

Repository
- > Lager für alle Ressourcen eines Projektes (inklusive Historie)
- > Oft an zentralem Ort realisiert → Repository Server

Wozu eine Versionskontrolle?



Lesen

> Harry und Sally erzeugen eigene Kopien

Editieren

> Beide arbeiten gleichzeitig auf ihren Kopien

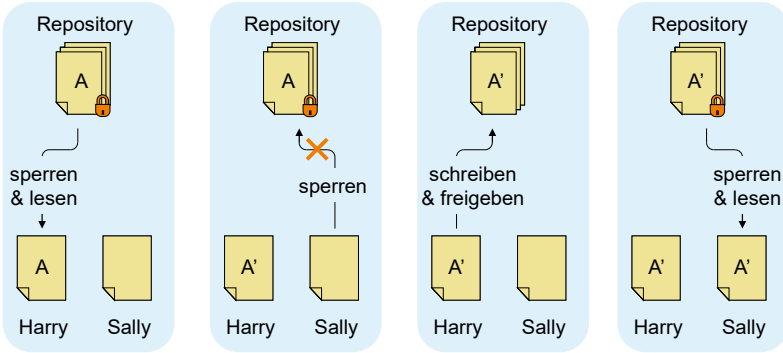
Schreiben

> Zurückschreiben von Harrys Änderungen

Überschreiben

> Verlust von Harrys Änderungen

Lösung: Sperren → Ändern → Freigeben



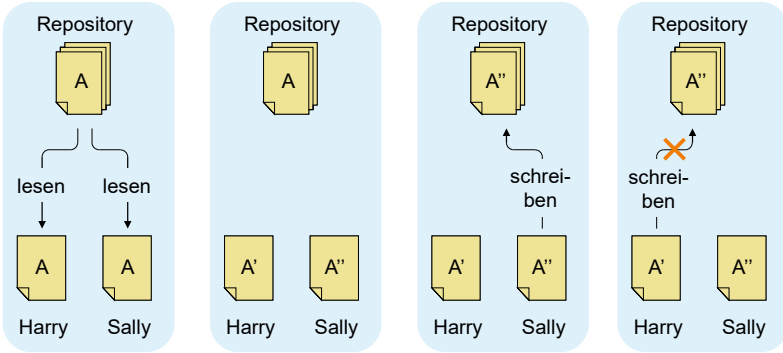
Sperren
> Harry sperrt Dokument zur Bearbeitung

Ändern / Warten
> Sallys Sperrversuch scheitert
> Warten auf Freigabe

Freigeben
> Harrys Freigabe erfolgt nach Änderung

Sperren
> Sally sperrt Dokument zur Bearbeitung

Lösung: Kopieren → Ändern → …



Kopieren

> Harry und Sally erzeugen eigene Kopien

Editieren

> Beide arbeiten gleichzeitig auf ihren Kopien

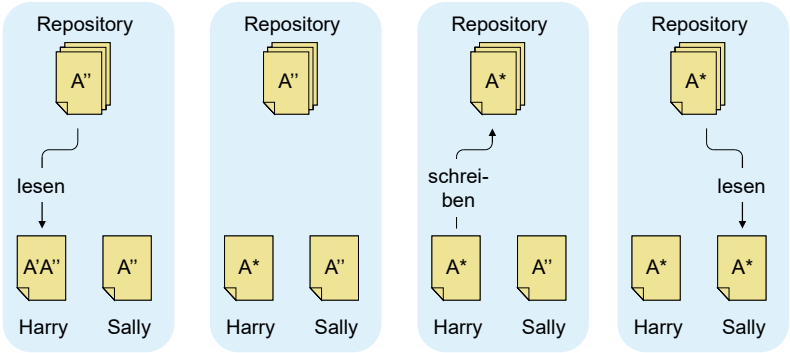
Schreiben

> Sally schreibt Änderungen zuerst zurück

Schreiben

> Harrys Schreibversuch schlägt fehl (da nicht mehr aktuell)

… → Zusammenführen



Update
> Harry vergleicht seine Version mit aktueller

Zusammenführen
> Erstellung einer gemeinsamen Version
> Konfliktpotential

Schreiben
> Harry schreibt zusammengeführte Version zurück

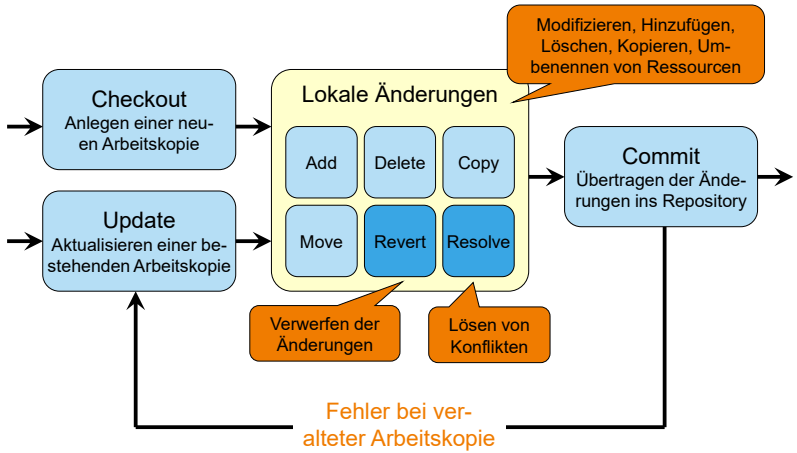
Lesen
> Sally erhält neue Version vom Repository

Subversion

- Versionsmanagementsystem
  - Zentrales Repository für Ordner und Dateien
  - Versionsmanagement → Revisionen des gesamten Repositorys
  - Historie (aller Revisionen) einer Ressource verfügbar
    → Speicherung des Originals und sämtlicher Änderungen
  - Unterstützung von Kopieren/Ändern/Zusammenführen
  - Explizites Sperren von Ressourcen ebenfalls möglich

- Subversion im Web
  - Webseite → http://subversion.tigris.org/
  - Download des Subversion Servers und Kommandozeilen-Clients
  - Dokumentation → http://svnbook.red-bean.com/

Subversion Workflow



Konflikte

- Scheitern der automatischen Zusammenführung von Revisionen beim Aktualisieren → Konflikt
- Zusätzliche Kopien bei Dateien mit Konfliktstatus
  - `DATEI`: Datei mit enthaltenen Konfliktmarkierungen
  - `DATEI.mine`: Eigene Arbeitskopie der Datei vor Aktualisierung
  - `DATEI.rXX`: aktuelle Revision XX aus dem Repository
  - `DATEI.rYY`: Ausgangsrevision YY (<XX) der Arbeitskopie
- Konflikte werden abschnittsweise innerhalb der Datei markiert

```
... Textzeilen der Datei ohne Konflikt
<<<<<<< .mine
Zeilen der Arbeitskopie
=======
Zeilen der aktuellen Repository-Revision
>>>>>>> .r42
Textzeilen der Datei ohne Konflikt ...
```

Auflösen von Konflikten

- Manuelles Lösen des Konfliktes durch Benutzer
  - Editieren der entsprechenden Datei oder
  - Verwerfen eigener Änderungen → Revert oder
  - Ersetzen der Datei durch gewünschte Revision
- Revert
  - Verwirft lokale Änderungen der Ressource
  - Kann keine gelöschten Verzeichnisse wiederherstellen
- Resolve
  - Mitteilung über erfolgte Lösung des Konfliktes
    → Bietet keinerlei semantische Konfliktbehandlung
  - Löscht nur Konfliktstatus der Ressource
  - Löscht zusätzlich angelegte Hilfskopien

Unser Repository

- Playground (zum rumprobieren ohne etwas kaputt machen zu können):
  `https://svn.informatik.uni-rostock.de/lehre/ip2024/playground`
- Gruppenverzeichnis (zur Abgabe der Hausaufgaben):
  `https://svn.informatik.uni-rostock.de/lehre/ip2024/groups/<nr>`
- Öffentliches Verzeichnis (für VL/ÜB-Material):
  `https://svn.informatik.uni-rostock.de/lehre/ip2024/public`

Entwicklungsumgebung@Home

- Windows
  - MinGW64 herunterladen, entpacken und den Pfad des `bin`-Verzeichnisses zur `PATH`-Umgebungsvariable hinzufügen
  - SmartSVN herunterladen, entpacken und den Pfad des Verzeichnisses zur `PATH`-Umgebungsvariable hinzufügen
  - Alternative: Turtoise SVN
  - Visual Studio Code mit Erweiterungen ms-vscode.cpptools, johnstoncode.svn-scm; optional analytic-signal.preview-pdf etc.
- Linux (Ubuntu, andere ähnlich)
  - `sudo apt install build-essential subversion`
  - `sudo snap install -classic code` oder andere Wege
  - VS Code Erweiterungen wie oben

Entwicklungsumgebung@ITMZ

- PC-Pool (R310, R201, R239): Einloggen mit IMTZ-Account “ab1234” und Passwort
- von zu Hause: Einloggen auf unicomp.uni-rostock.de
- Software wie auf voriger Folie vorinstalliert
- Hinweis: Nutzung von “Visual Studio Code (pre installed extensions)”

→ Demonstration am Beamer
Genauere Infos bzw. Textversion s. Stud.IP/Praktikum

Aufgaben

1. Legen Sie im Unterordner `playground` des Hausaufgaben-SVN eine Datei an und comitten Sie diese.
2. Lassen Sie eine/n KommolitonIn diese Datei bearbeiten.
3. Updaten Sie ihre Arbeitskopie und sehen sich die Änderungen an der Datei an. Sehen Sie sich die aufgezeichnete Historie (Log) der Revisionen an.
4. Wiederholen Sie die vorigen Schritte, aber ergzeugen Sie einen Konflikt, indem Sie selber auch Änderungen an der selben Stelle der Datei vornehmen und lösen Sie den Konflikt.