Server und manueller Spieler

Plenum Programmierpraktikum

2006-11-09

Organisatorisches

- Scheine von anoshkin, bartoszek, belskyy, borsch, drickl, dumro, esmarch, gottscha, grytsak, kunzh, landmann, langal, leiyi, nikolovn, perdeckj, spannbau, utzlu, wangn, wangch.

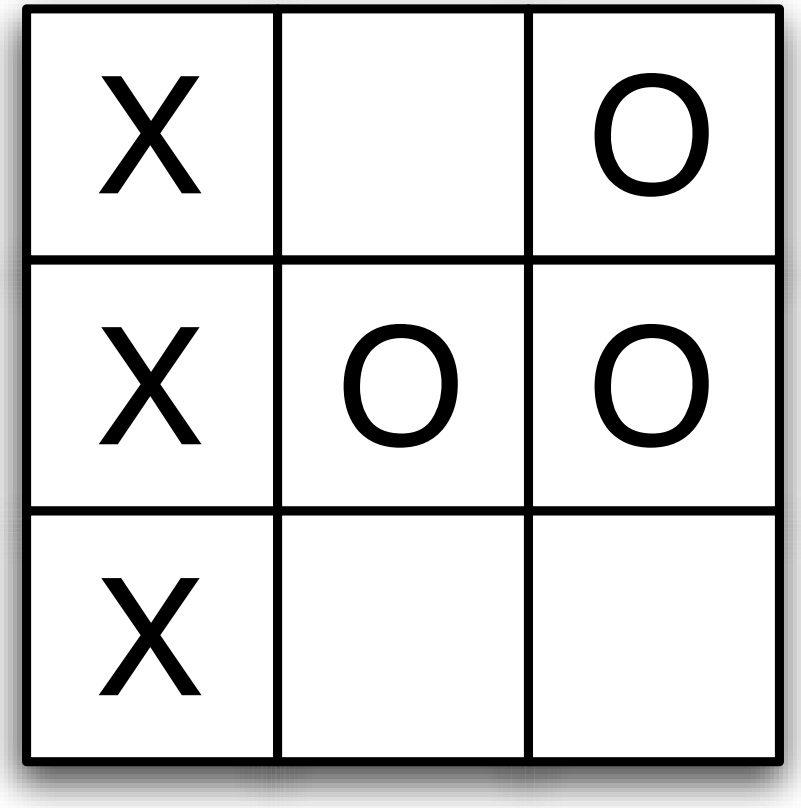
- Nächstes Plenum: Um 9:15 Uhr (eine Stunde später).

- Treffen zur Gruppenbildung: Nächste Woche, direkt im Anschluss an Info III (wahrscheinlich am Montag).

- Wanted: Design-Fragen!

- Folien: Vorabversion gibt es immer Mittwoch nacht.

Das nächste Projekt



User-Stories

1. Tic-Tac-Toe soll verteilt implementiert werden. Es gibt einen Server und mehrere manuelle Spieler (Clients).

2. Schwarz (X) fängt immer das Spiel an.

3. In der Loginphase nimmt der Server Spieler an.

4. In der Spielphase startet der Server einen Thread pro Spiel.

5. Ein manueller Spieler wird über die Konsole bedient.

6. Es wird das standardisierte $PlayerInterface$-Protokoll verwendet.

Protokoll

- Nach der Kontaktaufnahme zum Server ist der Player komplett passiv und wartet auf die Aufrufe vom Server.

- Die möglichen Aufrufe sind in einem Interface definiert.

Protokoll

```
public interface PlayerInterface {
    public String getRules();
    public String getName();
    public void openGame(PieceColor color);
    public void trackMove(Coord coord);
    public Coord computeMove(long timeLimit);
    public void closeGame(PieceColor winner, String message);
}
```

- Aufruf-Beispiel: `closeGame[tab]BLACK[tab]finished`
- Antwort-Beispiele: “OK” (void), “Tictac” (String), “1,2” (Coord).

Zeitplanung

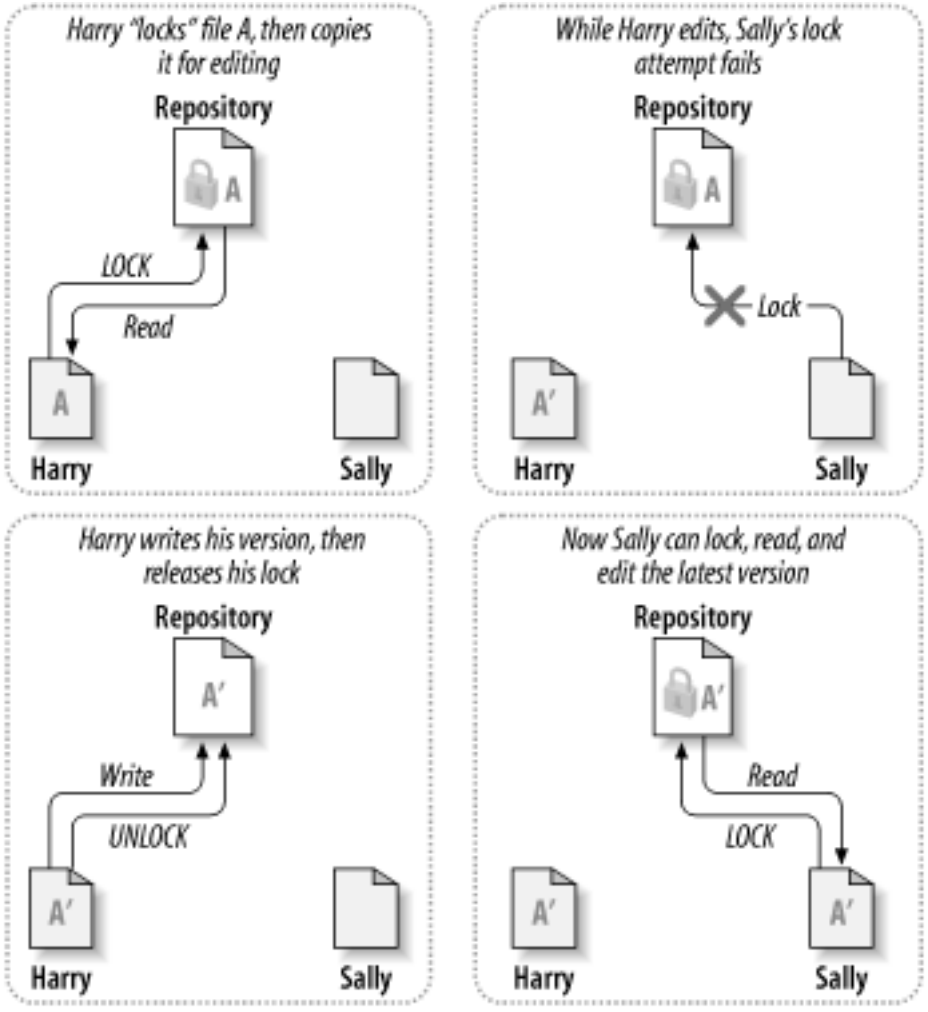
In der Gruppe diskutieren und planen:

- User-Stories zerlegen in $Tasks$.
- Pro Task: „Ideale Zeit“ schätzen (wie lange man benötigt, wenn man nicht gestört wird und keine größeren Nachforschungen erforderlich sind). Diese Zeit als Punkte vergeben.
- Punkte mit Faktor multiplizieren: Berechne ideale Zeit in tatsächliche Zeit um. Als Daumenregel rechnet man anfangs, dass ein Drittel der Zeit aus „Ablenkungen“ besteht. Das ist eine gute Vorhersage, wie viel der Arbeit man pro Woche schafft.
- Danach messen: Wie lange haben wir wirklich gebraucht? Damit kann man mit der Zeit den Faktor genauer bestimmen.

Versionskontrolle

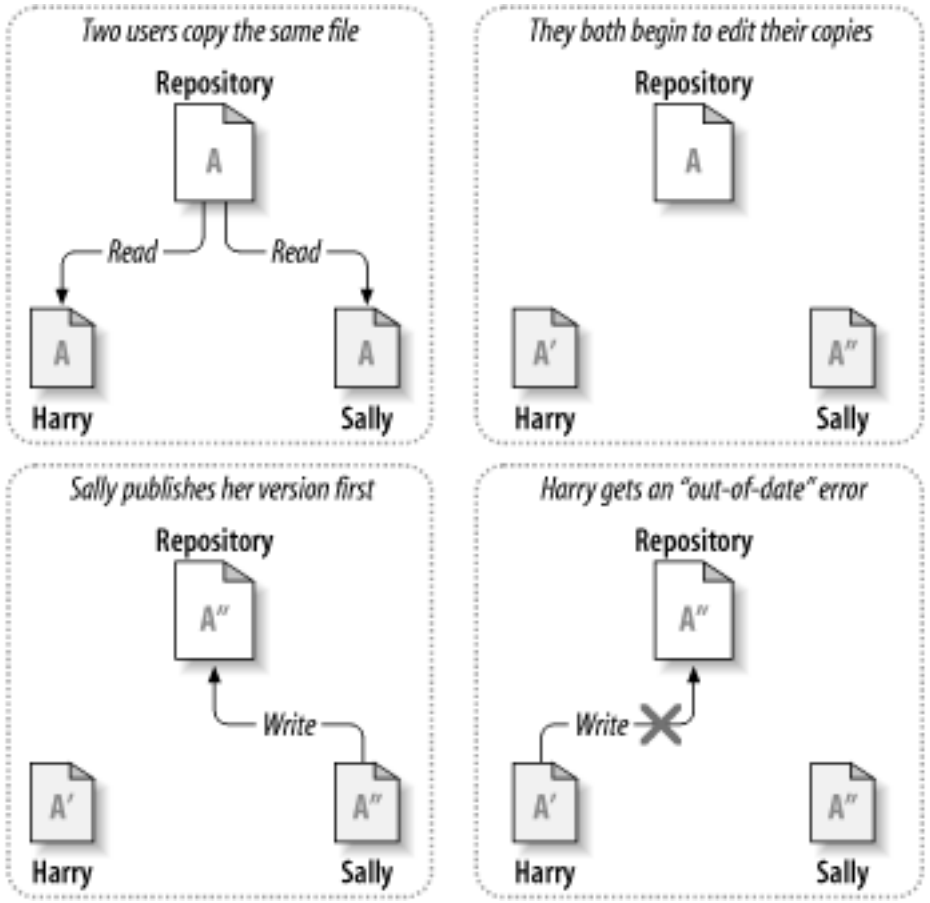
- Problem: Mehrere Leute sollten am selben Quellcode arbeiten.
- Lösung: Versionskontrolle
- Weitere Vorteile (auch wenn man alleine arbeitet!)
  - Unlimited Undo
  - Backup
  - Verteilung der Daten

Lock-Modify-Unlock

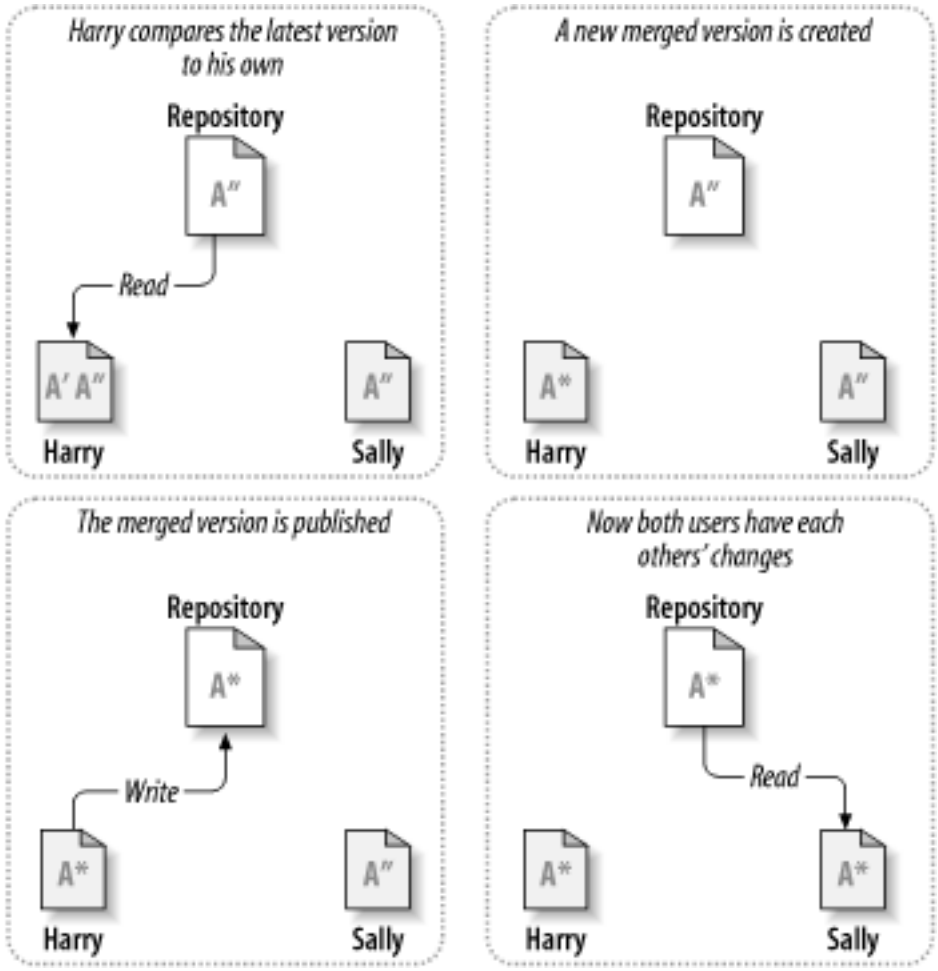


Quelle: „Version Control with Subversion“

Copy-Modify-Merge (1/2)



Copy-Modify-Merge (2/2)



Subversion (SVN)

- Standard, Open Source, Nachfolger von CVS.
- Eigener Server nötig (gibt es im CIP)
- Merge meist automatisch, überlappende Konflikte werden angezeigt und können manuell behoben werden.
- Clients: Web-basiert, Kommandozeile, grafisch oder Eclipse-Plugin.

Subversion (SVN)

Arbeitsweise:

- Working-Copy auschecken.
- Working-Copy ändern: Dateien editieren, löschen, umbenennen. Neue Dateien zur Working-Copy hinzufügen.
- Commit: Änderungen in der Working-Copy ins Repository stellen.
- Zwischendurch: Per Update Änderungen von anderen aus dem Repository in die Working-Copy übernehmen.

Kommandozeilen-Befehle

Aufruf: `svn <kommando>`, Hilfe: `svn help`

- **checkout:** Erzeuge eine *Working-Copy* des Repositories.
- **update:** Hole Änderungen vom Repository.
- **add:** Stelle Dateien und/oder Verzeichnisse ins Repository.
- **commit:** Schicke Änderungen zum Repository.

Befehle

- **tag:** Versehe den momentanen Zustand im Repository mit einer Markierung. Zu diesem markierten Zustand kann man später jederzeit zurückkehren.
- **status:** Was hat sich geändert? Was wurde gelöscht? Was ist noch nicht im Repository?
- **revert:** Kehre zur ursprünglichen Working-Copy zurück.
- **diff:** Vergleiche aktuellen Stand mit der ursprünglichen Working-Copy.

SVN-Eclipse-Plugin: Subclipse

- Tutoren richtet im CIP-Pool ein Repository pro Gruppe ein, das durch einen URL identifiziert wird.
- Subclipse installieren: "Help ▶ Software Updates ▶ Find and Install. . . "
- Restliche Befehle im Kontextmenü:
  - Team ▶ (meistens: "Synchronize with Repository")
  - Compare With ▶
  - Replace With ▶
- Zusätzlich: SVN-Perspektive für weitergehendes Management.

Aufgaben

- Es werden keine Lösungen mehr besprochen. Die Abgabe gegenüber dem Tutor wird wichtiger.
- „Aktuelles“ auf der PP-Homepage beobachten.
- Der Tutor ist Kunde, wenn Zweifel auftauchen der Form „wie genau sollen wir das machen?“.
  $\Rightarrow$ Klarstellungen zu jeder User-Story protokollieren.
- User-Stories können im Forum diskutiert werden.
- Nächstes Plenum: Techniken zur Teamarbeit und Implementierung.

Aufgabe: Gruppenarbeit

- Gruppengröße ist genau 6 Leute.
- Gruppe registrieren: E-Mail mit Logins (einer pro Zeile, im Anschluss zusätzlich die vollen Namen) an A.R.
- Jede Gruppe bekommt einen Tutor zugeteilt
  $\Rightarrow$ Zeitpunkt für wöchentliches Treffen ausmachen.
- Tutor schickt SVN-URL
  $\Rightarrow$ Ein Eclipse-Projekt anlegen, ins SVN stellen.
- Zeitschätzung
- Abgabe: nächste Woche, beim ersten Tutorentreffen.

Aufgabe: Implementierung

- User-Stories erfüllen: Server und manueller Spieler mit Textoberfläche.
- Zusätzlich: Unit-Tests.
- Abgabe: In drei Wochen (KW 48).

Referenzen

- Versionskontrolle
  - http://www.johnnysthoughts.com/subversion-quick-reference/
  - „Version Control with Subversion“, Collins-Sussman, Fitzpatrick, Pilato. http://svnbook.red-bean.com/
  - Subversion im CIP: http://www.rz.ifi.lmu.de/Dienste/Subversion
  - Eclipse-Plugin „Subclipse“: http://subclipse.tigris.org/

- Ausblick: Es gibt noch andere Systeme zur Versionskontrolle, mit sehr interessanten Features. http://www-128.ibm.com/developerworks/linux/library/l-vercon/