„Gesamtsystem in der Cloud“ – Neue Möglichkeiten für Kollaboration und Outsourcing

Johannes Kreutz

Der vorliegende Beitrag behandelt den Aspekt der simultanen Bearbeitung von Rechenmodellen zur Strukturanalyse durch mehrere Beteiligte. Mit den Möglichkeiten von Versionskontrollsystemen, einer Technik aus der Softwareentwicklung, ergeben sich insbesondere für Modelle auf der Basis von Textdateien Vorteile wie Sicherheit und Nachvollziehbarkeit der Änderungen und dadurch neue Möglichkeiten für Kollaboration und Outsourcing. Dies führt zu einer größeren Effizienz und Flexibilität im Zusammenhang mit der Erstellung und Pflege von Rechenmodellen. Im Beitrag werden Versionskontrollsysteme in ihrer Funktionsweise erläutert und ihre Vorteile und Einschränkungen in der Anwendung für die Eingabe von strukturellen Rechenmodellen aufgezeigt.

This work covers the aspect of simultaneous editing of calculation models for structural analysis by numerous coworkers. Version control systems, a technique developed for software development enable for big advantages like security and reproducibility of changes and by that allow for new possibilities in collaboration and outsourcing. This leads to a greater efficiency and flexibility concerning the preparation and maintenance of structural models. This contribution explains version control systems with respect to their mode of operation and outlines the emerging advantages and restrictions when they are used for the input of structural models.

1 AUSGANGSSITUATION

Die immer weiter steigende Verfügbarkeit von Rechenleistung und der stetige Fortschritt in der Bedienbarkeit von Finite Elemente Programmen hinsichtlich Visualisierung, Bedienbarkeit, Ein- und Ausgabe haben dazu geführt, dass räumliche Gesamtmodelle immer öfter das Mittel der Wahl sind, wenn es um die Berechnung von komplexeren Tragwerken geht. Die Erstellung der Gesamtmodelle ist dabei ein arbeitsintensiver und komplexer Prozess. Diese Tatsache gilt unabhängig davon, welche Art der Eingabe (grafisch oder textbasiert), oder welches Finite-

Elemente-Paket verwendet wird. Trotz dieser Komplexität ist es in Ingenieurbüros üblich, dass der Projektingenieur selbst die Modelle erstellt und sich in die entsprechenden Software-Pakete einarbeitet. In anderen Branchen wie z.B. dem Maschinenbau ist die Arbeitsteilung und die Fremdvergabe von Modellierungs- und sogar Berechnungsaufgaben generell weiter fortgeschritten. Es gibt zum Teil sehr ausgefeilte Ansätze, Kollaboration an einem 3D-CAD-Modell innerhalb einer Architekten- oder Konstrukteurs-Arbeitsgruppe über das lokale Netzwerk oder das Internet umzusetzen. Beispielhaft seien hier die CAD-Produkte *ARCHICAD BIM-Server*, *AutoDesk Revit Server* und *Allplan Workgroup* genannt. Sie arbeiten nach dem Lock-Modify-Unlock-Prinzip (Erläuterung siehe Abschnitt 2.2) auf Objektebene. Dem Verfasser ist jedoch kein Produkt bekannt, das speziell die kollaborative Erstellung von Modellen für die Strukturanalyse für Zwecke des Bauingenieurwesens ermöglicht.

Ein zentrales, serverbasiertes 3D-Modell (Datenbank), auf das alle Beteiligte, (Architekten, Tragwerks- und andere Fachplaner) lesend und schreibend zugreifen, wird wohl auf längere Sicht nicht verfügbar sein und ist vom Denkansatz auch umstritten. Die *OpenBIM*- bzw. *buildingSMART Initiative* [1] geht z.B. einen anderen Weg und setzt auf das sogenannte Referenzmodell, das vom Architekten erstellt und über möglichst gute Schnittstellen (IFC-Klassen) in Softwarelösungen anderer Gewerke wie z.B. des Tragwerksplaners im- und exportiert werden kann. Auch wenn die Verbreitung zunimmt, ist mit einer flächendeckenden Umsetzung solcher Arbeitsweisen in Deutschland in der näheren Zukunft noch nicht zu rechnen [2].

Vorgenannte Punkte verdeutlichen, dass die Eingabe von Rechenmodellen auf Basis von Textdateien nach wie vor verbreitet und auch sinnvoll und effizient ist, sofern die verwendete Software wie z.B. *ANSYS* mit *APDL* oder *SOFiSTiK* mit *CADINP* eine funktionsreiche und gut dokumentierte Eingabesprache anbietet. Weitere Vorteile hat die textbasierte Eingabe durch die effiziente Abbildung von wiederkehrenden Strukturen und die Möglichkeit, „Makros“ zu erstellen. Der vorliegende Artikel soll beleuchten, dass die Eingabe über eine Textdatei noch einen Vorteil mit sich bringt. Zeilenorientierte Textdateien eignen sich besonders gut für eine sogenannte Versionsverwaltung (engl. version control system, VCS). Dies ermöglicht die simultane Bearbeitung durch mehrere Personen bei gleichzeitiger optimaler Kontrolle und soll im folgenden Abschnitt erläutert werden.

2 VERSIONSVERWALTUNGSSYSTEME

2.1 Grundlagen und Aufgabenbeschreibung

Ein VCS ist ein System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird. Alle Versionen werden in einem Archiv (engl. Repository) mit Zeitstempel und Benutzerkennung gesichert und können später wiederhergestellt werden.

Versionsverwaltungssysteme werden typischerweise in der Softwareentwicklung eingesetzt, um Quelltexte zu verwalten. Versionsverwaltung kommt auch bei Büroanwendungen oder Content-Management-Systemen zum Einsatz [3].

Die Aufgaben des Systems sind neben der Protokollierung aller Änderungen und der Wiederherstellung von alten Ständen die Koordinierung bzw. Ermöglichung des gemeinsamen Zugriffs von mehreren Beteiligten auf die gleiche Datei bzw. Dateien. Auch die simultane Bearbeitung mehrerer „Entwicklungszweige“ (engl. Branches) ist möglich. Die Programme sind ursprünglich als Kommandozeilenprogramme aufgesetzt, für die verbreiteten Open Source VCS-Pakete wie z.B. GIT [4] und Subversion (SVN) [5] sind aber auch komfortable Integrationen in das Windows Betriebssystem als Open Source Programme verfügbar [6] [7]. Der Status der jeweiligen Datei (unverändert, verändert, neu hinzugefügt, oder in Konflikt) wird hierbei direkt mit Symbolen am Datei-Icon visualisiert (s. Abbildung 1) und Aktionen sind direkt über das Kontextmenü verfügbar.

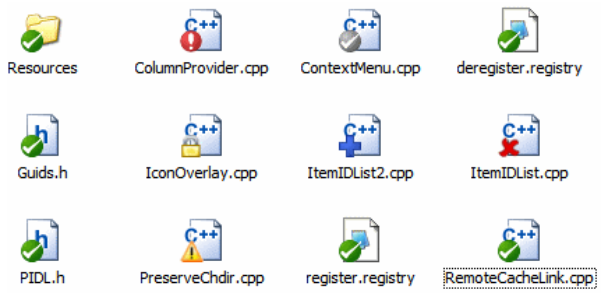

Abbildung 1: Tortoise: Icon-Overlays informieren über den aktuellen Status jeder Datei [6].

2.2 Funktionsprinzip im Detail

Prinzipiell funktioniert eine Versionshaltung für alle Arten von Dateien. Das Prinzip ist, dass es ein (zentrales) Archiv (Repository) gibt. Von diesem wird bei jedem Projektbeteiligten lokal eine Arbeitskopie der Daten anlegt, welche dann mit beliebigen Programmen bearbeitet werden kann. Nach der Bearbeitung werden die Daten wieder an das Repository übertragen und damit den anderen Beteiligten zugänglich gemacht. Was bei gleichzeitiger Bearbeitung einer Datei dann ohne Versionsverwaltung passieren würde, ist in Abbildung 2 zu sehen: Die zuletzt übertragene Änderung überschreibt alle vorherigen.

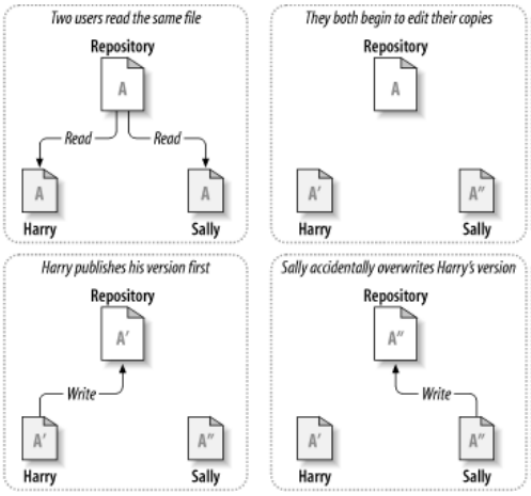

Abbildung 2: Kollaboration ohne Versionsverwaltung [5].

Wird hingegen eine Versionsverwaltung verwendet, so sind zunächst zwei Prinzipien zu unterscheiden:

- Lock Modify Unlock: Einzelne Dateien müssen vor einer Änderung durch den Benutzer gesperrt und nach Abschluss der Änderung wieder freigegeben werden. Die zugrunde liegende Philosophie wird pessimistische Versionsverwaltung genannt [3], eine beispielhafte Prinzipskizze ist in Abbildung 3 zu finden. Bei diesem Prinzip gibt es keine Einschränkung hinsichtlich des Dateityps, es ist aber anzumerken, dass eine gleichzeitige Bearbeitung derselben Datei nicht möglich ist (genau das wird ja durch das „Lock“ verhindert). Es ist auch anzumerken, dass es Systeme gibt, die die Locks nicht auf Datei-, sondern auf Objektebene setzen. z.B. ArchiCAD BIM Server oder auch die Tabellenkalkulation von Google Drive, die einzelne Zellen sperrt.).
- Copy Modify Merge: Ein solches System lässt gleichzeitige Änderungen durch mehrere Benutzer an einer Datei zu. Anschließend werden diese Änderungen automatisch oder manuell zusammengeführt (engl. merge). Problematisch bei diesem System sind Binär- (nicht-text-) Dateien1, da diese, sofern kein passendes Werkzeug verfügbar ist2, nicht automatisch zusammengeführt werden können. Die zugrunde liegende Philosophie wird als optimistische Versionsverwaltung bezeichnet [3]. Eine beispielhafte Prinzipskizze ist in Abbildung 4 zu finden. Die Zusammenführung funktioniert in vielen Fällen vollautomatisch, manchmal (bei einer Änderung in der gleichen Zeile einer Datei) ist eine manuelle Bearbeitung jedoch unausweichlich (siehe Abbildung 6).

[1] Die meisten gängigen Dateiformate sind binär: .doc(x), .xls(x), .dwg, und .pdf ebenso wie sämtliche Bilddateiformate.
[2] In der Regel ist kein Werkzeug verfügbar, das zwei geänderte Dateien zusammenführen kann. *Microsoft Word* stellt solch eine Funktion aber z.B. für seine Dateiformate zur Verfügung und setzt dabei die bekannten „Änderungen verfolgen“-Funktionalitäten ein.

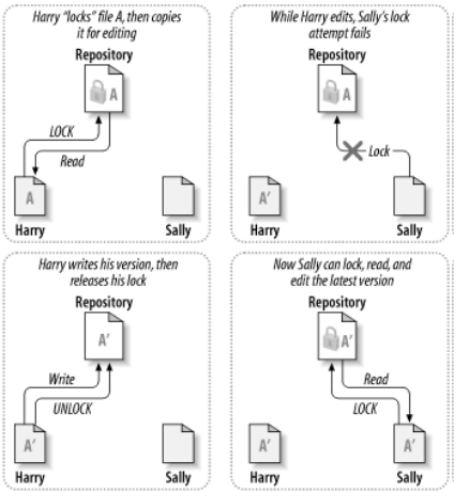

Abbildung 3: Schematisches Prinzip Lock Modify Unlock [5]

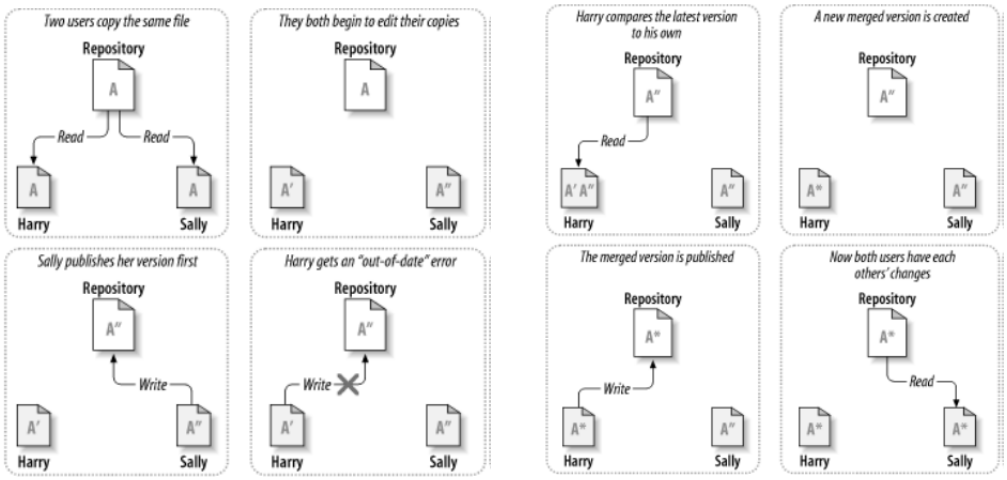

Abbildung 4: Schematisches Prinzip Copy Modify Merge [5]

Letztere Variante ist ungleich mächtiger. Sie ermöglicht zum einen eine gleichzeitige Bearbeitung durch mehrere Beteiligte und zum anderen eine „inkrementelle“ Übertragung und Speicherung der Daten, also nur in Form von Änderungen. Dadurch ist jeder Zwischenstand verfügbar und die einzelnen Änderungsschritte (im Quellcode, s. Abbildung 5) bleiben nachträglich immer nachvollziehbar. Die Menge der zu übertragenden Daten ist im Normalfall sehr klein. Im Folgenden soll nur noch das Copy Modify Merge Prinzip behandelt werden.

VCS sind weiterhin zu unterscheiden in zentralisierte (z.B. SVN) und verteilte Systeme (z.B. GIT). Auf diesen Unterschied soll hier aber nicht weiter eingegangen werden, der interessierte Leser sei auf [4] verwiesen.

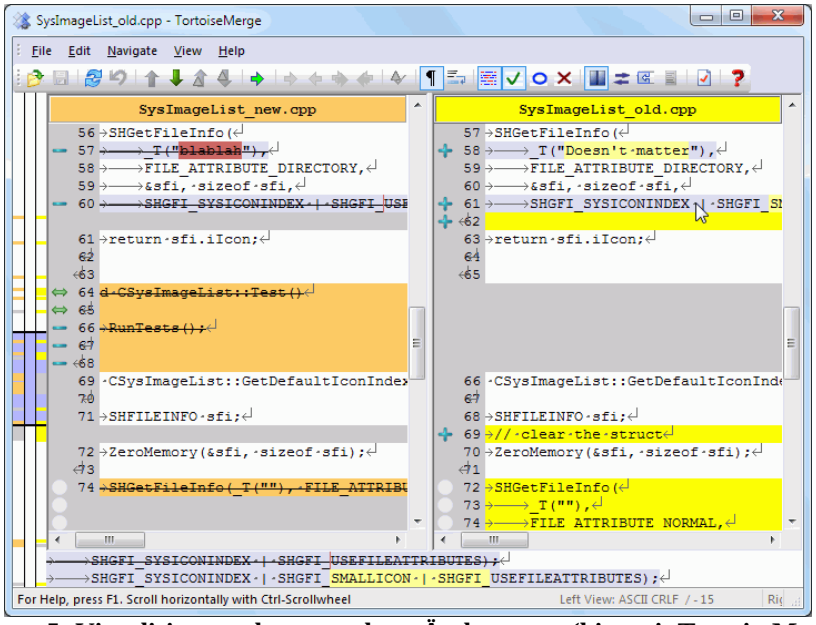

Abbildung 5: Visualisierung der gemachten Änderungen (hier mit TortoiseMerge [6]).

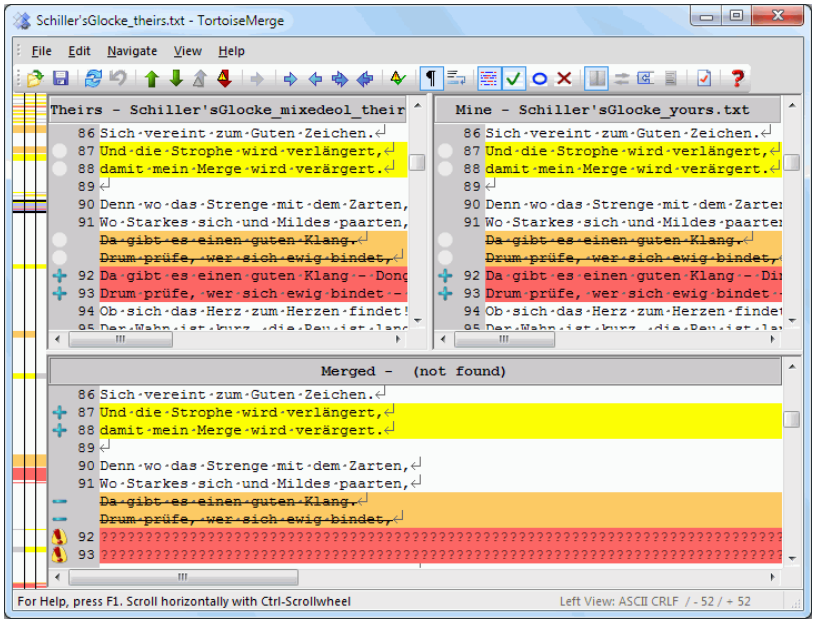

Abbildung 6: Manuelles Bearbeiten von Konflikten (hier mit TortoiseMerge [6]).

2.3 Anbindung an Projektsteuerungssoftware (Ticketsystem / Tracker)

Die oben erwähnten Systeme GIT und SVN können zudem an webbasierte Projektsteuerungssysteme, wie z.B. Redmine [8], angebunden werden, was die Anforderungen an eine Projektarbeitsumgebung vervollständigt (s. Abbildung 7). So können beispielsweise Aufgaben, Meilensteine und eine automatische, aufgabenbezogene Buchführung der geleisteten Arbeitsstunden aller Projektbeteiligten in ein und demselben System gemeinsam mit dem statischen Modell verwaltet werden. Eine Webbasierte Architektur ermöglicht auch den Zugriff von Smartphone-Apps z.B. zum Zweck der Projektbearbeitung und zum Buchen von Stunden.

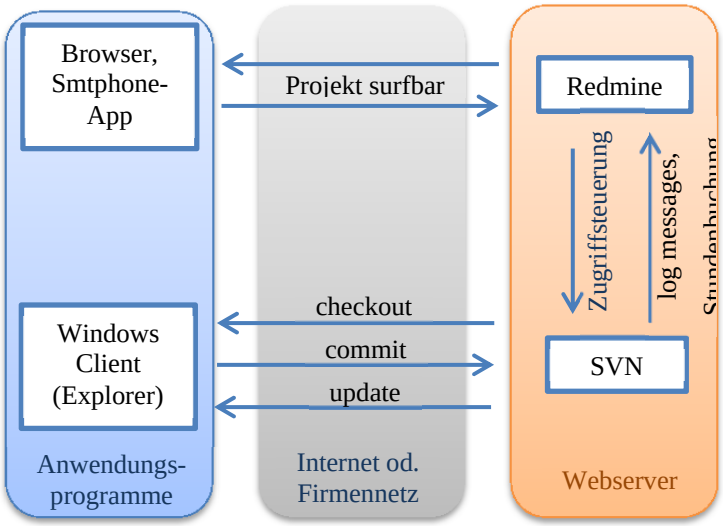

Abbildung 7: Beispielhafte Konfiguration von SVN/Redmine

Die Serverseite (rechter Bereich in Abbildung 7) kann sowohl innerhalb des Firmennetzwerks als auch auf einem Server im Internet aufgesetzt werden. Im Internet werden auch fertig gehostete Lösungen angeboten, z.B. Plan.io [9].

3 FAZIT: NEUE MÖGLICHKEITEN

Die Verwendung der zuvor erläuterten Technik ermöglicht es ganz prinzipiell, dass mehrere Personen an einem Modell arbeiten. Damit es möglichst nicht zu Konflikten kommt, müssen innerhalb des Modells die Aufgaben- bzw. Arbeitsbereiche einigermaßen klar zwischen den Beteiligten aufgeteilt und koordiniert werden. Eine „aufgeräumte Struktur“ der Eingabesätze ist unerlässlich.

Die Verwendung von VCS erlaubt einen höheren Grad an Flexibilität und Spezialisierung bei gleichzeitiger Kontrolle. Durch die Tatsache, dass alle Änderungen vorgehalten werden, ist es keinem Beteiligten möglich, das System z.B. durch Fehlbedienung endgültig zu beschädigen. Die entstandene Flexibilität kann auch dazu genutzt werden, z.B. die Modellierungsleistung im Sinne der Arbeitsteilung bzw. Spezialisierung an externe Mitarbeiter, Mitarbeiter in Heimarbeit oder externe Dienstleister zu vergeben, wodurch sich sowohl bei kleineren wie auch bei größeren Modellen Effizienzsteigerungen erreichen lassen. Durch die Verwendung einer webbasierten Projektsteuerungssoftware (s. Abschnitt 2.3) haben alle Beteiligten auch im Fall einer Fremdvergabe der Dienstleistung jederzeit die Kontrolle über das Modell und einen guten Überblick über den Fortschritt des Modells und die nächsten Arbeitsschritte.

4 LITERATUR

[1] http://www.buildingsmart.org/openbim (Abgerufen: 17. November 2013, 15:20 UTC)

[2] *von Both, P., Koch, V., Kindsvater, A* Abschlussbericht BIM - Potentiale, Hemmnisse und Handlungsplan Fachgebiet Building Lifecycle Management am Karlsruher Institut für Technologie, 2012

[3] Seite „Versionsverwaltung“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 29. Oktober 2013, 09:48 UTC. URL: http://de.wikipedia.org/w/index.php?title=Versionsverwaltung&oldid=123924642 (Abgerufen: 13. November 2013, 21:19 UTC)

[4] *Chacon, S.;* Pro Git, Apress, New York, 2009

[5] *Pilato, C. Michael, Collins-Sussman, Ben und Brian W. Fitzpatrick;* Version Control with Subversion, O’Reilly, Sebastopol, 2004
(creative commons Lizenz 2.0 http://creativecommons.org/licenses/by/2.0/)
Online als pdf verfügbar unter http://svnbook.red-bean.com/de/1.6/svn-book.pdf

[6] http://tortoisesvn.net/ (Abgerufen: 14. November 2013, 19:10 UTC)

[7] http://code.google.com/p/tortoisegit/ (Abgerufen: 14. November 2013, 19:11 UTC)

[8] *Lang, Jean-Philippe;* Redmine Projekthomepage http://www.redmine.org (Abgerufen: 23. August 2013, 10:10 UTC)

[9] http://plan.io; (Abgerufen: 14. November 2013, 19:39 UTC)