Reengineering-Projekte in der Ausbildung

Klaus Bothe
Institut für Informatik
Humboldt-Universität zu Berlin
bothe@informatik.hu-berlin.de

1 Überblick

Vorlesungen zum Thema Software-Reengineering bzw. Wartung haben mit einem negativen Image zu kämpfen: schlecht strukturierte Altsysteme, oftmals in unmodernen Programmiersprachen entwickelt, sind Gegenstand der Untersuchungen [4].

Eine andere Situation kann sich bei Reengineering- bzw. Wartungsprojekten ergeben: hohe Motivation, verbunden mit der Ausprägung von Fähigkeiten zur Teamarbeit [6, 7, 9, 11].

In unserem Beitrag soll ein mittlerweile seit fünf Semestern laufendes Projekt [3] vorgestellt und einige Aspekte vertiefend diskutiert werden.

2 Motivation durch reale Projekte

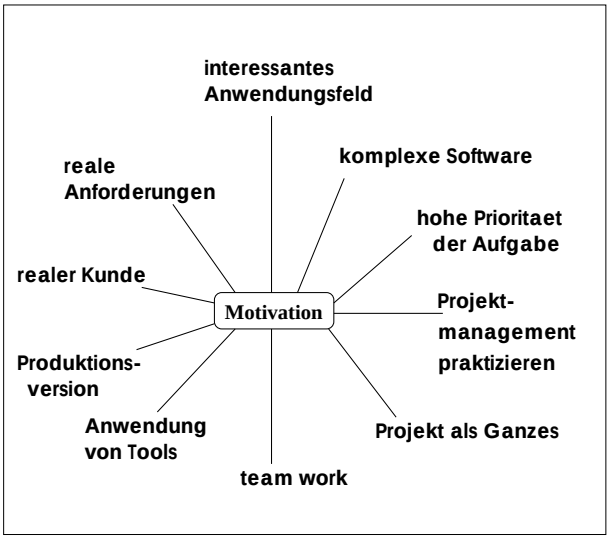

Abbildung 1: Aspekte der Motivation durch ein reales Reengineering-Projekt

Seit Ende 1998 wird ein in der Experimentalphysik eingesetztes umfangreicheres Softwaresystem in einem studentischen Projekt bearbeitet. Dabei stehen sowohl reine Reverse Engineering-Aktivitäten (Entwicklung von Beschreibungen auf höherer Abstraktionsebene) als auch weiterführende Reengineering-Aufgaben (Restrukturierung, Systemerweiterung) im Mittelpunkt. Motivationsprobleme gibt es kaum - eine Reihe motivierender Faktoren (Abb. 1) führt im

Gegenteil dazu, daß die studentischen Teilnehmer einen überdurchschnittlich hohen Einsatz zeigen.

3 Reengineering-Aktivitäten: Projektaufgaben

Quellfiles im Umfang von 27 000 LOC lagen als einzige Softwaredokumente vor. In einer Kombination von Top-down- und Button-up-Programmverstehen wurde die Funktionalität und Struktur des Systems erschlossen (Abb. 2). Ein großes Problem - in einem studentischen Projekt wie auch in Softwarefirmen - war die fehlende Dekomposition in Komponenten mit klaren Schnittstellen, um unabhängige Teamarbeit zu ermöglichen. Nicht konfliktfrei war die Suche nach den aktuellen Programmquellen, da ein Versionsmanagement bisher nicht vorlag.

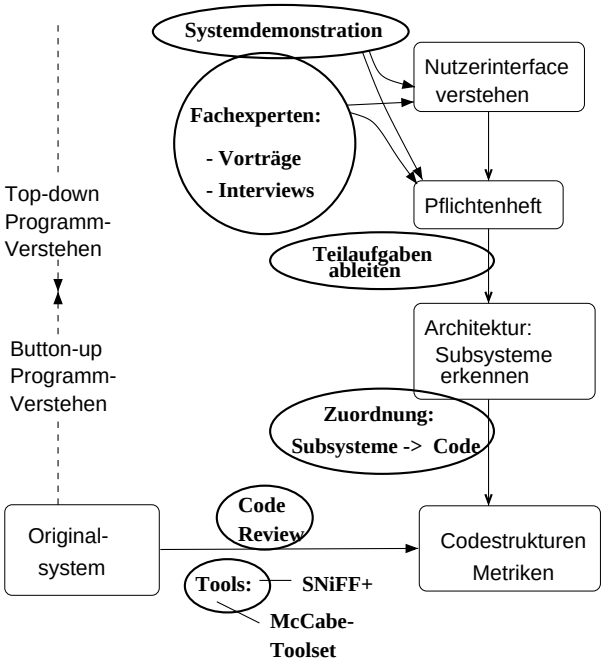

Abbildung 2: Projekt-Aktivitäten

4 Probleme mit Reengineering-Tools

Eine Vielzahl von SW-Werkzeugen wird als Reengineering-Tools angeboten [2, 1, 5, 8, 10], wobei der Einzugsbereich im einzelnen höchst unterschiedlich ist. So gibt es Tools zur Strukturanalyse von C++-Programmen [8], zur Ermittlung von Metriken [5], zur Wiedergewinnung objektorientierter Architekturen auf der Grundlage eines objektorientierten Quellcodes [10] sowie zur Ableitung von Architekturinformationen aus beliebigen (auch imperativen) Programmstrukturen [2, 1].

Kaum eines dieser Tools war (zumindest problemlos) auf unser C++-System (Grundlage: Borland C++-Compiler) anwendbar. Die Gründe waren vielfältig: Das Bauhaus-System [1] akzeptiert nur C-Programme, URCA [2] und die McCabe-Tools [5] akzeptieren zwar C++, jedoch nicht in der Borland-Version. Dasselbe Problem traf auch auf weitere Tools (u. a. zum Test) zu.

Mit Erfolg konnte dagegen SNiFF angewandt werden, um Querbezüge zwischen Bezeichnerdeklaration und -nutzung zu gewinnen.

5 Weitere Pläne

Folgende Aufgaben stehen aus

- Entwurf einer ordentlichen SW-Architektur: Restrukturierung und Wrapping; besonders wichtig scheinen hier die Architekturkomponenten zu sein
- Erweiterungen: nach Wünschen der Nutzer (Physiker)
- Sicherheit: unkontrollierte Systemabstürze sind zu beheben.
- Test: Aufbau eines Testsystems unter Nutzung von Umgebungssimulatoren
- Portierung: von Borland C++ nach Visual C++

Literatur

[1] Bauhaus, Universität Stuttgart

[2] D. Bojic: URCA-Tool, 2000

[3] K. Bothe: Reverse Engineering: the Challenge of Large-Scale Real-World Educational Project, Conference on Software Engineering Education and Training, Charlotte, USA, Febr. 2001

[4] R. Koschke: Vorlesungen zum Thema Software-Reengineering, 2. Workshop Software Reenginnering, Bad Honnef, Mai 2000

[5] McCabe Toolset, 1995

[6] Pierce: Teaching Software Engineering Principles using Maintenance-Based Projects, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997

[7] K. Sikkel, T. Spil, R. van de Weg: Replacing a Hospital Information System: an Example of a Real-World Case Study, 12th Conference on Software Engineering Education and Training, New Orleans 1999

[8] SNiFF+ for C++

[9] J. Slimnick: An Undergraduate Course in Software Maintenance and Enhancement, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997

[10] Together C++

[11] C. Wohlin: Meeting the Challange of Large-Scale Software Development in an Educational Environment, 10th Conference on Software Engineering Education and Training, Virginia Beach, 1997