5th International Workshop on Software Configuration Management

- Reisebericht -

Hans-Ulrich Kobialka
GMD/SET
53754 Sankt Augustin
email: kobialka@gmd.de

Dieser Workshop fand am 23 + 24. April 1995 mit einigen anderen im Vorfeld der International Conference on Software Engineering (ICSE) in Seattle statt. Er hat eine lange Tradition, die bis in das Jahr 1988 zurückreicht. Ein Kennzeichen dieses Workshops ist die sehr starke Industriebeteiligung nicht nur bei den eingeladenen Teilnehmern (40 in diesem Jahr) sondern auch bei den Vortragenden. Dies liegt daran, daß kommerzielle Configuration Management (CM) Systeme vorgestellt werden, Anwender von ihren Erfahrungen berichten und wissenschaftliche Arbeiten meist von Forschungsgruppen kommen, die eigene CM-Systeme gebaut haben und diese teilweise in Projekten, public domain oder sogar als kommerzielles Produkt im Einsatz haben. Das Wichtigste auf diesem Workshop sind natürlich die Diskussionen; ich möchte daher meine inhaltlichen Bemerkungen im folgenden an einigen Diskussionsthemen ausrichten.

Version Selection

Einige CM-Systeme (u.a. ClearCase und Continuus) bieten zum Teil sehr mächtige Mechanismen um den „Workspace“ eines Entwicklers, d.h. die Menge der Versionen (Stücklisten), die gerade für seine Arbeiten relevant sind, mittels Auswahlregeln zu berechnen. Bill Schweitzer (Tandem) behauptete, daß dies zu kompliziert ist und ein einfacheres Benutzungsmodell ausreicht und für die Entwickler leichter handhabbar ist. Bei Tandem hat jede Entwicklungsaktivität als „Workspace“ eine (virtuelle) Kopie der gerade aktuellen Systemkonfiguration (Synonyme: Bill of material, Baseline). In diesem „Workspace“ können isoliert von anderen Aktivitäten Änderungen vorgenommen, übersetzt und getestet werden. Am Ende aber auch während der Entwicklungsaktivität

- können geänderte Versionen in der Baseline (bzw. einem anderen „Workspace“) importiert werden um Konflikterkennung, -behebung (Merge) und Test lokal durchzuführen,
- eigene Änderungen als neue Versionen in die Baseline integriert werden.

Diese Vorgehensweise ist auch bei anderen Unternehmen anzutreffen und ist natürlich auch bei regelbasierter Versions-Selektion möglich [Die Behauptung, daß Anwender Selektions-Regeln eigentlich fast nur so einsetzen, blieb von dem ClearCase-Vertreter aber auch anderen Teilnehmern unwidersprochen]. Allerdings sind Selektions-Regeln für diese Verwendung a) zu mächtig/fehleranfällig in der Benutzung und b) modellieren nicht ausreichend die Existenz von mehreren gleichzeitigen Entwicklungsaktivitäten (z.B. zur Beantwortung der Fragen „Welche Entwicklungsaktivitäten sind zur Zeit aktiv“ oder „Was hat Entwicklungsaktivität A geändert“). Geoff Clemm (Bellcore) hat ein System namens Asgard entwickelt (inhouse release), das aufsetzend auf ClearCase dem Benutzer eine solche Sicht auf Aufgaben ermöglicht.

Distributed CM, Merging

Systeme werden durch mehrere Personen entwickelt, die u.U. auch weit voneinander entfernt arbeiten. Je nachdem wieviel Aufwand man vorher in die Entwicklungsaufgaben aufgeteilt hat, entsteht ein Aufwand, wenn man die Änderungen verschiedener Personen später integriert (Merging).

Das Konzept von ClearCase MultiSite erlaubt den konfliktfreien Austausch von Änderungen zwischen mehreren ClearCase Datenbanken, zwischen denen vorher Absprachen getroffen wurden (Sperren von ClearCase Branches). Die Technik (Verwendung von Command Logs) vereinfacht das Problem der verteilten Datenhaltung auf sehr angenehme Weise, wobei allerdings das einfache Datenmodell von ClearCase auch eine Rolle spielt [z.B. für ein anspruchsvolleres Beziehungskonzept (doppelseitige Referenzen) müßte der Mechanismus erweitert werden].

Falls eine Datei von mehreren Personen unabhängig voneinander verändert wurde, müssen diese Versionen später integriert werden (Merging). Jim Buffenberger hielt einen hervorragenden Vortrag über Merging, speziell semantisches Merging. Heute werden meist zeilenbasierte Techniken verwendet, die laut einigen Teilnehmern schon eine sehr hohe Konfliktlösungsquote haben („90%").

CM Processes

Es gab zwei Erfahrungsberichte über den Einsatz von CaseWare (Vorläufer des Continuus CM-Systems) und Adele. Hintergrund waren Projekte bei Boing (CaseWare) und Ericson (Adele). Die Anpaßbarkeit der Systeme wurde diskutiert. Letztendlich konnten die Systeme an die Anforderungen der Organisationen angepaßt werden (z.T. auch umgekehrt). Einzig allein die Unterstützung von CM-Prozessen erschien unmöglich (Adele) bzw. erforderte einen sehr hohen Aufwand (3 Personenjahre, CaseWare).

CM vs. Software Architecture

CM Systeme betrachten ein Software-System als eine Menge von Dateien, die untereinander in Abhängigkeiten stehen, die beispielsweise in einem Makefile aufgelistet sind. Diese Abhängigkeiten und ihre Auswertung z.B. durch make haben recht wenig mit der Architektur des Software-Systems und den darin bestehenden semantischen Abhängigkeiten zu tun (man denke z.B. an ein objekt-orientiertes System). Hier haben wir also redundante Beschreibungen, die aufwendig sind in ihrer Erstellung/Wartung und zu fehlerhaften bzw. überflüssigen Übersetzungen führen können.

Yi-Jing Lin stellte einen Ansatz vor, in dem Architekturelemente, in diesem Falle von C++ (Klassendefinitionen & Methiodenimplementierungen) versioniert werden statt wie bislang Dateien. Der Programmierer sieht in dieser Entwicklungsumgebung keine Dateien sondern nur Architekturelemente, die er editiert/versioniert. Die Aufteilung in Dateien, Include-Anweisungen, Makefiles etc. werden unsichtbar für den Benutzer generiert. Die hierbei resultierende Aufteilung kann sehr feingranular sein.

Introducing CM Systems to Managers

Christopher Adams erzählte, warum es ihm in verschiedenen Firmen, in denen er (beratend?) tätig war, nie (!?) gelungen war ein CM System für ein Projekt zu erwerben. Als Gründe führte er u.a. an

- die hohen Kosten bei der Anschaffung (Kauf, Training, Betrieb). Die Kosten gerade bei neuen innovativen Systemen so hoch, daß dies über der Ebene der Projektleiter entschieden werden muß. Hierbei sind sich die Verantwortlichen oft nicht bewußt, welche Kosten augenblicklich wegen fehlender oder veralteter CM-Technik in ihrer Organisation entstehen.
- Return of Investment. Mehrere Teilnehmer des Workshops sagten, daß aus ihrer Erfahrung Manager die Anschaffung eines CM-Systems innerhalb eines Projektes amotisiert sehen wollen, sei es aus Kostendruck oder dem Streben nach kurzfristigem Erfolg. Ein großes Problem besteht darin, daß über die Kostenersparnis durch CM-Systeme keine brauchbaren Zahlen vorliegen, geschweige denn über den Unterschied durch die Ersetzung eines veralteten CM-Systems durch ein neues. Oft führen CM-Anbieter an „Mit unserem System können Sie wie zuvor weiterarbeiten". Gegenfrage der Manager: „Warum soll ich dann dafür soviel Geld ausgeben?"
- CM-Systeme sind keine Werkzeuge für Manager. Dies würde die Identifikation eines Managers mit der neuen Technologie und den dadurch nötigen oder möglichen Arbeitsabläufen verbessern. Das Management muß insbesondere bei der Einführung von CM voll dahinterstehen, da auch auf Seiten der Entwickler oft große Vorbehalte existieren.

Adams forderte, daß CM-Systeme als Baukasten-Systeme entworfen und verkauft werden sollten. Eine Firma sollte sich zunächst eine Basis-Ausstattung für wenig Geld kaufen können, um dann Zusatz-Funktionalität schrittweise dazukaufen zu können. Die CM-Anbieter unter den Workshopteilnehmern führten dagegen an, daß sie natürlich ihre imensen Entwicklungskosten finanzieren müssen und daß CM-Kunden sich mit Teillösungen oft nicht zufrieden geben, sondern eine möglichst optimale CM-Lösung wollen.

Other CM Domains

CM wird nicht nur in der Software-Entwicklung benötigt sondern auch in anderen Bereichen wie z.B. Maschinenbau, Multi-Media-Entwicklung oder Hardware-Software-Codesign. Es ist zu erwarten, daß in diesen Bereichen

- große Datenmengen zu verwalten sind (großes Gesamtvolumen wie auch große Objekte),
- Datenbanken weit mehr genutzt werden als in der Software-Entwicklung,
- Werkzeuge wie make keine so dominierende Rolle wie in der Software-Entwicklung besitzen,
- Versions-Attribute und -Beziehungen weit mehr benutzt werden [nach Auskunft von Atria verwenden ClearCase-Anwender Attribute und Hyperlinks nur sehr wenig]. Verteilung und Merging können dann schwieriger werden.
- mehrstufige Entwurfsprozesse vorherrschen, in denen auf jeder Stufe die Anzahl der Varianten und die Abhängigkeiten zwischen Ergebnissen weit höher sind als in der Software-Entwicklung.

CM Systems on Customers Site

Sobald ein Software-System ausgeliefert ist, kann es vorkommen, daß der Kunde Fehler findet oder Änderungen verlangt. Daher müssen insbesondere bei großen Systemen gleich ein (kleines) CM-System mitgeliefert werden, welches folgende Aufgaben hat:

- Konfiguration des Software-Systems entsprechend verschiedener (Installations-)Parameter, die kundenspezifisch eingestellt werden können, und Verwalten dieser Parameter.
- Verwalten von Änderungen (Versionen), die vom Software-Anbieter (sei es übers Netz oder durch einen Vertreter vor Ort) oder auch durch den Kunden selber vorgenommen werden.

Dies ist insbesondere notwendig, wenn es sich bei dem Anbieter um einen Auftragnehmer handelt, der Software erstellt, die nach einer Übergangsphase in den Besitz des Auftraggebers übergeht und kein einheitliches CM-System auf beiden Seiten zugrundeliegt.

Der 6th International Workshop on Software Configuration Management findet wieder im Zusammenhang mit der ICSE im nächsten Jahr in Berlin statt..