Software Configuration Management

Seminar

Internetbasierte Wissensverarbeitung

WS 1999/2000

Beitrag von Markus Heidenreich

Inhaltsverzeichnis

Vorwort

Da gerade in der heutigen Zeit viele zusammenarbeitende Softwareentwickler benötigt werden, um immer komplexer werdende Applikationen zu entwerfen, geht der Trend mehr und mehr in die Richtung des räumlich getrennten Arbeitens.
Begünstigt wird diese Entwicklung nicht zuletzt durch die Möglichkeiten der Kommunikation und des Datenaustauschs, die durch das Internet geboten werden.
Auf dieser Basis sollen Werkzeuge konzipiert und entwickelt werden, die eine effiziente verteilte Softwareentwicklung ermöglichen.
Die Nutzung des Internet zu diesem Zweck löst das Verbindungsproblem für sehr große Entfernungen, die Nutzung von Webservern und -browsern wird der Anforderung der Betriebssystemunabhängigkeit und der Realisierung der Verteiltheit im Sinne des Client/Server-Prinzips gerecht.
Unter dem Oberbegriff "Software Configuration Management" versteht man die Menge aller Aufgaben, die bei der Produktverwaltung im Bereich der Softwareherstellung anfallen.
In dieser Ausarbeitung sollen zunächst die Anforderungen an ein webbasiertes SCM-System formuliert, einige technische Möglichkeiten genannt und verschiedene existierende SCM-Produkte, die eine Web-Schnittstelle bieten auf die Anforderungen überprüft und miteinander verglichen werden.

1. Motivation

Programmierer und Softwarearchitekten stehen gerade in einer sehr schnelllebigen Zeit, wie heute vor der Herausforderung, solch umfangreiche Produkte anzubieten, deren Entwicklung nur durch die ständige Zusammenarbeit vieler Spezialisten, ja sogar vieler Spezialistenteams aus aller Welt bewältigt werden kann.

Da ein entstehendes Projekt den Teams nicht in disjunkten Teilen zugewiesen werden kann, besteht die Notwendigkeit der Verwaltung eines gemeinsamen Entwicklungsablaufes.

So müssen Regeln eingehalten werden, die beispielsweise eine Überspeicherung und den damit verbundenen Verlust der Arbeit anderer verhindern und auf diese Weise eine effiziente Zusammenarbeit ermöglichen sollen.

Wir versuchen nun, solche Regeln zu definieren und weitere Anforderungen an ein System zur Verwaltung dieser weltweiten Zusammenarbeit formulieren.

2. Anforderungsanalyse

Softwareprodukte besitzen gerade in der Entwicklungsphase eine Menge Dateien in verschiedenen Versionen. Einige der Versionen werden von unterschiedlichen Programmierern implementiert, sodass die Notwendigkeit besteht, auf eine Versionshistorie jedes einzelnen Moduls zurückgreifen zu können.

Jede ausgelieferte Version einer Applikation soll dem Entwicklungsteam jederzeit zur Verfügung stehen. Man spricht in diesem Zusammenhang zwar von der Versionsnummer einer bestimmten Datei, werden allerdings mehrere Versionen zusammengefasst gesehen so bezeichnet man die Menge dieser Dateien als eine bestimmte Konfiguration. In unserem Fall besteht die Notwendigkeit, zu alten Konfigurationen untergeordnete Entwicklungsstufen zu erstellen, beispielsweise Service-Pakete, sog. Bug-Fixes. Diese Fähigkeit muss nicht nur ein global eingesetztes SCM-System bieten, auch Systeme in kleinen Umgebungen müssen zum effizienten Arbeiten ein solches Konfigurationsmanagement beherrschen.

Hier ist nicht nur die Speicherung und Verwaltung der Objekte selbst von Interesse, es besteht überdies die Notwendigkeit, objektbeschreibende Daten (sogenannte Metadaten), wie Versionsbezeichnung, Name des Eigentümers, Kommentare zum Objekt usw. bereitzuhalten und in einer bestimmten Struktur zu speichern.

Werden Projekte in Teams erarbeitet, so muss jedes Team bzw. jedes Mitglied jederzeit geregelten Zugriff auf bisher erarbeitete Inhalte besitzen. Hier ist die Verwaltung von Zugriffsrechten für die einzelnen Benutzer sehr wichtig. Sind diese Teams nun über die ganze Welt verteilt, so müssen verschiedene Hardwarevoraussetzungen und betriebssystembedingte Unterschiede, sowie Probleme, die sich aus den verschiedenen Sprachen, Zeitzonen, Entwicklungsmethoden und Ansichten ergeben überwunden werden.

Um ein SCM-System in globalen Umgebungen sinnvoll einsetzen zu können, muss es der Anforderung genügen, alle gestellten Anfragen und Aufgaben verteilt bearbeiten zu können. Hier muss ständige und stabile Verfügbarkeit garantiert werden können.

Durch ein SCM-System sollen die Benutzer nicht von ihrer Hauptaufgabe abgelenkt werden, es soll also die Verwaltung der Daten vorwiegend selbst übernehmen und eine zwar funktional orientierte, aber dennoch benutzerfreundliche, übersichtliche und komfortable Benutzeroberfläche bieten.

Zusammenfassend soll ein SCM-System

- automatisches Versions- und Konfigurationsmanagement beherrschen
- benutzerfreundlich angelegt sein
- jederzeit stabile Verfügbarkeit bieten
- nach Prinzipien der Verteiltheit aufgebaut sein
- hardware- und betriebssystemunabhängig arbeiten
- weltweiten Zugriff erlauben.

3. Ansätze zur Erfüllung der Anforderungen

Um der Anforderung der Verteiltheit sinnvoll zu genügen, bietet sich hier die Konzeption nach dem Client/Server-Prinzip an, wo die Benutzer als Clients fungieren und ein zentraler Speicherort als Server die geforderten Dienste bereitstellt.

Ideal eignet sich zur Herstellung der Verbindung zwischen dem Entwickler und einem zentralen Speicherort für die Daten des Projekts das Web, da es ständig zugängig ist und mit den Komponenten der Webserver und -browser aus seiner Natur das Client/Server-Prinzip erfüllt.

Die Unabhängigkeit von eingesetzter Hardware und verwendetem Betriebssystem ist durch den Einsatz von Webbrowsern unter Benutzung standardisierter Protokolle gewährleistet, da zu allen gängigen Betriebssystemen und Hardwarearchitekturen Implementierungen der genannten Komponenten frei erhältlich sind.

Eine Garantie für die unterbrechungsfreie Verfügbarkeit eines Webservers kann zwar nicht übernommen werden, aber durch die später in Abschnitt 3 vorgestellten erweiterten Server-Modelle kann ein maximaler Grad an Fehlertoleranz erzielt werden.

Die Balacierung der Zugriffsrechte einzelner Benutzer sollte sinnvollerweise auf dem Server vorgenommen werden, an dem sich die Benutzer zu authentifizieren haben. Entweder übernimmt das SCM-System selbst die Verwaltung der Zugriffsberechtigungen oder nutzt ein geeignetes Dateisystem aus und somit die Fähigkeit der Rechteverwaltung des Serverbetriebssystems an sich.

Natürlich steigt die Anzahl der Anforderungen bezüglich der Einbruchsicherheit, sowie des Abhörschutzes von Kanälen, wenn das SCM-System über ein öffentliches Medium, wie das Web betrieben werden soll. Durch die Möglichkeit der Nutzung des Web für jedermann kommen solche Sicherheitsrisiken ins Spiel, die es beispielsweise durch den Einsatz kryptographischer Methoden auszugrenzen gilt.

In [2] werden SCM-Systeme in Generationen eingeteilt und unterschieden, ob die serverseitige Speicherung der eigentlichen Projektdaten und/oder der Metadaten der einzelnen Objekte, wie Versionsbezeichnung, Name des Eigentümers, Kommentare zum Objekt, usw. in einem reinen Dateisystem erfolgt oder ob die Informationen in einer oder mehreren Datenbanken eingebunden sind.

Bezüglich des Erscheinungsbildes des Systems für den Benutzer - dem sogenannten Frontend - müssen natürlich gegenüber lokal betriebenen Softwareprodukten Abstriche gemacht werden. Sollen beispielsweise Java-Applets in einem Webbrowser verwendet werden, sind zu hohe Last und die damit verbundenen unnötigen Wartezeiten durch zu verspielte graphische Layoutfinessen nicht erwünscht.

Es würde sich für graphisch aufwendige Oberflächen eher eignen, lokale Clientsoftware zu verwenden, die ihrerseits das Layout bietet und sich nur notwendige Daten über die Netzwerkverbindung verschafft.

4. Diskussion verschiedener Serverkonzepte

Die beiden wichtigsten Fragen für SCM-Systeme allgemein stellen sich in der Art der Speicherung der Projektdaten (sowohl der Objekte als auch deren Metadaten) und der Repräsentation dieser Daten gegenüber dem Benutzer, der dieses verteilte System intuitiv bedienen und ohne großen Aufwand sicher damit arbeiten soll.

Im ersten Punkt der Art der Speicherung aller anfallender Daten sind der Phantasie natürlich keine Grenzen gesetzt. Man unterscheidet drei Konzepte, die natürlich beliebig erweiterbar und kombinierbar sind.

1. Single-Server-Modell
2. Replikationsmodell
3. Peer-to-Peer-Modell

Zur ersten Variante gibt es keinen starken Erklärungsbedarf, da ein Hauptserver alle Clients bedient und hier alle Daten gespeichert werden. Diese Variante birgt viele Nachteile, da der Server nur an einem Ort placiert werden kann und die Geschwindigkeit des Zugriffs für örtlich weit getrennte verschiedene Teams sehr unterschiedlich sein kann. Entwickler, die eine sehr langsame Netzwerkverbindung zum Server haben werden in ihrer Arbeit regelrecht behindert. Genauso bietet die Single-Server Lösung keinerlei Fehlertoleranz. So ist keine unterbrechungsfreie Nutzung der Projektdaten möglich, wenn der einzige datentragende Server beispielsweise abstürzt oder seine Netzverbindung gekappt ist.

Das Replikations- sowie das Peer-to-Peer-Modell beruhen auf der Idee, mehrere Server einzusetzen, von denen jeder einen gewissen Benutzerkreis bedient und deshalb auch geographisch in dessen Nähe placiert wird.

Beim Replikationsmodell trägt jeder Server alle Daten. So ist es notwendig, dass nach jedem daten- oder metadatenändernden Interaktionsprozess ein Synchronisationsvorgang zwischen den Servern angestoßen wird, was mit einer hohen Netzlast verbunden ist, da jeder Server zu jeder Zeit alle Daten auf aktuellem Stand bereithalten soll. Durch das dauernde Abgleichen werden ständig Daten über das Netz geschickt, was starke Sicherheitseinbußen mit sich bringt, denn was ständig über externem Weg versandt wird, wird auch dauernd der Abhör- oder Abänderungsgefahr ausgesetzt. Der starke Vorteil des Replikationsmodells ist die gebotene Fehlertoleranz. Ist einer der Server nicht erreichbar, so stehen die benötigten Daten auf allen anderen Servern zur Verfügung. Natürlich sind hier Probleme zu verwalten, wie konkurrierende Sperrversuche auf identische Objekte, deren Lösung meist eine Erhöhung von Wartezeiten und somit Effizienzverluste bedeuten.

Das Peer-to-Peer-Modell verfolgt die Idee, jeden Server nur diejenigen Daten tragen zu lassen, die von dem jeweils bedienten Benutzerkreis am meisten benötigt werden. So sind die Daten anderer Server referenziert und werden nur im Bedarfsfall von dort angefordert. Man ist so in der Lage, einen höheren Sicherheitsstandard zu bieten, als das Replikationsmodell, da allenfalls Metadaten - wie Erstellungsdatum, Eigentümer, Sperrzustand - und nicht die wesentlich sicherheitsrelevanteren Projektinhalte an sich abgeglichen werden müssen. Dagegen wird hier an der Stelle der Fehlertoleranz gespart, da die Daten jeweils nur von einem der Server gespeichert werden. Natürlich bietet man immerhin bessere Fehlertoleranz, als das Single-Server-Modell, da bei Absturz eines Servers hier nur ein Teil der Projektdaten vorübergehend nicht zur Verfügung steht. Um die Fehlertoleranz zu erhöhen, kann man sich hier überlegen, jeden Server gesondert durch einen sogenannten Backup-Server zu spiegeln, was allerdings die Anzahl der benötigten Server und somit den dafür zu bezahlenden Preis verdoppelt.

5. Beschreibung und Bewertung verschiedener SCM-Systeme

Selbstverständlich würde eine umfassende Beschreibung mehrerer SCM-Systeme und ihrer Web-Schnittstellen den Umfang dieser Ausarbeitung sprengen. Es sollen hier schließlich keine Bedienungsanleitungen gegeben, sondern die verwendeten Grundkonzepte der einzelnen Produkte bzw. Projekte beleuchtet und auf die in Abschnitt 1 formulierten allgemeinen Anforderungen überprüft werden. Bestimmte Kriterien, können natürlich nicht objektiv beurteilt werden (beispielsweise das GUI an sich), dennoch kann die Funktionalität mehr oder weniger Positives aufweisen. Ebenfalls kann der folgende Abschnitt nicht als "Produktberatung" dienen, da verschiedene Entwickler auch verschiedene Anforderungen an ein SCM-System herantragen. So wäre ein System, das nach dem Replikationsmodell arbeitet trotz der idealen Fehlertoleranz für ein Unternehmen völlig uninteressant, das an ein Accounting- und Billing-System angeschlossen ist und für jegliche erzeugte Netzlast bezahlen müsste. Hier würde sich besser ein Peer-to-Peer- oder, je nach weiteren Gesichtspunkten, sogar ein Single-Server-System eignen.
Da die Webanbindung bei den meisten SCM-Systemen einen Aufsatz auf dem eigentlichen System darstellt, muss nach der Funktionalität des SCM-Systemes selbst und des Bedienkomforts des Webfrontends unterschieden werden.

5.1. Kurzüberblick einiger Produkte

Viele webbasierten SCM-Produkte kommunizieren mit dem Benutzer unter Verwendung reiner HTML-Seiten, was zur Aktualisierung der angezeigten Informationen nachteiligerweise ein erneutes Laden der Seite bedeutet. Solche Systeme werden später noch betrachtet. Zunächst ein kurzer Überblick über drei nicht sehr verbreitete Systeme:

Bei WWCM (World Wide Configuration Management) werden die Interaktionen zwischen Server und Clients mit Hilfe von Java-Applets geregelt, was eine Aktualisierung der Inhalte ohne Benutzereingriffe gewährleistet. Zum Dateitransfer wird hier auf eine Fremdapplikation zurückgegriffen, was mit erheblichen Stabilitätseinbussen verbunden ist.

Der Adele Configuration Manager bietet statt der reinen Speicherung der Projektdaten in einem Dateisystem ein verteiltes Datenbankkonzept an, für dessen Verwaltung (Aktualisierung und Synchonisation) die Benutzer selbst verantwortlich sind. Aus diesem Grund besteht die Gefahr des schnellen Auftretens von Inkonsistenzen.

Als Vertreter des Peer-to-Peer-Modells kann Network-Unified Configuration Management (NUCM) genannt werden, das konzeptionell unter Umständen mehrere Server zu einer logischen Speichereinheit zusammenfasst und so dem Benutzer präsentiert. Es besteht zur Erhöhung der Fehlertoleranz die Möglichkeit, jeden Serverplatz um einen gespiegelten Server zu erweitern, was in den meisten Fällen aus Kostengründen nicht in Erwägung gezogen werden kann.

5.2. WebConnect (Starbase Corp.)

Bei WebConnect handelt es sich um eine Erweiterung des bereits bestehenden SCM-Systems StarTeam von Starbase, die es Benutzern erlaubt, über das Web auf StarTeam zuzugreifen. WebConnect unterstützt die einfachen Operationen des Check-in und -out, sowie des Sperrens und Freigebens von Objekten.

WebConnect arbeitet nach dem Single-Server-Modell, wodurch keinerlei Fehlertoleranz geboten ist. Das Versionsmanagement für Objekte übernimmt der Server eigenständig. Zu bedauern ist, dass nur der Administrator und nicht die Benutzer selbst Projekte oder Subprojekte anlegen können.

Zu Beginn einer Sitzung wird eine Authentifizierung durch Eingabe von Benutzername und Passwort durchgeführt.

Im folgenden soll ein optischer Überblick von WebConnect gezeigt werden, woraufhin die dargestellte Benutzeroberfläche beurteilt und auf Ihre Funktionalität überprüft werden soll.

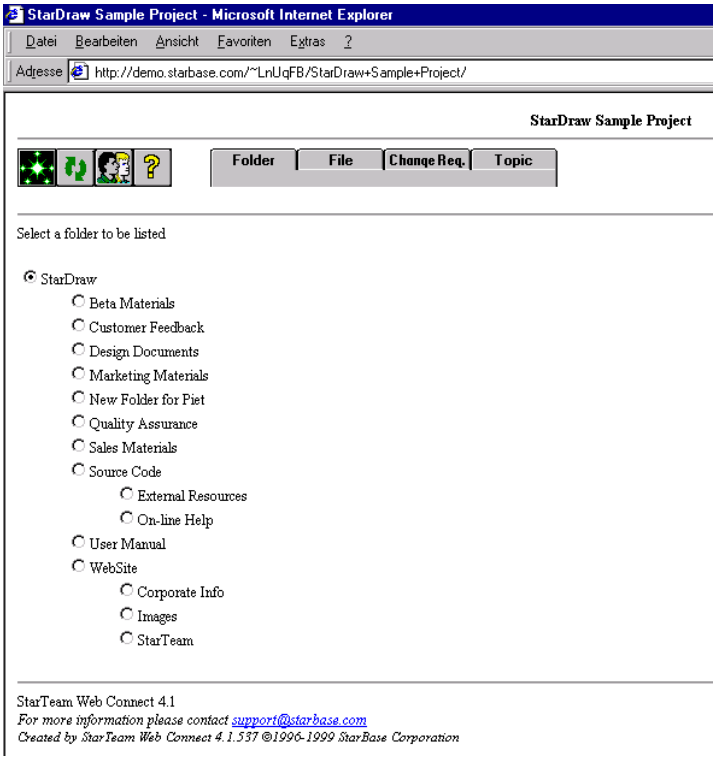

Abb. 4.2.1. Sicht auf die Projektstruktur

Die Sicht auf ein Projekt und seine Unterprojekte stellt sich als stilisierte Verzeichnisstruktur dar. Man kann hier das gewünschte Verzeichnis zur Anzeige der Inhalte auswählen. Hat man ein Verzeichnis gewählt, so lässt sich über das kleine Registerkatenmenü in die Datei-Anzeige wechseln. Man bekommt hier eine Liste der zum (Teil-) Projekt gehörenden Objekte, sowie Informationen über deren aktuelle Version, Datum und Uhrzeit der letzten Änderung sowie Sperrzustand.

Eine detailliertere Ansicht bekommt der Benutzer durch Auswahl des Dateinamen.

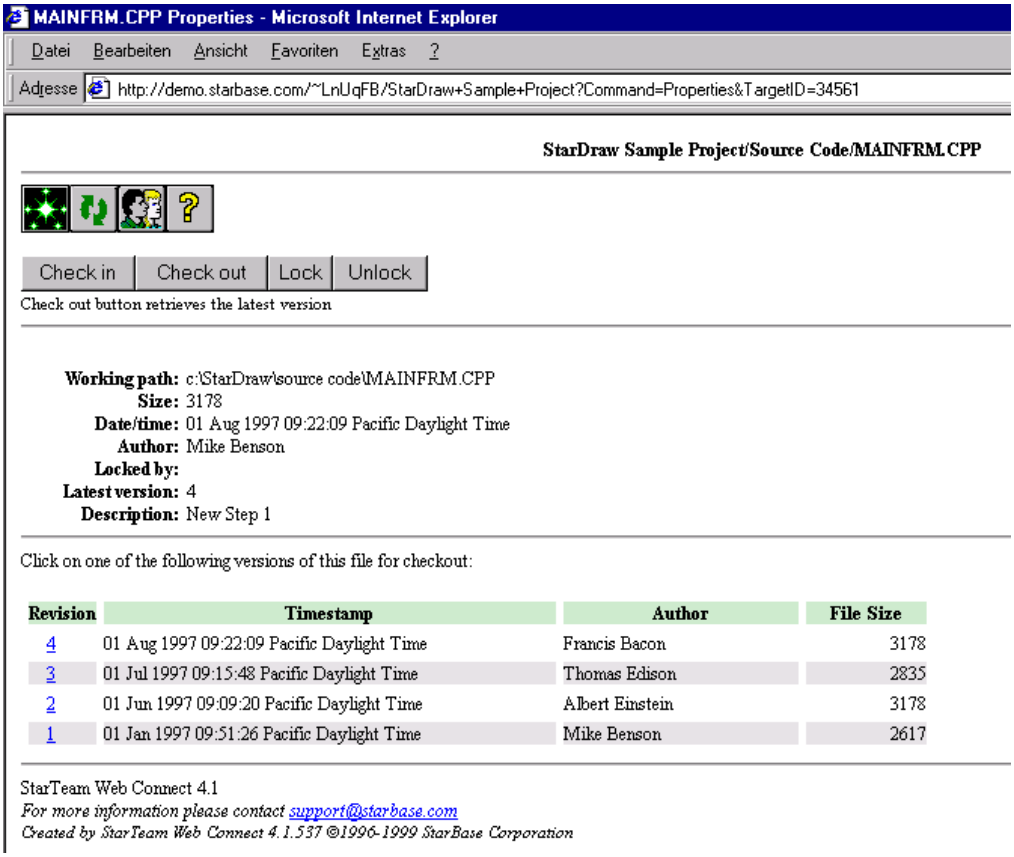

Abb. 4.2.2. Detailanzeige von Objekteigenschaften

Hier bekommt der Benutzer die Möglichkeit, zum Check-in und -out sowie zum Sperren oder Freigeben eines gesperrten Objekts.

Es fällt auf, dass zum Browser des Benutzers reine HTML-Seiten übermittelt werden, die einmal übertragen nur manuell erneut geladen werden können. So kann es passieren, dass ein Objekt längst schon wieder freigegeben ist, obwohl die statische HTML-Seite das Objekt noch immer als gesperrt ausweist. Zwar gibt es in der Sprache HTML die Möglichkeit, Seiten in gewissen Zeitintervallen aufzufrischen, jedoch würde das Flackern der sich ständig neu aufbauenden Webseite keinen Bedienungskomfort darstellen.

Dem Benutzer wird hier keine Möglichkeit geboten, einen eigenen Arbeitsbereich anzulegen, der vom Server mitverwaltet werden könnte - eine Art Home-Verzeichnis.

5.3. Source Integrity Web (Mortice Kern Systems Inc.)

Bei MKS Source Integrity Web handelt es sich, ähnlich wie bei Starbase's Web Connect um einen Aufsatz auf ein bestehendes SCM-System, der weltweiten Zugriff über das Web erlaubt.

Die graphische Benutzeroberfäche gestaltet sich durch eine Menüführung bei MKS benutzerfreundlicher und übersichtlicher, als bei Starbase.

Die bessere Bedienbarkeit und das kompaktere Erscheinungsbild kann jedoch nicht über die Verwendung reiner HTML-Seiten hinwegtäuschen, deren Nachteile bereits genannt wurden.

MKS arbeitet auch nach dem Single-Server-Modell, das in Abschnitt 2 beschrieben wurde und somit die dort genannten Nachteile mit sich bringt.

Allerdings bietet Source Integrity Web die Möglichkeit der Assoziation von Dateitypen mit lokalen Applikationen. Hierdurch wird dem Benutzer eine effizientere Bearbeitung der innerhalb eines Projektes anfallenden Aufgaben ermöglicht.

Durch das Konzept der "Sandbox" erhält der Benutzer eine Möglichkeit, zentral gespeicherte eigene Arbeitsverzeichnisse zu erstellen. MKS unterstützt auch die Erstellung von Projekten und Subprojekten durch Benutzer.

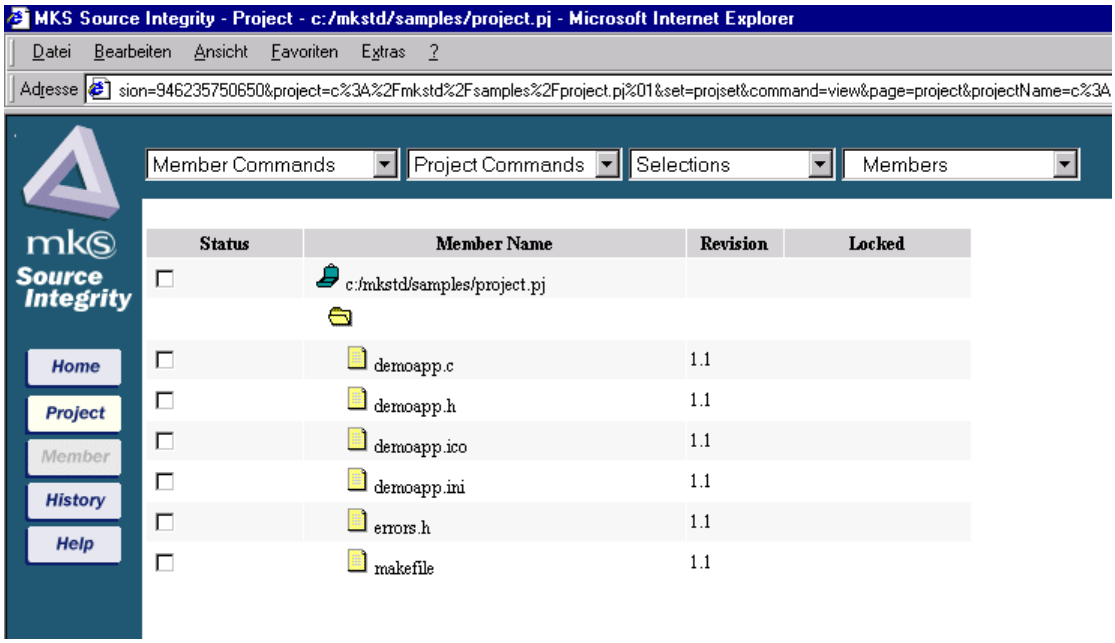

Abb. 4.3.1. Sicht auf ein Projekt

Zur Authentifizierung und Verwaltung von Zugriffsberechtigungen benutzt Source Integrity Web die Benutzerdatenbank des Microsoft Windows NT Servers, auf dem es installiert ist. Somit ist die Steuerung und Überprüfung von Zugriffsrechten auf das Dateisystem des Servers verlagert. Es empfiehlt sich also, ein Dateisystem wie NTFS zu wählen, das bis zur Dateiebene genaue Vergabe von Berechtigungen unterstützt.

MKS bietet dem Benutzer ein sehr umfangreiches HTML-basiertes Hilfesystem.

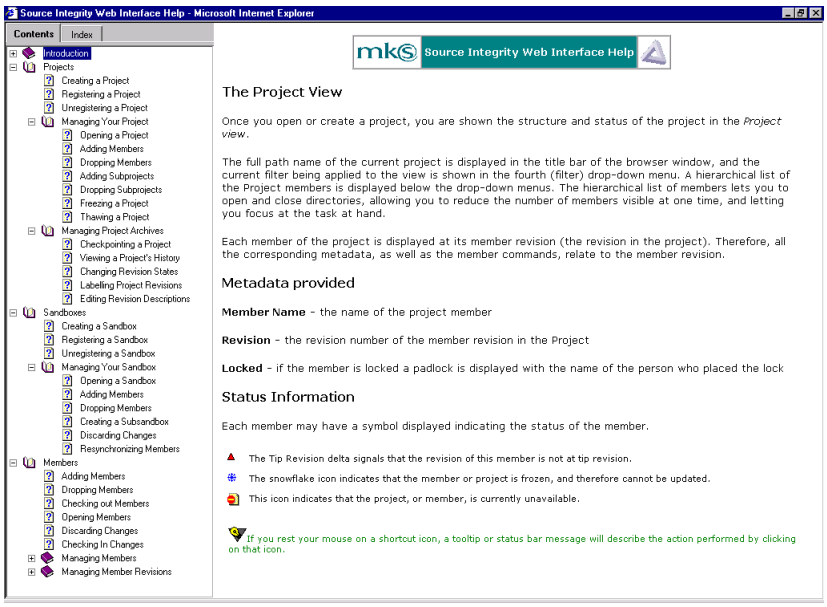

Abb. 4.3.2. MKS-Hilfesystem

Bei der ersten Anmeldung am Source Integrity Web Server wird der Benutzer aufgefordert, eine Clientsoftware zu installieren, die er von dem MKS-Server erhält.

5.4. DISC (Projekt der University of Calgary)

DISC unterscheidet sich grundsätzlich von den bisher vorgestellten Systemen. Zunächst wählt es eine Mischung aus dem Replikations- und dem Peer-to-Peer-Modell. Um einen möglichst hohen Stand an Fehlertoleranz bei gleichzeitiger Kosteneffizienz zu gewährleisten, fügt man zu einem nach dem Peer-to-Peer-Modell aufgebauten Szenario einen zentralen Backup-Server hinzu. Man gewinnt so den großen Vorteil des Replikationsmodells der größeren Ausfallsicherheit, muss aber auch die Nachteile der Netzlast durch die Aktualisierung des Backup-Servers in Kauf nehmen.
Ein weiteres Problem stellt sich durch die hier notwendig werdende Verwaltung möglicher Konkurrenzsituationen bei gleichzeitigen check-out des gleichen Objektes durch mehrere Benutzer.

Für Interaktionen zwischen den Benutzern und dem System wird clientseits ein JAVA Runtime Environment (JRE) Plugin verwendet, womit man den Nachteil reiner HTML-Seiten umgeht. Das Transaktionsmanagement übernimmt ein Enterprise Java Bean (EJB) Tool namens GemStone.

Die Speicherung erfolgt serverseits, wie auch bei den vorhergehenden SCM-Systemen in einem einfachen Dateisystem. Allerdings wird intern nicht projekthierarchisch gespeichert, da diese Art der Speicherung für Objekte, die in mehreren Projekten verwendet werden sollen einen zu hohen Verwaltungsaufwand mit sich bringen würde.

Die notwendigen Metadaten werden allerdings nicht mit den Objekten im Dateisystem, sondern in einer Datenbank gehalten, die von GemStone verwaltet wird.

Ein Interaktionsmanagement, das Konkurrenzsituationen vermeiden soll (sperren oder check-out von Objekten anderer Server muss angemeldet werden, damit keine kritische Situation, wie das doppelte check-out eines Objekts vorkommt) wird ebenso, wie der Synchonisationsprozess zwischen den einzelnen bedienenden und dem Backup-Server von GemStone übernommen.

Ein großer Nachteil zeigt sich bei dem Versuch des Einsatzes sicherer Methoden zum Dateitransfer und der damit verbundenen Notwendigkeit des Empfangs eines Schlüssels. Eine Software, die den Schlüsselempfang verwalten könnte, darf rechtlich nicht in Verbindung mit einem SCM-System vertrieben werden. So müsste sich jeder Benutzer eine solche Software eigens besorgen und installieren, was durch die Verteilung einer Datei umgangen wird, die diese Schlüssel enthält. Hat ein Benutzer allerdings bereits eine solche Datei für andere Zwecke, entsteht das Problem des dauernden Umbenennens und den damit verbundenen Einbussen an Bedienerfreundlichkeit.

6. Abschließende Bewertung

Die in Abschnitt 2 genannten Anforderungen werden durch die vorgestellten SCM-Systeme weitgehend abgedeckt. Bei näherer Betrachtung stellen sich jedoch wieder neue Anforderungen heraus, die nicht zuletzt subjektiver Natur sind und von Benutzer zu Benutzer unterschiedlich stark gewichtet werden.

Als sinnvoll erweist sich zur Anzeige der Benutzeroberfläche die Verwendung eines Java-Applets - wie bei DISC - da hier für die Darstellung von Aktualisierungen im Projektinhalt keine erneute Übertragung einer HTML-Seite benötigt wird bzw. der Benutzer keine statisch alte Information vor sich hat. Die Benutzung eines solchen Applets bringt zwar noch die Nachteile der Verteilung des Übertragungsschlüssels mit sich, hier verbieten aber rechtliche Gründe eine höhere Bedienerfreundlichkeit.

Wünschenswert wäre noch der Einsatz fehlertoleranter Server-Konzepte in kommerziellen SCM-Produkten, was bei DISC in netzlast- sowie kostengünstigerer Weise eingearbeitet ist.

Folgende Tabelle stellt die drei betrachteten Systeme vergleichend gegenüber :

	Web Connect	Source Integrity	DISC
Verteiltheit	+	+	+
weltweite Verfügbarkeit	+	+	+
Stabilität	+	+	+
Fehlertoleranz	-	-	+
Systemunabhängigkeit	+	o	o
Benutzerfreundlichkeit	-	o	+
Konfigurationsmanagement	o	+	+

Literaturverzeichnis

[1] Brian Hoang, Frank Maurer.
DISC: A Distributed Web Based Software Configuration Management System.
http://sern.ucalgary.ca/~maurer/ICSE99WS/Submissions/hoang/Hoang.html

[2] Alex Lobba.
PC-Based Version Control.
http://www.silcom.com/~alobba/no_tables/pc_vc.html

[3] Mortice Kern Systems Inc.
Software Development Meets the Web.
http://www.mks.com/products/scm/si/2168.htm

[4] James J. Hunt, Frank Lamers, Jürgen Reuter, Walter F. Tichy.
Distributed Configuration Management via Java and the World Wide Web.
http://wwwipd.ira.uka.de/~jjh/publications/wwcm.ps