

# **OPEN SOURCE SOFTWARE**

## **BitKeeper und seine Besonderheiten gegenüber Subversion**

**Autor: Thorsten Stahl**

OPEN SOURCE SOFTWARE.....	1
BitKeeper und seine Besonderheiten gegenüber Subversion.....	1
1 Versionsverwaltungswerkzeuge.....	3
1.1 Funktionsweise.....	3
2 BitKeeper in der Open-Source-Softwareentwicklung.....	4
3 Konzepte.....	5
3.1 Die Konzepte Subversion.....	5
3.1.1 Repository .....	5
3.1.2 Versioning Model.....	5
3.1.3 Branching and Merging.....	5
3.2 Die Konzepte BitKeeper.....	6
3.2.1 Repository .....	6
3.2.2 Changesets .....	6
3.2.3 Revision Control.....	6
3.2.4 Open Logging .....	6
4 Vergleich .....	7
4.1 Repository Operationen.....	7
4.1.1 Atomic Commits.....	7
4.1.2 Dateien und Verzeichnisse verschieben oder umbenennen.....	7
4.1.3 Dateien und Verzeichnisse kopieren.....	7
4.1.4 Remote Repository Replikation.....	7
4.1.5 Inkrementelle Änderungen zu Eltern – Repositories.....	7
4.1.6 Repository Erlaubnisse.....	7
4.1.7 Changesets' Support.....	7
4.1.8 Zeilenweise lesen der Dateihistorie.....	7
4.1.9 Tabellarische Übersicht Repository Operationen.....	8
4.2 Features.....	8
4.2.1 Die Fähigkeit auf nur einem Verzeichnis im Repository zu arbeiten.....	8
4.2.2 Verfolgen von “Uncommitted” Veränderungen.....	8
4.2.3 Pro-File Commit Nachricht.....	8
4.2.4 Tabellarische Übersicht Features.....	8
4.3 Technischer Status.....	9
4.3.1 Dokumentation.....	9
4.3.2 Einfacher Einsatz.....	9
4.3.3 Kommandos.....	9
4.3.4 Netzwerk Unterstützung.....	9
4.3.5 Portabilität.....	9
4.3.6 Tabellarische Übersicht Technischer Status.....	9
4.4 User Interfaces.....	10
4.4.1 Web Interface.....	10
4.4.2 Availability of Graphical User-Interfaces.....	10
4.4.3 Tabellarische Übersicht „User Interfaces“.....	10
4.5 Lizenz.....	10
5 Fazit .....	11
6 Quellen.....	12
6.1 Bücher.....	12
6.2 Textquellen aus dem Internet.....	12
6.3 Links zu den verschiedenen Versionsverwaltungen:.....	13

# 1 Versionsverwaltungswerkzeuge

Was ist unter dem Begriff Versionsverwaltungswerkzeuge zu verstehen?

„Unter einer **Versionsverwaltung** versteht man ein System, welches typischerweise in der Softwareentwicklung zur Versionisierung und um den gemeinsamen Zugriff auf Quelltexte zu kontrollieren, eingesetzt wird. Hierzu werden alle laufenden Änderungen erfasst und alle Versionsstände der Dateien in einem Archiv mit Zeitstempel und Benutzerkennung gesichert.“

[Wiki]

Die eben zitierte Definition zeigt deutlich die Vorteile eines solchen Werkzeuges zum entwickeln von Software gerade dann, wenn mehrer Leute an einem Projekt arbeiten. Sofort ist erkennbar, wer wann was geändert hat. Bei Fehlentwicklungen im Code ist es somit leicht möglich, einen vorherigen Stand wieder herzustellen. Das impliziert auch den Vorteil einer solchen Software, wenn nur ein einzelner Programmierer am Software schreiben ist.

## 1.1 Funktionsweise

Die prinzipielle Funktionsweise der verschiedenen Versionsverwaltungswerkzeuge ist immer gleich.

In einem Repository (englisch für Ablage, Aufbewahrungsort...) werden die Dateien eines Projekts archiviert. Meist wird auch nur der Unterschied zwischen zwei Versionen gespeichert und dafür auch noch ein eigenes Dateiformat verwendet. Das spart zwar Platz, aber es wird immer die Versionsverwaltung benötigt, um auf einen Versionsstand zu zugreifen.

Um geänderten Source-Code kompilieren zu können, muss natürlich eine lokale Kopie der geänderten Datei auf dem Rechner vorhanden sein, an dem kompiliert werden soll. Und nicht nur das. Da in großen Projekten ja nicht nur eine Code-Datei vorhanden ist, sondern sehr viele, werden diese aus den verschiedensten Gründen in separate Verzeichnisse gelegt. Dafür ist eine lokale Kopie des ganzen „Projektverzeichnisbaumes“ notwendig. Diese wird Arbeitskopie genannt. Die Arbeitskopie auf dem lokalen Rechner muss natürlich dementsprechend aktualisiert werden. Damit ist eine Hauptaufgabe der Versionsverwaltungssoftware die Kopie und das Repository zu synchronisieren. Dafür gibt es verschiedene Kommandos. Beispielsweise überträgt Checkout aus dem Repository die aktuelle Arbeitskopie. Checkin dagegen überträgt die Arbeitskopie in das Repository. Natürlich verwenden die verschiedenen Tools für die eben beschriebenen Aktionen auch unterschiedliche Bezeichnungen. Manche benutzen Aktualisieren für Checkout oder Commit für Checkin. Das Prinzip ist dabei natürlich immer dasselbe.

Das Versionsverwaltungstool ermöglicht dem Programmierer durch den beschriebenen Aufbau jederzeit das Wiederherstellen einzelner Dateien oder eines früheren Projektstandes. Außerdem kann, da jede Änderung mitprotokolliert wird, der Werdegang der einzelnen Dateien eingesehen werden und so festgestellt, wer wann und was geändert hat. Durch die Archivierung ist ebenfalls der Zugriff auf komplette Release-Stände möglich. So besitzt man automatisch ein Releasearchiv. Die Entwicklung verschiedener Softwarezweige ist ebenfalls Problemlos möglich.

Der Hauptverwendungszweck ist allerdings nach wie vor die Koordinierung des gemeinsamen Zugriffs auf Dateien. Dabei gibt verschiedene Arten dies zu koordinieren. Der Einfachste ist wohl **Lock Modify Write**. Dabei wird die Datei durch einen Benutzer gesperrt. Kein anderer kann hierbei auf die Datei zugreifen, bis der erste diese

geändert und wieder freigegeben hat. Eine weitere Art der Koordinierung ist **Copy Modify Merge**. Eine Datei (vielmehr ihre Kopie) wird von mehreren Benutzer gleichzeitig bearbeitet. Das Verwaltungstool, oder ein „Softwareadministrator“ bzw. Chiefarchitect, Chefprogrammierer etc, fügt die verschiedenen Kopien wieder zusammen.

Von Versionsverwaltungssystemen gibt es mittlerweile schon eine ganze Menge. Die Bekantesten Vertreter sind wohl Alienbrain, BitKeeper, Rational ClearCase, AccuRev, CMVVC, CVS, Darcs, Git, GNU arch, in-Step, monotone, RCS, SCCS, Subversion und Visual SourceSafe.

Einige unterscheiden sich wegen ihrer Verfügbarkeit: Open Source vs. kommerzieller Software. Andere Werkzeuge unterscheiden sich auch in ihrem Benutzerinterface. Manche bieten eine graphische Oberfläche und einige nur die Kommandozeile. Aber die Unterschiede reichen noch viel weiter. Obwohl alle Versionskontrollsysteme in ihren Grundfunktionen dem oben beschriebenen Prinzip entsprechen, haben die Entwickler der einzelnen Tools noch zusätzliche Raffinessen und Funktionen hinzugefügt, um jede Art der Softwareentwicklung mit einem Versionsverwaltungs-Werkzeug auszustatten.

Ein sehr gutes und viel verwendetes Tool ist CVS und dessen Weiterentwicklung Subversion. Sie bieten allerhand Funktionalität und stehen unter der GNU – Lizenz und können damit frei verwendet werden...

## 2 BitKeeper in der Open-Source-Softwareentwicklung

...

Da verwundert es natürlich schon, was einen Linus Torvalds dazu bewegt, obwohl er ja gemeinhin als Guru in der Open – Source – Gemeinde gilt, seit 2002 ein kommerzielles Versionsverwaltungstool für seine Kernelentwicklung zu benutzen, obwohl es aus dem Open – Source Lager ja fast einen unerschöpflichen Vorrat an diesen Werkzeugen gibt. Dieser Missstand veranlasst sogar GNU – Urvater Richard Stallman zu einem Kommentar:

*"Es gibt Menschen wie Torvalds, der unsere Gemeinschaft zwingt, ein nicht-freies Programm zu nutzen und fordert sofort eine technisch bessere Lösung bereitzustellen oder den Mund zu halten" so Stallman*  
[Golem]

Der Artikel geht noch viel weiter in die Details der Fehlentwicklung in der Open – Source – Gemeinde ein.

Doch dies soll nicht Inhalt dieser Ausarbeitung sein. Vielmehr sollen hier die Unterschiede zwischen den sehr verbreiteten Subversion und BitKeeper herausgearbeitet werden. In den Quellenangaben finden sich einige Links, mit denen sich der interessierte Leser über die Diskussion um BitKeeper bei der Entwicklung von Freier Software informieren kann.

## 3 Konzepte

Der Vergleich der beiden Versionskontrollsysteme soll in diesem Abschnitt nicht an deren Einfachheit der Installation oder deren Benutzerfreundlichkeit bezüglich der GUI – ausführliches darüber in Abschnitt 4 - erfolgen, sondern an deren Konzepte und Funktionalitäten. Dazu sind die von Herstellern in ihren Handbücher, Manuals und User-Dokumentationen beschriebenen und angepriesenen Konzepte herausgearbeitet und in den folgenden Abschnitten niedergeschrieben worden.

### 3.1 *Die Konzepte Subversion*

Die hier im Abschnitt 3.1 beschriebenen Konzepte sind aus [Sub] entnommen, das, von dem in der Quellenangabe vermerkten Link, als freies Buch heruntergeladen werden kann.

Subversion ist eine freies Versionskontrollsystem, welches Datei und Verzeichnisse in einem zentralen Repository verwaltet. Das zentrale Repository ist nichts anderes als ein gewöhnlicher Fileserver, welcher zusätzlich noch jede Veränderung der Dateien und Verzeichnisse vermerkt. Durch das Vermerken der Änderungen, ist es Subversion möglich, einen alten Stand wieder herzustellen.

#### 3.1.1 **Repository**

Da Subversion als zentrales System konzipiert ist, ist der Kern von Subversion das Repository. Die Funktionen von Subversion sind dementsprechend auf dieses Repository abgestimmt.

#### 3.1.2 **Versioning Model**

Die Hauptaufgabe eines Versionsverwaltungstools ist es, den gemeinsamen Zugriff auf Dateien zu verwalten. Mit „Lock-Modify-Unlock“ und “Copy-Modify-Merge“ werden die in der Einleitung schon beschriebenen Konzepte unterstützt.

#### 3.1.3 **Branching and Merging**

Das Erstellen von Branches gehört ebenfalls zum Funktionsumfang von Subversion. Damit können verschiedene Entwicklungszweige aus einer gemeinsamen Basis heraus erstellt und verwaltet werden. Dazu wird das ursprüngliche Filesystem (Repository oder Teile davon) echt kopiert und existieren so als normales Filesystem. Allerdings ist dies kein internes Konzept der Verwaltungssoftware. Es ist einfach nur ein Kopieren eines Verzeichnisses.

Beim Merging werden nicht zwei Branches zusammengeführt, sondern es werden lediglich zwei Repositoryzweige verglichen und deren Unterschiede in einer Arbeitkopie zusammengelegt.

## **3.2 Die Konzepte BitKeeper**

Die beschriebenen Konzepte sind von [Bit] entnommen.

Anders als die meisten Versionskontrollsysteme ist BitKeeper als ein skalierbares System entworfen worden. Damit unterstützt es globales verteiltes Entwickeln. Dazu gehören auch Operationen die ausgeführt werden können, wenn ein Client nicht mit dem Zentralen System verbunden ist. Changesets (Erklärung siehe Abschnitt 4.1.7) und benannte Entwicklungsstränge (Branches) werden ebenfalls unterstützt.

### **3.2.1 Repository**

Jedes Repository ist eine geschlossene Einheit, welche alles enthält, was ein Programmierer zum arbeiten benötigt. Ein Entwickler kann davon einen Klon erstellen oder es löschen, ohne das dabei ein Seiteneffekt auf gemeinsam benutzte Repositories auftritt. Die Verbindung zwischen Repository und Klone kann als Eltern-Kind Beziehung betrachtet werden. Dabei werden Änderungen generell vererbt. Jedoch kann dies umgangen werden, wenn es gewünscht wird.

### **3.2.2 Changesets**

Changesets gruppieren gleichartige Änderungen. Deswegen spricht die Firma BitMover gern von einem Änderungs- Task und so von einem task – based – System. Ein Changeset erfüllt die Atomizität (Siehe Abschnitt 4) und kann auch wieder rückgängig gemacht werden.

### **3.2.3 Revision Control**

Überwacht werden Dateiinhalte, Dateinamen, Dateiflags, symbolische Links und Dateiberechtigungen.

### **3.2.4 Open Logging**

Um die dezentrale Struktur von BitKeeper zu ermöglichen ist natürlich trotzdem ein reger Nachrichtenaustausch mit einem zentralen System nötig damit Änderungen konsistent gehalten werden können. Dafür dient ein öffentlicher Web – Server. Das senden von Metadaten und deren anschließende Verarbeitung nennt BitMover Open Logging. Dadurch kann der Fortschritt von jedem Entwickler verfolgt werden. Gesendet werden ChangeSet – Dateien, Log – Messages und viele Files, welche nötig für die Konsistenz von Repository sind. Nicht gesendet werden Sourcecode – Dateien und Dateien, die von BitKeeper verwaltet werden.

## 4 Vergleich

Der Inhalt des Vergleiches in diesem Abschnitt ist Größtenteils von [CompVCS] entnommen und wurde, soweit wie möglich, eingedeutscht und bei Bedarf erweitert. In den Tabellen ist die englische Bezeichnung belassen worden. Um genauere Kommentare zu den einzelnen Eigenschaften zu erhalten bitte [CompVCS] nachschlagen.

### 4.1 *Repository Operationen*

Die hier beschriebenen Funktionen betreffen hauptsächlich die Synchronisation zwischen Lokaler Kopie und Repository.

#### 4.1.1 **Atomic Commits**

Wenn ein Versionsverwaltungstool diese Eigenschaft besitzt, so entsteht kein Schaden, wenn ein Checkin bzw. Commit erfolgen soll und dieser durch irgendeinen Einfluß, z.B. Netzwerkausfall etc., unterbrochen wird. Das Repository ist immer in einem konsistenten Zustand. Ist eigentlich ähnlich wie ein Transaktionsverfahren.

#### 4.1.2 **Dateien und Verzeichnisse verschieben oder umbenennen**

Diese Eigenschaft mag trivial klingen, aber man darf nicht vergessen, das ja jede Datei ein Logfile besitzt, indem ja der Werdegang dieser verzeichnet ist. Das Logfile muss beim Umbenennen und Verschieben beibehalten werden und auch das Umbenennen vermerken.

#### 4.1.3 **Dateien und Verzeichnisse kopieren**

Gleiches Problem wie oben bei 4.1.2 mit den Logfiles.

#### 4.1.4 **Remote Repository Replikation**

Die Anforderung beinhaltet die Aufgabe ein Repository, welches auf einem entfernten Server liegt, auf ein lokales System zu kopieren und dabei die volle Funktionalität aufrecht zu erhalten.

#### 4.1.5 **Inkrementelle Änderungen zu Eltern – Repositories**

Da ja ein Repository ein Projekt, oder auch Modul, Funktion o.ä., enthält fordert diese Eigenschaft eine Verbindung der einzelnen Archive.

#### 4.1.6 **Repository Erlaubnisse**

Hierunter ist eine kleine Zugriffskontrolle auf verschiedene Teile eines Repositorys zu verstehen. Somit können sensible Teile des Projekts nur von autorisierten Entwicklern bearbeitet werden.

#### 4.1.7 **Changesets' Support**

Changesets sind eine Menge von Veränderungen, welche voneinander abhängig sind. Mit einem Changeset kann die Modifizierung der Datei/en in einem „atomic package“ erfolgen.

#### 4.1.8 **Zeilenweise lesen der Dateihistorie**

Die Funktionalität wird zum Beispiel dafür benötigt, um der Werdegang einer Datei genauestens nachzuvollziehen - wer was und wann geändert hat.

### 4.1.9 Tabellarische Übersicht Repository Operationen

	Subversion	BitKeeper
Atomic Commits	Ja	Ja
Files and Directories Moves or Renames	Ja	Ja
File and Directories Copies	Ja	Ja
Remote Repository Replication	Ja. Extra package	Ja
Propagating Changes to Parent Repositories	Ja. Extra package	Ja
Repository Permissions	Nein	Ja
Changesets' Support	Teilweise unterstützt	Ja
Tracking Line-wise File History	Ja	Ja

## 4.2 Features

### 4.2.1 Die Fähigkeit auf nur einem Verzeichnis im Repository zu arbeiten

Bietet ein Versionskontrollsystem diese Eigenschaft, so muss nicht das ganze Repository als lokale Kopie auf dem Rechner liegen, sondern nur das entsprechende Verzeichnis.

### 4.2.2 Verfolgen von "Uncommitted" Veränderungen

Bei dieser Eigenschaft geht es darum, dass, wenn eine Arbeitsverzeichnis noch nicht auf das Repository übertragen wurde, die Filehistory auch in der lokalen Kopie nachvollzogen werden kann.

### 4.2.3 Pro-File Commit Nachricht

Wird dieses Feature unterstützt, so meldet das Tool nach jeder übertragenen Datei das erfolgreiche Übertragen.

### 4.2.4 Tabellarische Übersicht Features

	Subversion	BitKeeper
Ability to Work only on One Directory of the Repository	Ja	Nein
Tracking Uncommitted Changes	Ja	Ja
Per-File Commit Messages	Nein	Ja

### 4.3 *Technischer Status*

Beim technischen Status geht es um die Software an für sich, also die Qualität, und was man von einer derartigen Anwendung noch erwarten kann.

#### 4.3.1 **Dokumentation**

Ist die Software gut Dokumentiert? Kann man sich schnell einarbeiten...

#### 4.3.2 **Einfacher Einsatz**

Ist die Versionsverwaltung schnell einsetzbar. Müssen zusätzlich Dinge beachtet werden.

#### 4.3.3 **Kommandos**

Wie schnell lernt man mit dem Tool umzugehen? CVS war der defacto-Standard. Wenn sich ein Tool daran anlehnt, können viele damit umgehen, ohne sich erneut einlesen zu müssen.

#### 4.3.4 **Netzwerk Unterstützung**

Wie stark ist ein Netzwerkeinsatz mit in die Software integriert?

#### 4.3.5 **Portabilität**

Wie viel Betriebssysteme und Architekturen werden unterstützt?

#### 4.3.6 **Tabellarische Übersicht Technischer Status**

	Subversion	BitKeeper
Documentation	Sehr gut	Sehr gut
Ease of Deployment	Erfordert einen Apache 2 Modul...	Gut
Command Set	Anklicken genügt (GUI), ansonsten wie CVS	CVS artiger Kommando Standard, leicht zu lernen
Networking Support	Sehr gut	Gut
Portability	Exzellent	Sehr gut

## 4.4 *User Interfaces*

### 4.4.1 **Web Interface**

Unterstützt die Software ein Webinterface?

### 4.4.2 **Availability of Graphical User-Interfaces.**

Wie gut ist die Verfügbarkeit von GUI's?

### 4.4.3 **Tabellarische Übersicht „User Interfaces“**

	Subversion	BitKeeper
Web Interface	Ja	Ja
Availability of Graphical User-Interfaces.	Sehr gut	Gut

## 4.5 *Lizenz*

	Subversion	BitKeeper
Lizenz	Open Source	Kommerzielle Software

## 5 Fazit

Aus den Konzepten aus 3 ist ein großer Unterschied erkennbar. Subversion ist ein zentrales System wogegen BitKeeper ein skalierbares, dezentrales Tool ist. Auch der Vergleich in Abschnitt 4 zeigt einige Unterschiede zwischen BitKeeper und Subversion auf.

Subversion ist als Weiterentwicklung von CVS ein sehr gutes Tool für den normalen Softwareentwicklungsalltag ohne viel Schnickschnack. Es verbessert und erweitert den Urvater des Versionskontrollsystems CVS erheblich aber bietet keine Changesets und unterstützt nur simples Branching.

BitKeeper ist ein sehr professionelles Werkzeug zur Codeverwaltung. Es kommt der Kernelentwicklung von Linus Torvalds und seinen Mannes mit seiner dezentralen Struktur sehr entgegen und ist durch seine Zuverlässigkeit und sonstigen Features außer Konkurrenz.

Der entschiedenste Unterschied, weswegen sich Linus Torwald wohl 2002 für BitKeeper entschieden hat, soll hier, durch ein Statement von den Entwicklern von Subversion selbst, nochmals herausgestellt werden:

„Subversion was primarily designed as a replacement for CVS. It is a centralized version control system. It does not support distributed repositories, nor foreign branching, nor tracking of dependencies between changesets. Given the way Linus and the kernel team work, using patch swapping and decentralized development, Subversion would simply not be much help. While Subversion has been well-received by many open source projects, that doesn't mean it's right for *every* project.”

[SubTeam]

Als Weiterentwicklung von CVS war Subversion nie dafür ausgelegt, die in Kernelentwicklung benötigten Eigenschaften zu erfüllen. Verteilte Repositories, da ja ein Kernel viele sehr unterschiedliche Module enthält, ein ausgeklügeltes Branching oder etwa das Überwachen der Abhängigkeiten zwischen Changesets (siehe oben) bietet Subversion nicht.

Das veranlasste wohl Linus Torvalds dazu ein eigenes Versionsverwaltungstool „git“ als Ersatz für Bitkeeper zu entwickeln [ArHeGit].

## 6 Quellen

### 6.1 Bücher

[Sub]:

Collins-Sussman, Fitzpatrick, Pilato: Version Control with Subversion: For Subversion 1.1, <http://svnbook.red-bean.com> 2005

### 6.2 Textquellen aus dem Internet

[Bit]: <http://www.bitkeeper.com/UG/>

Beschreibung:

User Guide von BitMover über BitKeeper.

[SubTeam]: <http://subversion.tigris.org/subversion-linus.html>

Beschreibung:

Das Subversion – Entwicklerteam beschreibt selbst, warum ihr Versionsverwaltungstool nicht für die Kernelentwicklung geeignet ist.

[DiscBk]: <http://thread.gmane.org/gmane.linux.kernel/293914>

Beschreibung:

Diskussion mit Linus Torvald um den Nachfolger von BitKeeper.

[Wiki]: <http://de.wikipedia.org>

Beschreibung:

Freie Online Enzyklopädie.

[CompVCS]: <http://better-scm.berlios.de/comparison/comparison.html>

Beschreibung:

Vergleich von 13 bekannten Versionsverwaltungstools.

[ArtHeise]: <http://www.heise.de/newsticker/search.shtml?T=BitKeeper&button=los%21>

Beschreibung:

Verschiedene Artikel, die auf [www.heise.de](http://www.heise.de) über die Versionsverwaltung in der Kernelentwicklung geschrieben wurden.

[ArHeGit]:

<http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/58621&words=BitKeeper%20Bitkeeper>

Beschreibung:

Hinweis darauf, das Torvalds ein eigenes Versionsverwaltungstool entwickelt.

[KerTrap]: <http://kerneltrap.org/node/4982>

Beschreibung:

Hinweis darauf, das Torvalds ein eigenes Versionsverwaltungstool entwickelt.

[Golem]: <http://www.golem.de/0205/19887.html>

Beschreibung:

Richard Stallman ärgert sich über die Linuxentwicklung .

### 6.3 *Links zu den verschiedenen Versionsverwaltungen:*

- **Alienbrain** : <http://www.alienbrain.com/>
- **BitKeeper** : <http://www.bitkeeper.com/>
- **ClearCase** : <http://www-306.ibm.com/software/awdtools/clearcase/>
- **AccuRev** : <http://www.accurev.com/>
- **CMVVC** :
- **CVS** : <https://www.cvshome.org/>
- **Darcs** :
- **Git** : <http://www.git-source.org/>
- **GNU arch** : <http://www.gnu.org/software/gnu-arch/>
- **in-Step** : <http://www.in-step.de/>
- **monotone** :
- **RCS** : <http://www.gnu.org/software/rcs/rcs.html>
- **SCCS** : <https://www.cvshome.org/cyclic/cyclic-pages/sccs.html>
- **Subversion** : <http://subversion.tigris.org/>
- **Visual SourceSafe** : <http://msdn.microsoft.com/vstudio/previous/ssafe/>