Vorstellung SVN und Trac

Jan Zieschang, 01.06.2007, Berlin




Agenda

- Begriffe
- Vorstellung SVN
  - TortoiseSVN – das GUI
  - Unterschiede zu VSS
- Vorstellung TRAC
  - Trac Umgang/Komponente
- SVN und Trac bei SD&C / CapeVision
- Weitere Tools / Ausblick

Begriffe

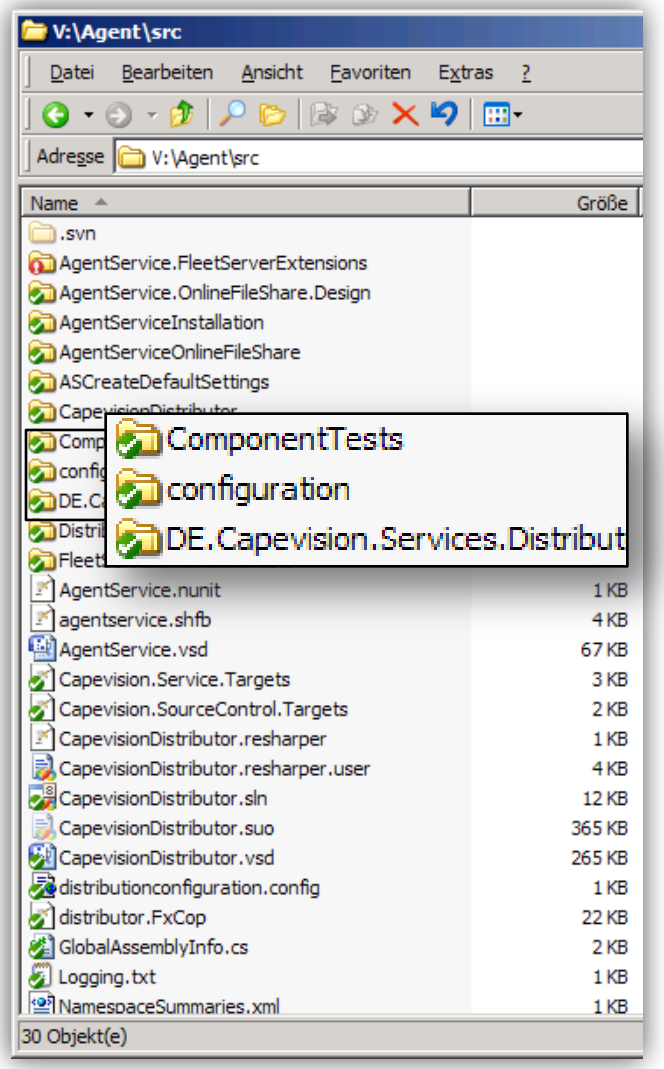
- SVN = Subversion
- Wiki = Einfaches und leichtes Bearbeiten von Webseiteninhalten mit dem Ziel der Wissensspeicherung
- Issue Tracking = Verwalten/Steuern von Projektbezogenen Aufgaben/Artefakten
- SCM = Software Configuration Management
- CI = Continues Integration
- WAN = Wide Area Network
- VSS = Microsoft Visual Source Safe
- Repository = Speicher-/Ablageort

Vorstellung SVN

- SVN = Subversion http://subversion.tigris.org/
- Open Source
- SVN ist ein Versionsverwaltungssystem/SCM
- Sehr Leistungsfähig
- Mehrere Schnittstellen
- Viele Plattformen
- Copy-Modify-Merge
- Sehr gute Skalierung über WAN
- Keine Veränderung der Quellen wie bei VSS

Vorstellung SVN

- Arbeiten mit SVN etwas umständlich, da nur Kommandozeilen Tools
- ABER
  - Viele Clients zum arbeiten verfügbar, beste TortoiseSVN
  - Integration in Explorer
  - Schnelle Anzeige des Status
  - Keine separate Anwendung notwendig
  - Einfaches Anlegen neuer Repositories
  - Einfaches Mergen von Versionen/Branches



Vorstellung SVN - Server

- Verschiedene Möglichkeiten einen Server zu erstellen/betreiben
  - Apache (häufigste) – nutzen der kompletten Apache Infrastruktur, Authentifizierung, Authorisierung, Datenkomprimierung, Datenfilterung, …
  - FileSystem/UNC-Shares – direkter Zugriff auf das Repository, Stuerung über FileShare-Berechtigungen
  - SVNSERVE – schneller und schlanker Server. Allerdings nur Standard SVN-Authentifizierung
- Gute Dokumentation online verfügbar
  - Erklärung vieler Grundprinzipien
  - Server-Einrichtung
  - Arbeiten mit SVN

Vorstellung SVN -Eigenschaften

- **Most current CVS features.** Subversion is meant to be a better CVS, so it has most of CVS's features. Generally, Subversion's interface to a particular feature is similar to CVS's, except where there's a compelling reason to do otherwise.

- **Directories, renames, and file meta-data are versioned.** Lack of these features is one of the most common complaints against CVS. Subversion versions not only file contents and file existence, but also directories, copies, and renames. It also allows arbitrary metadata ("properties") to be versioned along with any file or directory, and provides a mechanism for versioning the `execute' permission flag on files.

- **Commits are truly atomic.** No part of a commit takes effect until the entire commit has succeeded. Revision numbers are per-commit, not per-file; log messages are attached to the revision, not stored redundantly as in CVS.

- **Apache network server option, with WebDAV/DeltaV protocol.** Subversion can use the HTTP-based WebDAV/DeltaV protocol for network communications, and the Apache web server to provide repository-side network service. This gives Subversion an advantage over CVS in interoperability, and provides various key features for free: authentication, wire compression, and basic repository browsing.

Vorstellung SVN -Eigenschaften

- **Standalone server option.** Subversion also offers a standalone server option using a custom protocol (not everyone wants to run Apache 2.x). The standalone server can run as an inetd service, or in daemon mode, and offers basic authentication and authorization. It can also be tunnelled over ssh.

- **Branching and tagging are cheap (constant time) operations** There is no reason for these operations to be expensive, so they aren't.

- Branches and tags are both implemented in terms of an underlying "copy" operation. A copy takes up a small, constant amount of space. Any copy is a tag; and if you start committing on a copy, then it's a branch as well. (This does away with CVS's "branch-point tagging", by removing the distinction that made branch-point tags necessary in the first place.)

- **Natively client/server, layered library design** Subversion is designed to be client/server from the beginning; thus avoiding some of the maintenance problems which have plagued CVS. The code is structured as a set of modules with well-defined interfaces, designed to be called by other applications.

Vorstellung SVN -Eigenschaften

- **Client/server protocol sends diffs in both directions** The network protocol uses bandwidth efficiently by transmitting diffs in both directions whenever possible (CVS sends diffs from server to client, but not client to server).
- **Costs are proportional to change size, not data size** In general, the time required for a Subversion operation is proportional to the size of the *changes* resulting from that operation, not to the absolute size of the project in which the changes are taking place. This is a property of the Subversion repository model.
- **Choice of database or plain-file repository implementations** Repositories can be created with either an embedded database back-end (BerkeleyDB) or with normal flat-file back-end, which uses a custom format.
- **Versioning of symbolic links** Unix users can place symbolic links under version control. The links are recreated in Unix working copies, but not in win32 working copies.
- **Efficient handling of binary files** Subversion is equally efficient on binary as on text files, because it uses a binary diffing algorithm to transmit and store successive revisions.

Vorstellung SVN -Eigenschaften

- **Parseable output** All output of the Subversion command-line client is carefully designed to be both human readable and automatically parseable; scriptability is a high priority.
- **Localized messages** Subversion uses gettext() to display translated error, informational, and help messages, based on current locale settings.
- **Repository mirroring** Subversion supplies a utility, svnsync for synchronizing (via either push or pull) a read-only slave repository with a master repository.

Subversion Struktur

- Verschiedene Theorien über den Aufbau.
- Sinnvoll: ein Projekt ein Repository

- Gebräuchliche Struktur bei CV:
  - Trunk – enthält den aktuellen Arbeitsbaum
  - Tags – Alte Versionen, die bestehen bleiben sollen
  - Branches – ältere Versionen, an denen weitergearbeitet wird (bug fixes, features, …). Branches entstehen meistens aus Tags
  - Releases – Ablage von Übergaben/Auslieferungen an den Kunden mit Paketen. Allerdings keine Änderungen des Source Codes

Subversion post commit – Ticket change

- #TicketNummer als Ticket referenz
- Tickets beim commit modifizieren:
  - command #1
  - command #1, #2
  - command #1 & #2
  - command #1 and #2
  - command ticket:1
  - command ticket:1, ticket:2
  - command ticket:1 & ticket:2
  - command ticket:1 and ticket:2

Subversion post commit – Ticket change

- Kommandos
  - 'close‘ → schließen von Tickets
  - 'closed‘→ schließen von Tickets
  - 'closes' → schließen von Tickets
  - 'fix' → schließen von Tickets
  - 'fixed' → schließen von Tickets
  - 'fixes' → schließen von Tickets
  - 'korrigiert' → schließen von Tickets
  - 'addresses' → hinzufügen von Kommentaren zu einem Ticket
  - 're' → hinzufügen von Kommentaren zu einem Ticket
  - 'references' → hinzufügen von Kommentaren zu einem Ticket
  - 'refs' → hinzufügen von Kommentaren zu einem Ticket
  - 'see' → hinzufügen von Kommentaren zu einem Ticket
  - 'siehe' → hinzufügen von Kommentaren zu einem Ticket

Unterschiede zu VSS – Eure Meinung?

SVN

- Atomare Checkins
- SF.NET hat auf SVN umgestellt
- Branches kosten „kein“ Speicher
  Branch relativ schnell
- Berechtigung abhängig vom verwendeten Mechanismus, SVNAUTH-Nutzbar oder auch Apache-Steuerung
- Problemloser Zugriff aus dem Internet möglich
- Verknüpfung zwischen Code und Issues

VSS

- File basierende Checkins
- Probleme bei großen Repositories
- Branch verdoppelt die Größe des verzweigten Bereiches
- Berechtigung über Client steuerbar
- Zugriff aus dem Internet langsam oder nur mit Visual Studio möglich
- Integration nur mit kommerziellen Tools (Annahme)

Unterschiede zu VSS – Eure Meinung?

SVN

- Gleichzeitiges Arbeiten an einer Datei
- Nur Server Kommunikation zum aktualisieren oder senden veränderter Daten
- Kein Erwerb von Lizenzen notwendig
- Bergiffe teilweise anders
- Möglichkeit Kommentare beim Einchecken zu erzwingen

VSS

- Datei wird exklusiv von einem Bearbeitet
- Kommunikation zum Bearbeiten und zum Abschließen mit dem Server notwendig
- VSS Lizenz notwendig (VS oder separat)
- Für viele gewohnte Begriffe
- Kommentare für Ein- und Auschecken möglich

Vorstellung Trac

- Open Source Project (http://trac.edgewall.org/wiki/WikiStart)
- Issue / Project Tracking System
- Web-Interface
- Project Wiki
- Source-Browser
- Nahtlose Integration mit SVN
- Übernehmen von Check-In Comments
- Steuern von Tickets mit Check-In Comments

Trac Komponenten - WIKI

- Built-In Wiki Engine
- Einfache Kommandos zum Formatieren des Codes
- Möglichkeit Tickets, SCM-Artefakte zu referenzieren
- Nachschlagewerk für Projektbeteiligte
- Möglichkeiten von Exports (LaTeX und PDF -Plugins)
  http://www.trac-hacks.org/

Wiki Formatting
'''bold'''
''italic''
'''''bold italic'''''
__underline__
{{{monospace}}} or `monospace`
~~strike-through~~
^superscript^
,,subscript,,.
= Heading =
== Subheading ==
Line 1[[BR]]Line 2
* Item 1   * Item 1.1
1. Item 1   1. Item 1.1
Term 1::   definition 1

{{{
  def HelloWorld()
  print "Hello World"
}}}

||Cell 1||Cell 2||Cell 3|| ||Cell 4||Cell 5||Cell 6||
http://www.example.com/
WikiPageName
[http://www.example.com/ EX, inc.]
[wiki:TitleIndex Title Index]
!NoHyperLink
http://www.example.com/image.png
[[WikiMacroName]]
{{{ #!html Test }}}

TrackLinks

```
#1 -- Tickets
{1} -- Reports
r1, [1] -- Changesets
r1:3, [1:3] -- Revision Logs
CamelCase -- Wiki pages
milestone:"Beta Release"
source:trunk/COPYING
attachment:"file name.doc"
source:/trunk/COPYING#200
source:"/trunk/README FIRST"
```

WikiMacros

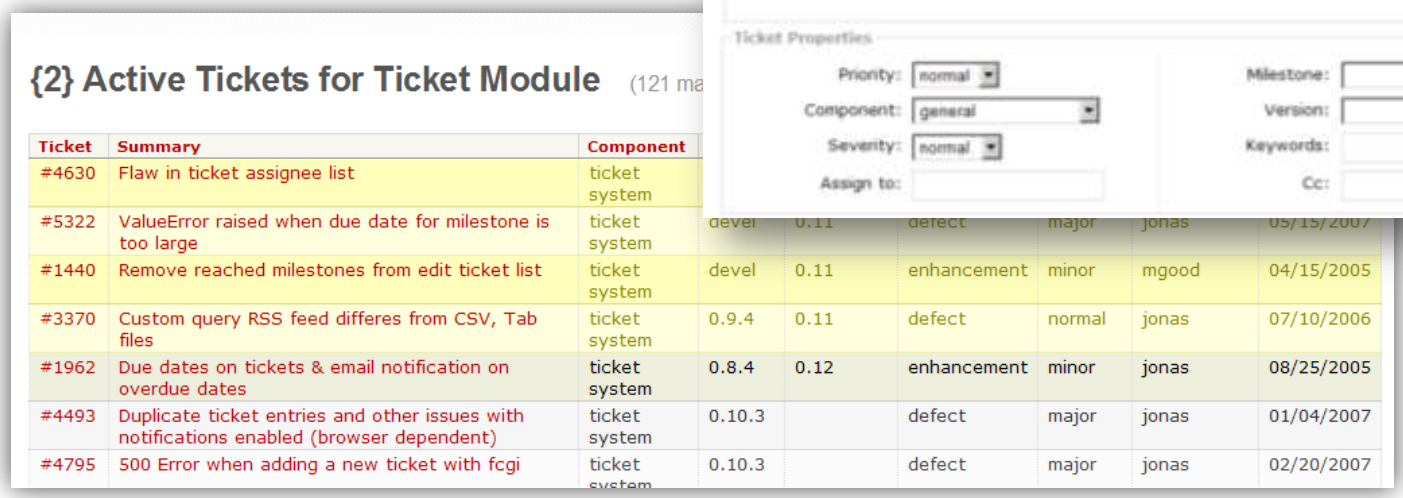
```
[[TicketQuery]]
[[TitleIndex]]
[[RecentChanges]]
[[PageOutline]]
[[Image(photo.jpg)]]
[[MacroList]]
[[TracGuideToc]]
[[Timestamp]]
```

WikiProcessors

```
html
rst
textile
c
cpp
python
perl
ruby
php
asp
sql
xml
```

Trac Komponenten – Issue Tracker

- Zentrales Element in Trac
- Tickettypen können definiert werden (Bug, Task, FR)
- Tickethistorie
- Benachrichtigungen
- Kommentare und Beschreibung nutzen die Wiki-Syntax

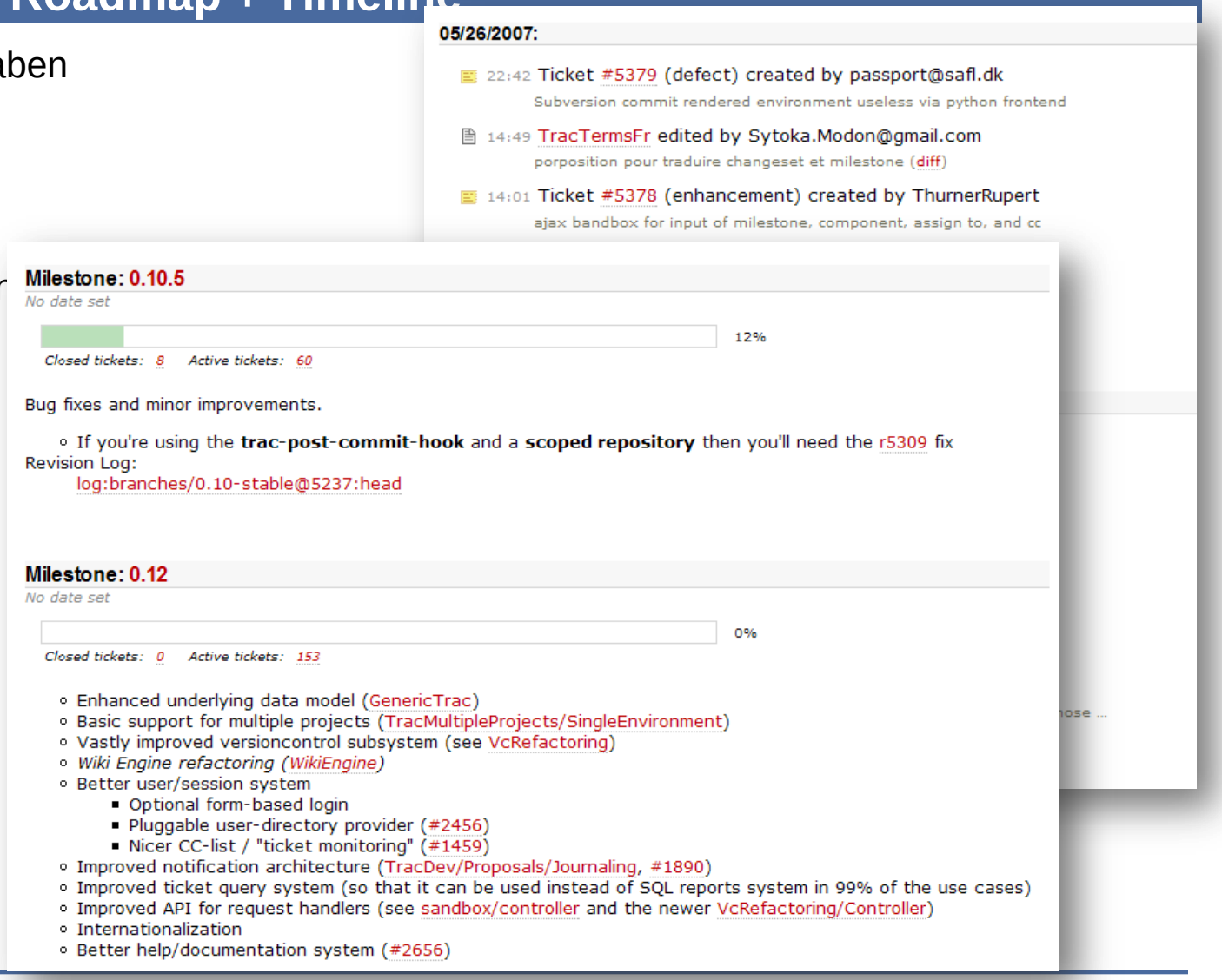


{2} Active Tickets for Ticket Module (121 ma

Ticket	Summary	Component					
#4630	Flaw in ticket assignee list	ticket system					
#5322	ValueError raised when due date for milestone is too large	ticket system	devel	0.11	defect	major	jonas	05/15/2007
#1440	Remove reached milestones from edit ticket list	ticket system	devel	0.11	enhancement	minor	mgood	04/15/2005
#3370	Custom query RSS feed differes from CSV, Tab files	ticket system	0.9.4	0.11	defect	normal	jonas	07/10/2006
#1962	Due dates on tickets & email notification on overdue dates	ticket system	0.8.4	0.12	enhancement	minor	jonas	08/25/2005
#4493	Duplicate ticket entries and other issues with notifications enabled (browser dependent)	ticket system	0.10.3		defect	major	jonas	01/04/2007
#4795	500 Error when adding a new ticket with fcgi	ticket system	0.10.3		defect	major	jonas	02/20/2007

Trac Komponenten – Roadmap + Timeline

- Übersicht über die Aufgaben
- Erfüllungsgrad und Terminübersicht
- Verteilung auf die einzelnen Komponenten
- Prioritätenansicht
- Übersicht der letzten Ereignisse
- Änderungsverfolgung



05/26/2007:

22:42 Ticket #5379 (defect) created by passport@safl.dk
Subversion commit rendered environment useless via python frontend

14:49 TracTermsFr edited by Sytoka.Modon@gmail.com
porposition pour traduire changeset et milestone (diff)

14:01 Ticket #5378 (enhancement) created by ThurnerRupert
ajax bandbox for input of milestone, component, assign to, and cc

Milestone: 0.10.5
No date set

12%

Closed tickets: 8 Active tickets: 60

Bug fixes and minor improvements.

- If you're using the **trac-post-commit-hook** and a **scoped repository** then you'll need the r5309 fix

Revision Log:
log:branches/0.10-stable@5237:head

Milestone: 0.12
No date set

0%

Closed tickets: 0 Active tickets: 153

- Enhanced underlying data model (GenericTrac)
- Basic support for multiple projects (TracMultipleProjects/SingleEnvironment)
- Vastly improved versioncontrol subsystem (see VcRefactoring)
- *Wiki Engine refactoring (WikiEngine)*
- Better user/session system
  - Optional form-based login
  - Pluggable user-directory provider (#2456)
  - Nicer CC-list / "ticket monitoring" (#1459)
- Improved notification architecture (TracDev/Proposals/Journaling, #1890)
- Improved ticket query system (so that it can be used instead of SQL reports system in 99% of the use cases)
- Improved API for request handlers (see sandbox/controller and the newer VcRefactoring/Controller)
- Internationalization
- Better help/documentation system (#2656)

Weitere Tools / Ausblick

- Einsatz von CCNET (mit MSBUILD)
  - CCNET ist ein CI-Tool
  - Check der Commits ins SVN -> Email bei Fehlern
  - Überprüfen der Applikation mit Unit-Tests
  - Erstellen von Applikationen, Anwendungspaketen, Dokumentationen
  - Update von TRAC-Eigenschaften
  - Automatisches Aktualisieren von Umgebungen

Fragen / Wünsche / Diskussionen / Meinungen

?