

	<h2>SVN Analysis</h2>	Fecha: 28/01/2009
Proyecto: TWINS		Autor: MAC
Página: 1 de 15		Revisión: 28/01/2009

## ÍNDICE

1.1.Introduction.....	2
1.1.Object.....	2
1.2.Scope.....	2
1.3.Need of a source version control.....	2
1.4.Previous situation.....	2
2.2.Analysis of SVN.....	4
2.1.MKS.....	4
2.2.Basics of Subversion.....	4
2.3.Comparison between MKS and SVN.....	7
3.3.SVN with no code files.....	9
3.1.Documentation.....	9
3.2.Hardware designs.....	9
4.4.Tools.....	10
4.1.Server.....	10
4.2.Client.....	10
5.5.SVN migration.....	11
5.1.Cvs2svn.....	11
6.6.SVN in numbers.....	12
7.7.Problems of SVN.....	13
8.8.Conclusion.....	14
9.9.Revision register.....	15

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 2 de 15		<b>Revisión:</b> 28/01/2009

# 1. Introduction

## 1.1. Object

The object of the present document is presenting an analysis of the use of the version control system (VCS) application called subversion or SVN. Experiences of its use over the last years will be exposed, as well as suggestions for improvements.

## 1.2. Scope

Inside ZIV there are lots of projects for which the use of an advanced VCS is a must. The tradition was to use MKS as the main source version controlling system.

The main purpose of the TWINS project in ZIV was to apply TWINS's techniques and tools in one project used as a pilot. However, as the use of SVN has been such satisfactory in it, it has been extended to a full line of projects.

## 1.3. Need of a source version control

A source version control is critical for ZIV due to the following reasons:

- Allows a historic following of the development of a product.
- Allows the code sharing between different products simplifying the correction of shared errors.
- Supports the modular design of software solutions.
- Supports the learnign of new team members and the spreading of good practices.

## 1.4. Previous situation

The situation previous to the use of subversion was a mixed scenario in which some projects used MKS as VCS for their source code while other projects just used no VCS. Some designs began to require the use of Linux as a development machine which turned

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 3 de 15		<b>Revisión:</b> 28/01/2009

into a problem due to lack of Linux version of MKS.

Another problem was the increase in the number of users of VCS in ZIV which implied more MKS licenses.

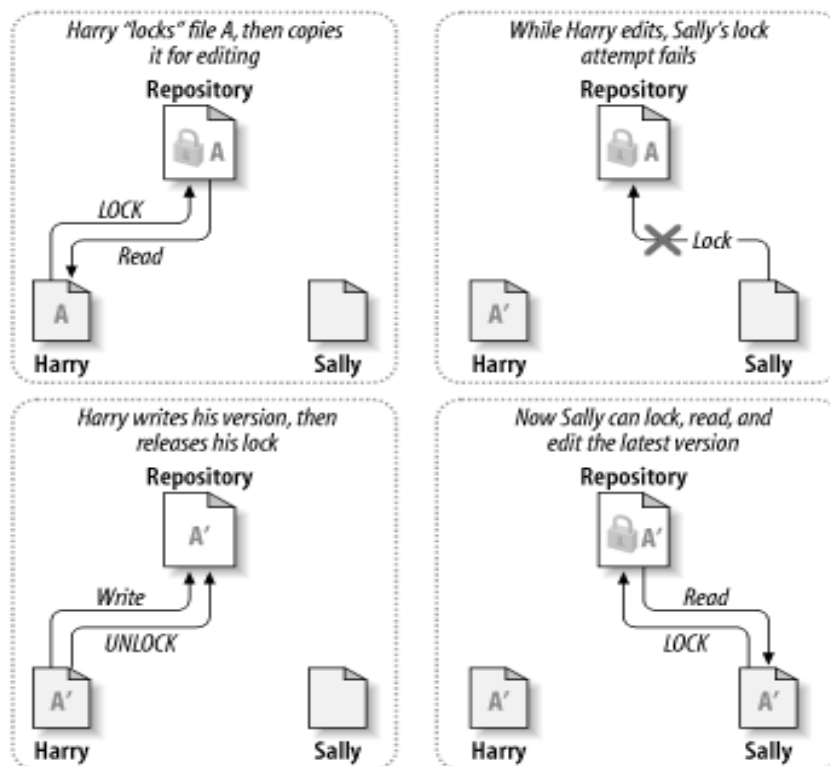
	<b>SVN Analysis</b>	Fecha: 28/01/2009
Proyecto: TWINS		Autor: MAC
Página: 4 de 15		Revisión: 28/01/2009

## 2. Analysis of SVN

### 2.1. MKS

MKS is a comercial tool based on RCS, improving some of its flaws (it works on files and not in projects) as CVS does (free improvement of RCS).

El MKS implements a lock-modify-unlock model.



This implies that a file is locked by a developer until he finishes doing his job. This can be valid for little developments, or not interlaced projects, but not for big ones.

Another problem of MKS is the lack of support for Linux systems.

### 2.2. Basics of Subversion

Subersion (SVN) is an VCS developed by Tigris (subversion.tigris.org), using a license

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 5 de 15		<b>Revisión:</b> 28/01/2009

Apache / BSD.

Its main purpose is being an evolution of CVS solving some of its traditional inconvenients.

SVN uses copy-modify-merge methodology.

SVN is supported in different operative systems (Windows, Linux...)

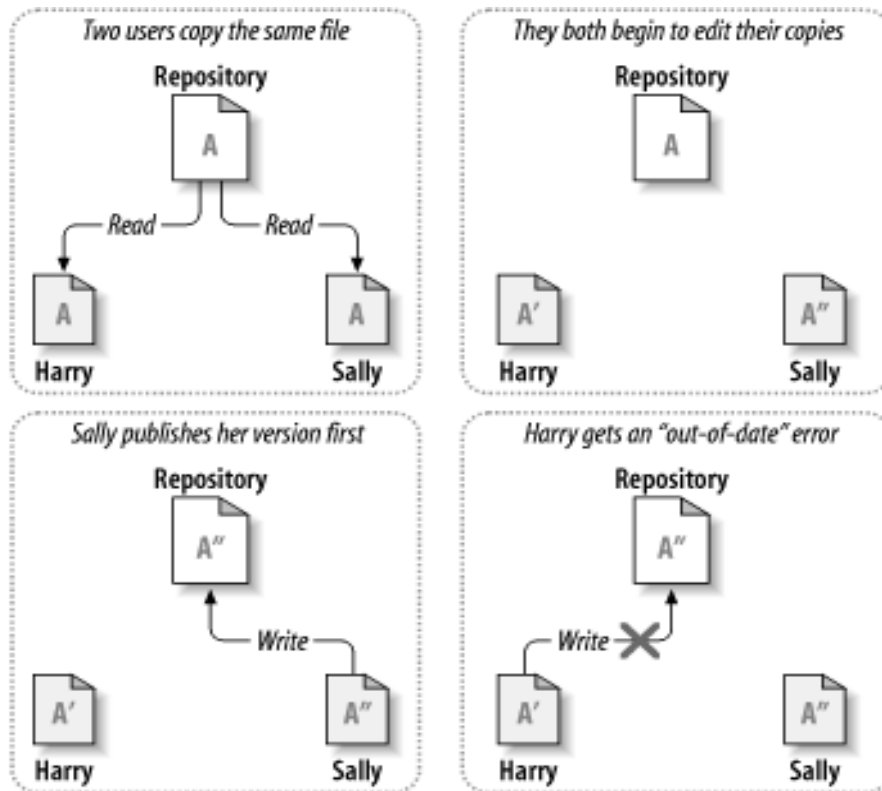
Due to the fact of its license, there are different graphical environments which become an alternative to the raw command line. As an example, TortoiseSVN (evolution of TortoiseCVS), integrates the commands of SVN into the Windows explorer

## 2.2.1 Copy-modify-merge

A copy-modify-merge methodology, simplifies the way in which two different developers access files. This is very important in projects where a developer must work on lots of files at the same time.

Imagine that Harry and Sally want to work on the same file "A". They both take this file, and work on it.

Harry has A' in his repository while Sally has A''.



If Sally commits her changes before Harry, the repository file changes from A to A''.

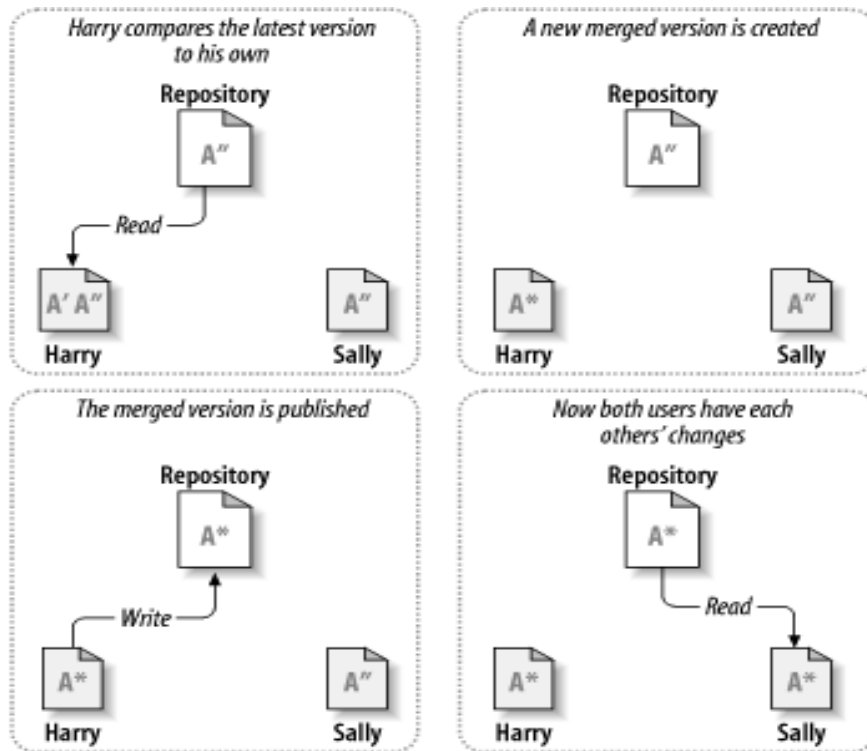
If Harry tries to commit, the SVN knows that he is trying to commit changes from an old file, so commit is not allowed.

Harry is informed of that fact and has to update his version of the file, so he updates his repository, and SVN has to transform changes from A to A' to changes from A'' to A'.

Two different situations can happen:

- Merge: SVN is able to identify the changes and apply them to A'' (this usually happens when two different parts of the file has been modified by Harry and Sally).
- Conflict: SVN cannot merge the different changes and offer Harry to compare the different releases (A, A', A''...) so that he manually resolves the conflict.

	<b>SVN Analysis</b>	Fecha: 28/01/2009
Project: TWINS		Autor: MAC
Página: 7 de 15		Revisión: 28/01/2009



## 2.3. Comparison between MKS and SVN

There are a lot of differences between both systems. We will mention the main ones:

- MKS considers the file as the fundamental element, whereas for SVN, the most important thing is the project.
- Using MKS, the decision on the check point of the whole project is a user decision... SVN stores a differential image of the whole project each time a file is committed (atomic commits) → SVN offers a better control of the whole project.
- Directories are part of SVN (in MKS they are not). That allows a restructuring of the directories in a project without losing the historic information.
- SVN optimizes the use of the bandwidth of the networks, allowing a greater number of offline operations (svn status, svn diff, svn revert).

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 8 de 15		<b>Revisión:</b> 28/01/2009

- This last characteristic enforces the immunity of the system against sporadic bad behaviour of the system.
- SVN splits the functions of status (current situation of the local files), and update (update of the files to their versions in the repository).
- SVN allows to store meta-information of the files, such as file attributes (executable or write flag). This makes easier the permission control in embedded systems.
- Branches and tags are nothing different than other kind of directories for SVN. This offers a great flexibility for their use.
- SVN merging mechanism is very advanced, as it is its conflict detection and resolution.
- SVN has an algorithm of binary differences, allowing a better management of non-text files.
- SVN offers a blame command that allows an easy check of the author (or modifier) and situation of every line in the project.

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 9 de 15		<b>Revisión:</b> 28/01/2009

### 3. SVN with no code files

Traditionally, ZIV has used the VCS just to control the versioning of the source code developed for its projects and products.

During the present project, we decided to include the versioning of two other types of documents:

#### 3.1. Documentation

Due to its text nature, the documentation of the projects seem to be a good candidate to be subversioned. Before SVN, documentation in ZIV was stored in net drives, and its versioning controlled just by revision registers.

With the use of SVN some efforts for using SVN for documentation (mostly in the engineering area) has been done demonstrating its value. However this use is not part of the ZIV procedures (yet).

#### 3.2. Hardware designs

Other different field in which SVN has been tried is the version control of hardware designs.

Until the moment, some netlist have been added to SVN improving the HW-SW codesign and helping in a better coordinate management of the project deliverables.

With the new adoption of Altium as the hardware design tool, we have found that this has an specific connection to SVN. We are trying to use it (with no success yet), and we hope to transform this into a big step into the HW-SW codesign paradigm.

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 10 de 15		<b>Revisión:</b> 28/01/2009

## 4. Tools

SVN (as other VCS systems) is composed of two different roles:

### 4.1. Server

An SVN server is required to store the repositories of the project and control the access to the files.

As we have said, SVN can run on different operating systems. In our case we decided to set an HP Server running a “Debian testing” due to the easy integration of svn (we had a default SVN package in our system).

### 4.2. Client

Each developer needs an SVN client to access the repository and work in their sandboxes. In ZIV we have different kind of users, so we did an analysis of the clients and take the following decisions:

- Use the raw SVN client for Linux users (also included in Debian testing).
- Use TortoiseSVN for Windows XP and Windows Vista
- Use of Cygwin + command line for Windows 98 users (TortoiseSVN does not work for it).

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 11 de 15		<b>Revisión:</b> 28/01/2009

## 5. SVN migration

Migration from MKS to SVN was not a must for this initial implantation of SVN, but it could be very interesting to change all the background information from old projects in MKS.

We have considered three different scenarios for old projects:

- Maintain the use of MKS until they are deprecated.
- Maintain the old information in MKS, but begin to develop using SVN from now on.
- Transform all the info of the MKS repository into an SVN repository using an external tool (cvs2svn?).

### 5.1. Cvs2svn

This tool is supposed to offer different ways to transform a cvs project into an SVN one:

- Just transform the trunk information.
- Selection of the branches to transform.
- Transform all trunk and branches.

However we have not been able to transform an MKS repository. It would be very interesting to develop (in TWINS or other projects) a tool that simplifies this kind of migration.

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Project:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 12 de 15		<b>Revisión:</b> 28/01/2009

## 6. SVN in numbers

The use of SVN has expanded in ZIV from its initial project. The “numbers” of SVN in ZIV are the following:

- Used in more than 20 different projects.
- Used by 18 different developers.
- The biggest project of SVN in ZIV has more than 9200 commits already from 10 different developers.
- 1 unavailability incidence in three years... Availability recovered in 2 hours. 0 lines of code lost.
- Controls code developed in Windows and Linux.
- Controls code developed for more that 5 different embedded architectures.

	<p style="text-align: center;"><b>SVN Analysis</b></p>	<p>Fecha: 28/01/2009</p>
<p>Proyecto: TWINS</p>		<p>Autor: MAC</p>
<p>Página: 13 de 15</p>		<p>Revisión: 28/01/2009</p>

## 7. Problems of SVN

Not everything in SVN is perfect. During these last years we have found the following problems:

No easy migration from MKS (not a problem at the moment), but a good tool for doing that could be interesting.

SVN is not a distributed VCS, so developers that have no access to the main repository (working in the field, at home without VPN...) have to commit different changes in one single operation when returning to connected mode, or working hard to split the changes. Next step in this evolution will be to adopt a distributed VCS that improves this lack. GIT, Bazaar and Mercurial are being considered but there is not a final decision yet.

	<b>SVN Analysis</b>	<b>Fecha:</b> 28/01/2009
<b>Proyecto:</b> TWINS		<b>Autor:</b> MAC
<b>Página:</b> 14 de 15		<b>Revisión:</b> 28/01/2009

## 8. Conclusion

SVN solves some of the problems of MKS.

SVN is easy to use and platform independent.

SVN uses the copy-modify-merge technic, so work in big teams or parallel work can be done without risking the security of the system.

SVN can be integrated partially or totally.

	<b>SVN Analysis</b>	Fecha: 28/01/2009
Proyct: TWINS		Autor: MAC
Página: 15 de 15		Revisión: 28/01/2009

## 9. Revision register

Revision	Modification	Author	Rev.	Aprov.
V1.0	Original version	MAC	MAC	