Системы контроля версий

Определение

- Система контроля версий - программа, специально предназначенная для работы с изменяющимися документами.

Проблемы

- Представим, что Вы разрабатываете проект состоящий из одного небольшого файла. После выпуска первой версии проекта перед Вами встает непростой выбор: необходимо исправлять проблемы о которых сообщают пользователи первой версии и в тоже время разрабатывать что-то новое для второй.

Проблемы

- Даже если надо просто исправлять возникающие проблемы, то велика вероятность, что после какого-либо изменения проект перестает работать и надо определить, что было изменено чтобы было проще локализовать проблему. Также желательно вести какой-то журнал внесенных изменений и исправлений, чтобы не делать несколько раз одну и туже работу.

Большие проблемы

- В простейшем случае вышеприведенную проблему можно решить хранением нескольких копий файлов, например, один для исправления ошибок в первой версии проекта и второй для новых изменений. Но что если проект состоит из нескольких тысяч файлов и над ним работает сотня человек? Если в этом случае использовать метод с хранением отдельных копий файлов (или даже только изменений) то проект застопорится очень быстро.

Возможности

- Хранение версий файлов, причем обычно хранятся только изменения между предыдущей и текущей версией и таким образом хранилище не растет слишком быстро;
- Возможность получить любые предыдущие версии хранимых файлов;
- Просмотр изменений внесенных между заданными в запросе версиями;
- Сохранение и просмотр комментариев и авторов к внесенным изменениям;

Внутренее устройство

- Обычно есть некое хранилище, где-нибудь в доступном всем участникам проекта месте, типа интернета или локальной сети, чтобы версионируемые документы можно было получить с любого компа и с любого компа отправить.

Внутренее устройство

- В этом хранилище хранятся все документы, причём для каждого документа хранится каждая его версия. То есть, если кто-то поменял что-то в документе, создаётся его новая версия, но старая тоже сохраняется, и её при желании можно получить.
-

Внутренее устройство

- Естественно, хранятся не версии документов целиком, а только изменения по отношению к предыдущей версии (это называется дельта-компрессией, а сами изменения - дельтой). Имея исходный документ и набор дельт для всех изменений можно построить любую версию документа.

Рабочая копия

- Для того, чтобы внести изменения в версионируемый документ, его надо сначала получить из хранилища, скопировав куда-то к себе. Такая копия документа называется рабочей копией. Пользователь вносит в документ какие-то изменения и помещает документ обратно в хранилище, где создаётся новая версия для этого документа.

Версионирование

- Операции, связанные с версионированием у пользователя обычно делаются какими-то клиентскими программами.
- В принципе, версионированием может заниматься сама программа, с которой работает пользователь, без явных операций получения/отправки копий документа, например Google Docs создаёт новую версию всякий раз, когда сохраняет изменения в документе, ещё некоторые версии борландовских сред разработки хранили где-то 20 последних версий редактируемых файлов во временных файлах.

Классификация

- Систем контроля версий тысячи.
- Они делятся на два больших класса - централизованные и распределённые.

Централизованные системы

- Централизованные системы имеют единый сервер, который хранит версионируемые документы, и с ним синхронизируются все пользователи.

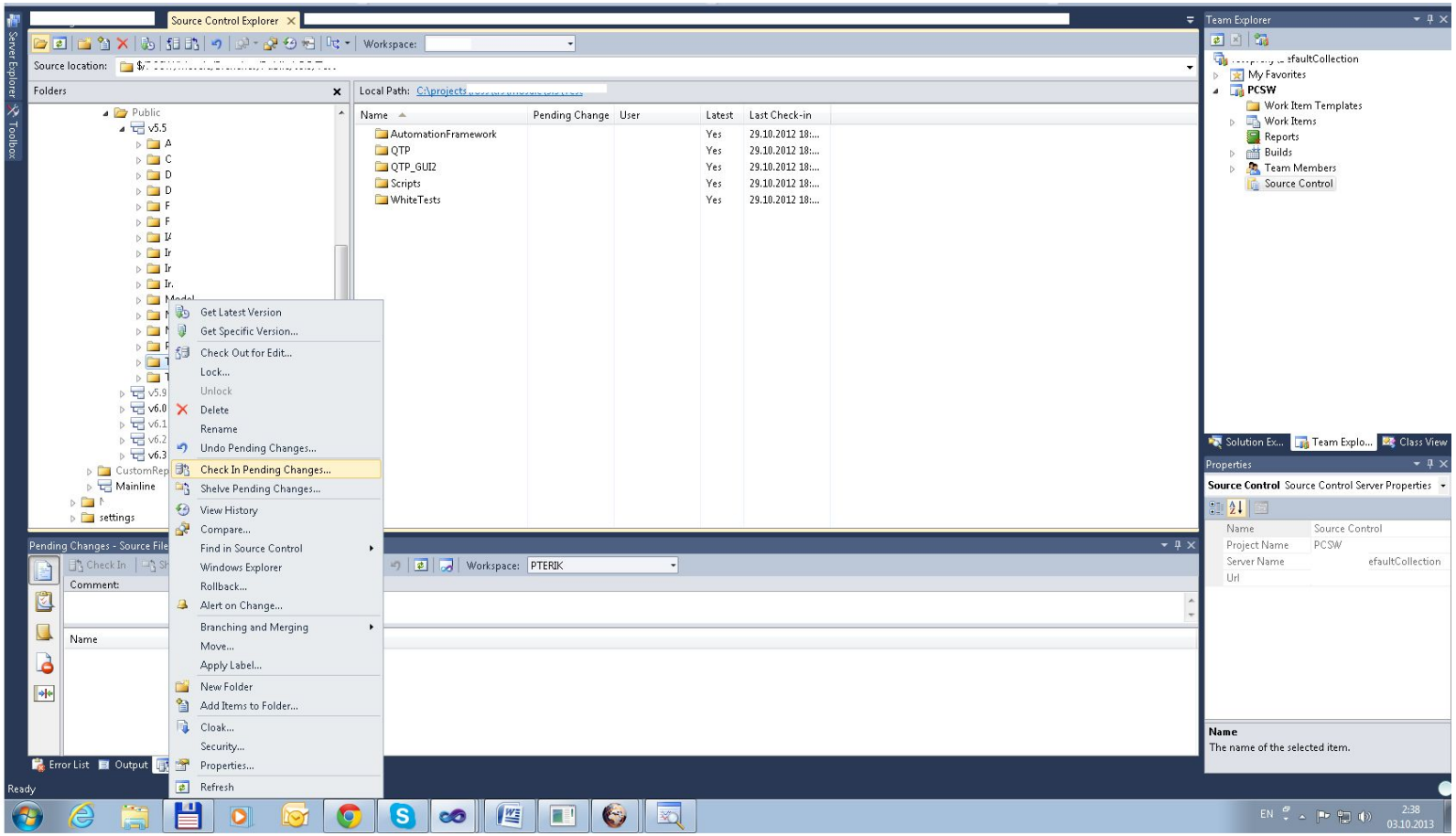
Распределённые системы

- Распределённые системы центрального хранилища не имеют, и пользователи синхронизируются непосредственно друг с другом, рабочая копия может быть полноценным хранилищем. Естественно, она запоминает локальные изменения и может грамотно применить исправления из другого хранилища.

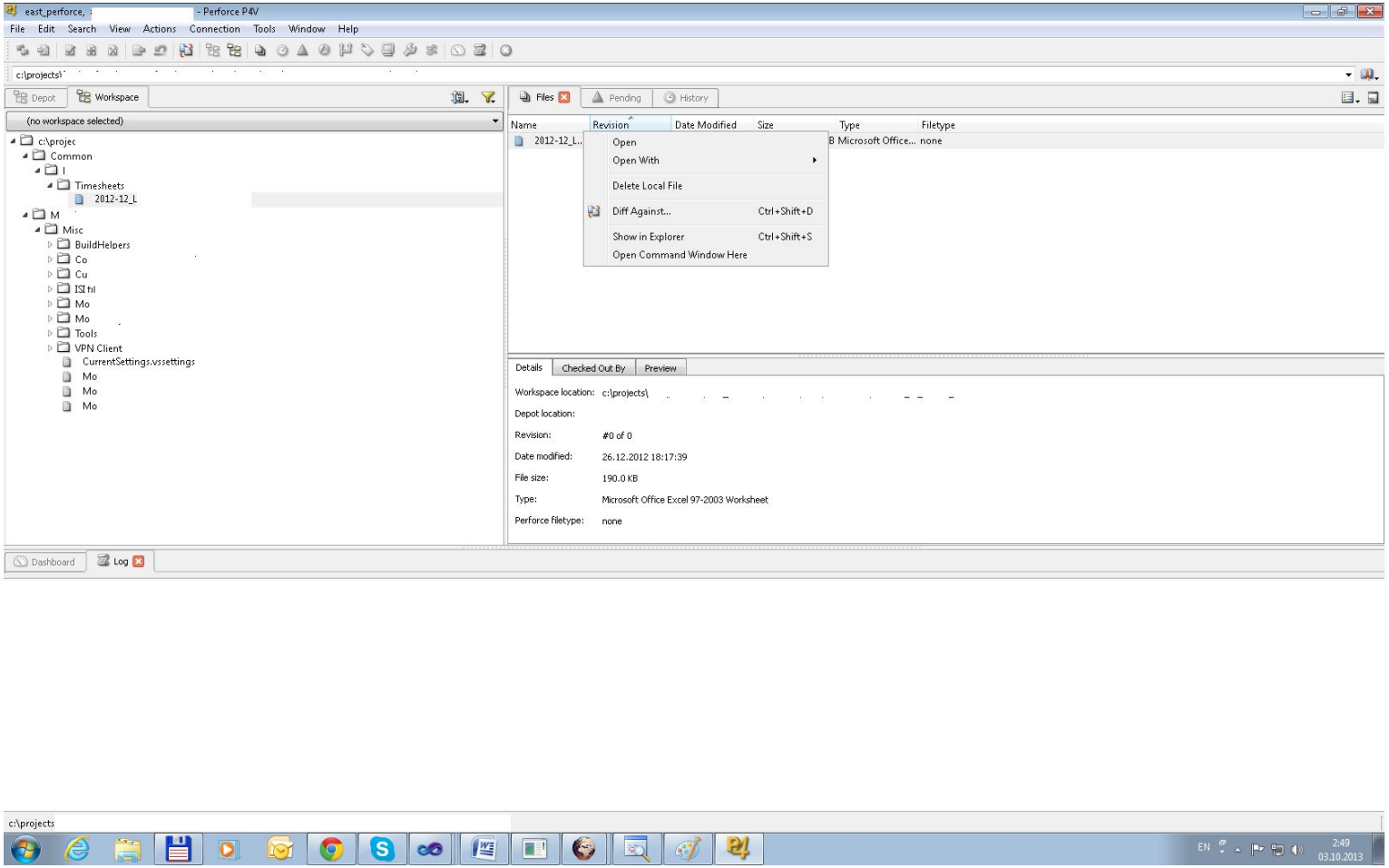
Обзор

- Мы будем пользоваться системой SVN - потому как она на данный момент является самой распространённой, она централизованная. Её потихоньку вытесняют распределённые системы Git и Mercurial.
- Ещё из популярных бывают Visual SourceSafe и TFS, Bazaar, Perforce, CVS.

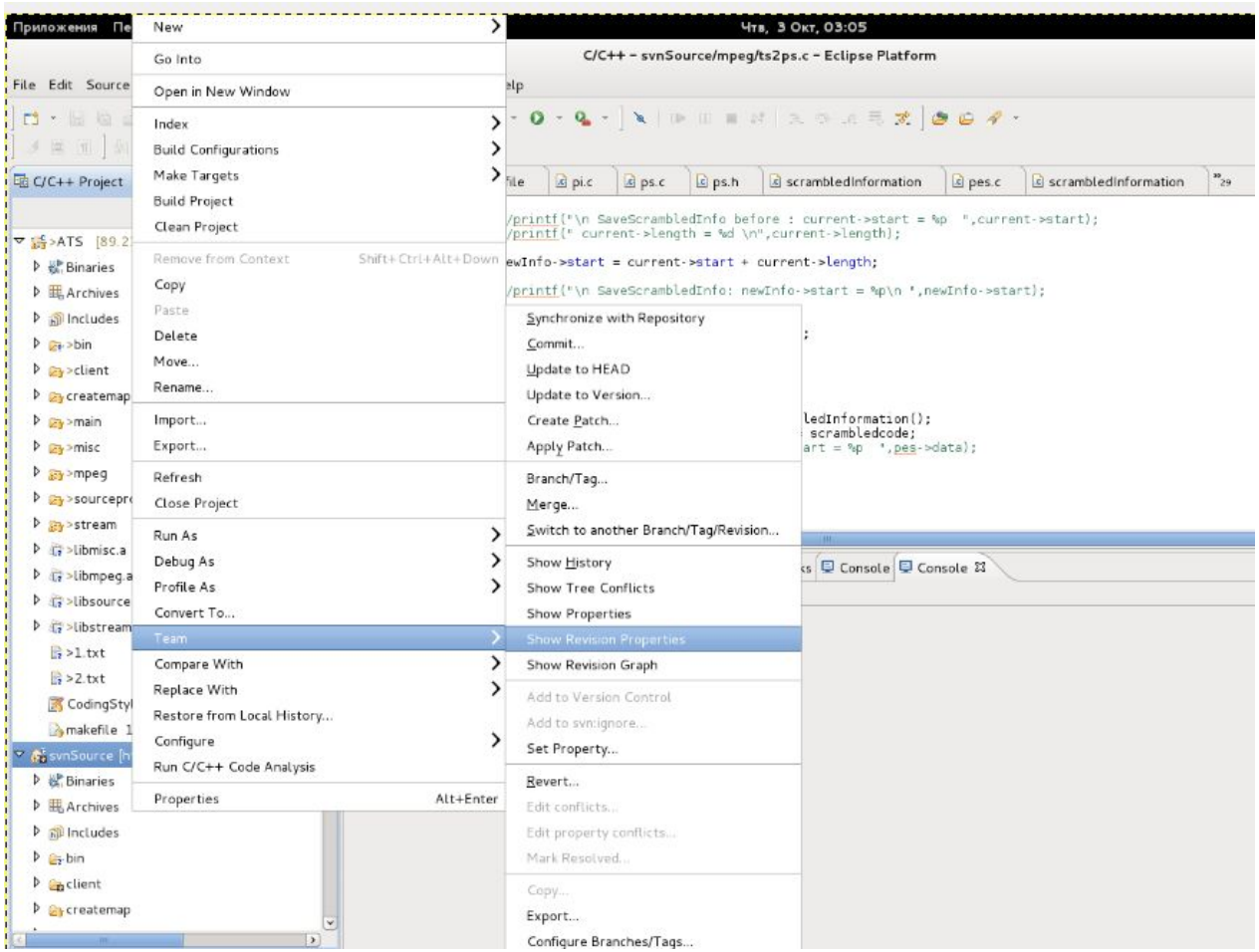
tfs


Source Control Explorer
Workspace:
Source location:
Folders
Local Path:
Name
Pending Change
User
Latest
Last Check-in
AutomationFramework
QTP
QTP_GUI2
Scripts
WhiteTests
Get Latest Version
Get Specific Version...
Check Out for Edit...
Lock...
Unlock
Delete
Rename
Undo Pending Changes...
Check In Pending Changes...
Shelve Pending Changes...
View History
Compare...
Find in Source Control
Windows Explorer
Rollback...
Alert on Change...
Branching and Merging
Move...
Apply Label...
New Folder
Add Items to Folder...
Cloak...
Security...
Properties...
Refresh
Team Explorer
My Favorites
PCSW
Work Item Templates
Work Items
Reports
Builds
Team Members
Source Control
Properties
Source Control Server Properties
Project Name
Server Name
Pending Changes - Source File
Comment:
Error List
Output
Ready

perforce


east_perforce, - Perforce P4V
File Edit Search View Actions Connection Tools Window Help
c:\projects\
Depot Workspace
(no workspace selected)
c:\projec
Common
I
Timesheets
2012-12_I
M
Misc
BuildHelpers
Co
Cu
ISI hl
Mo
Mo
Tools
VPN Client
CurrentSettings.vssettings
Mo
Mo
Mo
Files Pending History
Name Revision Date Modified Size Type Filetype
2012-12_I.. B Microsoft Office... none
Open
Open With
Delete Local File
Diff Against... Ctrl+Shift+D
Show in Explorer Ctrl+Shift+S
Open Command Window Here
Details Checked Out By Preview
Workspace location: c:\projects\
Depot location:
Revision: #0 of 0
Date modified: 26.12.2012 18:17:39
File size: 190.0 KB
Type: Microsoft Office Excel 97-2003 Worksheet
Perforce filetype: none
Dashboard Log
c:\projects
EN
2:49
03.10.2013

Svn,cvs


Приложения
Чтв, 3 Окт, 03:05
C/C++ – svnSource/mpeg/ts2ps.c – Eclipse Platform
File Edit Source
C/C++ Project
New
Go Into
Open in New Window
Index
Build Configurations
Make Targets
Build Project
Clean Project
Remove from Context Shift+Ctrl+Alt+Down
Copy
Paste
Delete
Move...
Rename...
Import...
Export...
Refresh
Close Project
Run As
Debug As
Profile As
Convert To...
Team
Compare With
Replace With
Restore from Local History...
Configure
Run C/C++ Code Analysis
Properties Alt+Enter
Synchronize with Repository
Commit...
Update to HEAD
Update to Version...
Create Patch...
Apply Patch...
Branch/Tag...
Merge...
Switch to another Branch/Tag/Revision...
Show History
Show Tree Conflicts
Show Properties
Show Revision Properties
Show Revision Graph
Add to Version Control
Add to svn:ignore...
Set Property...
Revert...
Edit conflicts...
Edit property conflicts...
Mark Resolved...
Copy...
Export...
Configure Branches/Tags...
Console
Console

Терминология

- Терминология у всех систем разная, причём иногда в одной системе может использоваться разная терминология для одного и того же действия, впрочем, запутаться всё равно невозможно.
- Итак, рассмотрим основные термины:

Репозиторий

- Хранилище версионируемых документов.

revision/version/changeset

- Версия документа или всего репозитория. SVN версионирует весь репозиторий, то есть если вы меняете один или несколько файлов и отправляете изменения на сервер, номер текущей ревизии продвигается для всех документов в репозитории, даже не изменившихся. Собственно, поэтому и changeset - набор изменений.

check-out

- в SVN это создание рабочей копии путём копирования документов с сервера. Можно получить конкретную папку из репозитория.
- Обычно чекаут копирует последнюю ревизию (называемую в свн головной), но можно попросить ревизию с конкретным номером.

- check-in/commit

- Отправка изменений на сервер.
- Вы можете изменять какие-то файлы в рабочей копии, добавлять новые файлы, удалять файлы, добавлять/удалять папки и т. д.
- Кстати, система контроля версий следит только за ВЕРСИОНИРУЕМЫМИ файлами в рабочей копии.

- check-in/commit

- Так, чтобы файл добавился на сервер, недостаточно его просто подложить в папку с рабочей копией, нужно ещё выполнить специальную команду, которая его добавит в версионируемые файлы.
- Удалять файлы так просто тоже не получится - надо сказать системе контроля версий, что мы удалили файл осознанно и хотим его в репозитории тоже удалить. Кстати, удаление из репозитория обратимо, как и любая команда системы контроля версий.

update

- Синхронизация рабочей копии с репозиторием. Тут и начинается самое интересное - представьте, что вы что-то поменяли у себя в своей рабочей копии, а в это время кто-то ещё поменял что-то на сервере.

update

- Если изменения были в разных файлах, то система контроля версий просто скопирует новые файлы поверх старых неизменившихся, а изменённые вами файлы не тронет.

update

- Если изменения были в одном файле, но в разных строчках, система сможет "смерджить" файл - поменять только те строчки, что изменились на сервере, сохранив ваши изменения.

update

- Если изменения были в одной и той же строчке, система не сможет понять, что надо сделать - то ли изменения на сервере свежее/правильнее локальных, то ли наоборот, то ли надо оставить оба изменения, но в любом случае сказать что-то пользователю надо, потому как высока вероятность, что изменилось что-то важное, над чем пользователь работал. Такая ситуация называется конфликтом. Конфликты разрешаются (resolve) ручками.

Конфликты

- Многие системы контроля версий имеют механизмы, позволяющие конфликтов в принципе избежать - вешать замки (lock) на файлы. Это называется блокирующей стратегией версионирования, и её, в частности, любили пользователи VSS.
-

Конфликты

- Когда пользователь начинает вносить какие-то изменения в файл в своей рабочей копии, он говорит об этом системе контроля версий, и она не даёт никому больше вносить в него изменения (нельзя закоммитить залоченный файл). Это хорошо в том смысле, что меньше головной боли с конфликтами, но плохо в том смысле, что более чем одному пользователю работать с файлом нельзя.
- SVN тоже поддерживает такую стратегию, но она обычно не используется.

revert

- Откатить изменения в рабочей копии. Вы что-то делали, поняли, что сделали полную фигню, жмёте revert, у вас получается такая рабочая копия, какая была на момент последнего апдейта, будто вы ничего не делали.

бранчи

- Положим, вам нужно изменить код, и у вас уйдёт неделя, чтобы сделать так, чтобы он хотя бы компилился. Если вы закоммитите свои изменения в некомпилябельном или совсем нерабочем состоянии, ваши товарищи при следующем апдейте получат всё это себе в рабочие копии и не смогут продолжать работать.

бранчи

- Если вы неделю не будете ничего коммитить, то вас уволят. За неделю у вас может слететь винч, и неделя рабочего времени будет мимо (тогда как серьёзные конторы обычно делают резервные копии репозиториев).
- Ещё есть проблема, что товарищи за неделю сделают столько всего, что вы потом ещё неделю будете разгребать конфликты.

бранчи

- Решение этой проблемы - создаёте себе в репозитории бранч. То есть, копируете свой код в отдельную папочку в репозитории и работаете с ним, коммитя туда и апдейтя оттуда.
- При этом ваш код живёт своей жизнью, код товарищей - своей, но вы можете делать апдейты и из ветки товарищей, получая свежие изменения. Потом, когда вы закончите, вы сможете автоматически слить ветки.

- тэги

- Символьные метки определённых ревизий. Например, ими можно пометить крупные релизы, чтобы их можно было легко найти и вернуться к ним.

Свойства

- Некоторые атрибуты файлов и папок, которые хранятся вместе с ними и могут использоваться самой системой контроля версий или тулами, которые с ней интегрируются.
- Например, кодировка файла и тип содержимого - может использоваться веб-фронтендом репозитория для корректного показа содержимого.
- Или флаг svn:ignore, который ставится на папку и говорит, какие файлы в ней не следует даже пытаться версионировать.

Хорошие практики использования систем контроля версий

- - выкладывать туда только исходный код и ресурсы типа картинок, конфигов, может быть тестовые файлы.
- Никогда не выкладывать файлы, которые могут быть сгенерены из тех, что есть в репозитории, в т.ч. .exe, .obj, для джавы - .class.

Хорошие практики использования систем контроля версий

- - структура папок в типичном репозитории SVN такая - для каждого проекта заводятся папки trunk - это главная версия, она всегда должна компилиться, и всегда более-менее работать, именно её надо получать, если хочется собрать какую-то прогу. рядом с ней лежит папка branches, куда, каждая в свою папку, складываются бранчи. Рядом - папка tags, куда складываются тэги. Это рекомендации, изложенные в SVN Book - кстати, довольно хорошей и подробной документации по Subversion.

Хорошие практики использования систем контроля версий

- Бывает так, что некоторыми из этих папок пренебрегают. Например, маленький проект с небольшим ожидаемым количеством версий и одним-двумя разработчиками может вообще бранчей и тэгов не иметь, даже trunk может не заводиться.

Клиенты

- черепаха (http://tortoisesvn.net/downloads.html)
- Слик (http://www.sliksvn.com/en/download)

Наш svn

- https://code.google.com
- Имя проекта

2c2013

- ***URL***

https://2c2013.googlecode.com/svn/

Задачи

1. Завести почтовый ящик gmail.com
2. Прислать его на почту el.al.sm@gmail.com
3. Получить доступ к svn
4. Структурированно выложить в svn все задачи за прошлый год.