آشنایی با Subversion بخش اول

ورژن کنترل به معنای هنر مدیریت تغییرات اطلاعات می‌باشد. بـرنـامه نویسان در حین طراحی یک پروژه اغلب ساعتهای زیادی را صرف ساختن فایل‌های جـدید کرده و پس از ساخت این فایلها تغییرات زیادی را طی روزهای متمادی در تک تک این فایل‌ها اعمال می نمایند. بنـابر این می‌توان گفت که هر فایلی از پروژه از بدو تولد تا زمان بلوغ (تکمیل شدن نهایی) بارها تغییر پیدا می‌کند که حتی در مواردی به خاطر اشتباهات برنامه نویسی، برنـامـه نویس نـاچـار می‌شود از نسخه قبلی و یا حتی چند نسخـه قبـلی یـک فـایـل استفاده کند. در برنامه نویسی کلاسیک ایرانی، برنامه نویس اگر خیلی محتاط باشد قبل از اعمال تغییرات در یک فایل، یک کپی از آنرا در مکانی دیگر ذخیره کرده و تغییرات خود را در فایل اصلی اعمال می‌نماید تـا اگـر در حین برنامه نویسی مشکلی پیش آمد و او نتوانست آن فایل را از آنی که قبلا بوده است بهتر نماید، لااقل نسخه قبلی را کـه در جـایـی ذخیره کرده بود به مکان اصلی برگردانده و در واقع کار او پیامد منفی نداشته باشد. این روش حتی برای برنامه نویس تنـها و تک نفره ما نیز نمی‌تواند چـندان مفید باشد. به عنوان مثال ممکن است او پس از انجام چندین سری تغییرات در طـی روزهـای مختـلف و هـر بار پاک کردن آن فایل پشتیبان کپی شده در مکان دیـگـر، متـوجـه شـود کـه کـلا با اصـل تغییر مخالف بوده و همان فایل اولی که چند روز پیش با آن کـار می‌کـرده را دوباره می‌خـواهد به سیستم برگرداند. تنها مشکل اینجاست که او هر بار پس از انجام تغییرات و اطمینان از صحت فایل، آن فایل پشتیبان را پاک کرده است و به بیان دیـگر او هر بار تنها یک نسخه از آخرین ورژن فایل مورد بحث را در جای دیگر نگاهداری می‌کرده است. بنـابر این توقع بسیار نابجایی است اگر از او بخواهیم که مثلا نسخه ۵۰ روز پیش از یک فایل را به سیستم باز گرداند، به ایـــن دلیل کـه او بـارها ایـــن فـایل را تغییر داده و هر بار برای هر تغییر فایل پشتیبان جدید را جایگزین فایل پشتیبان قبلی کرده است.

اگـر بخـواهیم کمی واقـع بینانه تر به قضیه نگاه کنیم، مساله از اینی که هست هم بسیار مشکل تر خواهد شد! کدام پروژه را میشناسید که تنـها یـک بـرنـامه نـویس داشته باشد؟! یـک پـروژه IT معـمولا از چندین و چند برنامه نویس و گـرافیـست و تهیه کننده خبر و ... تشکیل شده است. در سیستـم بـرنامه نویسی کلاسیک ایرانی تمام این افراد را در اتاقی گرد هم جمع کرده و از آنها می‌خـواهند کـه در مجـاورت و مصاحبت همدیگر کار خود را انجام دهند، در صورتی که قـرار دادن برنامه نویسان و کلا افراد تولید کننده در کنار همدیگر بصورتی که دائم ناچار به صحبت و مداخله در کار یکدیگر باشند بسیار کار اشتباهیست و بـازدهـی را بسیار پایین می‌آورد. وقتـی کـه ما صحبت از ماژولاریتی در برنامه نویسی می‌کنیم، وقتی کـه می‌گوییم هـر قسمت از بـرنامه باید مستقل از قسمت دیگر طراحی شود و بصورت یک جعبه سیاه عمل کند، نباید فراموش کنیم که عین همین مفاهیم نیز در وهله اول باید برای خود ما صادق باشد تا بتوانیم آنها را عملا در زندگی و برنامه خود پیاده کنیم.
ولی آیا واقعا می‌توان بـرنامـه نـویسان را از هـم جدا کرده و از آنها توقع داشت که بدون دخالت در کار یکدیگر کار کنند؟ فرض کنید دو برنامه نویس در آن واحد در حال کار بـــر روی یک قسمت از پروژه باشند، و حـتی بدتر فرض کنید این دو در حـال کـار بر روی یک فایل مشترک باشند. خوب، چه تضمینی وجود دارد که این دو از نتایج کار یکدیگر مطلع شوند قبل از آنکه تغییرات یکدیگر در فـایـل مشترک را از بیـن بـبـرند و یـا اینکه دوباره کاری انجام دهند و برنامه نویس اول کاری را که برنامه نویس دوم همین الان تمام کرده را مجدد انجـــام دهـد؟ او کـه نمی‌دانـد برنامه نویس دوم مثلا خط پنجم تا صدم را اصلاح کرده است، خوب، او ممکن است عین همین کار را دوباره انجام دهد و یـا حتی بدتر ممکن است تغییرات خود را در خـطوط اول تا پنجم جوری اعمال نماید که کل کار برنامه نویس دوم زیر سوال برود، به گونه‌ای کـه مثلا متغیرهایی که برنامه نویس دوم در کار خود استفاده کرده است را، او در خطوط قبل تر پاک نماید!

با تمام این تفاسیر چیزی کـه مشخص است این است که ما به یک سیستم مکانیزه و خودکار نیاز داریم تا بتواند امکانات زیر را برای ما فراهم کند:

- ذخیره تمام نسخه های یک فایل از بدو ایجاد تا هنگام پایان توسعه آن فایل
- قابلیت دادن آمـارها و گزارشات مختلف از یک فایل/شاخه خاص، چون مشخص کردن تمام تغییرات داده شده روی آن فایل/شاخه، نمایش پیغامهای log و نام کاربرانی که این تغییرات را اعمال کرده اند و تاریخ هر تغییر.
- امکان بـرگـرداندن هـر فایل به نسخه‌های قبـلی، بـه صورتی که کاربر بتواند فایل a را مثلا به ۳۰ نسخه قبل تر برگرداند.
- امکان مقـایسه بین ورژنهای مختلف یک فایل و نمایش تغییراتی که فایل طی عبور از نسخه x به نسخه y داشته است.
- جلوگیـری از تصـادم دو نسخه مختلف یک فایل؛ بـصـورتی که اگر دو کاربر در آن واحد مشغول کار بر روی آن فایل باشند، تغییرات کاربر اول موجب از بین رفتن تغییرات کاربر دوم نشود و بالعکس.
- امـکان مطلع کردن کاربران مختلفی که بر روی پروژه مشترکی کار می‌کنند از تغـییری که در یکی از فایلهای پروژه توسط یکی از کاربران به وجود آمده است (به عنوان مثال توسط email، فاکس و ...).
- امنیـت و دقت بسیار بالا، در حـدی کـه کل پروژه در این سیستم ذخیره شود و کاربران با خیال راحت بتوانند با آن کار کنند بدون خطر از میان رفتن فایلها و یا گم شدن فایلی خاص.
- امـکان مشخص کردن مدیران پروژه بصورتی که کاربران پس از معرفی هر تغییر به فایل یا فایلهای سیستم منتظر تایید مدیران بمانند قبـل از اینـکه ایـن تغیـیرات واقعا در سیستم اعمال شود (هر چند که در صورت بروز هر گونه مشکلی ما قطعا می‌توانیم سیستم را به ورژنهای گـذشته برگردانیم و از فایلهای ورژنهای گذشته استفاده کنیم، چون همانطور که در بالا ذکر شد این سیستم باید بتواند در هر لحظه ما را در تونل زمان به عقب و یا جلو ببرد)

حال که به ضرورت کنترل ورژن و سیستمی که بتواند ایـن امـکانات را بـرای مـا بوجود آورد پی بردیم، در قسمت بعد به

معرفـــی روش‌های مختلف کنـترل ورژن و نرم افزارهایی کــه در ایـن زمینه وجود دارند و جدیدترین آنها یعنی subversion می‌پردازیم. لازم به ذکر است که هر چند ما در تمامی مثال‌های بـالا از یک پروژه برنامه نویسـی و برنامه نویسـان تحت آن به عنوان کاربران خود یاد کردیم، ولی ناگفته پیداست کــه این سیستم برای هر پروژه ای که تیمی بیشتر از یک نفر دارد و کار آنها باعث تغییرات منظمی در فایل‌ها می‌شود، می‌تواند مفید باشد. فـایـلهای مـورد بحث می‌توانند فایلهای متنی، عکس، موسیقی، فیلم و یا هـــر فایل دیگری باشنـد. بــه بیان دیگر این سیستم برای تمام توسعه دهندگان مفید است، اعم از موسیقی دانان، تهیه کنندگان فیلم، شیمیدانان و یا هـــر گــروه توسعه دهنده دیگری. دلیـل استفاده از پروژه‌های برنامه نویسـی و بالاخص برنامه نویس به عنوان کاربر در اینجا همانا این حقیقت است که ایــن گــروهها یـکی از فعالترین گروهها در زمینه توسعه و تغییرات منظم و مورچه ای در فایلها می‌باشند.

بیژن هومند