مبانی مدیریت پیکربندی

دفتر بهبود فرایندهای نرم‌افزار
همکاران سیستم

۱. چکیده

در منابع مختلف اعم از کتاب، مقاله، متدولوژی و استانداردهای متنوع توسعه نرم‌افزار، عموماً به موضوع مدیریت پیکربندی [۱] در سطحی انتزاعی؛ بدون ارائه‌ی مثال، و با ادبیاتی فشرده پرداخته شده است. به‌گونه‌ای که تشخیص مصداق‌های آن نیاز به تجربه عملی توسعه نرم‌افزار در تیم‌های کوچک و بزرگ داشته و برای کسانی که به تازگی وارد تیم‌های بزرگ در حرفه تولید نرم‌افزار می‌شوند نیاز به تفسیر و تبیین بیشتری وجود دارد. گاهی مدیریت پیکربندی تا سطح استفاده از یک ابزار خاص (مثل Git و TFS و ClearCase و SVN و غیره) بدون توجه به نیازمندی‌های اصلی کاهش پیدا می‌کند. این مقاله بردا شتی آزاد از منابع مختلف و حا صل تجربه نگارنده در تیم‌های مختلف تو سعه نرم‌افزار ا ست و در آن تلاش شده است تا موضوع مدیریت پیکربندی به زبانی ساده و قابل درک؛ به شکلی توضیح داده شود که تصویری ملموس و کاربردی از مطالب ارائه شده در منابع مختلف به د ست دهد. گفتنی ست مسائل و روش‌های پیکربندی محصولاتی که به صورت غیر متمرکز توسعه می‌یابند یا هم‌زمان با تیم توسعه، توسط مشتری یا شخص ثالث نیز در حال توسعه هستند از محدوده‌ی مقاله خارج شده است. در این مقاله صرفاً به تبیین نیازمندی‌های اصلی دیسیپلین مدیریت پیکربندی در فرآیند توسعه نرم‌افزار در تیم‌های متمرکز با برنامه انتشار یکپارچه، پرداخته شده و از ذکر مسائل و راه‌حل‌هایی که به سادگی و اختصار قابل تشریح نیستند، صرف‌نظر شده است.

۲. مقدمه

دیسیپلین مدیریت پیکربندی یکی از ساز و کارهای موثر برای حفظ هماهنگی تیم‌های توسعه نرم‌افزار است. تولید نرم‌افزار یک فعالیت تیمی است. طول عمر نرم‌افزارها از مدت حضور یک نفر در یک تیم و در یک شرکت بیشتر است. اندازه محصولات و متناسب با آن اندازه تیم‌ها بزرگ و بزرگتر شده است. ترکیب تیم‌ها در طول چرخه حیات نرم‌افزار مداوم تغییر می‌کنند. اطلاعات بین افراد توزیع شده است و مستندسازی جزئی این اطلاعات به گونه‌ای که در زمان لازم به دست افراد مرتبط با آن برسد به سادگی امکان‌پذیر نیست. از این‌رو، کارکردن چند نفر روی یک محصول نیاز به ساز و کارهایی دارد تا تیم را به یک هویت واحد بدل کند به‌گونه‌ای که با وجود تغییر اعضای تیم، این هویت در طول زمان ثابت بماند، خاطراتش از دست نرود و به شکلی قابل بازیابی یا استنتاج باشد که همه افراد اطلاعات لازم را در زمان مناسب در دسترس داشته باشند و با انرژی کمتری هماهنگی افراد حفظ شود. یکی از مهم‌ترین تسهیل‌کننده‌های هماهنگی داخل تیمی و بین تیمی در دیسیپلین مدیریت پیکربندی نهفته است.

در مجموع می‌توان گفت مدیریت پیکربندی مجموعه روش‌ها، ابزارها، توافقات، رویه‌ها و بطور مختصر دیسیپلینی است که به‌عنوان راه‌حلی برای مسائل مرتبط با پیکربندی محصول تدوین و اجرا می‌شود. از آنجا که معمولاً جنبه‌های مختلف مسئله با یکدیگر در تناقض قرار می‌گیرند معمولاً بین راه حل‌های مختلف سبک و سنگین کرده و حتی ممکن است در مقاطع مختلف توسعه پروژه به راه حل‌های مختلف با میزان رسمیت متفاوت برسیم.

وجود سیستم مدیریت پیکربندی به منظور کنترل فرآورده‌های زیادی که توسط افراد زیادی در یک پروژه مشترک تولید می‌شود بسیار ضروری است. این کنترل به جلوگیری از گیج شدن‌های پرهزینه کمک کرده و از ناسازگار نبودن فرآورده‌های حاصل، اطمینان حاصل می‌کند. این ناسازگاری‌ها ممکن است در نتیجه‌ی برخی از انواع مشکلاتی که در ادامه آمده است به‌وجود آید:

- **به‌هنگام سازی‌های هم‌زمان**[1]: وقتی چند نفر به‌طور هم‌زمان روی یک فرآورده (مثلا یک فایل .java یا .cs حاوی کد برنامه یا یک فایل اکسل یا یک فایل .xml یا یک فایل حاوی مدل طراحی) کار کنند ممکن است تغییرات یکدیگر را ازبین ببرند.
- **آگاه سازی محدود**[2]: وقتی چند نفر روی یک محصول کار می‌کنند از تغییرات یکدیگر مطلع نمی‌شوند.
- **تعدّد نسخه‌ها**[3]: ممکن است از یک محصول نسخه‌های مختلفی در تیم توسعه و در محیط مشتری وجود داشته باشد.

در بخش ۳ ابتدا مسئله‌هایی را معرفی می‌کنیم که مدیریت پیکربندی لازم است تا راه‌حلی برای آنها داشته باشد. سپس جنبه‌های مختلف هر مسئله را با راه حل‌های ساده‌ای که آنها را به درستی پوشش نمی‌دهند پررنگ تر کرده و به تدریج همه

[1] Simultaneous Update

[2] Limited Notification

[3] Multiple Versions

جنبه‌های هر م سئله را رو شن می‌کنیم. در بخش ۴ به معرفی کلا سیک سی ستم مدیریت پیکربندی برگرفته از منابع دیگر می‌پردازیم. و در نهایت به جمع‌بندی مطلب خواهیم پرداخت.

۳. مسئله‌ها

در این بخش در قالب مطرح کردن چند مسئله ساده و وارد کردن تدریجی پارامترهای تاثیرگذار در پیچیدگی مسئله؛ سعی خواهیم کرد مفاهیم پایه و دغدغه‌های اصلی مدیریت پیکربندی را با هم مرور کنیم.

۳,۱.مسئله‌ی کار کردن هم زمان چند نفر روی یک نسخه محصول

وقتی چند نفر روی یک پروژه کار می‌کنند، شاید بدترین راه‌حل آن است که هر کس نسخه‌ای از فایل‌ها را روی فایل سیستم خود کپی کند، روی آنها کار کند و بعد در پی آن باشد که تغییرات خود را به دیگران منتقل کرده و تغییرات دیگران را دریافت کند. (برای انجام درخواست‌های تغییر[۵] در محصول نهایی علاوه بر محتوای فایل‌ها ممکن است لازم باشد نام یا مسیر [۶] آنها و ساختار دایرکتوری نیز تغییر کند.) در این روش، نگرانی دائمی از بین رفتن تغییرات، کشف دیرهنگام ناسازگاری تغییرات و افزایش شدید هزینه یکپارچه‌سازی، در کنار فشار زمان انتشار، و مشتریان زیادی که حداقل‌های کیفیت محصول را از یک تیم مهندسی انتظار دارند، یک کار تیمی لذت بخش را به کابوسی برای اعضای کلیدی تیم بدل می‌کند.

اولین راه‌حل برای تبادل فرآورده‌های در حال توسعه، استفاده از یک مخزن مشترک برای نگهداری فرآورده‌های پروژه است. مثلاً همه‌ی فراورده‌ها را روی یک سرور شبکه قرار دهیم و همه افراد تیم تغییرات را روی آن مخزن انجام داده و محصول را از آنجا کامپایل کنند. در این صورت با ذخیره کردن هر فایل، تغییرات به سرعت به دیگران منتقل می‌شود. به این ترتیب اشتباهات هم به سرعت به دیگران منتقل می‌شوند. اما از آنجا که همواره «تغییر»، پیش از «کامپایل»، «ساخت» و «تست» انجام می‌شود و معمولاً روی بیش از یک فرآورده اثر می‌گذارد درنتیجه هر تغییر شما حتی وقتی که خودتان از ناتمام بودن آن اطمینان دارید و هنوز حتی کامپایل نکرده‌اید به دیگران منتقل شده و موجب ایجاد وقفه در کارشان خواهد شد. بنابراین مایل هستید فضای کاری[۷] شما از دیگران مستقل باشد تا تغییرات ناتمام شما به فضای کاری آنها منتقل نشود. به این ترتیب درصورتی که شما مجاز باشید فرآورده‌هایی که تغییر می‌دهید را به‌صورت محلی نگهداری کرده و پس از حصول اطمینان از صحت تغییرات خود، آنها را در مخزن مشترک کپی کنید در این صورت توانسته‌اید از انتشار اشکالات احتمالی در فرآورده‌هایی که هنوز کامپایل و تست نکرده‌اید به فضای کاری دیگران جلوگیری کنید و متقابلاً از تاثیر تغییرات ناتمام دیگران برای مدتی در امان باشید. چرا که شما از نسخه‌های مخزن مشترک در کنار نسخه‌های محلی خودتان استفاده می‌کنید و دیگران هم تغییرات را ابتدا به‌صورت محلی اعمال کرده و سپس در مخزن مشترک کپی می‌کنند.

اما کپی کردن در مخزن مشترک تنها وقتی امکان‌پذیر است که مطمئن باشید از زمانی که فرآورده را از مخزن مشترک برداشته و در فضای کاری محلی خودتان کپی کرده‌اید تاکنون فرد دیگری آن فرآورده را تغییر نداده است. زیرا در این‌صورت با کپی کردن روی مخزن مشترک، تغییرات دیگران از بین خواهد رفت. برای رفع این مشکل:

۱. یک راه‌حل این است که فرآورده‌ها را بین افراد تیم تقسیم کرده و دسترسی کپی هر فرآورده روی مخزن مشترک را تنها به صاحب آن فرآورده بدهید. (یعنی روی مخزن مشترک هیچ دو نفری به یک فرآورده دسترسی نوشتن نداشته باشند) به این ترتیب هر کس موظف است مدیریت فرآورده‌های تحت اختیار خودش را انجام دهد.

۲.ویژگی روش قبل این است که هر کاری روی یک فرآورده مشخص داشته باشیم باید به یک فرد مشخص ارجاع دهیم. در این صورت انجام کارهایی که به‌صورت انفرادی قابل انجام است سربار هماهنگی با دیگران را به دنبال خواهد داشت. همچنین گاهی شاهد ناسازگاری‌های کوتاه مدت در مخزن مشترک خواهیم بود. در تکمیل روش قبل یک روش روان‌تر این است که صاحب فرآورده را به‌صورت پویا تعیین کنید مثلا بگوییم هر کس با checkout کردن یک فرآورده صاحب آن می‌شود (وقتی یک فرآورده به منظور تغییر از مخزن مشترک در فضای کاری محلی قرار بگیرد اصطلاحا checkout شده‌است. کپی کردن نتیجه تغییرات فرآورده‌ها از فضای کاری محلی در مخزن مشترک، checkin نامیده می‌شود). در این صورت مدیر پیکربندی همواره باید فهرستی از اقلام و صاحب آن را نگهداری کند تا بتواند از بروز مشکل به‌هنگام سازی همزمان جلوگیری کند.

روش دیگر آن است که به همه اجازه تغییر همزمان فرآورده‌ها داده شود. در این صورت مثلاً صاحب هر فرآورده از تغییراتی که دیگران به‌صورت محلی در آن فرآورده داده‌اند مطلع شود و همه تغییرات را با هم ادغام کند و نتیجه را در مخزن مشترک کپی کند.

در مجموع می‌توان گفت هر checkin و checkout باید با هماهنگی مدیریت پیکربندی انجام شود. بسته به دیسیپلینی که مستقر کرده‌اید این مدیر ممکن است یک ابزار، یک نقش متمرکز در تیم، نقشی توزیع شده در همه توسعه دهندگان، یا ترکیبی از آنها باشد. به هر حال هر گونه دیسیپلینی که پیشنهاد کنید و با هر ابزار و روشی که آن را پیاده سازی کنید اگر احتمال مطلع نشدن از تغییرات دیگران وجود داشته باشد با مشکل «**آگاه سازی محدود**» روبرو هستید و اگر احتمال ازبین رفتن ناخواسته برخی تغییراتی که افراد مختلف به‌صورت همزمان روی یک فرآورده انجام داده‌اند وجود داشته باشد با مشکل «**به‌هنگام سازی‌های همزمان**» مواجه هستید. و اگر احتمال اشتباه در شناسایی پیکربندی[۸] مجموعه فرآورده هایی که باید تغییر کند وجود دارد با مشکل «تعدّد نسخه‌ها» روبرو هستید.

۳,۱,۱. سطح انزوا

اگر مایل هستید همزمان با کپی کردن دیگران در مخزن مشترک(checkin)، فضای کاری محلی شما هم به‌هنگام شود، می‌توان گفت شما به یک فضای کاری dynamic نیاز دارید. اما اگر مایل هستید به دستور شما بخش‌هایی از فضای کاری محلی‌تان مطابق با آخرین تغییراتی که دیگران checkin کرده‌اند به‌هنگام شود، در واقع فضای کاری snapshot مورد نظر شماست.(در ابزار clearcase به‌جای workspace از اصطلاح view استفاده شده است) انتخاب یکی از این دو روش به دیسیپلین شخصی هریک از اعضای تیم مربوط است. صرف‌نظر از نوع فضای کاری که انتخاب کرده‌اید، پیش از کپی کردن در مخزن مشترک (checkin)، باید از سازگاری تغییرات خودتان با آخرین پیکربندی محصول در مخزن مشترک مطمئن شوید. برای این منظور لازم است که تغییرات دیگران که به تغییرات شما وابسته‌اند را در فضای کاری محلی خود دریافت کنید. (در ابزارهای مختلف اصطلاحات مختلفی برای این کار وجود دارد مثل UpdateView یا GetLatestVersion)

۳,۲.مسئله‌ی کارکردن هم زمان یک نفر روی چند نسخه محصول

ممکن است مشتری در حال استفاده از ویرایش ۱,۰ محصول باشد در حالی که شما در حال توسعه ویرایش ۲,۰ محصول هستید. در این صورت اگر بخواهید مشکلات ویرایش اول را پیش از انتشار ویرایش دوم رفع کرده و تغییرات را بصورت patch یا hotfix منتشر کنید به این معنی است که روی هر دو نسخه بصورت همزمان کار می‌کنید. این موضوع نسخه های داخلی را هم که برای یکپارچه سازی یا تست آماده میکنید شامل میشود.

یک راه حل ساده آن است که ویرایش‌های مختلف را بعنوان محصولات مستقل در نظر بگیرید اما وقتی شما روی چند ویرایش از محصول بصورت همزمان کار می‌کنید، معمولا مایل هستید تغییراتی که در هر یک از ویرایش‌ها می‌دهید به ویرایش نهایی هم منتقل شود لذامثل این است که در مسئله ۳,۱ هر ویرایش را در فضای کاری یکی از افراد تیم نگهداری کرده و تغییرات هر ویرایش را از روی فضای کاری خودش منتشر کنید. همچنین تغییرات هر ویرایش را در مخزن مشترک checkin کنید و نسخه نهایی را از مخزن مشترک منتشر کنید. در این صورت ملاحظات زیر به مسئله افزوده می‌شود:

- از آنجا که ممکن است در هر یک از ویرایش‌ها یک فرآورده به‌طور مستقل تغییر کند در نتیجه باید اجازه تغییر یک فرآورده در هر یک از فضاهای کاری بطور هم‌زمان داده شود.
- به دلیل اینکه یک نفر هستید و طبیعتا مایل هستید روی یک ایستگاه کاری کار کنید باید بتوانید چند فضای کاری را روی یک ایستگاه ایجاد کنید.
- تغییرات همه ویرایش‌ها در مخزن مشترک ادغام می شوند اما باید توجه داشته باشید که به هنگام سازی فضای کاری نسخه‌های منتشر شده با آخرین وضعیت مخزن مشترک بی‌خطر نیست. چرا که شما مایل نیستید کارهایی که در فضای کاری ویرایش ۲,۰ انجام شده و در مخزن مشترک checkin شده است با به‌هنگام سازی فضای کاری ویرایش ۱,۰ به ویرایش ۱,۰ منتقل شود.

ریشه‌ی جدا سازی نسخه‌ها در برنامه‌ریزی انتشار است. اگر لازم نباشد پیش از انتشار نسخه دوم چیزی منتشر شود نگهداری از پیکربندی نسخه اول فایده‌ای در حد تهیه نسخه پشتیبان خواهد داشت. نسخه پشتیبان یک نسخه زنده و قابل تغییر نیست و مزیت آن نیز غیر قابل تغییر بودن آن است.

۳,۳.مسئله‌ی حفظ سوابق تغییرات در مخزن مشترک

پیش از این اشاره کردیم که یکی از اهداف اصلی سیستم مدیریت پیکربندی جلوگیری از گیج شدن‌های پرهزینه است. از طرف دیگر گیج شدن افراد می‌تواند ریشه در این واقعیت داشته باشد که به خاطر آوردن اینکه چه فرآورده‌هایی تغییر کرده است و دلیل تغییرات آنها چه بوده است یا اینکه یک کار مشخص در کدام نسخه‌ی در حال توسعه انجام شده است کار دشواری است. در نتیجه حفظ سوابق تغییرات پیکربندی محصول می‌تواند از گیج شدن جلوگیری کرده تا پیکربندی محصول از کنترل تیم خارج نشود.

۳,۴.مسئله اقلام پیکربندی[1] پایگاه داده

اگر روی نرم‌افزاری کار می‌کنید که برای نگهداری اطلاعات از پایگاه داده استفاده می‌کند، ساختار پایگاه داده و برخی از داده‌های ذخیره‌شده در آن بخشی از پیکربندی پروژه شماست. در نتیجه حفظ سازگاری آنها با کدهای برنامه یکی از دغدغه‌های شما خواهد بود. استفاده از یک پایگاه داده مشترک بین همه اعضای تیم مشکل به‌هنگام سازی هم‌زمان را در پی دارد و استفاده از پایگاه داده مستقل برای هر فرد مشکل آگاه‌سازی محدود را به همراه خواهد داشت.

برگرداندن تغییرات پایگاه داده به سادگی برگرداندن تغییرات در کدهای برنامه نیست. به‌ویژه زمانی که اسکریپت‌هایی برای ارتقاء به نسخه‌های میانی یا نهایی می‌نویسید که علاوه بر ساختار پایگاه داده ممکن است لازم باشد داده‌ها را هم دستکاری کند. در نوشتن این اسکریپت‌ها ممکن است پیش‌فرض‌هایی داشته باشید که در سایر نسخه‌ها و فضاهای کاری دیگران صحیح نباشد. اطمینان از اینکه کارهایی که روی پایگاه داده انجام می‌دهید اولاً در محصول نهایی منتشر خواهد شد و ثانیاً یکپارچگی محصول را تضمین می‌کند؛ پیچیده‌تر از تغییر در سایر کدهای برنامه است. اینکه از پایگاه داده چه چیزی با چه اندازه‌ای به‌عنوان قلم پیکربندی انتخاب شود، یکی دیگر از مصداق‌های پیچیده مدیریت پیکربندی است که باوجود ظاهر پیچیده آن راه‌حل‌های ساده‌ای دارد.

۳,۵.مسائل جایگزین شدن تیم به‌جای فرد

اگر مسائل قبلی به‌جای هر نفر یک تیم را جایگزین کنید (مثلاً مسئله کارکردن هم‌زمان چند «تیم» روی یک نسخه محصول)، آن‌گاه همان مفاهیم قبلی کاربرد خواهند داشت. (توجه کنید که گرچه مفهوم checkin و checkout و فضای کاری محلی و مخزن مشترک و سطح انزوا در هر دو دسته مسئله یکسان است اما ممکن است پیاده‌سازی تیمی این مفاهیم کاملا متفاوت باشد):

۱. هنوز نیاز به یک مخزن مشترک به منظور آن‌که هر یک از تیم‌ها آخرین تغییرات معتبر خود را در آن checkin کنند تا در اختیار سایر تیم‌ها قرار گیرد و نیاز به یک فضای کاری محلی برای تیم جهت نگهداری نسخه‌های checkout شده، وجود دارد.

۲. متناظر با سطح انزوای فضای کاری هر فرد، سطح انزوای یک تیم نیز قابل تعریف است. درنتیجه یک تیم ممکن است متناسب با سطح انزوای مورد نظر خود، از فضای کاری Dynamic یا Snapshot استفاده کند. به این معنی که تیم مایل است از انتشار اشکالات احتمالی در فرآورده‌هایی که هنوز تست نکرده است به فضای کاری سایر تیم‌ها جلوگیری کند و متقابلاً از تاثیر تغییرات ناتمام سایر تیم‌ها برای مدتی در امان باشد.

از طرف دیگر با جایگزین کردن «تیم» به‌جای «فرد» ملاحظات دیگری به مسائل قبلی افزوده می‌شود:

[1] Configuration Item

۳,۵,۱. مسئله حقوق دسترسی

صرف‌نظر از موضوعات امنیتی که ممکن است برای شرکت شما بسیار مهم باشد، تعیین حقوق دسترسی به ابزارها و فرآورده‌ها به منظور جلوگیری از اشتباهات سهوی در یک کار تیمی نیز به همان میزان دارای اهمیت است.

هر فرآورده، توسط یکی از نقش‌های تیم، توسعه می‌یابد. همچنین بیشترین حجم تبادل فرآورده‌ها بین افرادی که نقش یکسانی دارند صورت می‌گیرد. درنتیجه بهتر است حقوق دسترسیِ افراد یک تیم که یک نقش دارند، مشابه باشد. از سویِ دیگر مایلیم هر فرد تنها به فرآورده‌های تیم خود دسترسی داشته باشد. لذا افرادی که یک نقش دارند ولی در دو تیم قرار گرفته‌اند دسترسی‌های متفاوتی دارند. بنابراین در طرح مدیریت پیکربندی لازم است راه‌حلی برای تبادل اطلاعات بین نقش‌های یکسان در دو تیم وجود داشته باشد.

۳,۵,۲. مسئله فضای کاری محلی تیمی

تا اینجا در هر فضای کاری تنها یک نفر تغییر را ایجاد می‌کرد اما وقتی تیم را جایگزین فرد کنیم به فضای کاری خاصی نیاز است تا بتواند امکان کار کردن چند نفر بطور هم‌زمان را فراهم کند. در ادامه، این مسئله به مسائل کوچکتری شکسته می‌شود:

۳,۵,۳. پیاده سازی فضای کاری محلی تیمی

در ابزارهای VersionControl متمرکز، فضای کاری هر تیم می‌تواند به‌صورت یک شاخه[۱۰] DVP منشعب شده از شاخه Main پیاده سازی شود. به این ترتیب در داخل هر تیم، شاخه DVP هر تیم در واقع مخزن مشترک افراد آن تیم است و شاخه Main مخزن مشترک بین تیم‌ها است. checkin کردن از فضای کاری تیم (شاخه DVP) در این مخزن مشترک (شاخه Main)، همان ادغام[۱۱] کردن DVP در Main است. می‌توان چنین تصور کرد که اقلامی که در DVP تغییر کرده‌اند اما هنوز در Main ادغام نشده‌اند در واقع در فضای کاری تیمی checkout هستند.

۳,۵,۳,۱. مسئله حفظ سوابق تغییرات در فضای کاری محلی تیمی

مسئله حفظ سوابق تغییرات روی مخزن مشترک تیم موضوعیت دارد اما برای نسخه‌های محلی هر فرد در فضای کاری انفرادی او سابقه تغییرات فرآورده‌ها اهمیت زیادی ندارد چون کارهای جاری افراد تا حدود زیادی در ذهن باقی می‌مانند. چرا که حافظه افراد متناسب با حجم کار انفرادی و نهایتاً به کمک ابزارهای ابتدایی به‌اندازه کافی قابلیت جلوگیری از گیج شدن فرد را تا پایان هر کار دارد. اما حفظ سوابق تغییرات برای نسخه محلی تیم در فضای کاری تیمی اهمیت دارد. چرا که هر یک از افراد تیم تنها در جریان بخشی از داده‌های پروژه قرار گرفته لذا افراد جزئیات کارهای جاری تیم را زودتر از کارهای انفرادی خود از یاد می‌برند.

در طرح پیکربندی هر گاه شاخه‌ای از شاخه اصلی یا فرعی جدا می‌کنید حتما سیاست‌های حاکم برآن از قبیل طول عمر، نوع تغییرات، زمان، شرایط و جهت ادغام تغییرات بین شاخه‌ها را هم مشخص کرده‌اید توجه کنید فقط شاخه اصلی است که

طول عمر آن از طول عمر محصول شما کمتر نیست لذا باید مطمئن شوید سوابق تغییرات در این شاخه به شکل صحیحی نگهداری می‌شود.

۳,۵,۳,۲. مسئله‌های مرتبط با نوع پیاده‌سازی فضای کاری تیمی

مشابه با ایجاد یک فضای کاری جدید، ایجاد یک شاخه جدید در پیکربندی به معنی نگهداری دو نسخه متفاوت به‌صورت هم‌زمان است. برخلاف فضای کاری، هرچه ایجاد یک شاخه جدید آسان است نگهداری از آن بطور موازی با شاخه قبلی دشوار است. دلیل این تفاوت آن است که:

- **زمان به‌هنگام سازی فضای کاری** انفرادی را خود فرد به تنهایی انتخاب می‌کند و نیازی به هماهنگی با دیگران ندارد. اما زمان به‌هنگام سازی فضای کاری یک تیم به هماهنگی نیاز دارد.
- به همین دلیل هر فرد فضای کاری خود را در زمان‌های کوتاه به‌هنگام می‌کند و با دیگران همگام می‌شود لذا معمولاً تغییرات خود را روی آخرین نسخه‌ها انجام داده و نیازی به ادغام با کار دیگران نخواهد بود. اما تیم به‌دلیل سربارهایی که هماهنگی با دیگران (مثلاً اطلاع از اینکه دیگر هم‌تیمی‌هایش چه تغییراتی داده‌اند و چگونه یکپارچگی این تغییرات با سایر تیم‌ها باید تصحیح شود) به‌همراه دارد، ترجیح می‌دهد فضای کاری خود (شاخه DVP)را به‌هنگام نکند(یعنی تغییرات Main را در DVP ادغام نمی‌کند) به عبارت دیگر تیم ترجیح می‌دهد دوره‌های به‌هنگام سازی فضای کاری تیم بلند مدت‌تر باشد. لذا تغییرات را پیش از آنکه آخرین نسخه‌ها را دریافت کرده باشد انجام می‌دهد و درآینده با ادغام‌های دشوارتری مواجه خواهد بود.
- در ابزارهای مدیریت پیکربندی، از به هنگام نبودن فضای کاری انفرادی (در نتیجه تغییراتی که دیگران در مخزن مشترک checkin کرده‌اند)به سادگی آگاه می‌شوید. اما روش آگاه شدن از به هنگام نبودن فضای کاری تیمی دشوار است.
- در اکثر ابزارهای مدیریت پیکربندی، Checkin کردن تغییرات فضای کاری انفرادی در مخزن مشترک تیم در یک مرحله انجام می‌شود اما checkin کردن تغییراتی که در فضای کاری تیم (شاخه Dvp)انجام شده به مخزن مشترک بین تیم‌ها در چند مرحله انجام می شود. لذا ممکن ا ست برخی از **مراحل** Checkin **کردن تغییرات فضای کاری** تیم در مخزن مشترک فراموش شود.
- ادغام با کار دیگران به صورت محلی و در همان فضای کاری روزمره انجام شده لذا بطور مستمر و غیر مستقیم تست می‌شود. اما تست ادغام در مخزن مشترک تیمی باید در فضای کاری دیگری انجام شود.**تفاوت فضای کاری روزمره با فضای کاری تست** موجب می‌شود که تست‌های غیر مستقیم انجام نشود.
- معمولا در زمانی که خطا/درخواستی در نسخه ۱,۰ گزارش می شود هنوز آن خطا/درخواست در نسخه ۲,۰ هم مصداق دارد. لذا احتمال اینکه در هر یک از دو نسخه یک فرآورده را با رویکردهای مختلف تغییر داده باشیم زیاد است. توجه کنید که این موضوع به تفاوت فضای کاری تیمی و انفرادی مربوط نیست. بلکه ریشه در **کار کردن هم‌زمان روی چند نسخه از محصول** دارد.

۳,۵,۴. مسئله هویت انفرادی تیم

در هر دو دسته مسئله نهایتاً عملیات مرتبط با مدیریت پیکربندی را یک فرد انجام می‌دهد در حالی که در دسته مسائل تیمی وقتی که یک تیم را به‌جای فرد قرار می‌دهیم اطلاعات لازم برای انجام این عملیات را کل تیم در اختیار دارد لذا تمهیداتی برای آنکه کل اطلاعات به یک فرد به نیابت از کل تیم منتقل شود یا بخش‌هایی از عملیات بین افراد توزیع شود و مدیر پیکربندی نقش هماهنگ کننده را ایفا کند، ضرورت خواهد داشت..

۳,۶.مسئله‌ی ساخت محصول

گرچه فعالیت‌های ساخت محصول در بعضی از منابع در رویه‌های دیسیپلین پیاده‌سازی در نظر گرفته می‌شود و به‌طور کامل در دیسیپلین مدیریت پیکربندی قرار ندارد اما به دلیل ارتباط نزدیکی که با طرح پیکربندی محصول دارد و اینکه بستر نامناسب این کار می‌تواند بسیار پرهزینه و گیج‌کننده باشد که درادامه به اختصار به آن خواهیم پرداخت.

تصور کنید مجموعه فرآورده‌های توسعه‌یافته در مخزن مشترک را در اختیار دارید، چگونه آنها را به یک محصول قابل اجرا تبدیل می‌کنید؟ یک روش اولیه این است که با بررسی آنها، تکنولوژی‌های استفاده شده و ابزارهای کامپایل و اجرای آنها را حدس بزنید، امتحان کنید و در نهایت با سعی و خطا موفق شوید مراحل ساخت را تشخیص دهید. برای کسی که از ابتدا در تیم توسعه بوده است این کار ساده تر خواهد بود. اما فرض کنید مثلاً در مخزن پروژه با بخشی روبرو می‌شوید که از تکنولوژی قدیمی‌تر استفاده کرده است. آیا چون این بخش قبل از سایر بخش‌ها توسعه یافته اینچنین است؟ آیا این بخش هنوز هم جزئی از محصول است؟ چگونه مطمئن می‌شوید چیزی از قلم نیافتاده است؟ گاهی ممکن است جزئیات مراحل ساخت چنان پیش پا افتاده باشد که اگر هر روز هم اقدام به ساخت محصول می‌کنید باز هم چیزهایی را فراموش کنید. روش بهتر آن است که به دنبال دستورالعمل ساخت بگردید و مطابق دستورالعمل پیش بروید. این دستورالعمل در نسخه‌های مختلف محصول تغییر می‌کند چگونه نسخه‌های قبلی را می‌سازید؟ (ساخت نسخه‌های قبلی از این جهت اهمیت دارد که ممکن است بخواهید خطای کوچکی را روی آنها رفع کرده و منتشر کنید) لذا بهتر است این دستورالعمل را جزئی از پیکربندی محصول در نظر بگیرید. و آن را هم همراه با توسعه محصول توسعه دهید. اجرای دستی این دستورالعمل امکان بروز خطا دارد است و زبان‌های طبیعی ابهامات زیادی دارند. لذا در دستورالعمل ساخت محصول هرچه گام‌های بیشتری خودکار باشد احتمال بروز اشکالات ساخت کمتر است همچنین نحوه ساخت، به زبان رسمی‌تری مشخص شده و ابهامات کمتری دارد.

گاهی ساخت محصول نیاز به جمع آوری نسخه‌های مختلف فرآورده‌ها از شاخه‌های مختلف پیکربندی محصول دارد در این موارد احتمال گیج شدن بیشتر است. گاهی زمان ساخت محصول زیاد است و تلاش زیادی صرف می‌کند. لذا ممکن است مایل باشید اقلامی که تغییر نکرده‌اند؛ دوباره ساخته نشوند. گاهی گونه‌های مختلفی از محصول وجود دارد و ساخت هر یک از این گونه‌ها ملاحظات خاص خود را دارد. اشتباهاتی که در ساخت محصول رخ می‌دهد در عین سادگی بسیار محتمل و پرهزینه هستند. لذا در دیسیپلینی که مستقر می‌کنید باید از صحت و تمامیت ساخت محصول مطمئن شوید.

ساخت‌های خودکار مکان مناسبی برای اعمال سیاست‌های پیکربندی محصول است. بطور مثال اگر تغییرات نسخه قبل باید در نسخه‌های بعد انجام شده باشد و در حین ساخت خودکار به شواهدی برخورد می‌کنید که نشان‌دهنده عدم ادغام

تغییرات نسخه های قبل در نسخه در حال ساخت است با توقف ساخت می‌توانید از رعایت این سیاست پیکربندی محافظت کنید.

۳,۷.مسئله انتشار محصول

آنچه منتشر می شود باید تست شده باشد. به عبارت بهتر تنها آنچه تست شده قابل انتشار است. لذا پیکربندی انتشار محصول باید همان پیکربندی محصول ساخته شده برای تست باشد. به این ترتیب دو راه حل زیر متصور است:

- همان محصول ساخته شده برای تست، منتشر شود.
- اگر محصولی که تست شده است به منظور انتشار، مجددا ساخته خواهد شد باید مطمئن شوید مخزن ساخت محصول جهت انتشار همان مخزن ساخت محصول جهت تست است. و در هر دو نسخه، فضای کاری ساخت، نمایی بدون تغییر از مخزن است. در این صورت اگر رویه ساخت، خودکار باشد تا حدود زیادی می‌توان اطمینان داشت که نتایج یکسان خواهد بود.

برای مثال انتشار همه یا بخشی از محصول از فضای کاری محلی هر یک از افراد تیم احتمال وجود تاثیرات جانبی کارهای ناتمام محلی بر آنچه منتشر می شود را افزایش می‌دهد. همچنین امکان ردیابی اقلام پیکربندی به محصول منتشر شده را از بین خواهد برد. لذا بهتر است همه بدانند پیکربندی محصول قابل انتشار در یک مخزن مشترک قرار دارد تا از فراموش شدن اعمال تغییرات محلی در مخزن مشترک جلوگیری شود.

یا مثلا اگر تغییراتی را که در شاخه A تست کرده‌اید جهت انتشار باید در شاخه B ادغام کنید. به‌دلیل پیچیدگی‌های ادغام، باید مطمئن شوید پیکربندی شاخه B یک کپی بدون تغییر از شاخه A است. که در این صورت به‌نظر می‌رسد یکی از دو شاخه زاید است.

۳,۸.مسئله‌ی گم شدن یا از کار افتادن ابزارهای جانبی

هدف پروژه، ساخت محصول است. محصول برآورده‌کننده نیاز همه‌ی ذینفعان است. تیم توسعه و تیم پشتیبانی ذینفعانی هستند که معمولاً کاربر ابزارهای جانبی محصول بوده و به این ابزارها از کانال‌های دیگری علاوه بر کانال‌های انتشار محصول دسترسی دارند.

این ابزارها ممکن است در دیسیپلین‌های مختلف توسعه نرم‌افزار به کار گرفته شوند. به‌عنوان مثال ابزارهای بررسی صحت داده‌ها در پشتیبانی، ابزارهای خودکارسازی همه یا بخش‌هایی از ساخت محصول، ابزارهای بررسی کیفیت کدها و ابزارهای مدیریت وابستگی‌ها از جمله ابزارهایی است که کاربر نهایی با آنها سر و کار ندارد. و معمولاً تواتر تغییر و توسعه و استفاده آنها کمتر از محصول اصلی است. و یا توسط ماشین استفاده می‌شوند و افراد به‌طور معمول با آنها سر و کار ندارند تا آنها را به یاد داشته باشند.

گاهی این ابزارها توسط یک نفر و در فضای رایانه محلی خودش توسعه می‌یابد. لذا اگر این ابزارهای جانبی در پیکربندی محصول، و یا در مراحل ساخت و تست محصول در نظر گرفته نشوند احتمالا در زمانی که به آنها نیاز داریم و می‌خواهیم آنها را توسعه دهیم و کارکردهایشان را تغییر دهیم، یا کسی از محل نگهداری کدها و مستندات و حتی فایلهای اجرایی آن

اطلاع ندارد (یعنی گم شـده‌اند) و یا اگر نسـخه ای از آنها وجود دارد یکپارچگی با محصـول را از دسـت داده‌اند. لذا اکیدا توصیه می‌شود ابزارهای جانبی جزئی از محصول تلقی شود و در پیکربندی محصول در نظر گرفته شده و به همراه محصول منتشر شوند.

۳,۹.مسئله‌ی بازگشت به عقب

وقتی و ضعیت مح صول مطابق آنچه برآورد کرده‌اید به پیش نمی‌رود و ضعیت فعلی ممکن ا ست از نظر شما از و ضعیت قبلی بدتر باشد. اما زمان انتشار، مطابق برنامه تقریباً ثابت است. لذا ممکن است ناگزیر به عقب برگردید. این کار به سادگی بازیابی یک نسخه پشتیبان نیست. چرا که هر نسخه پشتیبان ممکن است شامل موارد دیگری که می‌خواهید منتشر کنید نباشد و یا تست نشده باشد. به عبارت دیگر می‌خواهیم فقط بعضی تغییراتی که انجام شده است در محصولی که منتشر می‌شود اعمال نشود و بخواهید فقط وضعیت بخش‌هایی از محصول را به قبل از تغییر برگردانید. دلایل تصمیم برگشت به عقب در دو مسئله زیر به اختصار تشریح شده است:

۳,۹,۱. خطای تخمین

در هر چرخه [۱۲] از فرآیند توسعه‌ی محصول، زمان، کیفیت و محدودهی [۱۳] انتشار از پیش تعیین شده است. تنظیم همه تخمین‌ها با زمان انتشار دشوار است. گاهی تغییراتی را شروع کرده‌اید اما در زمان انتشار هنوز نیمه‌تمام است. گاهی تغییراتی را شروع می‌کنید که از درستی روش خود مطمئن نیستید و ممکن است در پایان کار ناگزیر به عقب برگردید. گاهی تغییراتی را اعمال کرده‌اید؛ اما در فرصت انتشار قادر به تست آن نیستید. بنابراین:

- یا زمان انتشار را جابجا می‌کنید.
- یا محدوده را تغییر می‌دهید (یعنی مایل هستید تغییرات شما با نسخه‌ی در حال انتشار منتشر نشود.)
- یا از کیفیت صرفنظر می‌کنید.
- مایل هستید ترکیبی از راهکارهای فوق را در پیش بگیرید (مثلا یک بخش از محصول را بدون تست منتشر کنید و تغییرات بخش دیگر را به همراه نسخه در حال انتشار منتشر نکنید)

تخمین اشتباه در زمان انتشار، مسیر توسعه، تامین کیفیت، و هر موضوع دیگری که رسیدن به زمان انتشار محصول را با مخاطره روبرو کند، از مصادیق خطای تخمین هستند.

۳,۹,۲. توسعه تدریجی و هم‌زمان

همه اعضای تیم، تغییرات را به‌صورت هم‌زمان در پیکربندی محصول اعمال می‌کنند و بخش‌های مختلف محصول به‌صورت تدریجی کامل می‌شوند. به دلیل انواع خطاهای تخمین همواره در مقطع انتشار بعضی بخش‌ها زودتر تکمیل شده‌اند و بعضی قطعات با تاخیر مواجه هستند. شاید به نظر برسد اولین راه‌حل تصمیم‌گیری در به تاخیر انداختن انتشار است. با اتخاذ این تصمیم دو راه دارید:

- برای منابع آزاد تا مقطع جدید انتشار، انجام تغییرات جدیدی را مجدداً برنامه‌ریزی کنید. در این صورت بعید نیست که در آن مقطع هم کارهای ناتمام جدیدی وجود داشته باشد.
- همه یا بخشی از منابع آزاد را در حالت انتظار نگاه داشته، یا همه یا بخشی دیگر را در فعالیت‌های جانبی درگیر کنید. با توجه به اهمیت بهینه‌سازی به‌کارگیری منابع [۱۴] در دیسیپلین مدیریت پروژه در اغلب سازمان‌ها، این روش قابل قبول نیست. (در بعضی پروژه‌ها و جوامع برنامه‌نویسی غیر انتفاعی که منابع اختصاصی و محدودیت زیادی در زمان انتشار نداشته باشند بهینه‌سازی به‌کارگیری منابع ملاحظات خاصی ندارد.)

تهیه‌ی نسخه پشتیبان اولین راه‌حل است. اما با بازآوری نسخه پشتیبان همه چیز به عقب باز می‌گردد و ممکن است مایل نباشید بخش‌هایی از پیکربندی محصول که مطابق با تخمین‌ها به پیش رفته است، به عقب باز گردد.

راه حل دیگر آن است که با تکمیل هر قطعه از محصول یک کپی از آن را نگهداریم و توسعه بعدی را ادامه دهیم و در مقطع انتشار بخش‌های تکمیل شده‌ای را که کپی کرده بودیم مونتاژ کنیم و محصول را بسازیم. در این صورت برای رفع خطاهای یکپارچه سازی به یک مخزن مشترک دیگر نیاز خواهد بود. چرا که نسخه پشتیبان یک نسخه زنده و قابل تغییر نیست. از طرف دیگر تهیه نسخه پشتیبان زمانی معقول است که مخاطرات فیزیکی تهدیدی برای انتشار محصول باشد. لذا به‌جای تهیه نسخه پشتیبان می‌توانید از مفهوم خط مبنا [۱۵] یا برچسب [۱۶] استفاده کنید. (ایجاد خط مبنا یا الصاق برچسب به معنی نشانه گذاری نسخه‌های مشخص چندین فرآورده با یک نام واحد است. ابزارهای مدیریت پیکربندی همچنین به منظور بازیابی فرآورده‌های برچسب گذاری شده امکاناتی را در اختیار توسعه دهندگان قرار می‌دهند. برای برچسب گذاری در ابزارهای مختلف قواعد و محدودیت‌هایی وجود دارد.)

خطوط مبنایی که روی پیکربندی هر قطعه از محصول ایجاد کرده‌اید، می‌تواند در جدا کردن یک شاخه جدید از پیکربندی محصول بعنوان مخزن مشترک جدید جهت یکپارچه سازی مجدد قطعات کمک نماید. در این مخزن مشترک نسخه خاصی از هر قطعه از محصول انتخاب شده است تا پس از رفع مشکلات یکپارچه‌سازی، تست و منتشر شود.

۴. سیستم مدیریت پیکربندی

حال که با مسائل مطرح در مدیریت پیکربندی تا اندازه‌ای آشنا شدیم، در این بخش به ارائه چند تعریف رسمی از سیستم مدیریت پیکربندی خواهیم پرداخت. در هر بند از این تعاریف، مسائل و مصادیق متعدد دیگری قابل ارائه است که در محدوده و هدف این مقاله قرار نگرفته است.

سیستم مدیریت پیکربندی برای مدیریتِ چندین گونه از محصول در حال توسعه، ردیابی اینکه چه نسخه‌هایی در یک ساخت مشخص استفاده شده است، اجرای ساخت برنامه‌های مستقل یا کل انتشار با توجه به مشخصات تعیین شده نسخه، و اعمال سیاست‌های توسعه خاص محیط مشتری کاربرد دارد. بعضی از مزایایی که به‌طور مستقیم توسط سیستم مدیریت پیکربندی فراهم می‌شود عبارتند از:

- پشتیبانی از روش‌های توسعه
- نگهداری از یکپارچگی محصول

- حصول اطمینان از صحت و تمامیت محصول
- فراهم کردن یک محیط پایدار که در آن محصول توسعه داده شود.
- اعمال محدودیت در تغییر فرآورده‌ها مبتنی بر سیاست‌های پروژه
- فراهم کردن امکان پیگیری اینکه هر فرآورده چرا، چه وقت و توسط چه کسی تغییر کرده است.

بطور کلی دغدغه‌های اصلی «مدیریت پیکربندی و تغییرات[1]» را می‌توان در موارد زیر خلاصه کرد:

- **مدیریت درخوا ست‌های تغییر**[2]: به زیرسـاخت سـازمانی لازم برای ارزیابی هزینه، زمانبندی، و تاثیرات یک تغییر درخوا ست شده روی مح صول موجود ا شاره می‌کند. همچنین به کارهای تیم مرور کننده و کمیته کنترل تغییر[3] ا شاره دارد.
- **مدیریت پیکربندی**: ساختار محصول را توصیف کرده و اقلام پیکربندی تشکیل دهنده آن را شناسایی می‌کند. اقلام پیکربندی به‌عنوان موجودیت های منفرد ن سخه پذیری در فرآیند مدیریت پیکربندی تلقی می شوند. همچنین مدیریت پیکربندی به تعریف پیکربندی‌ها، سـاخت و برچسـب گذاری و جمع آوری فرآورده‌های نسـخه گذاری شـده در داخل مجموعه‌های تشکیل دهنده محصول و نگهداشت ردیابی بین این نسخه‌ها می‌پردازد.
- **ارزیابی وضعیت پیکربندی**[4]**(اندازه گیری**[5]**)**: یعنی تشریح وضعیت محصول بر اساس نوع و تعداد و نرخ و شدت خطاهای کشف و رفع شده در طی توسعه محصول. متریک‌هایی که از این منظر چه بصورت خام و چه از طریق ممیزی استخراج می‌شوند برای تعیین وضعیت تکمیل پروژه مفید است.
- **ردیابی تغییرات**[6]: یعنی بتوانیم تعیین کنیم روی اقلام چه کاری انجام شده است و به چه دلیلی و در چه زمانی انجام شده است. این اطلاعات نقش سابقه و منطق تغییرات را دارند.
- **گزینش نسخه**[7]: هدف از گزینش نسخه اطمینان از این ا ست که نسخه صحیحی از اقلام پیکربندی برای تغییر یا پیاده سازی انتخاب می‌شوند. گزینش نسخه بر پایه‌های مستحکم «شناسایی پیکربندی[8]» بنا شده است.

---

[1] Configuration and Change Management

[2] Change Request Management (CRM)

[3] Change Control Board (CCB)

[4] Configuration Status Accounting

[5] Measurement

[6] Change Tracking

[7] Version Selection

[8] Configuration Identification

- **ساخت محصول نرم‌افزاری**[1]: نیاز به خودکارسازی مراحل کامپایل و تست و بسته بندی محصول جهت توزیع را پوشش می‌دهد.
- **تهیه نسخه پشتیبان:** مخاطرات از دست رفتن هر یک از فرآورده‌ها از قبیل کدهای نوشته شده، نسخه‌های مختلف محصول، درخواستهای تغییر، سوابق تغییرات، نتایج تست، و به طور کلی نتیجه حاصل از تلاش تیم پروژه را کاهش می‌دهد.

۵. جمع بندی

در هر تیم توسعه نرم‌افزار، بسته به ساختار تیمی، نوع و معماری محصولی که توسعه می‌دهد، مقطع زمانی از چرخه حیات محصول که در آن قرار دارد، بخش‌هایی از توسعه محصول که به شخص ثالث برون‌سپاری می‌کند، و متناسب‌سازی‌هایی که در سایت مشتریان انجام می‌شود نیازمندی‌های متفاوتی برای پیکربندی محصول مطرح است که اگر به درستی شناسایی نشوند و یا راه حلی متناسب با هر مسئله وجود نداشته باشد با مشکلاتی روبرو خواهد شد که هزینه‌های کار تیمی را افزایش داده و نهایتاً خروجی تیم کمتر از مجموع خروجی افراد آن خواهد شد. هرچقدر تیم بزرگتر باشد، یکپارچگی بین تیم‌ها اهمیت بیشتری داشته باشد، نسخه‌های متعددی را پشتیبانی کند، قطعات محصول روی پلتفرم‌های مختلف با نسخه‌های متفاوت مستقر شود، تعداد مشتریان بیشتری داشته باشید حساسیت موضوع بیشتر است. برقرار کردن یک سیستم مدیریت پیکربندی متناسب با نیازمندی‌های مطرح در تیم، می‌تواند زیرساختی برای یک کار تیمی منسجم فراهم کند که هزینه هماهنگی‌های داخلی و خارجی تیم را در طول چرخه حیات نرم‌افزار به شدت کاهش دهد.

[1] Software Manufacture

۶. منابع

[1] “Rational Unified Process”, version 2003. IBM Rational Software (2003)

[2] Brad Appleton, “Software Configuration Management Patterns- Effective Teamwork, Practical Integration”, 2002

[3] “Open Unified Process”, 2006, IBM

[4] Philippe Kruchten, “The Rational Unified Process—An Introduction”- 2nd ed, Addison-Wesley-Longman, Reading, MA (2000)

[5] Internal Process Documents, Quality Management Systems Department, System Group 2008

[6] SEI,2002. “Capability Maturity Model® Integration (CMMISM)”,Version 1.1. SEI,CMU/SEI-2002-TR-029

[7] “TickIT: Guide to Software Quality Management System Construction and Certification using ISO 9001:1994” - Issue 1.0, DISC TickIT Office, November. 1995

[8] Microsoft corp., “Microsoft Solution Framework”, v4.0, 2005