فصل ۹ مدیریت پیکربندی نرم افزار

مفاهیم کلیدی (مرتب بر حروف الفبا)

اشیاء پیکربندی ، اقلام پیکربندی نرم افزار ، خطوط مبنا ، شناسایی ، کنترل تغییر ، کنترل دستیابی ، کنترل نسخه ، کنترل همگام سازی ، فرآیند مدیریت پیکربندی نرم‌افزار ، گزارش وضعیت ، وارسی پیکربندی

KEY CONCEPTS

access control , baselines , change control configuration , audit , configuration's , objecds , identification , SCIs , SCM process , status reporting , synchronization , control , version control

نگاه اجمالی

مدیریت پیکربندی نرم‌افزار چیست؟ وقتی‌که نرم‌افزار کامپیوتری را می‌سازید، تغییر رخ می‌دهد. و چون تغییر رخ می‌دهد شما باید آن را به‌خوبی کنترل کنید. مدیریت پیکربندی نرم‌افزار (SCM) عبارت است از مجموعه‌ای از فعالیت‌ها که از طریق شناسایی محصولاتی که احتمال تغییر آنها وجود دارد، ایجاد رابطه بین آنها، تعریف مکانیزم نهایی برای مدیریت نسخه های مختلف این محصولات، کنترل تغییراتی که اعمال می‌شوند، و رسیدگی و گزارش تغییراتی که ایجاد می‌گردند، کنترل می‌شوند.

چه کسی این کار را انجام می‌دهد؟ هر کسی که درگیر فرآیند مهندسی نرم‌افزار است تا حدودی درگیر SCM نیز می‌باشد، اما گاهی اوقات برای کنترل فرآیند SCM موقعیت‌های پشتیبانی ویژه‌ای ایجاد می‌گردند.

چرا حائز اهمیت است؟ اگر شما تغییر را تحت کنترل در نیاورید، تغییر شما را تحت کنترل قرار می‌دهد و این وضعیت اصلا خوب نیست. یک سری تغییرات کنترل نشده می‌توانند یک پروژه نرم‌افزاری را که خوب هم برنامه‌ریزی شده دچار اختلال و آشوب نمایند. به همین دلیل SCM بخش ضروری مدیریت پروژه خوب و مهندسی نرم‌افزاری کامل و قوی است.

مراحل آن کدامند؟ از آن جایی‌که که محصولات، زمانی تولید می‌شوند که نرم‌افزار ساخته می‌شود، هر یک باید به تنهایی مورد شناسایی قرار گیرند.

به محض این‌که این کار انجام می‌شود، می‌توان مکانیزم مربوط به نسخه و کنترل تغییر را ایجاد کرد. برای حصول اطمینان از این‌که با ایجاد تغییرات کیفیت محصول حفظ می‌شود، فرآیند مورد بررسی قرار

می‌گیرد، و برای حصول اطمینان از این‌که افرادی که باید از تغییرات آگاه شوند، آگاه خواهند شد، باید فرآیند به آنها شرح داده شود.

محصول کار چیست؟ طرح مدیریت پیکربندی نرم‌افزار، راهبرد پروژه را برای **SCM** مشخص می‌نماید. به‌علاوه وقتی که به **SCM** اصلی استناد می‌شود فرآیند کنترل تغییر موارد مورد تغییر نرم‌افزاری را، ایجاد نموده و درخواست تغییر مهندسی را گزارش می‌دهد.

چگونه می‌توانم اطمینان پیدا کنم که این کار را درست انجام داده‌ام؟ وقتی که می‌توان هر محصول را شرح داد، دنبال کرد و تحت کنترل درآورد؛ وقتی‌که می‌توان تغییر را پیدا کرد و تجزیه و تحلیل نمود؛ وقتی هر کسی که لازم است درباره یک تغییر اطلاعاتی داشته باشد، از این تغییر آگاه شود ـ در این‌صورت شما کارتان را درست انجام داده‌اید.

وقتی‌که نرم‌افزار کامپیوتر ساخته می‌شود تغییر اجتناب‌ناپذیر است. و تغییر سبب افزایش میزان سر در گمی مهندسین نرم‌افزاری می‌شود که بر روی یک پروژه کار می‌کنند. سردرگمی زمانی حاصل می‌شود که تغییرات قبل از ایجاد، مورد تجزیه و تحلیل قرار نگیرند، قبل از اجرا ثبت نشوند، به کسانی که باید بدانند گزارش نگردد، و یا به روشی که کیفیت را بهبود بخشیده و خطاها را کاهش دهد، کنترل نشود.

بابیچ [BAB86][1] این مسئله را به‌صورت زیر مورد بحث و بررسی قرار دهد:

هنر موزون ساختن توسعه و تکمیل نرم‌افزار بمنظور کاهش سر در گمی، مدیریت پیکربندی[2] نامیده می‌شود. مدیریت پیکربندی عبارت است از هنر شناسایی، سازماندهی و کنترل تغییراتی که بر روی نرم‌افزار ساخته شده توسط تیم برنامه‌ریزی اعمال می‌گردد. هدف از این کار عبارت است از افزایش بهره‌وری از طریق کاهش خطاها.

مدیریت پیکره‌بندی نرم‌افزار[3] (**SCM**) عبارت است از یک فعالیت جامع که در طول فرآیند نرم‌افزاری اجرا می گردد. از آن‌جایی‌که تغییر می‌تواند در هر زمانی رخ دهد، فعالیت‌های **SCA** برای (۱) شناسایی تغییر (۲) کنترل تغییر (۳) حصول اطمینان از این‌که تغییر به‌درستی اجرا می‌شود (۴) گزارش تغییر به کسانی که علاقه‌مند به دانستن آن هستند، انجام می‌گردد.

باید به این نکته توجه نمود که بین پشتیبانی از نرم‌افزار و مدیریت پیکربندی نرم‌افزار تفاوت آشکاری وجود دارد. پشتیبانی عبارت است از مجموعه‌ای از فعالیت‌های مهندسی نرم‌افزار که پس از تحویل نرم‌افزار به مشتری و استفاده از آن به وقوع می‌پیوندد. مدیریت پیکربندی نرم‌افزار عبارت است از مجموعه‌ای از فعالیت‌های پیگیری و کنترل که زمانی شروع می‌شوند که یک پروژه مهندسی نرم‌افزاری آغاز می‌گردد، و تنها زمانی پایان می‌یابند که نرم‌افزار از کار می‌افتد.

1.Babich, W.A.

2.configuration management

3.software configuration management (SCM)

۹-۱ مدیریت پیکربندی نرم افزار (SCM)

خروجـــی فرآیند نرم‌افزاری عبارت است از اطلاعاتی که می‌توان آن را به سه گروه کلی تقسیم نمود: (۱) برنامه‌های کامپیوتری (هم در سطح منبع و هم فرم‌های قابل اجرا)؛ (۲) اسنادی که برنامه‌های کامپیوتری را شرح می‌دهند (که هم افراد فنی و هم کاربران را مورد خطاب قرار می‌دهد) و (۳) داده‌ها (که درون برنامه‌گنجانده شده و یا در خارج از آن قرار دارد). قلم‌هایی که شامل تمام اطلاعات تولید شده تحت‌عنوان بخشی از فرآیند نرم‌افزاری هستند یک پیکربندی نرم‌افزاری[1] نامیده می‌شوند.

نقل قول

هیچ چیز ثابت نیست جز تغییر مراکلوس ۵۰۰ سال قبل از میلاد

در حالی‌که فرآیند نرم‌افزاری پیش می‌رود، تعداد قلم‌های وضعیت نرم‌افزاری (SCIS) به‌سرعت افزایش می‌یابند. ویژگی یک سیستم[2] منشأ یک طرح پروژه نرم‌افزاری[3] و ویژگی نیازمندیهای نرم‌افزاری[4] است. (همچنین اسناد مربوط به سخت‌افزار). از طرف دیگر این‌ها نیز منشأ ایجاد یک سلسله اطلاعات هستند. اگر هر SCI صرفاً منشأ SCIهای دیگر باشد، سر در گمی کمی حاصل خواهد گردید. متأسفانه متغیر دیگری وارد این فرآیند می‌شود که همان "تغییر"[5] است. تغییر می‌تواند در هر زمانی، و به هر دلیلی روی دهد. در واقع اولین قانون مهندسی سیستم [BER80][6] بیان می‌دارد که: مهم نیست که شما در کجای چرخه زندگی سیستم قرار دارید، سیستم تغییر خواهد کرد، و تمایل به تغییر آن در تمام چرخه زندگی ادامه پیدا خواهد کرد.

منشأ این تغییرات چه چیزی است؟ پاسخ به این سؤال مثل خود تغییرات بسیار متفاوت هستند. اما چهار منبع تغییر اساسی وجود دارد:

- شرایط جدید کاری و یا تجاری که تغییر در شرایط محصول و یا قوانین تجاری را دیکته می‌کند.
- نیازهای جدید مشتریان که اصلاحات داده‌هایی را که توسط سیستم‌های اطلاعاتی ایجاد می‌شوند، عملکردی که توسط محصولات عرضه می‌شوند، و یا خدماتی که توسط یک سیستم کامپیوتری ارائه می‌شوند را عرضه می‌نمایند.
- سازماندهی دوباره و یا گسترش/ کاهش تجارت که سبب بروز تغییرات در اولویت پروژه و یا ساختار تیم مهندسی نرم‌افزار می‌شود.

---

1. software configuration
2. System Specification
3. Software Project Plan
4. Software Requirements Specification
5. Change
6. Bersoff, E.H.

... مشکلات مربوط به بودجه و یا برنامه‌ریزی که سبب تعریف دوباره سیستم و یا محصول می‌گردد.

مدیریت وضعیت نرم‌افزار (SCM) عبارت است از مجموعه‌ای از فعالیت‌هایی که برای کنترل تغییر در طول چرخ زندگی نرم‌افزار کامپیوتری به‌وقوع می‌پیوندد.

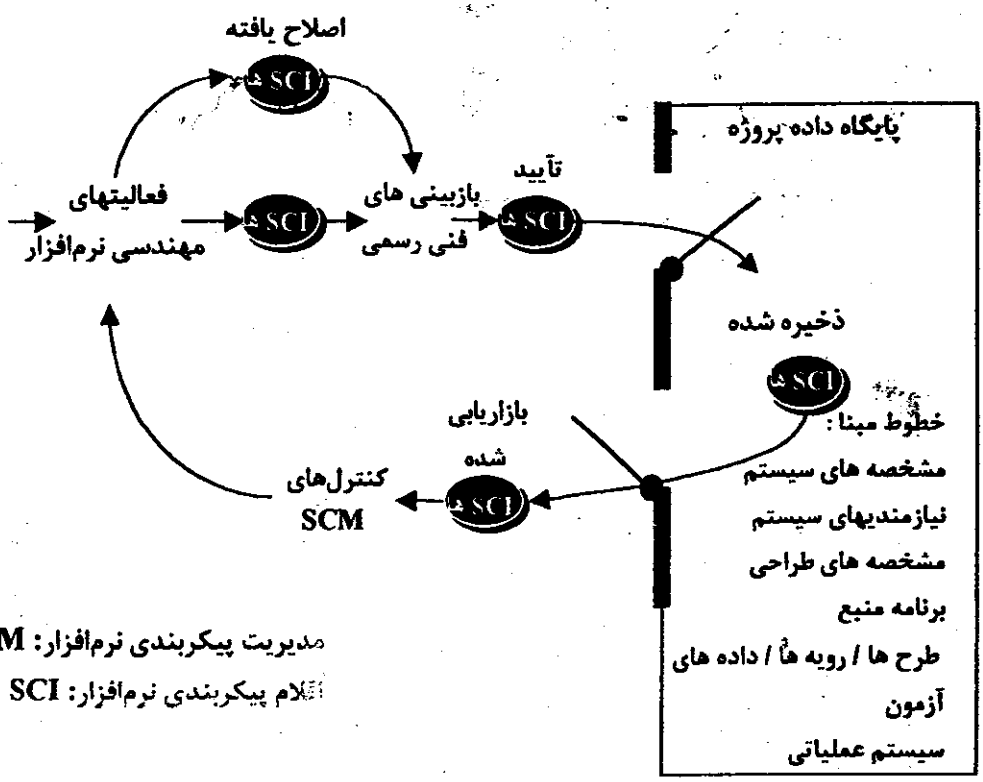
توصیه

بیشتر تغییرات نرم افزاری قابل توجیه اند. به جای آنکه در مقابل تغییرات، زانوی غم به بغل بگیرید، بهتر است از داشتن مکانیزمی برای تشخیص ضرورت و رفع و رجوع آنها اطمینان حاصل کنید.

۹-۱-۱ خطوط مبنا

خط مبنا عبارت است از یک مفهوم مدیریت وضعیت نرم‌افزار که به ما کمک می‌کند تا بدون این‌که توجه جدی به تغییر مجاز داشته باشیم، تغییر را کنترل کنیم. **IEEE (IEEE Std. 610.12-1990)** خط مبنا را به‌شرح زیر تعریف می‌کند:

یک ویژگی و یا محصولی که به‌طور اساسی مورد بررسی قرار گرفته و به تأیید رسیده، و پس از آن به‌عنوان پایه و اساسی برای مراحل تکامل بعدی مورد استفاده قرار خواهد گرفت، و فقط می‌تواند به‌واسطه شیوه‌های اساسی کنترل تغییر، تغییر یابد.



شکل ۹ـ۱ SCI های خط مبنا و پایگاه داده‌های پروژه

یکی از شیوه‌های توصیف خط مبنا از طریق قیاس می‌باشد:

درب‌های ورود به آشپزخانه در یک رستوران بزرگ را در نظر بگیرید. روی یکی از درب‌ها نوشته شده "خروج" (Out) و روی درب دیگر نوشته شده "ورود" (IN). درب‌ها دارای وسایل متوقف‌کننده‌هایی هستند که به آنها امکان می‌دهند فقط در جهت لازم باز شوند.

چنانچه یک پیشخدمت یک سفارش را در آشپزخانه بردارد، آن را در داخل سینی بگذارد و سپس متوجه شود که ظرف را اشتباهی برداشته، خیلی سریع و بدون تشریفات ظرف اشتباهی را سر جایش گذاشته و ظرف درست را برمی‌دارد و از آشپزخانه خارج می‌شود.

اما اگر او آشپزخانه را ترک کند، ظرف را به مشتری بدهد و سپس متوجه بشود که ظرف را اشتباه آورده است، باید مراحل زیر را طی کند: (۱) به فیش نگاه کند تا مطمئن شود آیا اشتباهی رخ داده یا خیر؛ (۲) از مشتری عذرخواهی کند؛ (۳) از طریق درب ورودی وارد آشپزخانه شود؛ (۴) اشتباه را توضیح دهد، و ...

یک محصول کار مهندسی نرم افزار تنها هنگامی قابل توجه است که بازبینی شده و تایید گردد.

یک خط مبنا مانند درب‌های آشپزخانه در رستوران هستند. قبل از این‌که یک قلم وضعیت نرم‌افزاری تبدیل به یک خط مبنا شود تغییر به سرعت و به‌طور غیر رسمی ایجاد می‌گردد. اما به‌محض این‌که یک خط مبنا تثبیت شد، ما به‌طور مجازی از میان یک در دارای حرکت نوسانی، که به یک طرف باز می‌شود عبور می‌نماییم. تغییرات باید ایجاد گردند، اما برای ارزیابی و تعیین هر تغییر باید از یک رویه‌رسمی استفاده نمود.

در بافت مهندسی نرم‌افزار، یک خط مبنا عبارت است از یک کیلومتر شمار در تکمیل و توسعه نرم‌افزار که درجه‌های آن عبارتند از ارائه یک و یا چند قلم پیکربندی نرم‌افزاری و تأیید این SCIهایی که از طریق یک بررسی فنی رسمی حاصل می‌گردند (فصل ۸).

توصیه

اطمینان حاصل نمایید که پایگاه داده های پروژه، به طور متمرکز و در جایگاهی کنترل شده، پشتیبانی می شوند.

به‌طور مثال، عناصر یک ویژگی طراحی[1] مستند شده و مورد بررسی قرار گرفته است. خطاهایی مشاهده شده و اصلاح می‌گردند. به محض این‌که تمام بخش‌های ویژگی مورد بررسی قرار گرفته، اصلاح شده و به تأیید برسد، ویژگی طراحی تبدیل به یک خط مبنا می‌شود. تغییرات بیشتر بر روی شکل برنامه (که در ویژگی طرح مستند شده) را می‌توان فقط پس از این‌که هر یک مورد بررسی قرار گرفته و به تأیید رسید ایجاد نمود، گرچه خطوط مبنا را می‌توان در هر مرحلهٔ جزیی تعریف نمود، معمول‌ترین خط مبنای نرم‌افزاری در شکل ۹-۱ نشان داده شده‌اند.

پیشرفت وقایع که منجر به خط مبنا می‌شود نیز در شکل ۹-۱ نشان داده شده است. کارهای مربوط به مهندسی نرم‌افزار سبب ایجاد یک و یا چند SCI می‌شوند. پس از این‌که SCIها مورد بررسی قرار گرفته و تأیید شدند، در یک پایگاه داده پروژه[2] قرار می‌گیرند (که همچنین کتابخانه پروژه و یا انبار

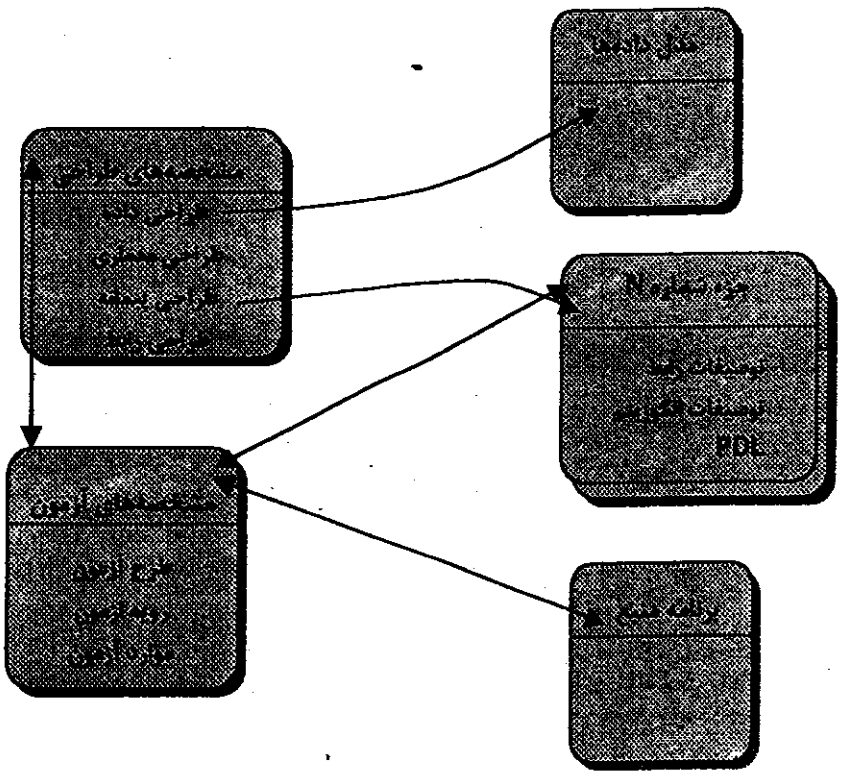
1.Design Specification

2.project database

نرم‌افزار[1] نامیده می‌شود). وقتی‌که یکی از اعضای تیم مهندسی نرم‌افزار می‌خواهد بر روی SCI دارای مبنا، تغییر ایجاد کند این SCI از روی پایگاه داده‌های پروژه بر روی فضای شخصی مهندس کپی می‌شود. اما این SCI استخراج شده را تنها زمانی می‌توان تغییر داد که کنترل SCI ادامه پیدا کند (بعداً در این فصل به آن خواهیم پرداخت). فلش‌های تیره که در شکل ۹-۱ می‌بینید نشان‌دهنده مسیر اصلاح یک SCI دارای خط مبنا هستند.

۹-۱-۲ اقلام پیکربندی نرم افزار

ما قبلاً یک قلم پیکربندی نرم‌افزار را بدین‌صورت تعریف نمودیم: اطلاعاتی که به‌عنوان بخشی از فرآیند مهندسی نرم‌افزار به‌وجود می‌آیند. در نهایت یک SCI را می‌توان بخش جداگانه‌ای از یک ویژگی بزرگ و یا یک مورد آزمون در مجموعه وسیعی از آزمونها در نظر گرفت. در واقع، یک SCI عبارت است از یک سند، مجموعهٔ کاملی از موارد آزمون، و یا جزء معروفی از یک برنامه (به‌طور مثال یک تابع C++ و یا یک بسته نرم‌افزاری Ada).



شکل ۹ـ۲ اشیاء پیکربندی

1:project library

اقلام پیکربندی نرم افزار

در واقع SCIها برای تشکیل اشیاء پیکربندی[1] سازمان‌دهی می‌شوند که می‌توانند در پایگاه داده‌های پروژه با یک نام جداگانه به فهرست درآینده شئ پیکربندی دارای نام و صفت می‌باشد و از طریق یکسری روابط با سایر شی‌ها ارتباط پیدا می‌کند. با توجه به شکل ۹-۲ اجزای پیکربندی، **ویژگی طراحی، مدل داده‌ها، جزء N، برنامهٔ منبع و مشخصات آزمون** هر یک به‌طور جداگانه تعریف می‌شوند. اما همان‌گونه فلش‌ها نشان می‌دهند هر یک از اجزا به دیگر اجزا مرتبط هستند. یک فلش خمیده نشان‌دهنده یک "رابطه ترکیبی"[2] است. یعنی الگوی داده‌ها و جزء N، بخشی از ویژگی طراحی شئ هستند. یک فلش صاف دو سر نشان‌دهنده وجود ارتباط درونی است. چنانچه بر روی جز برنامه اصلی ایجاد گردد این رابطه درونی به مهندس نرم‌افزار امکان می‌دهد تا تعیین نماید چه اجزای دیگری (و چه SCIهایی) ممکن است تحت تأثیر قرار بگیرند.[3]

۹-۲ فرآیند مدیریت پیکربندی نرم افزار

مدیریت پیکربندی نرم‌افزار رکن مهمی از تضمین کیفیت نرم‌افزار است. مسئولیت اولیه آن، کنترل تغییر می‌باشد. اما SCM همچنین مسئول شناسایی SCIهای جداگانه و نسخه های گوناگون نرم‌افزار، رسیدگی به پیکربندی نرم‌افزار برای حصول اطمینان از این‌که به‌طور مناسبی تکمیل شده و گزارش تمام تغییرات اعمال شده به پیکربندی.

ارجاع به وب

مجله زرد (انواع آدرس های مرتبط) مربوط به مدیریت پیکربندی — شامل لیست کاملی از منابع مدیریت پیکربندی نرم افزار در آدرس زیر وجود دارند: www.cs.colorado.edu/users/ander/configuration-management.html

با بحث و بررسی در مورد SCM یکسری سؤالات بسیار پیچیده مطرح می‌شود:

- یک سازمان چگونه می‌تواند نسخه‌های بسیار زیاد موجود یک برنامه (و مستندات آن) را به‌گونه‌ای شناسایی و کنترل نمایند که تغییر به‌طور مؤثری سازگاری پیدا کند.
- یک سازمان چگونه می‌تواند تغییرات را قبل از این‌که نرم‌افزار در اختیار مشتری قرار بگیرد کنترل نماید.
- چه کسی مسئول تأیید و اولویت‌بندی تغییرات است؟
- چگونه ما می‌توانیم اطمینان حاصل کنیم که تغییرات به‌طور مناسبی صورت پذیرفته‌اند؟
- برای ارزیابی تغییرات به‌وجود آمده دیگر، از چه مکانیزمی استفاده می‌شود؟

این سؤالات ما را بر آن می‌دارد که پنج کار SCM را تعریف کنیم یعنی: شناسایی[4]، کنترل نسخه[5]، کنترل تغییر[6]، بررسی پیکربندی[1] و گزارش دادن[2].

1.configuration object

2.compositional Relation

۳. این رابطه ها در پایگاه داده ها تعریف خواهند شد. ساختار پایگاه داده ای پروژه با جزئیاتی قابل توجه، در فصل ۳۱ تشریح گردیده اند.

4.identification

5.version control

6.Change control

۹-۳ شناسایی اشیاء در پیکربندی نرم افزار

برای کنترل و اداره قلم‌های پیکربندی نرم‌افزار، هر یک از آنها باید به‌طور جداگانه نام‌گذاری شده و سپس با استفاده از یک رهیافت "شیءگرا" سازمان‌دهی گردند. دو نوع شئ قابل شناسایی هستند [CHO89][3]: شی‌های پایه[4] و شی‌های مجتمع.[1] یک شئ پایه عبارت است از "واحد متن" که توسط یک مهندس نرم‌افزار و به هنگام تحلیل، طراحی، کددهی و یا آزمون ایجاد می‌شود. به‌طور مثال یک شئ اولیه باید بخشی از یک خصوصیات نیازمندیها و یا برنامه نوشته شده یک جزء و یا مجموعه‌ای از موارد آزمون که برای آزمون برنامه به‌کار می‌رود باشد. یک شئ مجتمع مجموعه‌ای از شی‌های پایه و سایر شی‌های مجتمع است. با توجه به شکل ۹-۲، "ویژگی طراحی" یک شئ مجتمع است. از نظر عقلانی می‌توان آن را به‌عنوان فهرست نام‌گذاری شده (شناسایی شده) اشاره‌گرهایی در نظر گرفت که شی‌های پایه همچون مدل داده‌ها و جزء N را مشخص می‌نمایند.

هر شئ دارای مجموعه‌ای از ویژگی‌های مشخص است که آن را به‌طور منحصربه‌فردی شناسایی می‌نماید: یک نام، یک توصیف، فهرستی از منابع، و یک "تحقق". نام شئ عبارت است از یک رشته کاراکتر که شئ را به‌طور مبهمی شناسایی می‌نماید. توصیف شئ عبارت است از فهرستی از قلم‌های داده‌ها که موارد زیر را شناسایی می‌نماید:

روابط داخلی که بین اشیاء پیکربندی برقرار شده، مهندس نرم افزار را قادر می سازد که آثار تغییرات را مورد ارزیابی قرار دهد.

- نوع SCI (به‌طور مثال سند، برنامه، داده) که به‌وسیله شئ مشخص می‌شود.
- شناسه پروژه
- اطلاعات مربوط به نسخه و / یا تغییر

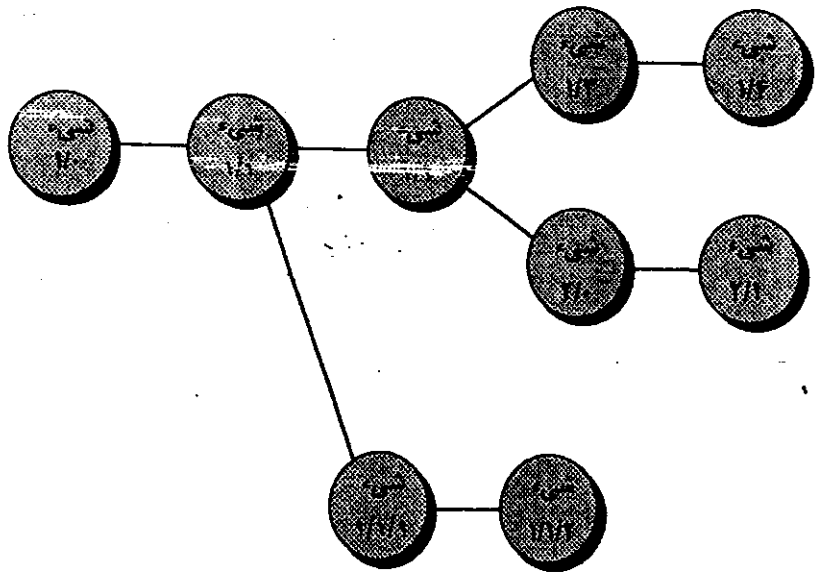
منابع موجودیت‌هایی هستند که ارائه شده، پردازش شده، ارجاع داده شده و یا توسط شئ درخواست می‌گردند.[CHO89] به‌طور مثال انواع داده‌ها، توابع ویژه و یا حتی اسامی گوناگون را می‌توان منابع شئ در نظر گرفت. تحقق، نشانگری است در مورد "واحد متن" یک شئ پایه و بی‌اعتبار در مورد یک شئ مجتمع.

1.configuration auditing

2.reporting

3.Choi, S.C.

4.basic objects



شکل ۹-۳ گراف تکامل

در شناسایی شئ پیکربندی همچنین باید روابطی را در نظر گرفت که در بین اجزای شناخته شده، وجود دارند. یک شئ می‌تواند به‌عنوان «بخشی ـ از» یک شئ مجتمع مشخص گردد. رابطهٔ «بخشی ـ از» یک سلسله شی‌ها را مشخص می‌نماید. به‌طور مثال با استفاده ازعلایم ساده:

نمودار ER ۱/۴ «بخشی ـ از» مدل داده‌ها

مدل داده‌ها «بخشی ـ از» خصوصیات طراحی

ما یک سلسله مراتب از SCIها ایجاد می‌کنیم.

فرض این‌که تنها رابطه بین شی‌ها در سلسله مراتب شی‌ها در طول مسیرهای مستقیم درخت سلسله مراتب هستند غیر واقعی می‌باشد. در بسیاری از موارد، شی‌ها در امتداد انشعابات سلسله مراتب شئ با هم مرتبط هستند. به‌طور مثال یک مدل داده به نمودار جریان داده‌ها (با فرض استفاده از تحلیل ساختاری) و همچنین به مجموعه‌ای از مورد مورد آزمون در خصوص یک کلاس معادل ویژه، مرتبط می‌باشد. این روابط ساختاری متقابل را می‌توان به روش زیر نشان داد:

ارجاع به فصلها

مدل های داده ای و نمودار جریان داده ها در فصل ۱۲ توضیح داده شده اند.

مدل داده‌ها «ارتباط دارد با» مدل جریان داده‌ها:

مدل جریان داده‌ها «ارتباط دارد با» مورد آزمونی کلاس m

در مورد اول رابطه بین شئ مرکب است، در حالی‌که رابطه دوم بین یک شئ مجتمع (مدل داده‌ها) و یک شئ پایه (مورد آزمونی گروه m) می‌باشد.

۱. مفهوم یک شئ متراکم و مرکب [GUS89] در حقیقت بازنمایی نسخه کاملی از یک پیکربندی نرم افزاری محسوب می گردد.

روابط بین شی‌های پیکربندی را می‌توان با استفاده از یک زبان برهم‌بندی پیمانه‌ای[1] (MIL) نشان داد [NAR87][2]. یک MIL وابستگی بین شی‌های پیکربندی را توصیف می‌کند و کمک می‌کند تا هر نوع نسخه ای از یک سیستم به‌طور خودکار ساخته شود.

در طرح شناسایی شی‌های نرم‌افزار باید به این مسئله واقف بود که شی‌ها در طول فرآیند نرم‌افزاری به‌وجود می‌آیند. پیش از آن‌که یک شئ به‌صورت خط مبنا درآید ممکن است چندین بار تغییر کند، و حتی پس از این‌که یک خط مبنا ایجاد گردد ممکن است چندین مرتبه تغییر حاصل گردد. می‌توان برای هر شئ یک نمودار تکاملی ایجاد نمود [GUS89][3]. نمودار تکاملی تاریخچه تغییر شئ را توصیف می‌کند و در شکل ۹-۳ آمده است.

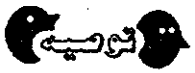
ابزار CASE - SCM

شی پیکره‌بندی ۰-۱ دست‌خوش تغییراتی شده و تبدیل به شئ ۱-۱ می‌شود. در نسخه های ۱-۱-۱ و ۱-۱-۲ که به‌دنبال یک ارتقاء اصلی یعنی شی ۱-۲ می‌آید، اصلاحات و تغییرات جزیی حاصل می‌شود. تکامل شی ۱-۰ از طریق شی ۱-۳ و ۱-۴ صورت می‌گیرد، اما در عین‌حال یک تغییر اساسی در مورد شی سبب به‌وجود آمدن یک مسیر تکاملی جدید می‌شود، یعنی نسخه ۲-۰. در حال حاضر از هر دو نسخه پشتیبانی می‌شود.

این امکان وجود دارد که هر نسخه ای دچار تغییر شود، اما نه الزاماً تمام نسخه ها. سازنده‌چگونه می‌توانند به تمام اجزاء، اسناد، موارد آزمون نسخه ۱-۴ رجوع کند؟ بخش فروش از کجا می‌داند که کدام مشتریان در حال حاضر دارای نسخه ۱-۲ هستند؟ چگونه می‌توانیم اطمینان حاصل کنیم که تغییرات اعمال شده بر روی کد منبع نسخه ۱-۲ در مستندسازی طرح متناظر به‌طور مناسبی منعکس گردیده است؟ عنصر کلیدی در پاسخ به کلیه سؤالات فوق، شناسایی می‌باشد.

به‌منظور کمک در امور شناسایی (و سایر SCMها) ابزارهای SCM خودکار گوناگونی ساخته شده است. در برخی موارد یک ابزار تنها برای تهیه کپی کاملی از جدیدترین نسخه طراحی می‌شود.[TIC82][4]

۹-۴ کنترل نسخه


" شمای " منتخب شما برای اقلام پیکربندی نرم افزار باید با شماره نسخه آن همخوانی داشته باشد.

"کنترل نسخه "[5] روش‌ها و ابزارها را ترکیب می‌نماید تا نسخه های گوناگونی از شی‌های پیکربندی را که طی فرآیند نرم‌افزار ایجاد شده‌اند مهار نمایند. کلم [CLE89][1] کنترل نسخه را در متن SCM به‌شرح زیر توصیف می‌کند:

1 Modul Interconnection Language (MIL)

2 Narayanaswamy, K.andW.

3 Gustavsson, A.

4 Tichy, W.F.

5.version control

مدیریت پیکربندی به یک کاربر امکان می‌دهد تا گزینه‌های پیکربندی گوناگونی از سیستم نرم‌افزاری را، از طریق انتخاب نسخه های مناسب مشخص نماید. این امر از طریق ربط دادن صفات با هر یک از نسخه های نرم‌افزاری و سپس مشخص [و ساخته شدن] یک پیکربندی از طریق توصیف مجموعه‌ای از صفات دلخواه، پشتیبانی می‌گردد.

"صفاتی" که در بالا به آنها اشاره شد می‌توانند به سادگی یک شماره نسخه خاص که به هر شی الحاق شده باشد، و یا به پیچیدگی یک‌سری متغیرهای بولی (سوئیچ‌ها) نشان‌دهنده انواع خاصی از تغییرات کاربردی اعمال شده بر سیستم، باشد. **[LIE89]**[2].

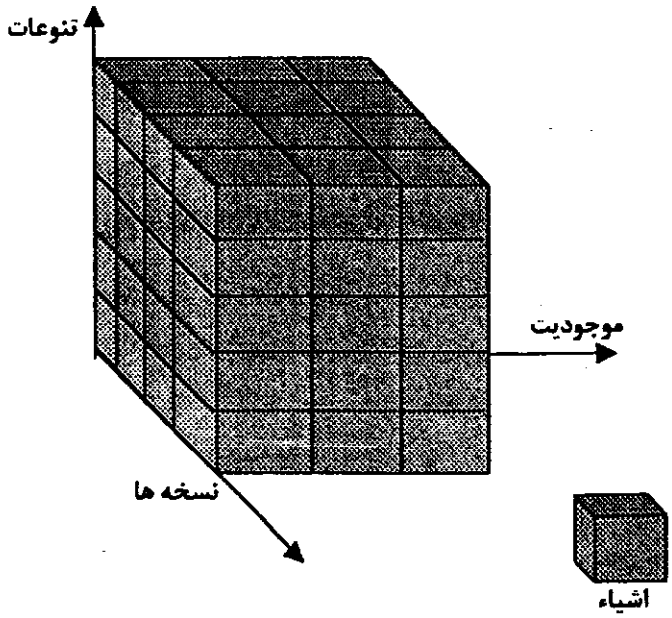
یکی از راه‌های نمایش نسخه های گوناگون یک سیستم نمودار تکاملی است که در شکل ۹-۳ آمده است. هر یک از گره‌های موجود در نمودار، یک شی مجتمع، یعنی نسخه کاملی از نرم‌افزار، می‌باشند. هر نسخه نرم‌افزار عبارت است از مجموعه‌ای از **SCI**ها (که منبع، اسناد، داده‌ها)، و هر نسخه می‌تواند ترکیبی از متغیرهای گوناگون باشد. برای نشان دادن این مسئله، یک نسخه از یک برنامهٔ ساده را در نظر بگیرید که از موجودیت‌های شماره ۱، ۲، ۳، ۴ و ۵ تشکیل شده است.[3] موجودیت[4] شماره ۴ تنها زمانی مورد استفاده قرار می‌گیرد که نرم‌افزار با استفاده از نمایشگرهای رنگی اجرا شود. موجودیت شماره ۵ زمانی مورد استفاده قرار می‌گیرد که نمایشگرهای تک رنگ در اختیار داشته باشیم. بنابراین دو نوع نسخه می‌توان تعریف نمود: (۱) موجودیت‌های ۱، ۲، ۳ و ۴؛ (۲) موجودیت‌های ۱، ۲، ۳ و ۵.

1.Clemm, G.M.

2.Lie, A.

۳. در اولین قدم واژه موجودیت یا نهاد یا هویت، به تمام اشیاء ترکیبی و اشیاء ساده که در قلم پیکربندی نرم افزار می تواند وجود داشته باشد، اطلاق گردیده است. برای مثال، یک موجودیت " ورودی " ممکن است از شش جزء متفاوت نرم افزاری ساخته شده باشد که هر کدام مسئول کارکردی فرعی از ورودی خواهند بود.

4.Entity



شکل ۹ـ۴ مخزن شیء ، بازنمای اشیاء ، تنوعات و نسخه [REI89]

برای ایجاد متغیر[1] مناسبی از یک نسخه مشخصی از یک برنامه، هر موجودیت را می‌توان یک "مجموعه‌ای از ویژگی‌ها" در نظر گرفت ـ فهرستی از ویژگی‌هایی که مشخص می‌کنند زمانی که یک نسخه خاصی از نرم‌افزار ساخته می‌شود آیا باید از موجودیت استفاده کرد یا خیر. برای هر متغیری یک یا چند صفت در نظر گرفته می‌شود. به‌طور مثال، برای مشخص کردن این‌که زمانی که نمایشگرهای رنگی مورد استفاده قرار می‌گیرند از چه موجودیتی باید استفاده کرد، باید از یک صفت رنگی استفاده نمود.

نقل قول

هرتغییر ولو در جهت بهبود بازتابی منفی و حاکی از عدم رضایت دارد. آرنولد بنت

روش دیگر مفهومی نمودن رابطه بین موجودیت‌ها، متغیرها و نسخه ها (اصلاحات) عبارت است از ارائه آنها به‌عنوان یک مخزن اشیاء[2] [REI89][3]. با مراجعه به شکل ۹ـ۴ می‌توان رابطه بین شی‌های پیکربندی و موجودیت‌ها، متغیرها و نسخه ها را به‌عنوان یک فضای سه‌بعدی نشان داد. یک موجودیت تشکیل شده است از مجموعه‌ای از شی‌ها که نسخه تجدیدنظر شدهٔ آنها در یک سطح قرار دارند. یک متغیر مجموعهٔ متفاوتی از شی‌هاست که نسخهٔ تجدیدنظر شده آنها در یک سطح است و بنابراین در موازات سایر متغیرها قرار دارد. وقتی‌که تغییرات عمده‌ای بر روی یک یا چند شی صورت می‌پذیرد، نسخه جدیدی تعریف می‌شود.

طی چند دههٔ گذشته چندین روش خودکار گوناگون برای کنترل نسخه پیشنهاد و ارائه شده است. تفاوت اصلی این شیوه‌ها در پیچیدگی صفاتی است که برای ساخت نسخه ها و متغیرهای خاصی از یک سیستم و روش کار فرآیند ساخت به‌کار می‌رود.

---

1. Variant
2. object pool
3. Reichenberger, C.

۹-۵ کنترل تغییرات

واقعیت "کنترل تغییر"[1] در یک بافت مهندسی نرم‌افزار مدرن توسط "جیمز باچ"[BAC98][2] به‌طور زیبایی جمع‌بندی شده است:

کنترل تغییر بسیار مهم است. اما نیروهایی که این کار را ضروری می‌نمایند، این کار را آزارنده نیز می‌کنند، ما در مورد تغییر نگران هستیم زیرا یک اختلال کوچک در کد می‌تواند اشکال بزرگی در محصول ایجاد نماید. اما از طرفی می‌تواند باعث یک شکست بزرگ شده و یا باعث‌به‌وجود آمدن قابلیت‌های جدید شگفت‌آوری گردد. ما در مورد تغییر نگران هستیم زیرا تنها یک اشکال ایجاد کننده می‌تواند سبب سقوط و نابودی پروژه گردد. اما ایده‌های درخشانی به ذهن این افراد خطور می‌کند و یک فرآیند کنترل تغییر مهم می‌تواند آنها را از انجام کارهای خلاق دلسرد کند.

نقل قول

هنر پیشرفت آن است که درخواست ها را در میان تغییرات و تغییرات را در میان درخواستها حفظ نماید.
آلفرد نوبث وایت هد

"جیمز باچ" اذعان می‌دارد که ما با یک عمل موازنه مواجه هستیم. وقتی‌که کنترل خیلی زیاد است ما مشکل ایجاد می‌کنیم، وقتی کنترل تغییر خیلی ناچیز است، باز هم مشکلات دیگری ایجاد می‌کنیم.

در یک پروژه مهندسی نرم‌افزاری بزرگ، تغییر کنترل نشده خیلی سریع سبب به‌وجود آمدن اختلال و آشوب می‌شود. در یک چنین پروژه‌هایی، کنترل تغییر، روش‌های انسانی و ابزارهای خودکار را با یکدیگر تلفیق می‌نماید تا مکانیزمی برای کنترل تغییر ارائه نماید. طرح فرآیند کنترل تغییر را در شکل ۹-۵ می‌بینید. یک درخواست تغییر[3] برای ارزیابی قابلیت فنی، تأثیرات جانبی بالقوه، تأثیر کلی بر روی سایر شی‌های پیکربندی و عملکرد سیستم، و هزینهٔ پیش‌بینی شدهٔ تغییر ارائه و ارزیابی می‌گردد. نتایج ارزیابی به‌صورت یک گزارش تغییر[4] ارائه می‌گردد که توسط یک مجوز کنترل تغییر[5] (CCA) مورد استفاده قرار می‌گیرد ـ توسط یک فرد و یا یک گروه که تصمیم نهایی در مورد وضعیت و فاصله اولویت تغییر را اتخاذ می‌کنند. برای هر تغییر به تأیید رسیده، یک ترتیب تغییر مهندسی[1] (ECO) ایجاد می‌شود. ECO تغییری را که قرار است به‌وجود آید توصیف می‌نماید؛ محدودیت‌هایی که باید اعمال شوند، و معیارهایی برای مطالعه و بررسی، شی که باید تغییر کند از پایگاه داده‌های پروژه بیرون کشیده شده، تغییر اعمال شده و فعالیت‌های SQA مناسبی اعمال می‌گردند. سپس شی وارد پایگاه داده‌ها شده و برای ایجاد نسخه بعدی نرم‌افزار از مکانیزم‌های نسخه مناسبی استفاده می‌شود. (بخش ۴-۹).

1.change control

2.Bach, J.

3.change request

با آنکه بسیاری درخواستهای تغییر در مرحله پشتیبانی و نگهداری (سیستم) دریافت می شوند، ما با نگرشی گسترده این مبحث را دنبال می کنیم.(با این نگرش) درخواست تغییر در هر زمان طی فرآیند نرم افزار ممکن است روی دهد.

4.change report

5.Change Control Authority (CCA)

فرآیند "بیرون کشیدن" و "وارد نمودن" دو رکن اصلی کنترل تغییر را انجام می‌دهد ـ کنترل دستیابی[2] و ***هم‌زمان کردن و همگام سازی تغییر***[3]. کنترل دستیابی تعیین می‌کند کدام مهندسین نرم‌افزار اختیار ارزیابی و اصلاح یک شی پیکربندی خاص را دارند. کنترل هم‌زمانی کمک می‌کند اطمینان حاصل شود که تغییرات مشابهی که توسط دو فرد مختلف انجام می‌شوند، بر روی هم کپی نشوند [HAR89][4].

توصیه

پریشانی، خطا را بدنبال دارد- که برخی از آنها بسیار خطرناکند.کنترل دسترسی و هماهنگی از اغتشاش جلوگیری می کنند. آندو را با هم اعمال کنید. حتی اگر رهیافت شما مبتنی بر همراه شدن با فرهنگ توسعه (کار فرهنگی) باشد.

طرح جریان ارزیابی و هم‌زمانی کنترل در شکل ۹-۶ آمده است. یک مهندس نرم‌افزار بر اساس درخواست تغییر به تأیید رسیده و **ECO** یک شی پیکربندی را بیرون می‌کشد. عمل ارزیابی کنترل، به ما اطمینان می‌دهد که مهندس نرم‌افزار اختیار بیرون کشیدن شی را دارد، و هم‌زمانی کنترل شی را در پایگاه داده‌های پروژه نگه می‌دارد[5]. به‌طوری‌که این شی به هیچ عنوان به روز نمی‌شود تا زمانی‌که نسخه‌ای که به‌تازگی بیرون کشیده شده جایگزین آن گردد. توجه داشته باشید که سایر نسخه‌ها را می‌توان بیرون کشید ولی سایر به روز رسانی‌ها را نمی‌توان ایجاد نمود. یک نسخه از شی دارای مبنا، که "نسخه استخراج شده"[6] نامیده می‌شود، توسط مهندس نرم‌افزار اصلاح می‌گردد. پس از **SQA** مناسب و آزمون، نسخه اصلاح یافته شی بیرون کشیده شده و قفل شی مبنای جدید باز می‌شود.

شاید برخی از خوانندگان از میزان تشریفات اعمال شده در اثر شرح فرآیند کنترل تغییر احساس نارضایتی کنند. این یک احساس غیرعادی نیست. بدون وجود حفاظت‌های مناسب، کنترل تغییر پیشرفت را کُند می‌نماید و سبب به‌وجود آمدن نوار قرمز غیر ضروری می‌شود. بیشتر سازندگان نرم‌افزاری که دارای مکانیزم‌های کنترل تغییر هستند (متأسفانه بیشتر آنها فاقد آن هستند) تعداد زیادی لایه کنترل ایجاد نموده‌اند که به آنها امکان می‌دهد از مشکلاتی که در بالا به آنها اشاره شد اجتناب نمایند.

توصیه

اندکی بیشتر از انچه می پندارید، نیاز است کنترل تغییرات داشته باشید. این امر بسیار مفید خواهد بود.

پیش از آن‌که یک **SCI** تبدیل به یک خط مبنا گردد، تنها نیاز به استفاده از کنترل تغییر غیر رسمی[7] وجود دارد. سازنده شی پیکربندی (**SCI**) مورد بحث، هرگونه تغییر منطقی را توسط پروژه و ابزارهای فنی اعمال می‌نماید (تا زمانی‌که تغییرات، نیازهای وسیع‌تر سیستم را که در خارج از حوزهٔ کاری سازنده وجود دارد تحت تأثیر قرار ندهد).

1.Engineering Change Order (ECO)

2.Access control

3.Synchronization control

4.Harter, R.

5.locks

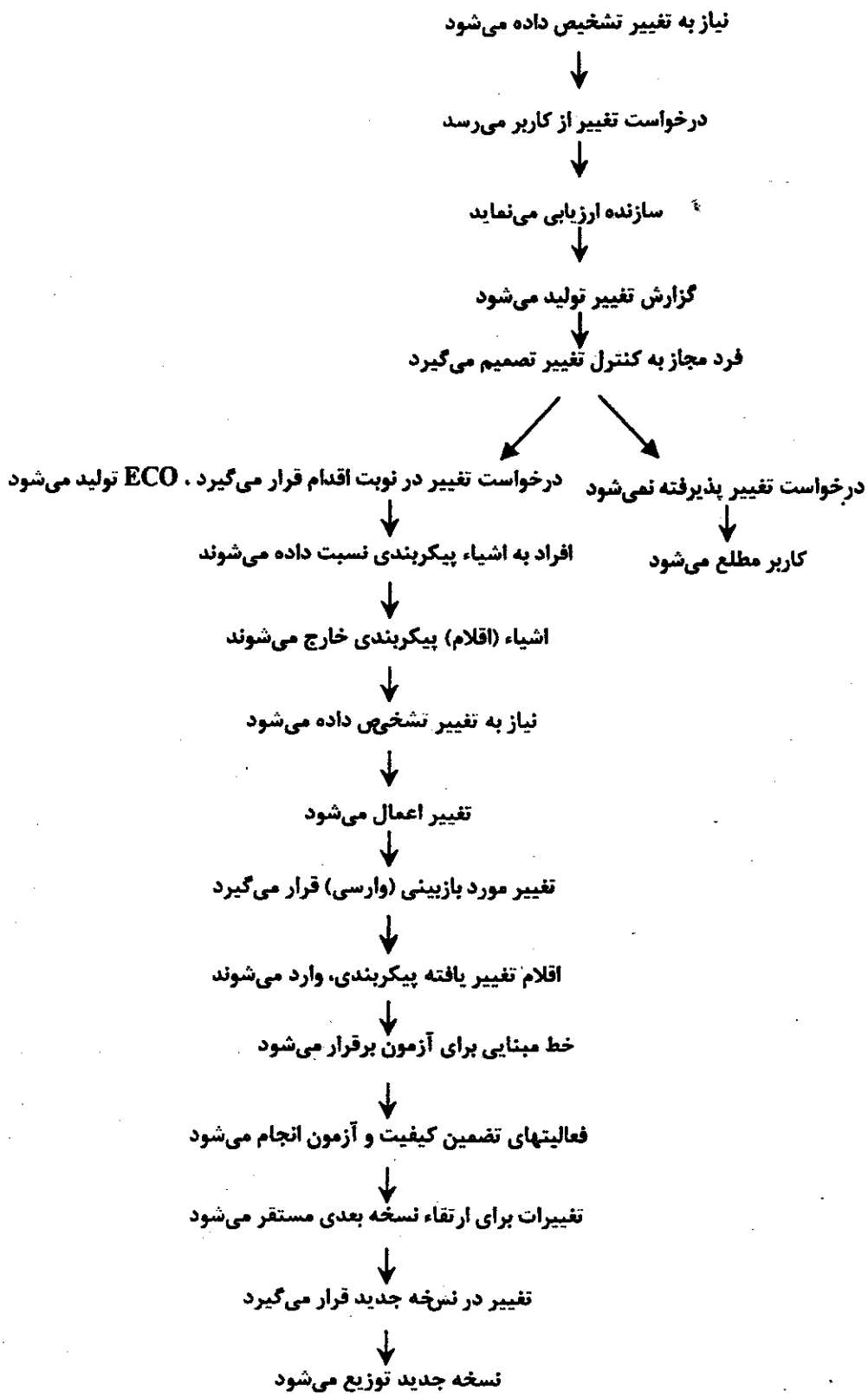
6.extracted version

7.informal change control

به محض این‌که شی تحت بررسی فنی رسمی قرار گرفته و به تأیید رسید یک خطا مبنا ایجاد می‌گردد. به محض این‌که یک SCI تبدیل به یک خط مبنا گردید کنترل تغییر سطح پروژه[1] به انجام می‌رسد. اما، برای ایجاد تغییر، چنانچه تغییر سایر SCIها را تحت تأثیر قرار دهد، سازنده باید از مدیر پروژه (در صورتی‌که تغییر موضعی باشد) و یا از CCA کسب اجازه نماید. در برخی موارد درخواست تغییر، گزارش تغییر و ECOها به‌طور رسمی انجام می‌شوند. اما ارزیابی هر تغییر صورت پذیرفته و تمام تغییرات دنبال شده و مورد بررسی قرار می‌گیرند. وقتی‌که محصول نرم‌افزاری در اختیار مشتریان قرار می‌گیرد، کنترل تغییر رسمی[2] صورت می‌پذیرد. شیوه کنترل تغییر در شکل ۹-۵ آمده است.

1.project level change control

2.formal change control



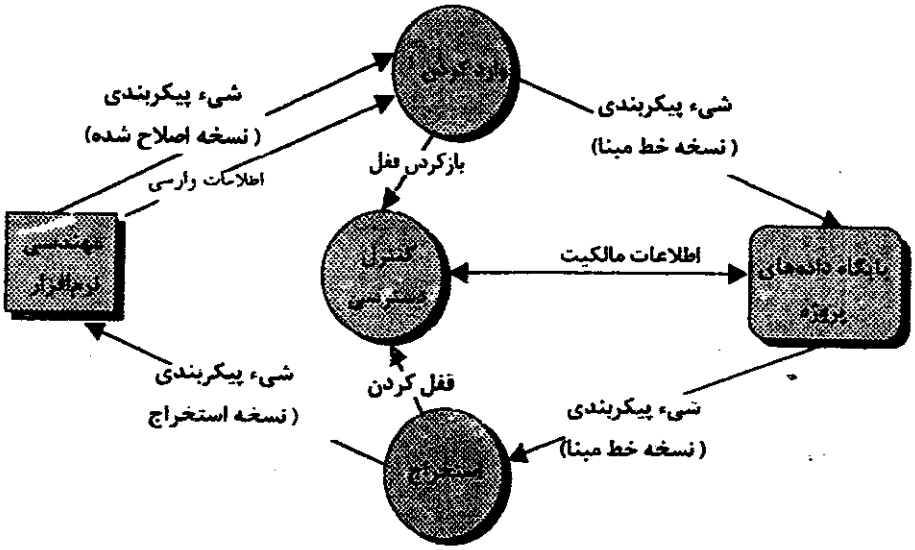
شکل ۹ـ۵ فرآیند کنترل تغییر

نقل قول

تغییرات ، احتناب ناپذیرند ولو برای ماشینهای خودکار(که با سکه ای نوشیدنی تحویل می دهند) .

بامپر استیکر

مسئول کنترل تغییر (CCA) نقش فعالی در لایه‌های دوم و سوم کنترل، ایفا می‌نماید. بسته به اندازه و ویژگی یک پروژه نرم‌افزاری CCA می‌تواند متشکل از یک نفر ـ مدیر پروژه ـ و یا چندین نفر باشد (به‌طور مثال نمایندگان مهندسی، پشتیبانی و فروش و ... نرم‌افزار، سخت‌افزار و پایگاه داده‌ها). نقش CCA عبارت است از داشتن یک دیدگاه جهانی، یعنی، ارزیابی تأثیر تغییر به جز SCI مورد سؤال، و اینکه تغییر چگونه می‌تواند درک مشتریان را نسبت به محصول اصلاح نماید؟ تغییر چگونه می‌تواند کیفیت و اعتبار محصول را تحت تأثیر قرار دهد؟ این سؤالات و سؤالات بسیار دیگر توسط CCA مورد بررسی قرار می‌گیرند.



شکل ۹-۶ کنترل دسترسی و هماهنگی

۹-۶ وارسی پیکربندی

شناسایی کنترل نسخه و کنترل تغییر به سازنده نرم‌افزار کمک می‌کند تا نظم را برقرار نماید، نظمی که در صورت عدم وجود، سبب آشفتگی و تزلزل موقعیت می‌شود. اما حتی موفق‌ترین مکانیزم‌های کنترل فقط تا زمانی تغییر را دنبال می‌نمایند که یک ECO ایجاد گردد. چگونه می‌توانیم اطمینان حاصل کنیم که تغییر به‌طور مناسبی صورت پذیرفته است؟ پاسخ به این سؤال از دو جنبه خواهد بود: (۱) بررسی‌های فنی رسمی و (۲) وارسی پیکربندی نرم‌افزار.

بررسی فنی رسمی (که به تفصیل در فصل ۸ شرح داده شده) به صحت فنی شیء پیکربندی که اصلاح گردیده می‌پردازد. بررسی‌کنندگان، SCIها، وارسایی‌ها و یا اثرات جانبی بالقوه را ارزیابی می‌نمایند بررسی فنی رسمی باید در مورد تمام و حتی جزیی‌ترین تغییرات صورت پذیرد.

"وارسی پیکربندی نرم‌افزار" بررسی فنی رسمی را از طریق ارزیابی ویژگی‌های یک شی پیکربندی که معمولاً طی بررسی در نظر گرفته نمی‌شوند، تکمیل می‌نماید. در وارسی سؤالات زیر پرسیده شده و به آنها پاسخ داده می‌شود:

چه سوالات اساسی در یک "وارسی پیکربندی" باید پرسیده شوند؟

۱- آیا تغییری که در ECO مشخص گردیده، اعمال شده است؟ آیا اصلاحات اضافی دیگری صورت پذیرفته است؟

۲- آیا برای ارزیابی صنعت فنی، بررسی فنی رسمی صورت پذیرفته است؟

۳- آیا پروسه نرم‌افزاری دنبال شده است و آیا استانداردهای مهندسی نرم‌افزار به‌طور مناسبی اعمال گردیده‌اند؟

۴- آیا تغییر در SCI مشخص گردیده است؟ آیا تاریخ تغییر و تغییردهنده مشخص گردیده است؟ آیا صفات و ویژگی‌های شی پیکربندی نشان‌دهندهٔ تغییر هستند؟

۵- آیا شیوه‌های SCM برای توجه به تغییر، ثبت آن، و گزارش آن دنبال شده است؟

۶- آیا تمام SCIهای مربوطه، به‌طور مناسب به روز شده‌اند؟

در برخی موارد سؤالات مربوط به وارسی به‌عنوان بخشی از یک بررسی فنی رسمی پرسیده می‌شوند. اما، وقتی که SCM یک فعالیت رسمی می‌باشد، حسابرسی SCM به‌طور جداگانه و توسط گروه تضمین کیفیت انجام می‌شود.

۹-۷ گزارش وضعیت

"گزارش وضعیت پیکربندی"[1] (که گاهی اوقات صورت وضعیت[2] نیز نامیده می‌شود) یکی از وظایف SCM است که به سؤالات زیر پاسخ می‌دهد: (۱) چه اتفاقی افتاد؟ (۲) چه کسی این کار را انجام داد؟ (۳) چه وقت این اتفاق افتاد؟ (۴) چه بخش‌های دیگری تحت تأثیر قرار خواهند گرفت؟

توصیه

یک "لیست آنانکه باید بدانند" برای هر قلم پیکربندی (SCI) تنظیم کنید و آنرا به هنگام سازید. هنگامی که تغییری به وجود می آید، اطمینان حاصل نمایید که هرآنکس نام او در لیست می باشد، از تغییر مطلع گردیده است.

جریان اطلاعات در مورد گزارش وضعیت پیکربندی (CSR) در شکل (۹-۵) آمده است. هر بار که یک SCI هویت جدید و یا به روزی را انتخاب می‌نماید، یک مورد CSR ایجاد می‌شود. هر بار که وارسی پیکربندی صورت می‌پذیرد، نتایج به‌عنوان بخشی از کار CSR گزارش می‌گردند. ممکن است خروجی CSR در یک پایگاه داده روی خط قرار بگیرد، [TAY85][3] که در این‌صورت سازندگان و یا اداره‌کنندگان نرم‌افزار می‌توانند اطلاعات مربوط به تغییر را به‌وسیلهٔ مقولهٔ کلید واژه، ارزیابی نمایند. به‌علاوه یک گزارش CSR بر طبق یک مبنای ثابت ارائه می‌شود و هدف از آن اینست که مدیر و سازندگان به ارزشیابی تغییرات ادامه دهند.

1.configuration status reporting

2.status accounting

3.Taylor, B.

گزارش وضعیت پیکربندی نقش مهمی در موفقیت پروژه های بزرگ نرم‌افزاری ایفا می‌نماید. وقتی‌که افراد زیادی درگیر یک پروژه می‌شوند، حرف هم‌دیگر را خوب درک نمی‌کنند. دو سازنده سعی می‌کنند یک SCI را با اهداف متفاوت و متضاد تغییر دهند. ممکن است یک تیم مهندسی نرم‌افزار ماهها وقت صرف ساخت نرم‌افزاری کند که مشخصات سخت‌افزاری آن منسوخ می‌نماید. فردی که متوجه تأثیرات جانبی جدی یک تغییر پیشنهادی می‌شود، از اعمال تغییر، آگاهی ندارد. CSR کمک می‌کند تا از طریق بهبود روابط بین تمام افراد این مشکلات برطرف گردند.

۹-۸ استاندارد های مدیریت پیکربندی نرم افزار

در بیش از دو دهه گذشته بسیاری استاندارد های مدیریت پیکربندی نرم افزار ارائه گردیده اند. بسیاری از استانداردهای اولیه، مانند MIL-STD-483 ، DOD-STD-480A ، و MIL-STD-1521A بر توسعه و ساخت نرم افزارهای نظامی متمرکز شده و تاکید داشته اند. با این وجود استانداردهای ANSI/IEEE ، مانند ANSI/IEEE Stds. No.828-1983, No. 1042-1987, و Std. No. 1028- 1988 [IEE94]، برای نرم افزارهای غیر نظامی به کار می رود و برای سازمانهای مهندسی نرم افزار کوچک و بزرگ (هردو) توصیه می شود.

۹-۹ خلاصه

مدیریت پیکربندی نرم‌افزار عبارت است از یک فعالیت جامع که در طول فرآیند نرم‌افزاری اعمال می‌گردد. SCM اصلاحاتی را که همواره به‌وقوع می‌پیوندد، در زمان ساخت نرم‌افزار و پس از ارائه آن به مشتریان، شناسایی کنترل، بررسی و گزارش می‌نماید. تمام اطلاعاتی که به‌عنوان بخشی از مهندسی نرم‌افزار تولید می‌شوند تبدیل به بخشی از پیکربندی یک نرم‌افزار می‌گردند. پیکربندی به روشی سازمان‌دهی می‌شود که کنترل منظم تغییر را امکان‌پذیر می‌سازد.

پیکربندی نرم‌افزار متشکل است از مجموعه‌ای از شی‌های مربوط به‌هم، که همچنین قلم‌های پیکربندی نرم‌افزار نیز نامیده می‌شوند، و در نتیجهٔ برخی فعالیت‌های مهندسی نرم‌افزار ایجاد می‌گردند. علاوه بر سندها، برنامه‌ها و داده‌ها، محیط ساخت که برای ایجاد نرم‌افزار به‌کار می‌رود، نیز می‌تواند تحت کنترل پیکربندی باشد.

به‌محض این‌که یک شی ایجاد گردید و تحت بررسی قرار گرفت، تبدیل به یک خط مبنا می‌شود. تغییراتی که بر روی یک شی مبنا اعمال می‌شوند، سبب می‌شوند تا یک نسخه جدیدی از آن شی به‌وجود بیاید. تکامل یک برنامه را می‌توان از طریق بررسی تاریخچهٔ اصلاح تمام شی‌های پیکربندی دنبال نمود. شی‌های پایه و شی‌های مرکب (مجتمع)، تشکیل یک مخزن اشیا را می‌دهند که نسخه ها و گونه‌های

متفاوت دیگر از روی آن ساخته می‌شوند. کنترل نسخه عبارت است از مجموعه‌ای از شیوه‌ها و ابزارها که برای کنترل استفاده از این شی‌ها به‌کار می‌روند.

کنترل تغییر، فعالیتی است که مربوط به روش کار می‌شود و با اعمال تغییرات به شیء، پیکربندی کیفیت و هماهنگی آن را تضمین می‌نماید. فرآیند کنترل تغییر، با یک درخواست آغاز شده، منجر به اتخاذ تصمیم برای ساخت و یا رد درخواست برای تغییر می‌شود، و با یک به روز رسانی کنترل شدهٔ SCIای که قرار است تغییر کند خاتمه می‌یابد.

وارسی پیکربندی عبارت است از یک فعالیت SQA که به حصول اطمینان از کیفیت به هنگام اعمال تغییرات کمک می‌نماید. گزارش وضعیت، اطلاعاتی را دربارهٔ هر تغییر به کسانی که نیازمند آگاهی از آن هستند ارائه می‌نماید.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۹-۱ چرا قانون اول مهندسی سیستم حقیقت دارد؟ آن چگونه بر ادراک ما از پارادایم های مهندسی نرم‌افزار تأثیر می گذارد؟

۹-۲ به زبان خودتان دلایل مربوط به خط مبنا را بحث کنید.

۹-۳ فرض کنید شما مدیر یک پروژه کوچک هستید، چه خطوط پایه ای برای پروژه تعریف می‌کنید و چگونه آنها را کنترل می‌کنید؟

۹-۴ یک سیستم پایگاه داده های پروژه طراحی کنید که مهندس نرم‌افزار را به ذخیره، ارجاع متقابل، ردگیری، به روز رسانی ، تغییرات و دیگر اقدامات لازم بر تمام اقلام پیکربندی مهم نرم افزار قادر سازد. پایگاه داده ها چگونه نسخه‌های گوناگون یک برنامه را پشتیبانی می کند؟ آیا شیوهٔ رفتار با برنامه ها و مستندات متفاوت خواهد بود؟ چگونه دو سازنده از اعمال تغییرات متختلف بر یک SCI یکسان اجتناب خواهند کرد؟

۹-۵ تحقیقی پیرامون پایگاه داده های شیءگرا انجام دهید و مقاله‌ای بنویسید بر این شرح که چگونه می توان آنها را در بافت مدیریت پیکربندی نرم افزار SCM مورد استفاده قرار داد.

۹-۶ از یک مدل E-R (فصل ۱۲) برای تشریح رابطه داخلی میان اقلام پیکربندی نرم افزار (اشیاء) که در بخش ۹-۱-۲ لیست شده، استفاده کنید.

۹-۷ یک ابزار موجود مدیریت پیکربندی نرم افزار را تحقیق کنید و شرح دهید که چگونه کنترل نسخه‌ها، تنوع‌ها و به‌طور کلی اشیاء پیکربندی را پیاده‌سازی می‌کند؟

۹-۸ روابط < قسمتی است از> و < ارتباط دارد با> رابطه ساده میان اشیای پیکربندی را نشان می دهند. پنج رابطهٔ اضافه دیگر شرح دهید که ممکن است در بافت پایگاه داده های پروژه مفید واقع شوند.

۹-۹ یک ابزار موجود مدیریت پیکربندی نرم افزار را تحقیق کنید و شرح دهید که چگونه مکانیک کنترل نسخه را پیاده‌سازی می‌کند. به جای آن، دو یا سه مقاله دربارهٔ SCM مطالعه کنید و ساختمان داده‌ها و مکانیسمهای مختلف به‌کار رفته در کنترل نسخه را تشریح کنید.

۹-۱۰ با استفاده از شکل ۹-۵ به‌عنوان راهنما، جزئیات بیشتری از تفکیک کاری را برای کنترل تغییرات توسعه دهید. نقش CCA را شرح دهید و قالب‌هایی را برای درخواست تغییر، گزارش تغییر و ECO پیشنهاد کنید.

۹-۱۱ یک لیست کنترل تهیه کنید که در حین وارسی پیکربندی به کار آید.

۹-۱۲ اختلاف میان یک وارسی مدیریت پیکربندی نرم افزار و بازبینی فنی رسمی چیست؟ آیا کارکرد آنها را می‌توان در یک بازبینی خلاصه نمود؟ مزایا و معایب آنها کدام است؟

فهرست منابع و مراجع

[BAB86] Babich, W.A., *Software Configuration Management,* Addison-Wesley, 1986.
[BAC98] Bach,]., "The Highs and Lows of Change Control," *Computer,* vol. 31, no. 8, August 1998, pp. 113-115.
[BER8O] Bersoff, E.H., V.D. Henderson, and S.G. Siegel, *Software Configuration Management,* Prentice-Hall, 1980.
[CH089] Choi, S.C. and W. Scacchi, "Assuring the Correctness of a Configured Software Description," *Proc. 2nd Intl. Workshop on Software Configuration Management,* ACM, Princeton, NJ, October 1989, pp. 66-75.
[CLE89] Clemm, G.M., "Replacing Version Control with Job Control," *Proc. 2nd Intl. Workshop on Software Configuration Management,* ACM, Princeton, NJ, October I 989,_ pp.162-169.
[GUS89] Gustavsson, A., "Maintaining the Evaluation of Software Objects in an Integrated Environment," *Proc. 2nd Intl. Workshop on Software Configuration Management,* ACM, Princeton, NJ, October 1989, pp. 114-1 17.
[HAR89] Harter, R., "Configuration Management," *HP Professional,* vol. 3, no. 6, June 1989.
[IEE94] *Software Engineering Standards,* 1994 edition, IEEE Computer Society, 1994.
[LIE89] Lie, A. et al., "Change Oriented Versioning in a Software Engineering Database," *Proc. 2nd Intl. Workshop on Software Configuration Management,* ACM, Princeton, NJ, October, 1989, pp. 56-65.
[NAR87] Narayanaswamy, K. and W. Scacchi, "Maintaining Configurations of Evolving Software Systems," *IEEE 7rans. Software Engineering,* vol. SE-13, no. 3, March 1987, pp. 324-334.
[RE189] Reichenberger,_ C., "Orthogonal Version Management," *Proc. 2nd Intl. Workshop on Software Configuration Management* ACM, Princeton, NJ, October 1989, pp. 137-140.
[TAY85] Taylor, B., "A Database Approach to Configuration Management for Large Projects," *Proc. Conf. Software Maintenance- I* 985, IEEE, November 1985, pp. 15-23.
[TIC82] Tichy, W.F., "Design, Implementation and Evaluation of a Revision Control System," *Proc. 6th Intl. Conf. Software Engineering,* IEEE, Tokyo, September 1982, pp. 58-67.

خواندنیهای دیگر و منابع اطلاعاتی

One of the few books that have been written about SCM in recent years is by Brown, et al. *(AntiPatterns and Patterns in Software Configuration Management,* Wiley, 1999). The authors discuss the things not to do (antipatterns) when implementing an SCM process and then consider their remedies.

Lyon *(Practical CM: Best Configuration Management Practices for the 2 1st Century,* Raven Publishing, 1999) and Mikkelsen and Pherigo *(Practical Software Configuration Management: The Latenight Developer's Handbook,* Allyn & Bacon, 1997) provide pragmatic tutorials on important SCM practices. Ben-Menachem *(Software Configuration Management Guidebook,* McGraw-Hill, 1994), Vacca *(Implementing a Successful Con*

figuration Change Management Program, I. S. Management Group, 1993), and Ayer and patrinnostro *(Software Configuration Management*, McGraw-Hili, 1992) present good overviews for those who need further introduction to the subject. Berlack *(Soft*

ware Configuration Management, Wiley, 1992) presents a useful survey of SCM concepts, emphasizing the importance of the repository and tools in the management of change. Babich [BAB86] provides an abbreviated, yet effective, treatment of pragmatic issues in software configuration management.

Buckley *(Implementing Configuration Management*, IEEE Computer Society Press, 1993) considers configuration management approaches for all system elements-hardware, software, and firmware-with detailed discussions of major CM activities. Rawlings *(SCM for Network Development Environments*, McGraw-Hill, 1994) is the first SCM book to address the subject with a specific emphasis on software development in a networked environment. Whitgift *(Methods and Tools for Software Configuration Management*, Wiley, 199 I) contains reasonable coverage of all important SCM topics, but is distinguished by discussion of repository and CASE environment issues. Arnold and Bohner *(Software Change Impact Analysis*, IEEE Computer Society Press, 1996) have edited an anthology that discusses how to analyze the impact of change within complex software-based systems.

Because SCM identifies and controls software engineering documents, books by Nagle *(Handbook for Preparing Engineering Documents: From Concept to Completion*, IEEE, 1996), Watts *(Engineering Documentation Control Handbook: Configuration Management for Industry*, Noyes Publications, 1993), Ayer and Patrinnostro *(Documenting the Software Process*, McGraw-Hill, 1992) provide a complement to more-focused SCM texts. The March 1999 edition of *Crosstalk* contains a number of useful articles on SCM.

A wide variety of information sources on software configuration management and related subjects is available on the Internet. An up-to-date list of World Wide Web references that are relevant to SCM can be found at the SEPA Web site: http://www.mhhe.com/engcs/compsci/pressman/resources/scm.mhtml