فصل ۳۰ مهندسی مجدد

مفاهیم کلیدی (مرتب بر حروف الفبا)
اصول مهندسی مجدد فرآیند تجاری(BPR) ، تجدید ساخت ، تجدید ساخت مستندات ، تحلیل موجودی ، ساختار داده ها ، سطح تجرید و انتزاع ، فرآیند مهندسی مجدد فرآیند تجاری (BPR) ، معماری خادم / مخدوم ، معماری های OO ، ملاحظات اقتصادی ، مهندسی رو به جلو ، مهندسی مجدد ، مهندسی معکوس

KEY CONCEPTS
Abstraction level , BPR principles , BPR process , c/s architectures , data structure , document restructuring , economics , forward engineering , inventory analysis , OO architectures , restructuring , reverse engineering , reengineering

نگاه اجمالی

مفهوم مهندسی مجدد چیست؟ هر نوع محصول فناوری را درنظر بگیرید که سرویس خوبی به شما داده است. شما به‌طور مرتب از آن استفاده می‌کنید ولی در حال کهنه شدن است. اغلب می‌شکند، تعمیر آن خیلی طول می‌کشد و دیگر نمایندهٔ فناوری جدید نیست. چه باید کرد؟ اگر این محصول سخت‌افزار است احتمالاً آن را به دور انداخته و یک مدل جدیدتر خواهید خرید. ولی اگر نرم‌افزار سنتی باشد دیگر این‌کار مقدور نیست. باید آن را دوباره بسازید. شما محصولی خلق خواهید کرد با کارکردهای اضافی، عملکرد بهتر، قابلیت اطمینان و قابلیت نگهداری بهتر و تمام این مراحل را مهندسی مجدد گویند.

چه کسی این‌امر را عهده دار است؟ در سطح تجاری، این‌کار از سوی کارشناسان تجاری (اغلب شرکت‌های مشاوره) انجام می‌شود. در سطح نرم‌افزاری، مهندسی مجدد توسط مهندسین نرم‌افزار انجام می‌شود.

چرا مهندسی مجدد از اهمیت برخوردار است؟ ما در دنیای سریعاً متحولی زندگی می‌کنیم. تقاضاها برای کارکردهای تجاری و فناوری اطلاعات که آنها را حمایت کند با سرعتی شگرف در تغییر است که فشار رقابتی شدیدی را روی سازمان تجاری اعمال می‌کند. هر دوی تجارت و نرم‌افزاری که آن را حمایت می‌کند باید مهندسی مجدد شوند تا هماهنگ با شرایط روز باشند.

چه مراحلی وجود دارند؟ مهندسی مجدد فرآیند تجاری (BPR) اهداف تجاری را روشن کرده، پروسه‌های تجاری فعلی را تعریف و ارزیابی می‌کند و پروسه‌های تجاری تجدیدنظر شده‌ای خلق می‌کند که

نیازها و اهداف جاری را بهتر برآورده می‌کنند. فرآیند مهندسی مجدد نرم‌افزار دربردارندهٔ تحلیل موجودی، ساخت مجدد سند، مهندسی معکوس می‌باشد. قصد این فعالیت‌ها خلق نسخه‌هایی از برنامه‌های فعلی است که قابلیت نگهداری بهتر و کیفیت بالاتر ایجاد می‌کنند.

محصول کار چه خواهد بود؟ انواع محصولات کاری مهندسی مجدد (مثلاً مدل‌های تحلیل، مدل‌های طراحی، روبه‌های آزمون) تولید می‌شوند. خروجی نهایی فرآیند تجاری مهندسی مجدد *و* یا نرم‌افزار مهندسی مجددی است که آن را حمایت می‌کند.

چگونه اطمینان حاصل کنم که آن را به‌درستی انجام داده‌ام؟ همان روش‌هایی را که در هر فرآیند مهندسی نرم‌افزار به‌کار می‌روند را به‌کار بگیرید ـ بازبینی‌های فنی رسمی، مدل‌های تحلیل و طراحی را ارزیابی می‌کنند، بازبینی‌های تخصصی قابلیت کاربردی و سازگاری تجاری را در نظر می‌گیرند، آزمون، به‌کار گرفته می‌شود تا اشتباهات در مضمون و محتوا کشف شود که شامل قابلیت کارکردی و عملیات درونی نیز می‌شود

در یک مقاله بنیادی نوشته شده برای هاروارد بیزنس ریویو[1] ، مایکل همر [HAM90][2] انقلابی در تفکر مدیریتی پیرامون پروسه‌های تجاری و محاسباتی، پایه‌گذاری کرد:

اینک زمان آن رسیده است که دیگر راه‌های مال‌وا را هموار نکنیم، به‌جای قرار دادن پروسه‌های کهنه در سیلیکون و نرم‌افزار، باید آنها را محو و پاک کرده و از نو شروع کنیم. ما باید امور تجاری خود را «مهندسی مجدد» نماییم. از فناوری اطلاعات مدرن برای طراحی مجدد اساسی پروسه‌های تجاری خود استفاده کنیم. به‌منظور دستیابی به پیشرفت‌های چشمگیر در عملکرد آنها هر شرکتی طبق قوانین غیر مدون بیشماری کار می‌کند... مهندسی مجدد تلاش می‌کند تا از قوانین کهنه دربارهٔ چگونگی سازمان‌دهی و اجرای امور تجاری خود رها شود.

مانند تمام انقلاب‌ها، ندای انقلابی «همر» به تغییرات مثبت و منفی منتهی شد. طی دههٔ ۱۹۹۰، بعضی شرکت‌ها تلاش به‌حقّی برای مهندسی مجدد داشتند و نتایج مربوطه به شرایط رقابتی بهتر منجر گردید. دیگران صرفاً متکی به کاهش سایز و بیمانکاری بودند (به‌جای مهندسی مجدد) تا خط کف خود را بهبود دهند. سازمان‌های متوسط با پتانسیل اندک برای رشد در آینده حاصل کار بودند. [DEM 95][3].

طی این اولین دهه از قرن بیست و یکم، حیال‌پردازی مرتبط با مهندسی مجدد رو به تحلیل رفته است ولی خود فرآیند شرکت‌های بزرگ و کوچک ادامه یافته است.

ارتباط بین مهندسی مجدد تجاری و مهندسی نرم‌افزار در یک «منظر سیستم» قرار گرفته است. نرم‌افزار اغلب تحقق قوانین تجاری است که همر در مورد آنها بحث کرده است. همان‌طور که این قوانین تغییر می‌کند، نرم‌افزار نیز باید تغییر کند. امروزه، شرکت‌های بزرگ ده‌ها هزار برنامه‌های کامپیوتری دارند

1. Harvard Business Review
2.Hammer ,M.
3.DeMarco,T.

که «قوانین تجاری کهنه» را حمایت می‌کنند. همان‌طور که مدیران برای اصلاح قوانین کار می‌کنند تا کارآیی و توان رقابتی بهتری بدست آید، نرم‌افزار باید به روز باشد. در بعضی موارد، این امر بدان معنی است که خلق سیستم‌های منکی به کامپیوتر الزامی است.[1] ولی در بسیاری دیگر موارد، به‌معنی آن است که کاربردهای فعلی اصلاح و تجدید ساخت گردند.

در این فصل، ما به گونه‌ای بالا به پایین مهندسی مجدد را بررسی خواهیم کرد که با مروری کوتاه بر مهندسی مجدد فرآیند تجاری آغاز و به بحث تفصیلی‌تری از فعالیت‌های فنی که هنگام مهندسی مجدد نرم‌افزار رخ می‌دهد، تداوم بیدا می‌کند.

۳۰-۱ مهندسی مجدد فرآیند تجاری (BPR)

نقل قول

(فرار کردن) در برابر فردا، با شیوه های تفکر دیروز، یعنی ترک زندگی رویایی با یک سکته.

جیمز بل

مهندسی مجدد فرآیند تجاری[2] (BPR) فراتر از حوزه‌فناوری‌های اطلاعاتی و مهندسی نرم‌افزار می‌رود. در بین بسیاری تعاریف (اکثراً به‌نوعی انتزاعی) که برای BPR پیشنهاد شده است، یکی در مجلهٔ فوریتون[3] چاپ شده است [STE93][4] «جستجو و پیاده‌سازی تغییرات عمده در فرآیند تجاری برای حصول نتایج راهگشا» ولی این جستجو چگونه انجام می‌شود و پیاده‌سازی چگونه حاصل می‌شود؟ مهم‌تر آن‌که، چگونه می‌توانیم مطمئن باشیم که «تغییر عمده» پیشنهادی در حقیقت به «نتایج راهگشا» به‌جای درهم ریختگی سازمانی منتهی خواهد شد؟

۳۰-۱-۱ فرآیند تجاری

یک پروسهٔ تجاری[5] «مجموعه‌ای از کارهای منطقاً به یکدیگر مرتبط است که انجام می‌شوند تا یک نتیجهٔ تجاری تعریف شده را حاصل کنند.» [DAV90][6] در محدوده‌ای فرآیند تجاری، افراد، تجهیزات، منابع مواد و رویه‌های تجاری ترکیب می‌شوند تا نتیجهٔ خاصی را حاصل کنند. نمونه‌های فرآیندهای تجاری شامل طراحی یک محصول جدید، خدمات خرید و عرضه، استخدام پرسنل جدید یا پرداخت عرضه‌کنندگان‌کننده می‌شود. هریک نیاز به مجموعه‌ای از وظایف داشته و هریک به منابع گوناگون درون کار تجاری ارتباط دارند.

هر فرآیند تجاری یک مشتری تعریف شده دارد. فردی که با گروهی که محصول را دریافت می کند (مثلاً، یک ایده، گزارش، طرح یا محصول). به‌علاوه، فرآیندهای تجاری از مرزهایی سازمانی عبور می‌کنند. آنها نیاز دارند که گروه‌های سازمانی متفاوت در وظائف«منطقاً مرتبط» که پروسه را تعریف می‌کنند،

---

۱. سیستم های مبتنی بر وب و برنامه های کاربردی در فصل ۲۹ (به همراه مثالهای متنوع) تشریح شده اند.

2.Business Process Reengineering
3. Fortum
4.Stewart,T.A.
5.Business Process
6.Davenport,T.H.

شرکت داشته باشند. در فصل ۱۰ ما توجه کردیم که هر سیستمی عملاً یک سلسله مراتب از زیر سیستم‌ها است. یک تجارت نیز استثناء نیست. کل تجارت به‌صورت ذیل قسمت‌بندی می‌شود:

- تجارت
- سیستم‌های تجاری
- فرآیند تجاری
- فرآیندهای فرعی تجاری

هر سیستم تجاری (که کارکرد تجاری[1] نیز نامیده می‌شود) تشکیل شده از یک یا چند فرآیند تجاری، و هر فرآیند تجاری با مجموعه‌ای از زیر فرآیندها تعریف شده است.

BPR در هر سطحی از سلسله مراتب می‌تواند به‌کار گرفته شود ولی همان‌طور که دامنهٔ عمل BPR وسیع می‌شود (به‌عبارتی همان‌طور که به‌طرف بالا در سلسله مراتب حرکت می‌کنیم)، ریسک‌های مرتبط با BPR به‌شدت افزایش پیدا می‌کند. برای همین خاطر است که اکثر تلاش‌های BPR روی فرآیندهای فرعی یا زیر فرآیندها متمرکز هستند.

۳۰-۱-۲ اصول مهندسی مجدد فرآیند تجاری

به‌طرق بسیار، BPR در تمرکز و دامنهٔ مهندسی فرآیند تجاری عین یکدیگر هستند (فصل ۱۰). هریک ترتیب ایده‌آل، BPR باید به روش بالا ـ پایین رخ بدهد، با شروع از تعریف اهداف تجاری اصلی و اهداف مربوطه و بعد روی مشخصات تفصیلی‌تر و وظایف متمرکز شده که به فرآیند تجاری ویژه مرتبط است. همر [HAM90][2] تعدادی از اصول را که فعالیت‌های BPR را هنگام شروع در سطح بالا (تجارت) پیشنهاد می‌کند.

پیرامون نتایج و نه امور، سازمان‌دهی کنید. بسیاری شرکتها فعالیت‌های تجاری تفکیک شده دارند تا هیچ فرد یگانه (یا سازمانی) مسئول (یا کنترل) ماحصل تجاری نباشد. در چنین موارد، تعیین وضعیت کار مشکل بوده و حتی رفع مشکلات فرآیند مشکل‌تر است، اثر چنین مشکلاتی پیش بینید. BPR باید فرآیندهایی را طراحی کند که این مشکل را نداشته باشند.

با افرادی کار کنید که از ماحصل فرآیند اجرا شده استفاده می‌کنند. قصد این توصیه، اجازه دادن به آنانی است که به محصول تجاری برای کنترل تمام متغیرهایی که اجازه می‌دهد تا محصول را به هنگام درآورند، نیاز دارند. هر چه اجراء جداگانهٔ کمتری در فرآیند هستند، راه برای رسیده به محصول سریع، هموارتر است.

به عنوان یک مهندس نرم افزار فعالیت مهندسی مجدد شما در انتهای این سلسله مراتب قرار دارد. اطمینان حاصل کنید که در سطوح بالاتر تفکر مناسب وجود داشته است. اگر اینگونه نباشد، کار شما با مخاطره همراه خواهد بود.

ارجاع به وب

اطلاعات جامعی در خصوص مهندسی مجدد فرآیند تجاری، در آدرس زیر یافت می شود:
www.brint.com/BPR.htm

1.business function
2.Hammer ,M.

کار پردازش اطلاعات را به کار واقعی که اطلاعات خام را بدست می‌آورد، پیوند زنید. همان‌طور که IT توزیع بیشتری پیدا می‌کند، می‌توان اکثر پردازش اطلاعات در سازمان را که داده‌های خام تولید می‌کند، مکان‌یابی کرد. این کنترل محلی، کرده و زمان ارتباطات را کم کرده و قدرت محاسبه را در دستان افرادی می‌گذارد که دارای علاقه‌ای پنهان به اطلاعات تولید شده دارند.

با منابع پراکندهٔ جغرافیایی به‌گونه‌ای رفتار کنید که گویا شده‌اند. ارتباطات متکی به کامپیوتر چنان پیچیده شده‌اند که گروه‌های متنوع از نظر جغرافیایی را می‌توان در «دفتر کار مجازی» همانند قرار داد. برای مثال، به‌جای داشتن سه شیفت مهندسی در یک مکان فقط یک شرکت جهانی می‌تواند یک شیفت در اروپا، شیفت دیگر در امریکای شمالی و شیفت سوم در آسیا داشته باشد. در هر مورد مهندسین در طول ساعات رور کار خواهند کرد و از طریق شبکه‌ها با باند عریض ارتباط خواهند داشت.

نقل قول

بمحض جایگزین کردن چیزهای کهنه با نو به آرامش می رسیم

فعالیت‌های موازی را متصل کنید به‌جای آن‌که نتایج آنها را یکپارچه سازید. زمانی‌که اجزای متفاوت به‌طور موازی کار می‌کنند ضرورت دارد که فرآیندی طراحی شود که متقاضی ارتباطات و هماهنگی‌های مستمر است در غیر این‌صورت مشکلات یکپارچه‌سازی قطعاً پیش خواهند آمد.

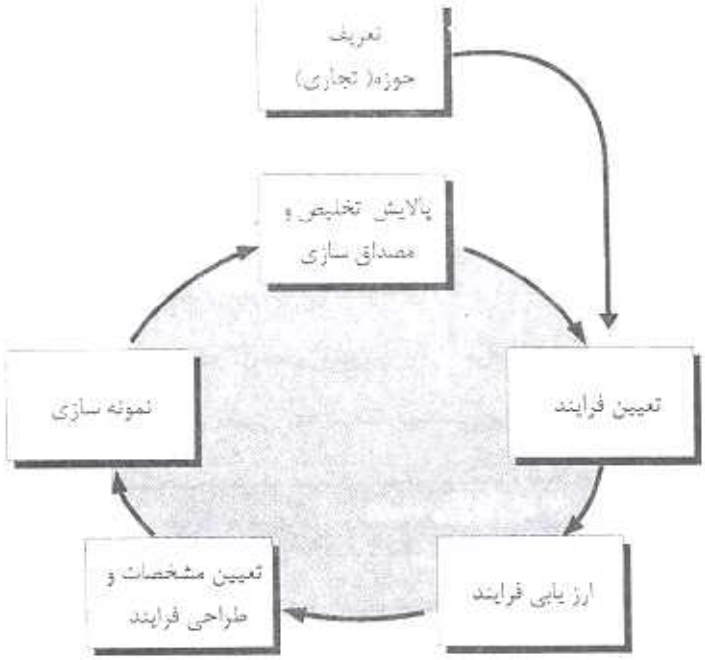
نقطهٔ تصمیم‌گیری را در جایی قرار دهید که کار انجام می‌شود و کنترل را در فرآیند اعمال کنید. با به‌کارگیری اصطلاحات طراحی برنامه‌ها، این حل یک معماری سازمانی بحث‌تر را با عوامل کاهش یافته، پیشنهاد می‌کند.

داده هارا یک بار و در منبع آن ذخیره کنید. داده‌ها باید به‌صورت On-Line ذخیره شوند تا زمانی‌که جمع‌آوری شدند هیچ‌گاه به وارد کردن دوباره نیازی نباشد.

هر یک از اصول فوق نمایانگر یک منظراز«تصویر بزرگ» BPR است. با هدایت این اصول طرح‌ریزان تجاری و طراحان فرآیند باید طراحی مجدد فرآیند را شروع کنند. در بخش بعدی، ما فرآیند BPR را با جزئیات بیشتری امتحان خواهیم کرد.

۳۰-۱-۳ یک مدل BPR

مانند اکثر فعالیت‌های مهندسی، مهندسی مجدد فرآیند تجاری تکرارشدنی است. اهداف تجاری و فرآیندهایی که آنها را حاصل می‌کنند باید با محیط تجاری متغیر سازگاری یابد.



شکل ۳۰-۱یک مدل مهندسی مجدد فرایند تجاری (BPR)

بدین‌خاطر، هیچ شروع و پایانی برای BPR نیست ـ یک پروسهٔ تکاملی است. مدلی برای مهندسی مجدد فرآیند تجاری در شکل ۳۰-۱ نمایش داده شده است. این مدل ۶ فعالیت را تعریف می‌کند:

تعریف تجارت. اهداف تجاری درون بافت ۴ محرک کلیدی قرار دارند: کاهش هزینه[1]، کاهش زمان[2]، ارتقاء کیفیت[3]، توسعه پرسنلی و تفویض قدرت[4]. اهداف که شاید در سطح تجاری تعریف شوند و یا برای یک جزء خاص ازتجارت.

شناسایی فرآیند. فرایندهایی که حیاتی بوده و باید اهداف آن حاصل شود که اهدافی تعیین شده در تعریف تجارت هستند، معین شده‌اند. سپس آنها با توجه اهمیت، نیاز به تغییر و یا هر طریقهٔ دیگری که برای امر مهندسی مجدد مناسب باشد. اولویت‌بندی می‌شوند.

ارزیابی فرآیند. فرآیند فعلی کاملاً تحلیل‌و اندازه‌گیری شده است. وظائف فرآیند تعریف شده‌اند، که شامل هزینه‌ها، زمان مصرف شده با امور فرآیند، خاطرنشان شده و مشکلات کیفیت/ عملکرد را منفک کرده‌ایم.

تعیین مشخصات پروسه و طراحی. بر پایه اطلاعات حاصله در طول اولین فعالیت‌های سه‌گانه BPR، مورد- کاربرد (فصل ۱۱) برای هر فرآیند که باید مجدداً طراحی شود، تهیه شده‌اند. در بافت و

1.cost reduction
2.time reduction
3.quality improvement
4.personel development and empowerment

مضمون BPR، کاربرد ـ موارد سناریویی را مشخص می‌کند که باردہ را به یک مشتری تحویل می‌دهد. با مورد- کاربرد به‌عنوان مشخصات فرآیند، یک مجموعهٔ جدید از وظائف (که با اصول متذکر شده در بخش ۳-۲-۱ هم‌خوانی دارند) برای فرآیند طراحی می‌شود.

ساخت نمونه اولیه. یک فرآیند تجاری دوباره طراحی شده باید قبل از آن‌که کاملاً در تجارت یکپارچه شود، نمونه اولیه‌اش ساخته شود. این فعالیت فرآیند را «می آزماید» تا بتوان بالایش‌ها را انجام داد.

پالایش و جایگزینی(پیاده سازی). بر پایه بازخور ناشی از نمونه اولیه، فرآیند تجاری پالایش شده و سپس درون یک سیستم تجاری قرار می گیرد.

فعالیتهای BPR مذکور گاهی اوقات در ارتباط با ابزارهای تحلیل جریان کاری به‌کار گرفته می‌شوند. قصد این ابزارها ساخت یک مدل از جریان کاری موجود در تلاش برای بهتر تحلیل کردن فرآیندهای فعلی است. به‌علاوه تکنیک‌های مدل‌سازی معمولاً مرتبط با فعالیتهای مهندسی فرآیند مانند برنامه‌ریزی راهبرد اطلاعات و تحلیل حوزهٔ تجاری (فصل ۱۰) می‌تواند برای پیاده‌سازی و اجرای اولین چهار فعالیتی به‌کار گرفته شد که در مدل فرآیند تشریح شده است.

۳۰-۱-۴ کلمات هشدار

نامعمول نیست که یک رهیافت تجاری جدید ـ در این مورد BPR ـ در درجهٔ اول به‌عنوان یک اکسیر تلقی شده و بعد چنان به‌شدت مورد انتقاد قرار بگیرد که تبدیل به یک موجود فرومایه گردد. طی گذشت سال‌ها، مجادله بر سر کارآیی BPR (مثلاً، [BLE93][1]، [DIC95][2]) شرکت گرفته است. در خلاصه‌ای بسیار خوب از مورد حمایت و عدم حمایت از BPR، ویز[WEI95][3] بحث را به‌طریق ذیل خلاصه می‌کند:

این‌که BPR را به‌عنوان یک هوس بیهودهٔ گلوله ـ نفره‌ای (راهکار برجسته ای که همه مشکلات را حل و فصل نماید .م) دیگر طرح نماییم، وسوسه‌انگیز است. از چندین دیدگاه، تفکر سیستم‌ها، مردم‌افزار، تاریخ ساده ـ شما باید نرخ‌های شکست بالا را برای این مفهوم پیش‌بینی کنید. نرخ‌هایی که ظاهراً توسط شواهد عملی و تجربی حاصل شده‌اند. برای بسیاری از شرکت‌ها، ظاهراً گلوله نقره‌ای گم شده است. برای دیگران، تلاش مهندسی مجدد به‌طور آشکار مفید بوده است.

BPR می‌تواند مؤثر باشد چنانچه افراد با انگیزه و آموزش دیده آن را به‌کار گیرند که بدانند مهندسی مجدد فرآیند یک فعالیت مستمر است. چنانچه BPR به‌طور مؤثر عمل و اجرا گردد. سیستم‌های اطلاعات در پروسهٔ تجارت بهتر یکپارچه می‌شوند. مهندس مجدد کاربردهای قدیمی‌تر می‌تواند در بافت راهبرد

1. Bleakley, F.R.
2. Dickinson, B.
3. Weisz , M.

تجاری یابه ـ گسترده مورد آزمون واقع شده، اولویت‌ها برای مهندسی مجدد نرم‌افزار به‌طور هوشمندانه‌ای مقرر شوند.

اما حتی اگر مهندسی مجدد تجارت، یک راهبرد است که توسط شرکتی مردود اعلام شده، مهندسی مجدد نرم‌افزار کاری است که باید[1] انجام شود. ده‌ها هزار سیستم‌های میراثی ـ کاربردهایی که برای موفقیت تجارت‌های بزرگ و کوچک حیاتی هستند ـ نیاز شدیدی به ترمیم و ساخت مجدد دارند.

۳۰-۲ مهندسی مجدد نرم افزار

این سناریو بسیار رایج است. یک برنامه کاربردی نیازهای تجاری شرکتی را برای ۱۰ یا ۱۵ سال برآورده کرده است. طی این مدت، صحیح عمل کرده، تطبیق‌پذیر بوده و در بسیاری دفعات ارتقاء یافته است. افراد به این کار با بهترین نیت‌ها نزدیک شدند ولی روش‌های به‌کارگیری مهندسی نرم‌افزار خوب همیشه به گوشه‌ای افتاده‌اند (تحت فشار موضوعات دیگر). اینک این برنامه کاربردی پایا نیست. هنوز عمل می‌کند ولی هر بار که تغییری امتحان می‌شود اثرات جانبی غیر منتظره و حدی پیش می‌آیند. با این حال، به‌کارگیری آن باید به رشد و تکامل خود ادامه دهد. چه باید کرد؟

نرم‌افزاری‌که قابل نگهداری نمی‌باشد یک مشکل جدید نیست. در حقیقت، تأکید گسترش یافته روی مهندسی مجدد نرم‌افزار توسط کوه یخ نگهداری نرم‌افزار رانیده شده است که برای سه دهه در حال ساخته شدن بوده است.

ارجاع به وب

SEI . منابع متعددی برا در حوزه مهندسی مجدد نرم افزار پیشنهاد نموده، در آدرس زیر قرار داده است :
www.sei.cmu.edu /reengineering

۳۰-۲-۱ نگهداری نرم افزار

سی سال پیش، نگهداری نرم‌افزار به‌عنوان یک «آبسبرگ» (کوه یخی) [CAN72][2] توصیف شد. ما امیدواریم که آن‌چه در درجه اول قابل مشاهده بود هنوز به قوت خود باقی باشد ولی می‌دانیم که، تودهٔ عظیمی از مشکلات بالقوه و هزینه در زیر سطح قرار دارد. در اوائل دههٔ ۱۹۷۵ آیسبرگ نگهداری آن‌قدر بزرگ بود که یک ناو هواپیمابر را غرق کند. امروزه، می‌تواند به‌راحتی کل نیروی دریایی را غرق کند!

نقل قول

قابلیت نگهداری برنامه و قابلیت فهم برنامه، مفاهیمی موازی است : هر قدر برنامه ای سخت تر درک شود، نگهداریش مشکل تر خواهد بود.

نگهداری نرم‌افزار فعلی می‌تواند ۶۰٪ تمامی تلاش‌ها را توجیه کند که از سوی یک سازمان توسعه‌دهنده، هزینه و اجرا شده است و این درصد همچنان هر چه که نرم‌افزار بیشتر تولید می‌شود، رو به افزایش است. [HAN93][3] خوانندگانی که در جریان نیستند، شاید سؤال کنند که چرا این مقدار نگهداری در مقیاس زیاد لازم است و چرا تلاش زیادی در این رابطه مصروف داشته شده است. اسبرن و چلینکوفسکی [OSB90][4] یک پاسخ جزئی و نه کامل ارائه داده‌اند :

1.must
2.Canning , R.
3.Manna,M.
4.Osborne, W.M and E.J.chikfskey

بیشتر نرم‌افزاری که امروزه بدان وابسته هستیم به‌طور متوسط ۱۰ تا ۱۵ سال عمر دارند. حتی زمانی‌که این برنامه‌ها با به‌کارگیری بهترین طراحی خلق شدند و فنون برنامه‌سازی شناخته شده در آن زمان [که اغلب چنین نبودند] نیز به‌کار رفتند. آنها خلق شدند زمانی‌که اندازهٔ برنامه و فضای ذخیره از ملاحظات اساسی بودند. سپس به پایگاه‌ها و سکوهای جدید نقل مکان کردند و تنظیم شدند برای تغییرات در فنآوری سیستم ماشین و تجهیزات، و ارتقاء پیدا کردند تا نیازهای جدید را برآورده کنند ـ تماماً بدون توجه کافی به معماری کلی.

نتیجه ساختارهای با طراحی ضعیف، برنامه‌سازی ضعیف، منطق ضعیف و مستندسازی ضعیف سیستم‌های نرم‌افزاری بود که اینک ما قرار است آنها را به مورد اجرا بگذاریم.

ماهیت همه جاحاضر "تغییر" زیربنای تمام امور نرم‌افزاری است. تغییر گریزناپذیر است زمانی‌که سیستم‌های منکی به کامپیوتر ساخته می‌شوند. بنابراین، باید مکانیسم‌هایی را راه‌اندازی کنیم برای ارزیابی، کنترل و انجام اصلاحات که نیازها برطرف شوند.*

براساس مؤلفهٔ پاراگراف‌های فوق، خواننده ممکن است اعتراض کند که: «... ولی من ۶۰٪ وقت خود را صرف رفع اشتباهات در برنامه‌ای‌که نوشته‌ام نخواهم کرد.» نگهداری نرم‌افزار البته به‌مراتب چیزی فراتر از «رفع اشتباهات» است. ما ممکن است نگهداری را بر پایه شرح چهار فعالیت تعریف کنیم. [Swa76][1] که پس از اتمام کار برنامه‌نویسی باید بدان‌ها پرداخت. در فصل ۲ ما چهار فعالیت متفاوت نگهداری را تعریف کردیم: *نگهداری تصحیحی*[2]، *نگهداری تطبیقی*[3]، *نگهداری تکمیلی*[4] یا ارتقایی و *نگهداری بازدارنده*[5] یا مهندسی مجدد[6]. فقط ۲۰٪ از کل کار نگهداری صرف «رفع اشتباهات» می‌شود. ۸۰٪ مابقی صرف تطبیق سیستم‌های فعلی با تغییرات محیط بیرونی می‌شود که ایجاد موارد تکمیلی مورد درخواست کاربران و مهندسی مجدد یک کاربرد برای استفاده درآینده از آن جمله است. زمانی‌که نگهداری دربردارندهٔ تمام این امور تلقی می‌گردد، بسیاً ساده می‌شود ادراک کرد که چرا تلاش زیادی را می‌طلبد.

۳۰-۲-۲ یک مدل فرآیند مهندسی مجدد نرم افزار

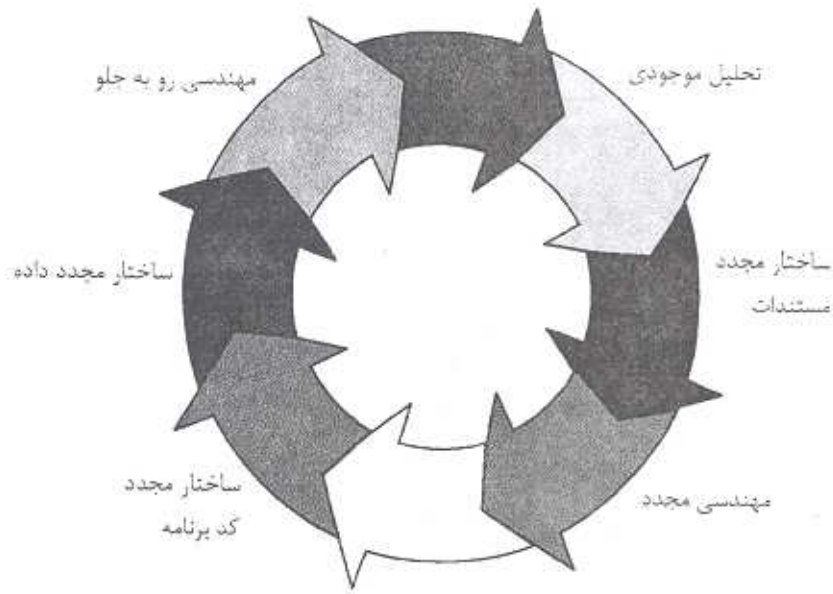
مهندسی مجدد زمان می‌برد، هزینهٔ زیادی دارد و منابعی را مصرف می‌کند که برای ـ رد دیگرضروری هستند. بنا به تمام این دلایل، مهندسی مجدد ظرف یک یا چند ماه و حتی چند سال حقق پیدا نمی‌کند. مهندسی مجدد سیستم‌های اطلاعات، فعالیتی است که منابع فنآوری اطلاعا ـ ، برای

---

* برای مطالعهٔ عمیق در خصوص دگرگونی و هدایت آن، خوانندگان علاقه‌مند می‌توانند به مطالعهٔ کتاب "هدایت تغ ـ (Driving Change) تألیف پروفسور وبند و برگردان مترجم، مراجعه نمایند.

1.Sw son,E.B.
2.corr ctive maintenance
3.adap ive maintenance
4.perfe tive maintenance
5.prev ntive maintenance
6.reengineering

سال‌ها مصرف می‌کند. بدین خاطر است که هر سازمانی به یک راهبرد پراگماتیک برای مهندسی مجدد نرم‌افزار نیاز دارد.



شکل۳۰-۲یک مدل فرایند مهندسی مجدد نرم افزار

یک راهبرد قابل اجرا در مدل فرآیند مهندسی مجدد لحاظ شده است. ما این مدل را بعداً در این بخش بررسی خواهیم کرد ولی در آغاز، چند اصل اساسی را مطرح می‌کنیم.

مهندسی مجدد یک فعالیت تجدید ساخت بوده و ما می‌توانیم مهندسی مجدد سیستم‌های اطلاعات را بهتر درک کنیم اگر یک فعالیت آنالوگی را در نظر داشته باشیم: تجدید بنای یک ساختمان. موقعیت ذیل را در نظر بگیرید:

بیایید تصور کنیم که شما خانه‌ای خریده‌اید که در کشوری دیگر واقع شده است. شما هیچ‌گاه به‌دقت آن را ندیده‌اید ولی آن را به قیمتی بسیار پایین خریده‌اید با این تذکر که شاید مجبور شدید آن را کاملاً بازسازی کنید. چگونه عمل خواهید کرد؟

- قبل از شروع به بازسازی، عقلانی به نظر می‌رسد که خانه را بازرسی کنید. تا دریابید که آیا به بازسازی نیازی هست یا خیر. شما (یا یک بازرس حرفه‌ای) یک لیست از معیارها تهیه خواهید کرد تا در نتیجه بازرسی شما نظام مند باشد.
- قبل از آن‌که خانه را کوبیده و از نو بازسازی کنید، اطمینان حاصل کنید که ساختار ضعیف باشد. اگر خانه به‌طور ساختاری سالم است شاید امکان تغییر مدل آن وجود داشته باشد، بدون آن‌که به بازسازی نیازی باشد (با هزینه‌ای خیلی کمتر، زمان بسیار کمتر).

• قبل از شروع بازسازی اطمینان حاصل کنید که چگونه خانه اولیه ساخته شده بود. از پشت دیوار نگاه کنید. سیم‌کشی‌ها، لوله‌ها و هر آن‌چه به‌طور ساختاری نوکاری شده است را دریابید. حتی اگر همه را به‌دور می‌اندازید، آنچه که بدست می‌آورید، به شما هنگام شروع کمک خواهد کرد.

• اگر بازسازی را شروع کردید، فقط مدرن‌ترین و با دوام‌ترین مواد و مصالح را به‌کار بگیرید. شاید کمی گران‌تر شود ولی به شما کمک خواهد کرد تا از نگهداری بر هزینه و زمان‌بر در آینده اجتناب کنید.

• اگر تصمیم گرفتید که بازسازی خانه را انجام دهید، انضباط را رعایت کنید. روش‌هایی را به‌کار بگیرید که به کیفیت بالا منتهی می‌شوند ـ امروز و در آینده.

هر چند اصول فوق بر روی بازسازی خانه متمرکز هستند، با این حال به‌طور برابر شامل مهندسی مجدد سیستم‌های مبتنی بر کامپیوتر و کاربردهایش نیز می‌شود.

برای اجرای این اصول، ما مدل فرآیند مهندسی مجدد نرم‌افزار را که شش فعالیت را تعریف می‌کند، نمایش داده شده در شکل ۳۰-۲ به‌کار خواهیم بست. در بعضی موارد، این فعالیت‌ها در یک توالی خطی رخ می‌دهند ولی این امر همیشه صدق نمی‌کند. مثلاً، شاید مهندسی معکوس (ادراک کارهای درونی یک برنامه) قبل از بازسازی مستندات نتواند شروع شود.

پارادایم مهندسی مجدد نمایش داده شده در شکل، یک مدل چرخه‌ای است. این بدان معنی است که هر فعالیت ارائه شده به‌عنوان قسمتی از پارادایم می‌تواند دوباره بازدید شود. برای هر چرخه خاص، فرآیند می‌تواند پس از یکی از این فعالیت‌ها به‌اتمام برسد.

تحلیل موجودی. هر سازمان نرم‌افزاری باید لیست موجودی از تمام برنامه های کاربردی را داشته باشد. این موجودی می‌تواند چیزی بیش از یک مدل صفحه گسترده در بردارندهٔ اطلاعاتی که توضیحات تفصیلی (مثلاً، اندازه، سن و حساسیت تجاری) هر برنامه کاربردی فعال، نباشد.

تحلیل موجودی

از طریق مرتب کردن این اطلاعات طبق اهمیت تجاری، طول مدت، نگهداری کنونی و دیگر معیارهای مهم داخلی، کاندیداهای مهندسی مجدد، ظاهر می‌شوند. منابع سپس می‌تواند به تقاضاهای کاندیدا برای کار مهندسی مجدد، اختصاص داده شود.

توجه به این‌که موجودی باید در یک چرخهٔ منظم بازدید مجدد شود، اهمیت دارد. وضعیت به‌کارگیری‌ها و کاربردها (مثلاً اهمیت تجاری) می‌تواند یک کارکرد زمانی را تغییر داده و در نتیجه، اولویت‌های مهندسی مجدد تغییر وضعیت خواهند داد.

تجدید ساخت مستندات. مستندسازی ضعیف علامت تجاری بسیاری از سیستم‌های میراثی است. ولی ما در مورد آنها چکار خواهیم کرد؟ گزینه های ما کدام ها خواهد بود؟

۱. ایجاد و خلق مستندسازی تاکنون بیش از اندازه وقت‌گیر بوده است. اگرسیستم کار می‌کند، ما با آنچه داریم، زندگی می‌کنیم. در بعضی موارد، این راهکار درست است. امکان آن‌که مستندسازی برای صدها برنامهٔ کامپیوتری دوباره خلق شود، وجود ندارد. اگر برنامه‌ای نسبتاً ایستا است، به

گامهایی که در اینجا برای یک خانه اشاره شده است، واضح و بدیهی است. اطمینان حاصل نمایید که ملاحظات شما در خصوص نرم افزار مشتمل بر گامهایی است که واضح و بدیهی می باشند. در مورد آن نیک بیاندیشید. پول زیادی درخطر است.

به اندازه ای مستندات تهیه کنید که برای درک کامل نرم افزار لازم است، نه حتی یک صفحه بیشتر.

آخر عمر مفید خود رسیده است و احتمال آن‌که تغییر شگرفی در آن حاصل شود، وجود ندارد، رهایش کنید.

۲. مستندسازی باید به‌روز و به‌هنگام باشد ولی ما منابع محدودی داریم. ما از یک رهیافت «مستند کردن هنگام ارجاع» استفاده می‌کنیم. شاید ضروری نباشد که کاملاً یک برنامه کاربردی را مستندسازی مجدد کنیم. ترجیحاً، آن قسمت‌های سیستم که به‌طور مستمر در حال حاضر دچار تغییر هستند به‌طور کامل مستندسازی می‌شوند. طی گذشت زمان، مجموعه‌ای از مستندسازی مفید و مرتبط بدست خواهد آمد.

۳. سیستم دارای اهمیت تجاری بوده و باید کاملاً مستندسازی شود. حتی در این حالت، یک رهیافت زیرکانه، کاستن مستندسازی در حداقل ضرورت است.

هر یک از این گزینه‌ها مناسب می‌باشد. یک سازمان نرم‌افزاری باید انتخابی داشته باشد که به بهترین نحو برای هر مورد مناسب است.

مهندسی معکوس[1] واژهٔ «مهندسی معکوس» ریشه‌اش در دنیای سخت‌افزار است. شرکتی یک محصول سخت‌افزاری رقابتی را از هم جدا می‌کند تا «اسرار*» طراحی ساخت رقیب را دریابد. اگر طراحی و مشخصات ساخت رقیب دردسترس باشند، این اسرار را به‌راحتی می‌توان فهمید. ولی این سندها اموال اختصاصی بوده و در دسترس شرکت مشغول به‌کار مهندسی معکوس نمی‌باشند.

به‌طور ماهیتی، مهندسی معکوس موفق یک یا چند مشخصهٔ ساخت و طراحی را بدست می‌دهد و این امر از طریق نمونه‌هایی واقعی که از محصول در دست است، انجام می‌شود.

مهندسی معکوس برای نرم‌افزار کاملاً مشابه است. در اغلب موارد، در هر حال، برنامه‌ای‌که باید مهندسی معکوس شود یک برنامهٔ رقابتی نیست. ترجیحاً، کار خود شرکت است (اغلب سال‌ها پیش انجام شده). «اسرار» مربوطه که باید درک شوند گنگ هستند زیرا هیچ مشخصاتی ارائه نشده است. بنابراین، مهندسی معکوس برای نرم‌افزار، فرآیند تحلیل یک برنامه در تلاش برای خلق یک بازنمایی از برنامه در سطح بالاتر انتزاع، نسبت به کد منبع است. مهندسی معکوس فرآیند بازیابی طراحی است. ابزارهای مهندسی معکوس، داده‌ها را استخراج کرده و اطلاعات طراحی رویه‌ای و معماری را از برنامه فعلی کسب می‌کنند.

تجدید ساخت برنامه. رایج‌ترین انواع مهندسی مجدد (عملاً، کاربرد واژه مهندسی مجدد در این مورد قابل بحث است) تجدید ساخت برنامه است. بعضی از سیستم‌های میراثی دارای یک آرایش برنامه نسبتاً محکم هستند ولی بیمانه های منفرد به‌طریقی برنامه‌نویسی شده‌اند که فهم آنها را دشوار می‌کند و

1. reverse engineering
* Secrets

شامل آزمون و نگهداری آنها نیز می‌شود. در چنین مواردی، برنامه درون بیمانه‌های مظنون را، می‌توان دوباره ساخت.

برای انجام این‌کار، کد منبع با به‌کارگیری یک ابزار تجدید ساخت، تحلیل می‌شود. موارد نقض برنامه‌ریزی ساختیافته مورد توجه قرار گرفته و بعد از آن، برنامه دوباره ساخته می‌شود (این‌کار می‌تواند به‌طور خودکار انجام شود). برنامهٔ تجدید ساخت شدهٔ حاصل، بازبینی و آزموده می‌شود تا اطمینان حاصل شود که هیچ آنومالی بوجود نیامده باشد. مستندات برنامه‌های داخلی به‌هنگام می‌شود.

تجدید ساختار داده‌ها. برنامه‌ای با معماری داده‌های ضعیف برای تطبیق و ارتقاء مشکل دارد. درحقیقت، برای بسیاری از کاربردها معماری داده‌ها بیشتر با دوام دراز مدت یک برنامه سر و کار دارد تا این‌که با خود کد منبع ارتباط داشته باشد.

برخلاف تجدید ساخت برنامه که در سطح نسبتاً پایین از انتزاع رخ می‌دهد، ساختار داده‌های یک فعالیت مهندسی مجدد در مقیاسی کامل است. در اغلب موارد، تجدید ساختار داده‌ها با یک فعالیت مهندسی معکوس شروع می‌شود. معماری داده‌های فعلی منفک شده و مدل‌های داده‌های ضروری تعریف شده‌اند (فصل ۱۲). اشیاء داده‌ای و صفات خاصه تعریف میشوند و ساختار داده‌های فعلی برای کیفیت، دوباره بررسی میشوند.

زمانی‌که ساختار داده‌ها ضعیف باشد (مثلاً فایل‌های تخت مرتباً استفاده می‌شوند، زمانی‌که رهیافت رابطه‌ای به‌شدت فرآوری و بردارش را ساده می‌کند)، داده‌ها تجدید مهندسی می‌شوند.

از آن‌جا که معماری داده‌ها اثر شدیدی روی معماری برنامه دارد و الگوریتم‌هایی که آن را عمومی می‌کند، تغییرات در داده‌ها به‌طور قطع به تغییرات معماری و یا سطح برنامه منجر خواهد شد.

مهندسی پیشرو. در یک دنیای ایده‌آل، برنامه‌های کاربردی با به‌کارگیری «موتور مهندسی مجدد» دوباره ساخته می‌شوند. برنامهٔ قدیمی باید وارد موتور شود، تحلیل شده دوباره ساختارمند گردد و بعد دوباره به شکلی به‌وجود آید بهترین جنبه‌های کیفیت نرم‌افزار را نمایش دهد. در کوتاه‌مدت، متحمل نیست که چنین «موتوری» ظاهر شود ولی فروشندگان CASE ابزارهایی را معرفی کرده‌اند که زیرمجموعه محدودی از توانمندی‌های معرفی شده در حوزه‌های کاربرد ویژه را فراهم می‌کند (مثلاً، کاربردهایی که با به‌کارگیری سیستم پایگاه داده‌های ویژه، اجرا می‌شوند). مهم‌تر آن‌که این ابزارهای تجدید مهندسی به‌طور فزاینده‌ای در حال پیچیده‌تر شدن هستند.

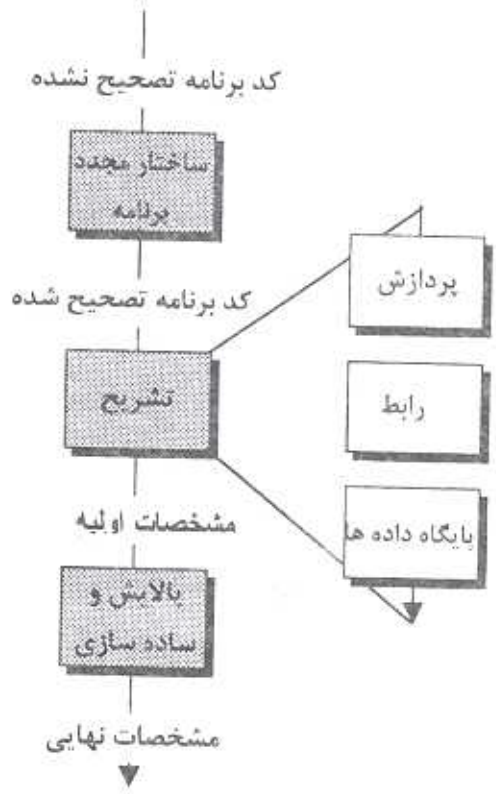
مهندسی رو به جلو(پیشرو)، که ترمیم[1] و بازیابی[2] نیز نامیده می‌شود [CHI90][3] نه فقط اطلاعات طراحی را از نرم‌افزار موجود استخراج می‌کند بلکه این اطلاعات را برای تعمیر یا تجدید ساخت سیستم موجود به‌کار می‌گیرد تا کیفیت کلی آن را ارتقاء دهد. در اغلب موارد، نرم‌افزار تجدید مهندسی شده،

1.renovation
2.reclamation
3.Chikofsky,E.J.

کارکرد سیستم موجود را دوباره پیاده‌سازی کرده و نیز کارکردهای جدید را اضافه کرده و / یا عملکرد کلی را بهبود می‌بخشد.

۳۰-۳ مهندسی معکوس

مهندسی معکوس تصویری از «دریچهٔ جادو» را متصور است. ما یک فهرست‌بندی بدون ساختار و منبع مستندسازی ناشده را وارد شکافی می‌کنیم و بعد از انتهای دیگر، مستندسازی شده‌های کامل برای برنامهٔ کامپیوتر خارج می‌شوند. متأسفانه، شکاف جادویی وجود خارجی ندارد. مهندسی معکوس می‌تواند اطلاعات طراحی را از کد منبع استخراج کند ولی سطح استخراج کامل بودن مستندسازی بسته به میزان کارکرد ابزارها و کار تحلیل‌گر انسانی همراه با هم و هدایت‌دهی فرآیند به‌شدت متغیر هستند [CAS 88].



شکل ۳۰-۳ فرآیند مهندسی معکوس

سطح انتزاع و تجرید یک فرآیند مهندسی معکوس و ابزارهای به‌کار رفته جهت اجرای آن، به پیچیدگی اطلاعات طراحی ارجاع می‌دهد در رابطه با اطلاعات طراحی که باید از کد منبع استخراج شود. به‌طور ایده‌آل، سطح تجرید باید در حد امکان بالا باشد. یعنی آن‌که، فرآیند مهندسی معکوس باید توانایی تجرید بازنمایی‌های طراحی رویه‌ای (سطح پایین تجرید) را داشته باشد، برنامه و اطلاعات ساختار داده‌ها

(به‌نوعی سطح بالاتر تجرید)، مدل‌های کنترل داده و جریان (یک سطح نسبتاً بالا از تجرید) مدل‌های رابطه – موجودیت [سطح بالای تجرید] . همان‌طور که سطح تجرید[1] بالا می‌رود، مهندس نرم‌افزار اطلاعاتی را بدست می‌آورد که درک برنامه را تسهیل می‌کند.

تکمیل بودن[2] یک فرآیند مهندسی معکوس به سطح جزئیاتی که در آن سطح تجرید فراهم شده است، اشاره می‌کند. در اکثر موارد، تکمیل بودن همراه با افزایش سطح تجریدکاهش پیدا می‌کند. مثلاً، یک فهرست‌بندی کد منبع مفروض، نسبتاً به آسانی ایجاد یک طراحی رویه‌ای کامل ساده است. بازنمایی‌های جریان داده‌های ساده را نیز می‌توان بدست آورد. ولی به مراتب از ایجاد یک مجموعه از داده‌های دیاگرام‌های جریان یا مدل‌های موجودیت- رابطه مشکل‌تر است.

کامل بودن، مقدار ریادی وابسته به تحلیل انجام شده شخصی است که عمل مهندسی مجدد را انجام می‌دهد. تعامل[3] به درجهٔ دخیل و یکپارچه شدن انسان با ابزارهای خودکار شده اشاره می‌کند تا یک فرآیند مهندسی معکوس مؤثر و کارآمد به‌وجود آید. در اغلب موارد، همان‌طور که سطح تجرید افزایش می‌یابد تعامل یا کامل بودن تحت فشار قرار می‌گیرد.

چنانچه هدایت‌دهی[4] فرآیند مهندسی معکوس یک طرفه باشد، تمام اطلاعات تجرید شده از کد منبع در اختیار مهندس نرم‌افزار قرار داده می‌شود که می‌تواند طی هرگونه فعالیت نگهداری از آن استفاده کند. اگر هدایت‌دهی دو طرفه باشد، اطلاعات، ابزار تجدید مهندسی را تغذیه می‌کند که در تلاش تجدید ساختاردهی و یا ایجاد مجدد برنامه قدیمی است.

فرآیند مهندسی معکوس در شکل ۳۰-۳ ارائه شده است. قبل از شروع فعالیت‌های مهندسی معکوس، کد منبع بدون ساختار[5] ("پلشت") درباره ساختارمند می‌شود (بخش ۳۰-۴-۱) تا فقط در بردارندهٔ سازه‌های برنامه‌نویسی ساختارمند گردد.[6] این امر خواندن کد منبع را ساده‌تر کرده و پایه‌ای برای تمام فعالیت‌های مهندسی معکوس متعاقب آن را بنا می‌نهد.

هستهٔ مهندسی معکوس، فعالیتی به‌نام استخراج تجریدها[7] است. مهندس باید برنامه قدیمی را ارزیابی کرده و از کد منبع (اغلب مستند نشده) مشخصات معنادار یک پردازش را که اجرا شده است، تجرید نماید و این امر شامل رابط کاربر ایجاد شده و ساختارهای داده‌های برنامه یا پایگاه داده‌های به‌کار رفته است.

ارجاع به فصلها

نمادهایی که اینجا توضیح داده شده، در فصل ۱۲ به تفصیل تشریح شده اند.

سه موضوع مهندسی معکوس قابل تمیز می باشند. سطح تجریدی، تکامل و جهت گیری.

ارجاع به وب

دی آر وی (DRPV) منابعی مفید را برای فعالیت مهندسی مجدد در آدرس زیر قرار داده است :
www.sel.iit.nrc.ca /project/dr/r.htm

1.abstraction level
2.completeness
3.interactivity
4.directionality
5.restructured

۶. برنامه می تواند به طور خودکار با استفاده از " موتور ساختار مجدد" بازسازی شود (استفاده از ابزارهای CASE که برای بازسازی کد برنامه به کار می رود)

7.extract abstractions

۳۰-۳-۱ مهندسی معکوس برای درک پردازش

اولین فعالیت مهندسی معکوس واقعی با تلاش برای فهم و استخراج تجریدهای رویه‌ای تعیین ارائه شده توسط کد منبع می‌باشد. برای درک تجریدهای رویه‌ای، کد (در سطوح مختلف تجرید) می‌شود: برنامه، سیستم، جزء، الگو و جملات در نظر گرفته می‌شوند. توالی کارکردی کلی تمام سیستم کاربردی باید قبل از انجام کارهای تفصیلی‌تر مهندسی معکوس، ادراک شود. این امر بافتی را به تحلیل بیشتر مقرر داشته و شناختی را نسبت به مباحث و موضوعات عملیات درونی برنامه‌های کاربردی سیستم فراهم می‌کند. هر یک از برنامه‌هایی که کاربرد سیستم را تشکیل می‌دهند، نمایشگر یک تجرید کارکردی در سطح بالایی از جزئیات می‌باشد. یک دیاگرام بلاک، برای بازنمایی تعامل بین این تجریدهای کارکردی، خلق می‌شود. از هر یک مقداری ربر ـ کارکرد تهیه کرده و بازنمایی یک تجرید رویه‌ای تعریف شده هستند. یک روایت پردازش‌گری برای هر جزء خلق شده است. در بعضی موقعیت‌ها ـ سیستم، برنامه و مشخصات جزء، از قبل وجود دارند. زمانی‌که چنین موردی وجود داشته باشد، مشخصات برای تطابق با کد فعلی بازبینی می‌شوند.[1]

مسائل، زمانی‌که که درون جزء در نظر گرفته می‌شود، پیچیده‌تر می‌شوند. مهندس در جستجوی بخش‌هایی از کد است که نمایشگر الگوهای رویه‌ای ژنریک هستند. تقریباً در تمام اجزاء، یک قسمت از کد، داده‌های مربوط به پردازش را تهیه می‌کند (درون بیمانه)، بخش متفاوتی از کد، پردازش را انجام می‌دهد و بخش دیگری از کد، نتایج پردازش برای صدور از جزء را تهیه می‌کند. درون هر یک از این اجزاء و بخش‌ها، زمانی‌که با الگوهای کوچک‌تر روبه‌رو می‌شویم مثلاً، اعتبار داده‌ها و کنترل مرزها، اغلب در محدودهٔ بخشی از کد واقع می‌شوند که داده‌های پردازش را تهیه می‌کنند.

برای سیستم‌های بزرگ، مهندسی معکوس عموماً با به‌کارگیری راهکار خودکار شده، انجام می‌شود. ابزارهای CASE برای «تشریح» معنا شناختی کد موجود به‌کار می‌روند. خروجی این فرآیند سپس به ابزارهای ساخت مجدد و ابزارهای مهندسی پیشرو انتقال می‌یابد تا فرآیند مهندسی مجدد کامل شود.

نقل قول

هیجانی که برای درک و فهم وجود دارد، درست به مانند همان هیجانی است که برای موسیقی وجود دارد. آن شور و شوق در کودکان مشترک است اما اندکی بعد، در بزرگسالان به خاموشی می گراید.

آلبرت اینشتین

۳۰-۳-۲ مهندسی معکوس برای درک داده ها

مهندسی معکوس داده‌ها، در سطوح متفاوت تجرید رخ می‌دهد. در سطح برنامه، ساختارهای داده‌های برنامه داخلی اغلب به‌عنوان قسمتی از یک تلاش مهندسی مجدد کلی، تحت مهندسی معکوس قرار می‌گیرند. در سطح سیستم، ساختارهای داده‌های عمومی (مثلاً، فایل‌ها، پایگاه‌های داده‌ها) اغلب مهندسی مجدد می‌شوند، تا بارادایم‌های مدیریت پایگاه داده‌های جدید دارای امکانات لازمه شوند (مثلاً حرکت از

---

۱. اغلب، مشخصه ها در ابتدای تاریخچه یک برنامه نوشته می شود و هرگز به هنگام نمی شود. و با ایجاد تغییرات، برنامه دیگر با مشخصه ها همخوانی نخواهد داشت.

فایل تخت به سیستم‌های پایگاه داده‌های رابطه‌ای یا شیءگرا). مهندسی معکوس ساختارهای داده‌های عمومی کنونی، مراحل معرفی یک پایگاه داده‌های گستردهٔ سیستمی را معین می‌کند.

ساختارهای داده‌های درونی. فون مهندسی معکوس برای داده‌های برنامهٔ داخلی روی تعریف کلاسهای اشیاء متمرکز است.[1] این امر توسط آزمون برنامه، با نیت گروه‌بندی متغیرهای مرتبط با برنامه انجام می‌شود. در بسیاری موارد، سازمان‌دهی داده‌ها درون کد، نوع داده‌های به‌دست آمده را تعیین می‌کند. برای مثال، ساختارهای رکورد، فایل‌ها، لیست‌ها و دیگر ساختارهای داده‌ها اغلب شاخص آغازین گروه‌ها را فراهم می‌کند. بروئر و لانو [BRE91][2] پیشنهاد می‌کنند که راهکار ذیل برای مهندسی معکوس کلاسها مناسب است:

۱- پرچم‌ها و ساختارهای داده‌های محلی درون برنامه را معین کنید که اطلاعات مهم دربارهٔ ساختارهای داده‌های عمومی (مثلاً، یک فایل یا پایگاه داده‌ها) را ثبت می‌کنند.

۲- رابطه بین پرچم‌ها و ساختارهای داده‌های محلی را تعریف کرده که شامل ساختارهای داده‌های عمومی نیز می‌شود. مثلاً، یک پرچم زمانی باید مقدار یک بگیرد که فایل خالی است، یک ساختار داده‌های داخلی می‌تواند به‌عنوان یک بافر که دربردارندهٔ آخرین ۱۰۰ رکورد حاصل از پایگاه داده‌های مرکزی است، خدمت کند.

۳- برای هر متغیری (درون برنامه) که نمایشگر یک آرایه یا فایل است، تمام متغیرهای دیگر که ارتباطی منطقی با آن دارند را فهرست کنید.

این مراحل یک مهندس نرم‌افزار را قادر می‌سازد تا کلاس‌های درون برنامه که با ساختارهای داده‌های عمومی تعامل می‌کنند را، معین کند.

ساختار پایگاه داده‌ها. بدون توجه به سازمان منطقی و ساختار فیزیکی، یک پایگاه داده‌ها تعریف اشیاء داده‌ای را ممکن می‌سازد که بعضی شیوه های استقرار رابطه بین اشیاء را حمایت می‌کنند. بنابراین، مهندسی مجدد یک شمای پایگاه داده‌ها، به‌نوعی دیگر از آن نیاز به ادراک اشیاء فعلی و روابط آنها دارد. مراحل ذیل [PRE94] را می‌توان برای تعریف مدل داده‌های موجود به‌عنوان یک پیش شرط اولیه مهندسی مجدد یک مدل پایگاه داده‌های جدید به‌کار برد:

۱- **یک مدل شیءای آغازین خلق کنید.** کلاسهای تعریف شده به‌عنوان مدل می‌توانند از طریق بازبینی رکوردهای درون یک پایگاه داده‌های فایل تخت یا جداول در یک شمای رابطه‌ای، حاصل شوند. اقلام درون رکوردها یا جداول صفات خاصه یک کلاس خواهند بود.

توصیه

اندک مسامحه ای در ساختار داده ها می تواند به مسائلی فاجعه آمیز در سالهای آتی منجر شود. به عنوان مثال مسئله Y2K (مشکل تاریخ سال ۲۰۰۰) را به خاطر آورید.

پرسش

کدام گامها توسط یک مهندس معکوس در ارتباط با ساختار پایگاه داده های موجود، می تواند برداشته شود؟

---

۱. برای دستیابی به یک بحث کامل از این مفاهیم شیءگرا، بخش چهارم این کتاب را مطالعه نمایید.

2.Bleakley , F.R.

۲- **کلیدهای کاندید را تعیین کنید.** صفات خاصهٔ مذکور برای تعیین این‌که آیا به‌کار رفته‌اند تا به رکورد یا جدول دیگری اشاره کنند یا خیر، آزمون می‌شوند. آنها که به‌عنوان اشاره‌گر خدمت می‌کنند، کلیدهای کاندید می‌شوند.

۳- **کلاسهای تجربی را پالایش کنید.** معین کنید که آیا کلاسهای مشابه می‌توانند به‌صورت یک کلاس واحد در هم ترکیب شوند یا خیر.

۴- **تعمیم‌ها را تعریف کنید.** کلاس‌هایی را که صفات خاصه ریادی دارند بررسی کنید تا معین شود که آیا یک سلسله مراتب کلاس باید از طریق تعمیم کلاس بالایی آن ساخته شود یا خیر.

۵- **شرکت‌پذیری‌ها را کشف کنید.** قیودی را به‌کار بگیرید که با راهکار CRC قابل قیاسند، هستند (فصل ۲۱) یا پیوندهای میان کلاس‌ها، برقرار شود و تعیین گردد.

زمانی‌که اطلاعات تعریف شده در مراحل فوق شناخته شده باشند، یک‌سری تغییر شکل‌ها [PRE94][1] را می‌توان برای نگاشتی از ساختار پایگاه‌ها به ساختار پایگاه داده‌های جدید به‌کار گرفت.

۳-۳-۳۰ مهندسی معکوس رابط کاربر

GUIهای پیچیده برای محصولات متکی به کامپیوتر و هر نوع سیستمی، جسته‌گسته شده‌اند. بنابراین ایجاد مجدد رابط‌های کاربر یکی از رایج‌ترین انواع امور مهندسی مجدد، شده است. اما قبل آن‌که یک رابط کاربر را بتوان دوباره ساخت، یک کار مهندسی معکوس باید انجام گیرد.

برای درک کامل یک اپرد- نیس کاربر موجود (UI)، ساختار و رفتار رابط باید مشخص شود. مرلو و همکارانش [MER93][2] سه سؤال اساسی را مطرح کردند که باید هنگام شروع مهندسی معکوس از نوع UI پاسخ داده شوند.

- اعمال اساسی (مثلاً ضربه به کلید و کلیک کردن موس) که رابط باید پردازش کند، چه هستند؟
- یک شرح فشرده از پاسخ رفتاری سیستم به این اعمال چیست؟
- منظور از «جانگزینی» چیست یا بطور دقیق‌تر، چه مفهوم معادلی از رابط در این‌جا موضوعیت دارد؟

پرسش

چگونه می توانم کار یک رابط کاربر موجود را درک نمایم؟

علامت‌گذاری مدل‌سازی رفتاری (فصل ۱۲) می‌تواند وسیله‌ای برای یافتن پاسخ‌ها به دو سؤال اول فوق فراهم کند. بیشتر اطلاعات ضروری برای ایجاد مدل رفتاری را می‌توان از طریق مشاهدهٔ علائم خارجی رابط موجود بدست آورد. ولی اطلاعات بیشتر که برای خلق مدل رفتاری ضروری است باید از کد استخراج شود.

1.Premerlani , W.J.
2.Merlo , E.

دقت در جایگزینی GUI که شاید رابط قدیمی را به‌طور دقیق بازتاب نکند، اهمیت دارد. (درحقیقت، شاید به‌شدت متفاوت باشد) اغلب ایجاد استعاره‌های تعاملی جدید ارزشمند است. برای مثال یک UI قدیمی می‌خواهد که یک کاربر فاکتور مقیاس را (با طیف ۱ تا ۱۰) فراهم کند تا تصویر گرافیکی کوچک یا بزرگ شود. یک GUI دوباره مهندسی شده شاید از یک میلهٔ لغزنده (Slide - Bar) و موس استفاده کند تا همان کارکرد را انجام دهد.

فلم‌های داده‌ای و اشیاء برای کسب اطلاعات دربارهٔ جریان داده‌ها و فهم ساختارهای داده‌های موجود که پیاده‌سازی شده‌اند، می‌باشد. این فعالیت را گاهی تحلیل داده‌ها[1] [RIC89] می‌نامند. زمانی‌که این کار انجام شد، طراحی مجدد داده‌ها شروع می‌شود. از ساده‌ترین شکل آن، یک مرحلهٔ استانداردسازی ثبت داده‌ها، تعاریف داده‌ها را روشن می‌کند تا سازگاری در بین اسامی فلم داده‌ها و با قالبهای رکورد فیزیکی درون ساختار داده‌های موجود با قالب فایل، حاصل شود. شکل دیگر تجدید طراحی که عقلانی کردن نام داده‌ها می‌باشد تضمین می‌دهد که تمام قراردادهای اسم‌گذاری داده‌ها با استانداردهای داخلی مطابق بوده و این‌که هنگام جریان داده‌ها از سیستم، حذف شده‌اند.

زمانی‌که تجدید ساختار کردن فراتر از استانداردسازی و عقلانی بودن می‌رود، اصلاحات فیزیکی در ساختارهای داده‌های فعلی انجام می‌شوند تا طراحی داده‌ها مؤثرتر گردد. این شاید به‌معنای تغییر وضعیت از یک قالب فایل به دیگری بوده یا در بعضی موارد، تغییر حالت از یک نوع پایگاه داده‌ها به نوعی دیگر باشد.

۳۰-۴ساختارسازی مجدد

ساختاردهی مجدد نرم افزار تلاش دارد که با اصلاح برنامه منبع یا داده هادر مقابل تغییرات آینده جوابگو باشد. به طور کلی ساختاردهی مجدد تمام معماری را تغییر نخواهد داد. بلکه بر جزئیات طراحی پیمانه های مستقل و ساختار داده ای محلی آنها متمرکز خواهد شد. اگر ساختار دهی مجدد از مرزهای پیمانه عبور کند و معماری نرم افزار را دستخوش تغییر سازد، تبدیل به مهندسی رو به جلو (پیشرو) خواهد شد(بخش ۳۰- ۵).

آرنولد [ARN89] مزایایی چند از ساختاردهی مجدد نرم افزار را برشمرده است:

- برنامه ها از کیفیت بالاتری برخوردار خواهند شد- مستندسازی بهبود خواهد یافت، پیچیدگی کاهش خواهد یافت، و با استانداردهای مدرن مهندسی تطابق پیدا خواهد نمود.
- با بهبود بهره وری و آموزش سریعتر، ناامیدی مهندسینی که بر یک برنامه کار می کنند کاهش خواهد یافت.
- نیروی لازم برای فعالیت های نگهداری کاهش خواهد یافت.

پرسش

تنظیم مجدد ساختار، چه مزایایی را در پی دارد؟

1.data analysis

- نرم افزار با سهولت بیشتری مورد آزمون واقع شده و اشکال زدایی می گردد.

ساختاردهی مجدد هنگامی روی می دهد که معماری اولیه یک برنامه کاربردی سخت باشد، کارهای فنی داخلی مورد نیاز باشد. و آن هنگامی آغاز می شود که تنها تغییر اندکی در پیمانه ها و داده ها مورد نیاز باشد.

ساختار مجدد برنامه هرچند تسکینی بر آلام منتج از خطاها و تغییرات جزئی باشد، مهندسی مجدد محسوب نخواهد شد. بهره واقعی هنگامی به دست می آید که ساختار داده و معماری مجدداً تنظیم گردند.

۳۰-۴-۱ ساختارسازی مجدد برنامه[2]

ساختارسازی مجدد برنامه به طراحی ای منجر می شود که کارکرد مشابهی با برنامه اصلی دارد اما از کیفیت بالاتری برخوردارست. به طور کلی فنون ساختاردهی مجدد برنامه ها، (مانند فنون ساده سازی منطقی وارنیر [WAR74]) با استفاده از جبر بولین منطق برنامه را مدل می کنند و یک سری قواعد تبدیل برای دستیابی به ساختار دهی مجدد، ایجاد می نمایند. هدف آن است که برنامه اسپاگتی گونه (درهم و بدون ساختار) به طراحی رویه ای و فلسفه برنامه سازی ساختیافته بهبود یابد.(فصل ۱۶)

فنون دیگر ساختارسازی مجدد با ابزارهای مهندسی مجدد انجام می گیرد. با نمودار تبادل منابع نگاشتی بین هر پیمانه برنامه با منابعش برقرار می سازد (نوع های داده، رویه ها، متغیرها) که با بازنمایی جریان منابع قابل تبادل بین آن پیمانه و دیگر پیمانه ها خواهند بود. معماری برنامه برای دستیابی به حداقل پیوستگی پیمانه ها ساختارسازی مجدد خواهد شد.

۳۰-۴-۲ ساختارسازی مجدد داده ها

پیش از ساختارسازی مجدد داده ها، یک فعالیت مهندسی معکوس به نام تحلیل کد منبع باید انجام گیرد. تمام جملات زبان برنامه سازی که مشتمل بر تعاریف داده یا توصیف فایلها و یا ورودی/خروجی یا توصیف رابط می باشند، باید ارزیابی شوند. این امر به مشخص کردن اقلام داده ای و اشیاء کمک می کند. و از این طریق جریان داده ها و فهم ساختار داده ای فعلی امکان پذیر خواهد شد. این فعالیت گاهی اوقات تحلیل داده ها [RIC89] نامیده می شود.

در حین کامل شدن تحلیل داده ها ساختارسازی مجدد داده ها آغاز می شود. در ساده ترین صورت گام استانداردسازی ثبت داده ها ، تعاریف داده ها را روشن می سازد تا سازگاری نام اقلام داده ای یا قالب فیزیکی ساختار داده ها موجود یا قالب فایل به دست آید. شکل دیگری از طراحی مجدد، عقلانی کردن (توجیه) نام داده ها نام دارد، که اطمینان می بخشد تمام نامهای داده ها با استانداردهای محلی تطابق دارد و نامهای مستعار حذف گردیده اند.

هنگامی که ساختارسازی مجدد به سمت استانداردسازی و عقلانی شدن پیش می رود، تغییرات فیزیکی ساختار داده ها باعث کارآیی هرچه بیشتر طراحی داده ها می شود. این به این معناست که یک

1.Ricketts,J.A.
2.code restructuring

شکل ترجمه از یک قالب فایل به دیگر قالب، یا در برخی موارد ترجمه از یک نوع ساختار پایگاه داده ها به نوعی دیگر صورت می پذیرد.

۳۰-۵ مهندسی پیشرو

برنامه‌ای با حربان کنترل که معادل گرافیکی یک کاسهٔ اسپاگتی است، با «پیمانه‌هایی» که ۲۰۰۰ جمله و دستورالعمل دارند، با خطوط توضیحی قابل فهم اندک در ۲۹۰/۰۰۰ جمله برنامه منبع و هیچ مستندسازی دیگری نباید اصلاح گردد تا نیازهای متغیر کاربر را تأمین کند. ما گزینه‌های ذیل را داریم:

پرسش

هنگامی که با یک طراحی ضعیف و یک برنامه پیاده سازی شده رو در رو می شویم، چه گزینه هایی پیش روی ماست؟

۱) ما می‌توانیم پس از اصلاح و در لابه‌لای آن بررسی کنیم، با طراحی و کد منبع تلویحی دست و پنجه نرم کنیم تا تغییرات ضروری را اعمال و پیاده‌سازی کنیم.

۲) می‌توانیم تلاش کنیم تا کارهای داخلی برنامه در تلاش برای انجام اصلاحات مؤثرتر فهمیده شوند.

۳) می‌توانیم بخش‌هایی از نرم‌افزار را که نیاز به اصلاحات دارند، دوباره طراحی، دوباره برنامه‌نویسی نموده، بیازماییم و برای این‌کار از رهیافت مهندسی نرم‌افزار برای تمام اجزای تحدیدنظر شده، استفاده کنیم.

۴) می‌توانیم برنامه را به‌طور کامل دوباره طراحی کرده، دوباره برنامه‌نویسی کرده و بیازماییم و برای این‌کار از ابزارهایی (ابزارهای مهندسی مجدد) استفاده کنیم تا به ما در فهم طراحی جاری یاری کنند.

هیچ گزینهٔ منفرد «درستی» وجود ندارد. شرایط می‌توانند اولین گزینه را دیکته کنند حتی اگر دیگر گزینه‌ها مطلوب‌تر باشند.

به‌جای انتظار کشیدن تا آن‌که تقاضای نگهداری، دریافت شود. سازمان پشتیبان یا توسعه‌دهنده‌ای که از نتایج تحلیل موجودی برای انتخاب برنامه‌ای‌که شرایط ذیل را داشته باشد، استفاده می‌کند:

۱) همچنان برای چند سال از قبل تعیین شده، مورد مصرف قرار می‌گیرد. ۲) این‌که در حال حاضر به‌طور موفقیت‌آمیزی به‌کار می‌رود. ۳) این‌که متحمل است دچار اصلاحات عمده شده یا در آینده نزدیک ارتقاء یابد. سپس، گزینهٔ ۲ و ۳ یا ۴ در فوق به‌کار خواهد رفت.

ترمیم

مهندسی مجدد بسیار شبیه به پاکیزه سازی دندان هاست. شما ممکن است به هزار دلیل آن را به تعویق بیاندازید و برای مدتی با مسامحه برخورد کنید، اما تاکتیک های تاخیری شما، عواقبی (تلخ) و درازمدت به دنبال خواهد داشت.

این راهکار نگهداری پیشگیرانه توسط Miller [MIL81] ابداع شد که آن را «تناسب مجدد ساختیافته» نام نهاد. او این مفهوم را اینگونه تعریف کرد :«به‌کارگیری فراروش‌های امروزی برای سیستم‌های دیروزی تا نیازمندیهای آینده برآورده شوند».

در نگاه اول، این پیشنهاد که یک برنامه بزرگ زمانی‌که یک نسخهٔ در حال کار قدیمی وجود دارد ایجاد شود، شاید خرج اضافی به‌نظر برسد. پیش از داوری در این مورد، نکات ذیل را در نظر بگیرید:

۱- هزینهٔ نگهداری یک خط برنامه منبع می‌تواند ۲۰ تا ۴۰ برابر هزینهٔ ایجاد اولیه همان خط باشد.

۲- طراحی دوباره معماری نرم‌افزار (برنامه و/ یا ساختار داده‌ها)، با به‌کارگیری مفاهیم طراحی مدرن، می‌تواند به‌شدت نگهداری آتی را تسهیل نماید.

۳- از آن‌جا که نمونه اولیه یک نرم‌افزار از پیش وجود دارد، بهره‌وری توسعه می‌تواند بسیار بالاتر از حد میانگین باشد.

۴- کاربر اینک با نرم‌افزار تجربه کسب کرده است. بنابراین، نیازمندیهای جدید و هدایت تغییر را می‌توان با راحتی بیشتری تعیین کرد.

۵- ابزارهای CASE برای مهندسی مجدد بعضی قسمت‌های کار را، خودکار انجام خواهند داد.

۶- یک پیکربندی نرم‌افزار کامل (مستندات، برنامه‌ها و داده‌ها) بر مبنای اتمام و تکمیل نگهداری پیشگیرانه وجود دارند.

زمانی‌که یک سازمان تولید نرم‌افزار، نرم‌افزاری را به‌عنوان محصول می‌فروشد، نگهداری پیشگیرانه به‌صورت «نسخه جدید» یک برنامه دیده و تلقی می‌شوند. یک سازندهٔ نرم‌افزار بزرگ، در منزل (مثلاً گروه تولید نرم‌افزار سیستم‌های تجاری برای یک شرکت تولیدات مصرفی بزرگ) می‌تواند ۵۰۰ تا ۲۰۰۰ برنامه تولیدی در حیطهٔ مسئولیت‌های خود داشته باشد. این برنامه‌ها می‌توانند از نظر اهمیت اولویت‌بندی شده و سپس به‌عنوان کاندیداهای نگهداری پیشگیرانه، بازبینی شوند. فرایند مهندسی پیشرو از اصول مهندسی نرم‌افزار استفاده می‌کند که شامل مفاهیم و روش‌های خلق دوباره یک برنامه کاربردی موجود نیز می‌شود. در اغلب موارد مهندسی پیشرو صرفاً یک معادل مدرن از برنامه قدیمی‌تر نیست. بلکه کاربر جدید و نیازهای فنآوری در تلاش مهندسی مجدد بکارجه می‌شوند. برنامهٔ ایجاد شده توانایی‌های کاربرد قدیمی‌تر را توسعه می‌دهد.

مهندسی نرم افزار خادم / مخدوم (C/S) در فصل ۲۸ توضیح داده شده است.

۳۰-۵-۱ مهندسی پیشرو برای معماری های خادم / مخدوم

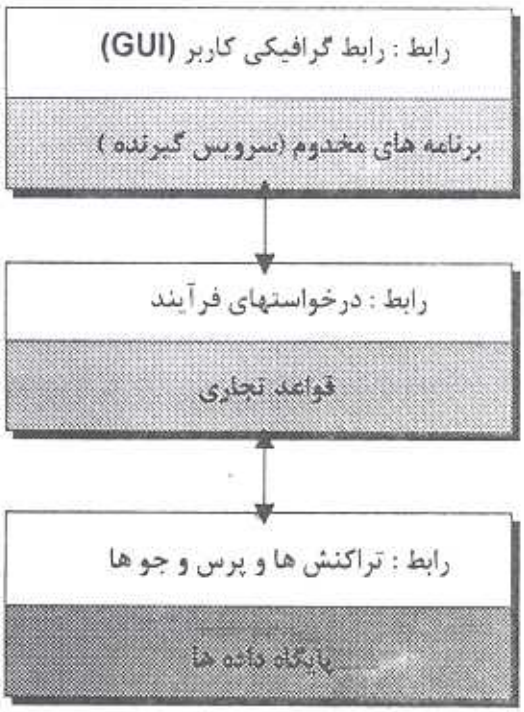
طی دههٔ گذشته، بسیاری از کاربردهای مین ـ فریم (کامپیوتر های عظیم) دوباره ایجاد شده‌اند تا معماری‌های خادم/مخدوم(C/S) را ندارکات کنند. به‌طور ماهیتی، منابع محاسباتی متمرکز (از جمله نرم‌افزار) در بین بسیاری از سکو های مخدوم توزیع شده‌اند. هر چند انواع مختلف محیط‌های توزیع شده را می‌توان دوباره طراحی کرده کاربرد معمول مین ـ فریم در معماری مشتری ـ خادم مجدداً ایجاد شده که ویژگی‌های ذیل را دارد:

- کارکرد برنامه کاربردی به هر کامپیوتر مخدوم منتقل می‌شود.
- رابط های جدید در سایت‌های مخدوم پیاده‌سازی می‌شوند.
- کارکردهای پایگاه داده‌ها برای خادم اختصاص می‌یابد.
- کارکرد تخصصی (مثلاً، تحلیل فشردهٔ محاسبه) ممکن است در سایت خادم باقی بماند.
- ارتباطات جدید، امنیت، آرشیو کردن و نیازمندیهای کنترل باید در سایت‌های مخدوم و خادم برقرار شوند.

در بعضی موارد، سیستم های خادم / مخدوم یا شئ گرا، طراحی می شوند تا جایگزین برنامه هایی در یک پروژه جدید باشند. مهندسی مجدد تنها هنگامی وارد عمل خواهد شد که یک سیستم قدیمی، با معماری جدید انسجام یابد. برخی موارد لازم است که شما سیستم پیشین را به کناری نهید و سیستمی جدید با قابلیت های تعریف شده بسازید.

مهم است که توجه داشته باشیم انتقال از کامپیوترهای بزرگ به محیط C/S نیاز به مهندسی مجدد تجارت و نرم‌افزار دارد. به‌علاوه یک *زیرساختار شبکه‌ای تجاری* [JAY94][1] باید مستقر گردد.

مهندسی مجدد برای کاربردهای C/S با یک تحلیل کامل از محیط تجاری شروع می‌شود که مین ـ فریم موجود را در برمی‌گیرد. سه لایه تجرید قابل تعریف و شناسایی می‌باشد. پایگاه داده‌ها در فونداسیون یک معماری خادم/مخدوم قرار می‌گیرد و تراکنش‌ها و پرس‌و جوهای برنامه‌های کاربردی خادم را مدیریت می‌کند. با این‌حال، این تراکنش‌ها و پرس و جوها باید در قالب مجموعه‌ای از قوانین تجاری کنترل شوند (تعریف شده توسط فرآیند تجاری فعلی یا مهندسی مجدد شده)، برنامه‌های کاربردی مخدوم مشتری توان کارکردی هدفمند را برای جماعت کاربر فراهم می‌کند.



شکل ۳۰ـ۴ برنامه های کاربردی اصلی مهندسی مجدد برای خادم / مخدوم

کارکردهای سیستم مدیریت پایگاه داده‌های موجود و معماری داده‌های پایگاه داده‌های موجود باید مهندسی معکوس شوند که به‌عنوان پیش شرط طراحی مجدد لایه فونداسیون پایگاه داده‌ها است. در بعضی موارد، یک مدل داده‌های جدید (فصل ۱۲) ایجاد می‌شود. در تمام موارد، پایگاه داده‌های C/S دوباره مهندسی می‌شود تا اطمینان حاصل شود که تراکنش‌ها به روشی سازگار انجام شده‌اند، و این‌که همگی به هنگام‌سازیها فقط از سوی کاربران مجاز صورت گرفته‌اند، این‌که قوانین تجاری مرکزی جنبهٔ اجرایی به

1.Jaychandra,Y.

خود گرفته‌اند (مثلاً، قبل از آن‌که یک رکورد فروشنده پاک شود، حادم اطمینان می‌دهد که حساب‌های قابل پرداخت مرتبط، قراردادها یا ارتباطات برای فروشنده وجود دارند)، این‌که استعلام‌ها به‌طور مؤثر تدارک می‌شوند و این‌که توانایی کامل بایگانی برقرار شده است.

لایه قوانین تجاری نمایان‌گر نرم‌افزاری است که در مخدوم و حادم مقیم است. این نرم‌افزار امور کنترل و هماهنگی را انجام می‌دهد تا تراکنش‌ها و برس و جوهای بین برنامه‌های کاربردی مخدوم و پایگاه داده‌ها قطعاً مطابق با پروسهٔ تجاری مقرر شده باشند.

لایه برنامه‌کاربردی مخدوم کارکردهای تجاری را پیاده می‌کند که توسط گروه‌های خاص کاربر - نهایی، خواسته شده‌اند. در بسیاری موارد، یک برنامه کاربردی مین‌فریم به تعدادی قسمت‌های کوچک‌تر تقسیم می‌شود و برنامه‌های کاربردی رو - میزی دوباره مهندسی می‌شوند. ارتباطات میان برنامه‌های کاربردی رو - میزی (زمانی‌که ضروری باشد) توسط لایه قوانین تجاری کنترل می‌شوند. یک بحث جامع دربارهٔ طراحی نرم‌افزار حادم/مخدوم و مهندسی مجدد در کتابهای مربوط به این موضوع در دسترس است. خوانندهٔ علاقه‌مند باید به [VAS93][1] و [INM93][2] و [BER92][3] مراجعه کند.

۳۰-۵-۲ مهندسی پیشرو برای معماری های شئ گرا

مهندسی نرم‌افزار شی‌ءگرا به توسعه پارادایم انتخاب برای بسیاری از سازمان‌های نرم‌افزاری شده است. ولی در مورد برنامه‌های کاربردی فعلی که با به‌کارگیری شیوه های متعارف به‌وجود آمده‌اند، چه خبر؟ در بعضی موارد، پاسخ آن است که چنین برنامه‌های کاربردی را همان‌طور که هستند رها کنید. ولی در دیگر موارد، کاربردهای قدیمی‌تر باید چنان مهندسی مجدد شوند که به‌راحتی در سیستم‌های شی‌ءگرا بزرگ یکپارچه شوند.

نرم‌افزار متعارف مهندسی مجدد شده در پیاده‌سازی شی‌ءگرا بسیاری از فنون مشابه مورد بحث قرار گرفته در قسمت ۴ این کتاب را، به‌کار می‌گیرد. اول، نرم‌افزار موجود مهندسی معکوس می‌شود تا داده‌ها، کارکرد و مدل‌های رفتاری مناسب حاصل شوند. اگر سیستم مهندسی مجدد شده فراتر از توان کارکردی یا رفتار برنامه کاربردی اصلی برود، مورد- کاربردها (فصل ۱۱ و ۱۲) ایجاد می‌شود.

مدل‌های داده‌های ایجاد شده در طرف مهندسی معکوس سپس به مدل‌سازیCRC (فصل۲۱) متصل شده تا پایه‌ای برای تعریف کلاس‌ها فراهم شود. سلسله مراتب گروه، مدل‌های موضوعی، مدل‌های رفتار - معطوف و زیر سیستم‌های تعریف شده‌اند و طراحی موضوعی شروع می‌شود.

همان‌طور که مهندسی پیشرو شی‌ءگرا به جلو می‌رود از تحلیل تا طراحی، یک مدل پروسهٔ CBSE (فصل ۲۷) را می‌توان پدید آورد. اگر کاربرد فعلی در دامنه و حوزه‌ای واقع است که پیش از این تحت

1.Vaskevitch , D.
2.Inmon ,W.H.
3.Berson,A.

اشغال کاربردهای شیءگرا بوده است، محتمل است که یک کتابخانهٔ اجزاء توانمند وجود داشته باشد که می‌تواند در طول مهندسی پیشرو به‌کار رود. برای آن کلاسهایی که باید از همان اول مهندسی شوند، شاید امکان داشته باشد که الگوریتم‌ها دوباره استفاده شوند و ساختارهای داده‌ها برای کاربرد متعارف فعلی مورد استفاده واقع شوند. به‌هر حال، این موارد باید دوباره طراحی شده تا با معماری شیءگرا مطابقت پیدا کنند.

۳۰-۵-۳ مهندسی پیشرو رابط های کاربر

همان‌طور که برنامه‌های کاربردی از مین ـ فریم به کامپیوتر رو ـ میزی منتقل می‌شوند، کاربرها، دیگر رابط های کاربر متکی به کاراکتر را تحمل نمی کنند. درحقیقت، یک بخش عمده از کلیه تلاش‌های انجام شده برای انتقال از مین ـ فریم (کامپیوترهای بزرگ) به محاسبات خادم/مخدوم می‌تواند صرف مهندسی مجدد رابط های کاربر برنامه کاربردی مخدوم گردد. مارلو و همکارانش [MER95][1] مدل ذیل را برای مهندسی مجدد رابط‌های کاربر ارائه کردند:

پرسش

برای مهندسی مجدد رابط کاربر کدام گامها باید برداشته شود؟

۱- درک رابط اصلی و داده‌هایی که بین آن و باقی‌ماندهٔ برنامه کاربردی حرکت می‌کنند. اگر یک GUI جدید ایجاد شود، داده‌هایی که بین GUI و باقی‌ماندهٔ برنامه حرکت می‌کنند باید با داده‌هایی که به‌طور جاری بین رابط متکی بر کاراکتر و برنامه حرکت می‌کند، سازگار باشد.

۲- مدل‌سازی مجدد رفتار رابط فعلی در یک سری تجریدهایی که در بافت یک GUI معنا و مفهوم دارند. هر چند که مدل محاوره می‌تواند به‌شدت متفاوت باشد، رفتار تجاری نمایش داده شده توسط کاربران رابط جدید و قدیم (زمانی‌که از لحاظ سناریوی به‌کارگیری در نظر گرفته می‌شود) باید یکسان باقی بماند. یک رابط دوباره طراحی شده باید هنوز هم به کاربر اجازه دهد تا رفتار تجاری مناسب را نمایش دهد. مثلاً، زمانی‌که یک پرسش پایگاه داده‌ها انجام می‌شود، رابط قدیمی نیازمند یک مجموعه طولانی از فرامین متکی به متن برای مشخص کردن پرس و جو می باشد. GUI مهندسی مجدد شده، می‌تواند نسبت به پرسش با چند تلنگر کوچک بر موشواره پرسش را پیدا کند ولی قصد و مضمون پرسش تغییری نمی‌کند.

۳- اصلاحاتی را وارد کار کنید که حالت تعامل را کارآتر نماید. شکست‌های ارگونومیک رابط فعلی مطالعه شده و در طراحی جدید GUI اصلاح گردند.

۴- GUI جدید را ساخته و یکپارچه کنید. وجود کتابخانه‌های کلاس و نسل چهارم ابزارها می‌توانند تلاش لازم برای ساخت GUI را به‌طور عمده کاهش دهد. در هر حال، یکپارچه کردن برنامه‌کاربردی فعلی نرم‌افزار می‌تواند وقت بیشتری بگیرد. باید دقت کرد تا اطمینان حاصل شود GUI تولید اثرات جانبی مخرب در برنامه کاربردی نمی‌کند.

1.Merlo,E.

۳۰-۶ مسائل اقتصادی مهندسی مجدد

در یک دنیای بی‌نقص و کامل، هر برنامه غیر قابل نگهداری بلافاصله بازنشسته و به کناری نهاده می‌شود تا با برنامه‌های کاربردی بسیار کیفی و دوباره مهندسی شده، جایگزین گردد برنامه‌هایی که توسط استفاده از روش‌های مهندسی نرم‌افزار مدرن پدید آمده‌اند ولی ما در دنیایی با منابع محدود زندگی می‌کنیم. مهندسی مجدد منابع دیگر موارد مصرف را می‌بلعد. بنابراین، قبل از آن‌که سازمانی تلاش کند تا برنامه کاربردی فعلی را دوباره مهندسی کند، باید تحلیل‌های هزینه/فایده انجام شوند.

یک مدل تحلیل مقرون به‌صرفه بودن (هزینه / فایده) برای مهندسی مجدد توسط اسنید [SNE95] ارائه شده است. نُه پارامتر تعریف شده‌اند:

P_1= هزینهٔ نگهداری سالانه جاری برای یک کاربرد

P_2= هزینهٔ عملیات سالانه جاری برای یک کاربرد

P_3= ارزش تجاری سالانه جاری یک کاربرد

P_4= هزینهٔ پیش‌بینی شدهٔ سالانه پس از مهندسی مجدد

P_5= هزینهٔ عملیات سالانه پیش‌بینی شده پس از مهندسی مجدد

P_6= ارزش تجاری سالانه پیش‌بینی شدهٔ پس از مهندسی مجدد

P_7= هزینه‌های مهندسی مجدد تخمین زده شده

P_8= تقویم زمانی مهندسی مجدد تخمین زده شده

P_9= فاکتور خطر مهندسی مجدد (P_9=1.0 اسمی است)

L= دورهٔ حیات کاری پیش‌بینی شدهٔ سیستم

هزینهٔ مرتبط با نگهداری مستمر یک برنامه کاربردی کاندیدا (به‌عبارتی مهندسی مجدد انجام نشده است) می‌تواند بدین‌طریق تعریف شود:

$$C_{maint} = [P_3 - (P_1 + P_2)] \times L \qquad (۱-۳۰)$$

هزینه‌های مرتبط با مهندسی مجدد با استفاده از رابطهٔ ذیل تعریف می‌شوند:

$$C_{reeng} = [P_6 - (P_4 + P_5) \times (L - P_8) - (P_7 \times P_9)] \qquad (۲-۳۰)$$

با به‌کارگیری هزینه‌های ارائه شده در این معادلات (۳۰-۱) و (۳۰-۲)، سود کلی مهندسی مجدد به قرار ذیل محاسبه می‌شود:

$$\text{سود/هزینه} = C_{reeng} - C_{maint}$$

تحلیل سود ارائه شده در معادلات فوق می‌تواند برای تمام کاربردهای بسیار در اولویت انجام شود که در طول تحلیل موجودی خواهد بود (قسمت ۳۰-۲-۲)، آن کاربردهایی که بیشترین سود را به‌واسطهٔ

نقل قول

شما می توانید هم اکنون مبلغی اندک به ما بپردازید، یا آنکه اندکی بعد، مبلغی بسیار به ما پرداخت کنید.

نوشته شده بر سر در یک تنظیم موتور خودکار

هزینه کردن نشان می‌دهند را می‌توان برای مهندسی مجدد، هدف قرارداد در حالی‌که روی دیگر موارد کار می‌شود و تا هنگامی‌که منابع تأمین می‌شود، به تعویق خواهند افتد.

۷-۳۰ خلاصه

مهندسی مجدد در دو سطح مقاومت انزاع و تجرید رخ می‌دهد. در سطح تجاری، مهندسی مجدد روی پروسه تجاری متمرکز می‌شود با این قصد که تغییراتی برای بهبود توان رقابتی در بعضی حوزه‌های کار تجاری، پدید آورد. در سطح نرم‌افزاری، مهندسی مجدد سیستم‌های اطلاعات و برنامه‌های کاربردی را امتحان و آزمون می‌کند با این قصد که آنها دوباره تجدید ساختار و ساختمان شوند تا کیفیت بالاتری را نمایش دهند. مهندسی مجدد پروسه تجاری (BPR) اهداف تجاری را تعریف کرده، پروسه‌های تجاری موجود را تعیین و ارزیابی می‌کند (در قالب اهداف تعیین شده)، پروسه‌های بازبینی شده را مشخص کرده و طراحی می‌کند که شامل نمونه‌های اولیه و بالایش و هماهنگ‌سازی آنها با یک کار تجاری می‌شود. نتیجهٔ BPR اغلب تعیین راه‌هایی است که فناوری‌های اطلاعات، بهتر بتوانند کار تجاری را حمایت کنند.

مهندسی مجدد نرم‌افزار در برگیرندهٔ یک‌سری فعالیت‌هایی است که شامل تحلیل موجودی، ساخت دوبارهٔ مستندات، مهندسی معکوس، برنامه‌های ساختاردهی مجدد داده‌ها و برنامه‌ها و مهندس پیشرو می‌شود. منظور از این فعالیت‌ها خلق نسخه‌هایی از برنامه‌های موجود است که کیفیت بالاتر و توان نگهداری بهتر نمایش می‌دهند ـ برنامه‌هایی که به‌خوبی فرن بیست و یکم هستند.

تحلیل موجودی یک سازمان را قادر می‌سازد تا هر کاربردی را به‌طور نظام مند ارزیابی کند با این هدف که تعیین نماید کدام کاندیداها برای مهندسی مجدد هستند. ساخت مجدد مستندات، چارچوبی از مستندسازی ایجاد می‌کند که برای پشتیبانی دراز مدت یک کاربرد لازم است. مهندسی معکوس بروسهٔ تحلیل یک برنامه است در تلاش برای کسب داده‌ها و اطلاعات معماری و رویه‌ای طراحی که مورد نیاز هستند. سرانجام مهندسی پیشرو برنامه‌ای را دوباره می‌سازد تا با به‌کارگیری روش‌های مهندس نرم‌افزار و اطلاعات حاصل در طول مهندسی معکوس، به اهدافش نایل شود.

مقرون به‌صرفه بودن مهندسی مجدد، مرتبط با پشتیبانی و نگهداری برنامه‌های کاربردی موجود است، یعنی هزینه‌های مرتبط با پشتیبانی و نگهداری یک برنامه‌کاربردی موجود با هزینهٔ پروژه‌ای مهندسی مجدد مقایسه شده و کاهش ناشی از آن در هزینه‌های نگهداری لحاظ می‌شود. تقریباً در تمام مواردی که یک برنامه عمری طولانی دارد و مدام نگهداری ضعیف نمایش می‌دهد، مهندسی مجدد یک راهبرد تجاری مقرون به‌صرفه را ارائه می‌کند.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۳۰-۱ شغل‌هایی که در ۵ سال گذشته داشته‌اید را در نظر بگیرید. نقش‌هایی را که در جریان کار بازی کرده‌اید را شرح دهید. از یک نمونه BPR شرح داده شده در قسمت ۳۰-۱-۳ استفاده کنید و به‌وسیلهٔ آن تغییراتی را برای پردازشی با سعی و تلاشی که باعث مؤثرتر شدن آن می‌شود، پیشنهاد می‌کنید.

۳۰-۲ تحقیقاتی را در مورد تاثیر و سودمندی مهندسی مجدد فرآیند تجاری انجام دهید و نظرات موافق و مخالف را برای این یافته خود ارائه کنید.

۳۰-۳ استاد شما برنامه‌هایی را که هدفی در طول دوره گسترش داده باشد را انتخاب می‌کند، برنامه‌هایتان را با بعضی دیگر در کلاس عوض کنید. در مورد برنامه‌هایتان توضیح ندهید و بحث نکنید. این باعث می‌شود در برنامه‌ای که دریافت کردید، آن‌را تکمیل کنید و افزایش دهید. (که به‌وسیلهٔ استادتان مشخص شده باشد).

الف- تمام وظائف مهندسی نرم‌افزاری که شامل خلاصه‌ای گذرا از آن است را به انجام رسانید. (اما نه به‌وسیلهٔ برنامه‌نویس آن).

ب- همهٔ اشتباهات و خطاهایی را که در طول تست یک برنامه با آن مواجه می‌شوید را نگه دارید.

ج- تجربهٔ بدست آمده خود را در کلاس مورد بحث قرار دهید.

۳۰-۴ در فهرست ویژگی‌های، تحلیل شده و معرفی شده در وبسایت SEPA جستجو کنید و سعی کنید که مقداری از سیستم عملیاتی برنامه را گسترش دهید، به قسمی که بتواند برنامه‌های موجود برای انتخاب نامزد برنامه‌ها در مهندسی مجدد را عملی کنید. سیستم شما باید در ماوراء یک تحلیل اقتصادی که در قسمت ۳۰- ۶ معرفی شده، گسترش پیدا کند.

۳۰-۵ گزینه هایی که برای مستند سازی می تواند مورد استفاده واقع شود، مانند مستند سازی با کاغذ و جوهر و موارد دیگر یا مستند ساری الکترونیکی مرسومی که بتواند به‌عنوان پایه و اساس برای بازسازی مدارک به‌کار گرفته شود، پیشنهاد دهید. (نکته: فکر جدید تکنولوژی‌های توصیفی جدید که می‌تواند در رابطه با اینترنت در برنامه استفاده شوند).

۳۰-۶ بعضی از افراد معتقدند که تکنولوژی هوش مصنوعی در مرحلهٔ تجربه‌سازی پردازش مهندسی معکوس افزایش پیدا خواهد کرد. تحقیق در این زمینه انجام دهید (مانند به‌کارگیری مهندسی معکوس) و صفحه‌ای خلاصه که حاوی این نکات باشد را بنویسید.)

۳۰-۷ چرا تکمیل یک برنامه برای دستیابی به یک مرحله (افزایشی) خلاصه مشکل است.

۳۰-۸ چرا در صورت افزایش تکامل، برنامه‌ها و فعالیت‌ها نیز باید افزایش پیدا کنند.

۳۰-۹ محصولاتی را در سه شاخه مهندسی ابزار برگشتی یافته و خصوصیات و ویژگی‌های آنها را در کلاس بیان کنید.

۳۰-۱۰ یک تفاوت بسیار کوچک بین دوباره‌سازی و مهندسی بیشرو وجود دارد. آن تفاوت چیست؟

۳۰-۱۱ نوشته‌ها را برای پیدا کردن یک یا چند صفحه در مورد مطالعاتی از پردازنده مرکزی + خادم / مخدوم بررسی کنید. خلاصه‌ای از آنها را تهیه کنید.

۳۰-۱۲ چگونه می‌توانید P_4 را از P_7 در مدل هزینه ـ سود (Cost-benefit) در بحش ۳۰-۶ مشخص کنید.

فهرست منابع و مراجع

[ARN89] Arnold, R.S., "Software Restructuring," *Proc. IEEE*, vol. 77, no. 4, April1989, pp.607-617.

[BER92] Berson, A., *Client/Server Architecture*, McGraw-Hill, 1992.

[BLE93] Bleakley, F.R., "The Best Laid Plans: Many Companies Try Management Fads, Only to See Them Flop," *The Wall Street Journal*, July 6, 1 993, p. 1.

[BRE91] Breuer, PT. and K. Lano, "Creating Specification From Code: Reverse-Engineering Techniques," *Journal of Software Maintenance: Research and Practice*, vol. 3, 1991, pp. 145-162.

[CAN72] Canning, R., "The Maintenance 'Iceberg'," *EDP Analyzer*, vol. 10, no. 10, October 1972.

[CAS88] "Case Tools for Reverse Engineering," *CASE Outlook*, CASE Consulting Group, vol. 2, no. 2, 1988, pp. 1-15.

[CHI90] Chikofsky, E.J., and J.H. Cross, II, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, January 1990, pp. 13-17.

[DAV90] Davenport, T.H. and J.E. Young, "The New Industrial Engineering: Information Technology and Business Process Redesign," *Sloan Management Review*, Sum-.ner 1990, pp. 11-27.

[DEM95] DeMarco, T., "Lean and Mean," *IEEE Software*, November 1995, pp. 101-102.

[DIC95] Dickinson, B., *Strategic Business Reengineering*, LCI Press, 1995.

[HAM90] Hammer, M., "Reengineer Work: Don't Automate, Obliterate," *Harvard Business Review*, July-August 1990, pp. 104-112.

[HAN93] Manna, M., "Maintenance Burden Begging for a Remedy," *Datamation*, April 1993, pp. 53-63.

[INM93] Inmon, W.H., *Developing Client Server Applications*, QED Publishing, 1993.

[JAY94] Jaychandra, Y., *Re-engineering the Networked Enterprise*, McGraw-Hill, 1994.

[MER93] Merlo, E., et al., "Reverse Engineering of User Interfaces," *Proc. Working Conference on Reverse Engineering*, IEEE, Baltimore, May 1993, pp. 171-178.

[MER95] Merlo, E., et al., "Reengineering User Interfaces," *IEEE Software*, January 1995, pp. 64-73.

[MIL81] Miller,J., in *Techniques of Program and System Maintenance*, (G. Parikh, ed.) Winthrop Publishers, 1981.

[OSB90] Osborne, W.M. and E.J. Chikofsky, "Fitting Pieces to the Maintenance Puzzle," *IEEE Software*, January 1990, pp. 10-11.

[PRE94] premerlani,W.J., and M.R. Blaha, "An Approach for Reverse Engineering of Relational Databases," *CACM*, vol. 37, no. 5, May 1994, pp. 42-49.

[RIC89] Ricketts, J.A., J.C. DelMonaco, and M.W. Weeks, "Data Reengineering for Application Systems," *Proc. Conf. Software Maintenance-1989*, IEEE, 1989, pp. 174-179.

[SNE95] Sneed, H., "Planning the Reengineering of Legacy Systems," *IEEE Software*, January 1995, pp. 24-25.

[STE93] Stewart, T.A., "Reengineering: The Hot New Managing Tool," *Fortune*, August 23, 1993,pp.41-48.

[SWA76] Swanson, E.B., "The Dimensions of *Maintenance*," *Proc. Second Intl. Conf Software Engineering*, IEEE, October 1976, pp. 492-497.

[VAS93] Vaskevitch, D., *Client/Server Strategies*, IDG Books, 1993.
[WAR74] Warnier, J.D., *Logical Construction of Programs*, Van Nostrand-Reinhold, 1974.
[WEI95] Weisz, M., "BPR Is Like Teenage Sex," *American Programmer*, vol. 8, no. 6, June 1995, pp. 9-15.

خواندنیهای دیگر و منابع اطلاعاتی

Like many hot topics in the business community, the hype surrounding business process reengineering has given way to a more pragmatic view of the subject. Hammer and Champy *(Reengineering the Corporation*, HarperCollins, 1993) precipitated early interest with their best selling book. Later, Hammer *(Beyond Reengineering: How the Processed-Centered Organization Is Changing Our Work and Our Lives*, Harper-Collins 1 997) refined his view by focusing on "process-centered" issues.

Books by Andersen *(Business Process Improvement Toolbox*, American Society for Quality, 1 999), Harrington et al. *(Business Process Improvement Workbook*, McGraw-Hill, 1 997), Hunt *(Process Mapping: How to Reengineer Your Business Processes*, Wiley, 1 996), and Carr and Johansson *(Best Practices in Reengineering: What Works and What Doesn It in the Reengineering Process*, McGraw-Hill, 1995) present case studies and detailed guidelines for BPR.

Feldmann *(The Practical Guide to Business Process Reengineering Using IDEFO*, Dorset House, 1998) discusses a modeling notation that assists in BPR. Berztiss *(Software Methods for Business Reengineering*, Springer, 1 996) and Spurr et al. *(Software Assistance for Business Reengineering*, Wiley, 1994) discuss tools and techniques that facilitate BPR.

Relatively few books have been dedicated to software reengineering. Rada *(Reengineering Software: How to Reuse Programming to Build New, State-of-the-Art Software*, Fitzroy Dearborn Publishers, 1999) focuses on reengineering at a technical level. Miller *(Reengineering Legacy Software Systems*, Digital Press, 1998) "provides a framework for keeping application systems synchronized with business strategies and technology changes." Umar *(Application (Re)Engineering: Building Web-Based Applications and Dealing with Legacies*, Prentice-Hall, 1997) provides worthwhile guidance for organizations that want to transform legacy systems into a Web-based environment. Cook *(Building Enterprise Information Architectures: Reengineering Information Systems*, Prentice-Hall, 1996) discusses the bridge between BPR and information technology. Aiken *(Data Reverse Engineering*, McGraw-Hill, 1996) discusses how to reclaim, reorganize, and reuse organizational data. Arnold *(Software Reengineering*, IEEE Computer Society Press, 1993) has put together an excellent anthology of early papers that focus on software reengineering technologies.

A wide variety of information sources on business process reengineering and software reengineering is available on the Internet. An up-to-date list of World Wide Web references can be found at the SEPA Web site:
http://www.mhhe.com/engcs/compsci/pressman/resources/reengineering.mhtml