مقالات تخصصی انگلیسی همراه با ترجمه فارسی

عرضه شده به صورت رایگان و اختصاصی در ایران عرضه

توجه !

این فایل از سری محصولات رایگان (فرمت PDF) ایران عرضه میباشد، لیکن

شما عزیزان میتوانید جهت تهیه مقالات تخصصی ترجمه شده این رشته به

صورت کامل و با فرمت ورد (قابل ویرایش) همراه با نسخه انگلیسی

مقاله از نشریات معتبر خارجی ISI و Sciencedirect

(Elsevier ،IEEE ،Springer ،Wiley)

اینجا کلیک نموده و با قیمت مناسب خریداری نمائید(تحویل آنی).

عنوان مقاله: تکنیک مهندسی مجدد و روش های آن

چکیده

مهندسی مجدد و الگوهای مهندسی مجدد از مفاهیم نسبتا جدیدی است که در ابتدا برای تاثیر گذاشتن بر جامعه ی مهندسی نرم افزار شروع شد. با تغییر منابع با هدف تجدید سازمان از سیستم های قدیمی به سوی تمرکز بر پیشرفت نرم افزار جدید، تجارت قادر خواهد بود زمان و منابع با ارزش زیادی را ذخیره نماید. اگرچه مزایای این رویکرد به وضوح دیده می شود اما برخی مشکلات از جمله بی تجربگی و کمبود ابزار مناسب، مهندسی مجدد سیستم را کمی مشکل می کند.

۳. تاریخچه ی مفاهیم مهندسی مجدد

در اوایل سال های انقلاب اطلاعاتی از طرف جوامع نیازی به مهندسی مجدد بیان نشده بود. در عوض، توجه جوامع به سوی کشف راه های جدید برای ساختن نرم افزارها و سخت افزارهای جدید بود. نکته ی حائز اهمیت این بود که جوامع چگونه پیشرفت سیستم جدید را که با سرعت فزاینده آشکار می شد و انتشار پیدا می کرد، اداره می کردند. تقریبا به سیستم های قدیمی که کم کم منسوخ می شدند هیچ توجهی نمی شد. به علاوه تجارت ها به سرعت در حال تغییر بودند و همراه با این تغییرات به نرم افزارهای اطلاعاتی مناسب نیاز شد که این دوران به عنوان " کمبود نرم افزار" شناخته می شد. زیرا نرم افزارها در حال ارتقا بودند ولی برای رفع نیازهای تجارت کافی نبود.

در اوایل دهه ی ۹۰، تمرکز به سرعت از پیشرفت نرم افزار جدید به مهندسی مجدد سیستم های قدیمی تغییر یافت (سیستم ها به مدت زیادی برای نگهداری و تعمیرات رشد کردند). در حقیقت در این زمان توجه زیادی به مهندسی مجدد شد و برحسب روش شناسی مهندسی مجدد پیشرفته و الگوهای آن، تجارت یکپارچه و ساختارهای آن، به رسمیت شناخته شد. مهندسی مجدد کلمه ی روز بود و مشاور مهندسی مجدد، به یک رشته تبدیل شد [BPR 99].

اما اعتقادی که مشاوران به مهندسی مجدد تجارت و نرم افزارهایش داشتند، به این آسانی هم نبود. بالغ بر نیمی از فرآیند مهندسی مجدد در آن زمان منجر به شکست شد. خیلی از آن ها به خاطر عدم تجربه و فقدان مشارکت خریدار و مشتری بود. این شکست ها هزینه های سنگینی به شرکت ها تحمیل کرد و پیشرفت سریع مهندسی مجدد را با وجود علاقه به توسعه ی مهندسی مجدد به پایان رسانید [BPR 99].

هنوز حدود ۸۰ درصد هزینه های بودجه ی سیستم اطلاعاتی تجارت مربوط به نگهداری سیستم های قدیمی می شود. به این معنی که فقط ۲۰ درصد کل هزینه، مربوط به توسعه ی سیستم جدید است. این نشان می دهد که مهندسی مجدد، سازماندهی مجدد و طراحی مجدد سیستم (یا تجارت) بسیار مهم است. اگر این هزینه ها کم شود سود زیادی برای کاربران نرم افزار به دست خواهد آمد. این حقیقت به بازگشت به جامعه ی فعال تحقیق مهندسی مجدد کمک می کند و همان طور که وارد قرن بیست و یکم می شویم، مشاهده می کنیم که مهندسی مجدد دوباره در حال پیشروی به سمت مهندسی نرم افزار است.[۱]

به هر حال از نقطه نظر تجارت، مهندسی مجدد نرم افزار ممکن است تنها روش برای ابقای سیستم های قدیمی باشد تا این سیستم ها بتوانند به وظایفشان عمل کنند. بنابراین ممکن است این رویکرد برای تطبیق با رویکردهای دیگر سیستم بسیار گران و مخاطره آمیز باشد.[۲]

مقاله اول ص ۲

مقاله دوم ص ۲

برای فهم بیشتر این موضوع، باید تخمین تقریبی از مسئله سیستم قدیمی بزنیم:

مقدار کد در سیستم های قدیمی بی اندازه است. در سال ۱۹۹۰ حدود ۱۲۰ بیلیون سطر از کد مبدا در جهان وجود داشت [Ulrich,1990]. اهداف این سیستم ها در COBOL که یک زبان برنامه نویسی خوب برای پردازش داده های تجارت می باشد، نوشته شده است. همین طور FORTRAN که زبانی برای برنامه نویسی علمی یا ریاضی است. اگرچه خیلی از این برنامه ها جایگزین شده اند اما بیشتر این برنامه ها هنوز در وظیفه ی خود باقی هستند. در ضمن پس از سال ۱۹۹۰ افزایش چشمگیری در استفاده از کامپیوتر برای فرآیند تجارت مشاهده شده است.

نگهداری از سیستم های قدیمی پرهزینه است. بنابراین مهندسی مجدد این سیستم ها عمر مفید آن ها را افزایش می دهد. مهندسی مجدد، ساختار سیستم را بهبود می بخشد، سیستم جدید را مستند و فهم آن را آسان می کند.[۱]

---

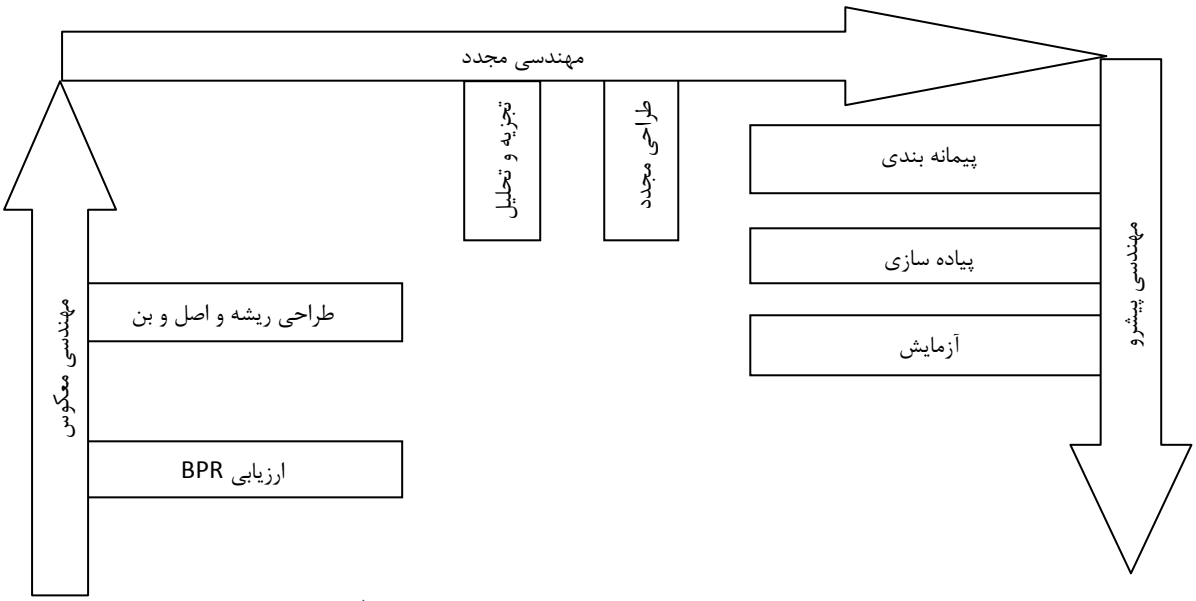
۱ مقاله دوم ص ۲

۴. مقدمه ی کوتاهی درباره ی مهندسی مجدد

مهندسی مجدد، آزمون مدل و کاربرد سیستم قدیمی موجود را به یکدیگر ربط می دهد و تکنیک های مختلفی را برای طراحی مجدد سیستم به کار می گیرد. هدف از این کار، استفاده ی مناسب و بهتر از قبل نرم افزار است. البته این کار، فرآیند آسانی نیست زیرا به روز کردن و اضافه کردن عامل جدید ممکن است باعث عدم صحت و درستی و همین طور به روزآوری مستندات سیستم شود. به خصوص اگر سیستم به مردم تفویض نشود، مهارتی در راه تفکر مهندسی مجدد به وجود نمی آید که در نهایت سیستم توانایی ترمیم سریع و کارآیی خود را در صورت بروز خرابی از دست می دهد.

کارگروه مهندسی مجدد باید توانایی دیدن جنگل انبوه را داشته باشد تا بتواند سیستم را توصیف کند. تصمیم، باید با دقت هر چه تمام تر گرفته شود. زیرا فرآیند مهندسی مجدد در مورد نگهداری آخرین کاربرها و مصرف کنندگان موجود، نیازمندی های آینده و حالت گذار از سیستم قدیمی به سیستم جدید صحبت می کند. اگر سیستم یکپارچه دوباره طراحی شود، دیگر نیازی به فرآیند مهندسی مجدد نخواهد بود. ولی فرآیند ضعیف نرم افزاری مهندسی مجدد همراه با تولیدات نهایی به آموزش مصرف کنندگان از دست رفته نیاز دارد.

تلاش در جنبه های بنیادی مهندسی مجدد مانند نگهداری، مشتری مداری، کارکردی کردن و دیگر نیازمندی ها هر اندازه هم که کوچک باشد سودمند است. این موضوع الگوی مهمی را در مهندسی مجدد بیان می کند: " تغییر، اما به مقدار اندک ممکن ".

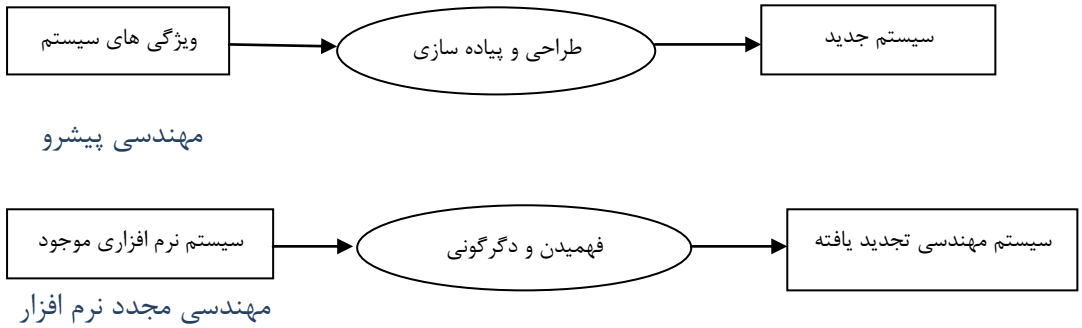


شکل ۱- مراحل فرآیند مهندسی مجدد[1]

[1] مقاله اول ص ۳

مهندسی مجدد سیستم نرم افزاری برای تکامل سیستم دو مزیت کلیدی نسبت به رویکردهای اصلی دیگر دارد:

۱- **ریسک کم:** در توسعه ی مجدد ریسک زیادی نهفته است که برای یک سازمان ضروری است. خطاها ممکن است در تشخیص سیستم، مسائل توسعه یافته و ... روی دهد.



شکل ۲- مهندسی پیشرو و مهندسی مجدد

۲- **هزینه ی کاهش یافته:** هزینه ی مهندسی مجدد نشان از کم بودن نسبت به هزینه ی توسعه ی سیستم جدید است. اولریچ [Ulrich,1990] مثالی از سیستم تجاری که برای اجرای آن ۵۰ میلیون دلار هزینه شد نقل می کند. این سیستم با ۱۲ میلیون دلار مهندسی مجدد شد. این ارقام بیان کننده ی کاهش ۴ برابری مهندسی مجدد در قبال دوباره نوشتن سیستم است.

عبارت مهندسی مجدد با مهندسی مجدد فرآیند تجارت آمیخته شده است [Hammer,1990]. مهندسی مجدد فرآیند تجارت به طراحی مجدد فرآیندهای تجارت برای کاهش شماری از فعالیت های زائد و تکراری و بهبود کارآیی فرآیند توجه دارد که معمولا متکی بر مقدمه یا افزایش پشتیبانی بر مبنای کامپیوتر برای پردازش است. فرآیند مهندسی مجدد اغلب یک محرک برای تکامل نرم افزار است. چون سیستم های قدیمی ممکن است وابستگی های مجازی را با فرآیندهای موجود ترکیب کنند. پس قبل از اینکه فرایند مهندسی مجدد اجرا شود، باید آن ها را کشف کرد و از بین برد. بنابراین وقتی مقیاس تغییرات توسط مهندسی مجدد فرآیند تجارت توانایی تطبیق با نگهداری برنامه ی نرمال را ندارد، نیاز به مهندسی مجدد نرم افزار در شرکت بیش از پیش آشکارتر می شود.

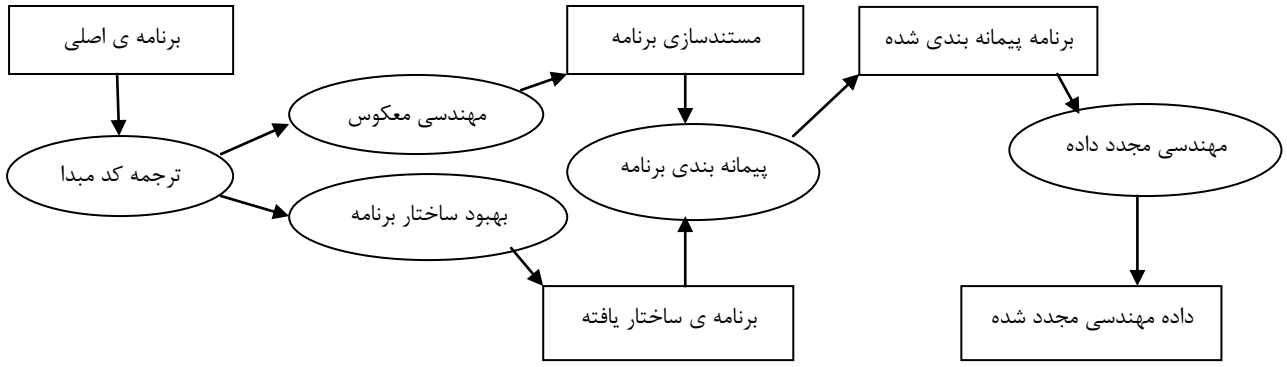
تمایز بین مهندسی مجدد و توسعه ی نرم افزار جدید نقطه ی شروعی برای پیشرفت است که این نقطه ی شروع با ویژگی های سیستم قدیمی به عنوان یکی از ویژگی های سیستم جدید می باشد.[۱]

---

[۱] مقاله دوم ص ۳

چیکوفسکی و کراس [Chikofsky and Cross,1990] توسعه ی متعارف و قراردادی را مهندسی پیشرو نامیدند تا آن را از مهندسی مجدد نرم افزار تمیز بدهند. این تمایز در شکل ۲ نشان داده شده است. مهندسی پیشرو با ویژگی های سیستم شروع می شود و طراحی و پیاده سازی سیستم جدید را درگیر می کند ولی مهندسی مجدد با سیستم موجود شروع می کند. شکل ۳ فرایند مهندسی مجدد را نشان می دهد. فعالیت های مربوط به این فرآیند عبارتند از:

۱- **ترجمه ی کد مبدا:** برنامه از زبان برنامه نویسی قدیمی به نسخه های مدرن به همان زبان یا زبان های دیگر تبدیل می شود.
۲- **مهندسی معکوس:** برنامه تجزیه و تحلیل می شود و اطلاعات برای مستندسازی استخراج می گردند.
۳- **بهبود ساختار برنامه:** کنترل ساختار برنامه برای سهولت خواندن و فهمیدن تجزیه و تحلیل و اصلاح می شود.
۴- **پیمانه بندی برنامه:** قسمت های مرتبط برنامه با هم گروه بندی و قسمت های زائد حذف می شوند.
۵- **مهندسی مجدد داده ها:** داده ها توسط برنامه ی تغییر یافته برای انعکاس تغییرات برنامه پردازش می شوند.

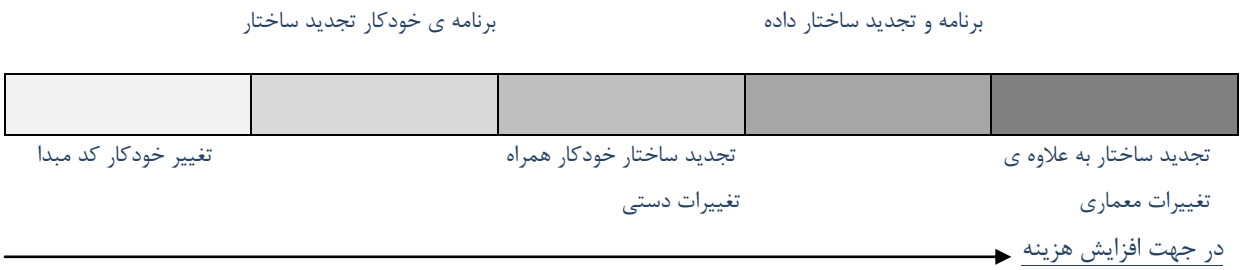


شکل ۳- فرآیند مهندسی مجدد[1]

هزینه های مهندسی مجدد به دامنه ی کار بستگی دارد. همان طور که در شکل ۴ مشاهده می شود، هزینه ها از چپ به راست افزایش می یابند. ترجمه ی کد مبدا به عنوان بخشی از تغییرات، گران ترین رویکرد به حساب می آید.

---

[1] مقاله دوم ص ۴



شکل ۴- رویکردهای مهندسی مجدد

عوامل مهمی که بر افزایش هزینه های مهندسی مجدد تاثیر دارند عبارتند از:

۱- *کیفیت نرم افزار مهندسی مجدد شده:* کیفیت پایین نرم افزار و مستندات مرتبط با آن.
۲- *ابزار در دسترس:* اگر برای مکانیزه کردن بیشتر تغییرات برنامه از ابزار موردی و ویژه ی آن قسمت استفاده شود.
۳- *گسترش داده های تبدیلی مورد نیاز:* اگر مهندسی مجدد نیاز به تبدیل و تغییر بخش اعظمی از داده ها داشته باشد.
۴- *دسترسی به کارمندان متخصص و خبره:* اگر کارمند برای نگهداری سیستم مسئولیت پذیر نباشد.

شایان ذکر است که مهم ترین زیان مهندسی مجدد نرم افزار محدودیت کاربردی آن است که از طریق مهندسی مجدد بهبود می یابد.

ترجمه ی کد مبدا

ساده ترین شکل مهندسی مجدد، ترجمه ی برنامه است. یعنی کد مبدا در یک زبان برنامه نویسی به صورت خودکار به کد مبدا در زبان برنامه نویسی دیگری ترجمه می شود. اما ساختار و سازماندهی برنامه تغییر نمی پذیرد. زبان هدف ممکن است زبان روزآمدی از زبان اصلی باشد. مانند: COBOL-74 به COBOL-85 . یا ممکن است ترجمه ای از زبان کاملا متفاوتی باشد. مانند: FORTRAN به C++ .[۱]

ترجمه ی سطح مبدا به دلایل زیر لازم است:

۱- به روز بودن پایگاه سخت افزاری: سازمان ممکن است در پی تغییر استانداردهای پایگاه سخت افزاری خود باشد. کامپایلرهای زبان اصلی ممکن است برای سخت افزار جدید قابل استفاده نباشند.
۲- فقدان مهارت کارمند: سازمان ممکن است برای نگهداری از زبان اصلی با فقدان کارمند آموزش دیده مواجه

---

۱ مقاله ی دوم ص ۵

باشد.

۳- تغییرات خطی مشی سازمانی: یک سازمان برای حداقل کردن هزینه های پشتیبانی نرم افزار ممکن است زبان خاصی را استاندارد کند. زیرا نگهداری از نسخه های قدیمی کامپایلرها خیلی گران است.
۴- فقدان پشتیبانی نرم افزار: تهیه کننده ی کامپایلر زبان ممکن است از تجارت کناره گیری یا تولید آن محصول را متوقف کرده باشد.[۱]

برای به دست آوردن دیدگاه جامعی از سیستم های قدیمی باید بخش هایی که فرآیند مهندسی مجدد را تعریف می کند بازگو کرد: مهندسی معکوس به استخراج عناصر و اطلاعات سیستم موجود، تجزیه و تحلیل و طراحی مجدد بستگی دارد. این مرحله داده ها را برای شناسایی جنبه های آن می گیرد. چون ممکن است بخش هایی از آن نیاز به جایگزینی یا طراحی مجدد داشته باشند.[۲]

مقاله ی دوم ص ۵ و ۶
مقاله اول ص ۳

۵. مهندسی معکوس

۵,۱– مقدمه

مهندسی معکوس در ابتدا برای تجزیه و تحلیل سخت افزار و پی بردن به مراحل طراحی آن به کار برده شد. اما امروزه در نرم افزار نیز کاربرد دارد. مهندسی معکوس عموما به دنبال بازیافت اطلاعات از سطوح پایین تا سطوح بالای یک مسئله می باشد. مانند یافتن طراحی اطلاعات از کد مبدا.[۱]

مهندسی معکوس چیزی شبیه به مهندسی مجدد نیست. هدف مهندسی معکوس نتیجه گرفتن طرح یا ویژگی های سیستم از کد مبداش است. هدف مهندسی مجدد تولید جدید و سیستم قابل نگهداری است در صورتی که مهندسی معکوس بخشی از فرآیند مهندسی مجدد است. مهندسی معکوس در حین فرآیند مهندسی مجدد برای بهبود طراحی برنامه استفاده می شود. مهندسان نیز از مهندسی معکوس قبل از سازماندهی مجدد ساختار سیستم برای درک بیشتر برنامه بهره می گیرند.

به هر صورت مهندسی معکوس همیشه از مهندسی مجدد تبعیت نمی کند:

۱- چون طراحی و ویژگی های سیستم به عنوان ورودی شرایط جایگزینی در برنامه را دارا هستند، مهندسی معکوس می شوند.
۲- طراحی ها و ویژگی هایی از سیستم که در نگهداری برنامه نقش دارند مهندسی معکوس می شوند. بنابراین لازم نیست دوباره کاری شود و کد مبدا سیستم را مهندسی مجدد کرد.[۲]

فرآیند مهندسی معکوس با مرحله ی تجزیه و تحلیل آغاز می شود. در این مرحله سیستم از طریق استفاده از ابزارهای خودکار برای کشف ساختارش تجزیه و تحلیل می شود. مهندسان اطلاعاتی را به سیستم کد مبدا و مدل ساختاری اش اضافه و اطلاعات زائد را حذف می کنند. سپس این اطلاعات به عنوان گراف مستقیم که به کد مبدا برنامه متصل است، نگهداری می شود. جست و جوگران منبع اطلاعاتی برای مقایسه ی ساختار گراف و کد استفاده می شوند. مستند کردن انواع برنامه، نمودارهای ساختار داده و ماتریس ها می توانند از گراف جهت دار تولید شوند. ماتریس ها، هویت در سیستم را نشان می دهند. فرآیند تولید سند به عنوان یکی از فرآیندهای تکراری در طراحی اطلاعات برای پالایش بیشتر اطلاعات که در مخزن سیستم نگهداری می شود، به کار می رود.

ابزارهای فهم برنامه در پشتیبانی از فرآیند مهندسی معکوس استفاده می شود که معمولا نماهای متفاوتی از سیستم را نشان می دهد. به عنوان مثال آن ها به کاربران اجازه می دهند تعریفی از داده ها ارائه دهند. چنین مثال هایی توسط کلیولند [Cleveland,1989]، امان و کک [Oman and Cook,1990] و نینگ و همکاران در سال

---

۱ مقاله ی اول ص ۴
۲ مقاله ی دوم ص ۶ و ۷

۱۹۹۴ مطرح شده است.

بعد از مستندسازی طراحی سیستم اطلاعات بیشتری برای طراحی مجدد ویژگی های سیستم به منبع اطلاعات افزوده می شود.[۱]

۵,۲– مزایا

مهندسی معکوس به معنی بازیافت اطلاعاتی است که سال ها از دست رفته بودند. بازیافت اطلاعات از سیستم های نرم افزاری قدیمی حائز اهمیت است. به خصوص اگر سیستم ها توسط افرادی که با سیستم آشنایی ندارند، نگهداری می شود.

در نرم افزارهای قابل استفاده ی مجدد یکی از نکات مهم، تعریف و توسعه ی عناصر قابل استفاده ی مجدد مانند اهداف، اجزای نرم افزار و بخش های مستندکردن سیستم است. با مهندسی معکوس تهیه ی انواع مستند سازی سیستم و شناخت اجزای قابل استفاده ی دوباره در سیستم میسر است. اهداف اصلی مهندسی معکوس چگونگی بهبود سیستم، حداقل کردن هزینه و مصرف و تسهیل استفاده ی مجدد از برنامه و نرم افزار است.

۵,۳– مسائل

در ذیل چند مسئله بیان شده است که باید در حین استفاده از مهندسی معکوس حل شوند:

اولین مسئله این است که شاید کد مبدا بسیار اندک ساختارمند شود، مشخصات از بین رفته دوباره طراحی شود یا ناقص بماند و یا مستند کردن سیستم منسوخ باشد. مسائل دیگر ترکیبی از این مدل ها و پیچیدگی ها می باشند.

وقتی سیستمی به وسیله ی اصول متناقض توسعه یافت، به طور حتم کار کردن با آن سیستم مشکل است. ابزار مؤثری برای پشتیبانی از فرآیند مهندسی معکوس وجود دارند اما همه ی آن ها قادر به استخراج اطلاعات کافی از منابع موجود و قابل دسترس نیستند (مانند کد مبدا). بنابراین بهترین راه برای پاسخ به این مسائل ترکیب ابزار کمکی اتوماتیک و حمایت تخصص انسانی می باشد.[۲]

مقاله ی دوم ص ۷

مقاله اول ص ۴

۶. تجزیه و تحلیل و طراحی مجدد

۶,۱– مقدمه

دومین مرحله از فرآیند مهندسی مجدد، تجزیه و تحلیل و طراحی مجدد است. این مرحله ارتباط نزدیکی با تجزیه و تحلیل و مراحل طراحی پروژه های مهندسی نرم افزار دارد. سیستم قدیمی از قبل ساختار طراحی و ویژگی های مربوط به خود را برای طراحی سیستم جدید دارد.

۶,۲– تجزیه و تحلیل

هدف این قسمت، تجزیه و تحلیل سیستم قدیمی به عنوان بخش بسیار مهمی از فرآیند مهندسی مجدد می باشد. زیرا بدون تجزیه و تحلیل مناسب و در نتیجه درک مناسب نرم افزار قادر نخواهیم بود پروژه ی خود را با استفاده از نتایج رضایت بخش کامل کنیم و به اتمام برسانیم. برون یابی نیاز فروشنده و / یا نیاز مصرف کننده برای ارائه ی تصویر شفافی از تولید نهایی، سود و ... ضروری است. کارگروه مهندسی مجدد با شناسایی این نیازها و وضع مجدد مقررات مشخص شده کار را آغاز می کند. به کار بردن مقررات صحیح به علت هزینه بر بودن هر قدم در حلقه ی مهندسی مجدد حائز اهمیت است [FAMOOS99].

همچنین بعضی پروژه ها در مهندسی مجدد سود کافی ندارند. خیلی از روش های استفاده شده در پروژه های مهندسی مجدد در مقابل سودآوری، هزینه زا هستند که طبق گفته ی برخی منابع این هزینه ها باید در طی فرآیند وجود داشته باشند. البته در بخش روش شناسی استاندارد بیشتر صحبت خواهد شد [FAMOOS99] [RM].

۶,۳– طراحی مجدد

طراحی مجدد، کاربرد نتایج تجزیه و تحلیل شده بر طرح موجود است تا با افزایش عناصر جدید طرح، تغییرات کارکردی سیستم را یکپارچه کند. طراحی مجدد مشاع زیادی در مرحله ی طراحی پروژه ی مهندسی نرم افزار دارد. تفاوت وضعیت ها در طراحی سیستم مهندسی تجدید یافته باید با طراحی قدیمی عناصر دقت نظر شود و این که چگونه این عناصر بر تغییرات طرح مؤثر خواهند بود.

در بعضی موارد طراحی مجدد به دلیل تضاد بین ساختار جدید طرح و طرح موجود مشکل است. اگر اثبات طرح ها خیلی ناسازگار باشد، چه طرح را به کلی تغییر دهید و چه رویکرد دیگری به کار گیرید، طرح شکست می خورد و بر عکس اگر طرح ها مناسب یکدیگر باشند، کاربرد نرم افزار جدید آسان می شود و زمان تحویل نرم افزار کوتاه خواهد شد. در این صورت طراحی سیستم جدید باید با دقت کافی به خصوص در مرحله ی تجزیه و تحلیل انجام شود.[۱]

---

۱ مقاله ی اول ص ۵ و ۶

کارگروه باید در جهت حداقل کردن تغییرات گام بردارد. چون تغییرات ممکن است از طریق عوامل محیطی برای نرم افزار ناسازگاری به وجود بیاورد. در نتیجه به روز کردن سیستم ها بسیار مهم است. چنین طراحی هزینه را برای فروشنده افزایش خواهد داد و برای تجارت در بلند مدت هزینه را به سوی ضرر سوق می دهد. دلیل دیگر برای محدود کردن تغییرات در دیدگاه ما نهفته است. کاربر نهایی، رقیبی برای تغییرات نیستند. زمان آموزش بلند مدت، به سرعت علایق کاربر را در استفاده از نرم افزار کاهش می دهد. و حتی ممکن است باعث ترک سیستم شود (شاید به سوی استفاده از نوع قدیمی اش).[1]

مقاله ی اول ص ۶

۷. مهندسی پیشرو

۷,۱– مقدمه

مهندسی پیشرو شبیه به فرآیند مهندسی نرم افزار است. این نوع مهندسی در طراحی جدید که در طی فاز طراحی و تجزیه و تحلیل از طراحی سطح بالا به سمت طراحی سطح پایین و پیاده سازی آن حرکت می کند، کاربرد دارد.

گام های مهندسی پیشرو عبارتند از [BPSARLSS94]:

۷,۲– پیمانه بندی برنامه

فرایندی برای تقسیم بندی اطلاعات است و برای ساخت گروه های طبقاتی با کارکرد مشابه و تولید عناصر قابل استفاده ی مجدد به کار می رود. نتایج پیمانه بندی، اندازه ی برنامه و پیچیدگی آن را کاهش می دهد که در نتیجه به معنی نگهداری و پشتیبانی آسان تر از برنامه است.[۱]

در تعریفی دیگر، پیمانه بندی برنامه، فرآیند سازماندهی مجدد برنامه است که بخش های آن برنامه با یکدیگر جمع و به عنوان یک مدل ارائه می شوند. برای مثال در یک برنامه که داده های زلزله نگار را پردازش می کند، همه ی عملیات های به هم پیوسته با هم جمع می شوند و خروجی آن که به صورت نمایش هندسی داده ها است، اجرا می شود.

انواع پیمانه که در طی فرآیند پیمانه بندی شکل می گیرند، عبارتند از:

- **چکیده های داده ها:** از به هم پیوستن مؤلفه های فرآیند به وجود آمده اند.
- **پیمانه های سخت افزاری:** این پیمانه ها به چکیده های داده ها بستگی دارند و همه ی توابعی را که برای کنترل دستگاه سخت افزاری نیاز است، جمع آوری می کنند.
- **پیمانه های کارکردی:** پیمانه هایی هستند که توابع مشابه هم که وظایف یکسانی را انجام می دهند، جمع آوری می کنند.
- **پیمانه های پشتیبانی فرآیند:** پیمانه هایی هستند که همه ی توابع مورد نیاز برای پشتیبانی از فرآیند تجارت خاصی را گروه بندی می کنند.[۲]

پیمانه بندی برنامه ها معمولا با ویرایش کدها صورت می پذیرد. برای پیمانه بندی برنامه باید ارتباط بین اجزا و نقش اجزا را شناخت. جست و جو و تجسم ابزار در پیمانه بندی مفید است ولی مکانیزه کردن کامل این فرآیند غیر ممکن است.

---

۱ مقاله ی اول ص ۷

۲ مقاله ی دوم ص ۱۰ و ۱۱

بازیافت چکیده های داده ها

بسیاری از سیستم های قدیمی برای ذخیره کردن فضای حافظه به استفاده از جدول (فهرست) سهام و نواحی داده های مشترک روی می آورند. اطلاعات در این نواحی در دسترس است و ممکن است توسط بخش های دیگر سیستم به روش متفاوت استفاده شود. تغییر دادن در این نواحی پرهزینه است. زیرا هزینه ی تغییر تجزیه و تحلیل، همه ی استفاده ی داده ها را تحت الشعاع قرار می دهد. برای کاهش هزینه ی تغییر فرآیند پیمانه بندی بر شناسایی چکیده های داده ها متمرکز می شود.

گام هایی که در تبدیل داده های کلی به انواع چکیده ها مؤثرند عبارتند از:

۱- **تجزیه و تحلیل کردن نواحی مشترک داده ها برای شناسایی چکیده ی داده های منطقی:** حالتی است که چندین چکیده در یک ناحیه ی داده با هم ترکیب می شوند و باید پس از شناسایی به طور منطقی بازسازی شوند.
۲- **خلق نوعی هدف برای هر کدام از چکیده ها:** اگر زبان برنامه نویسی امکاناتی برای مخفی سازی داده ها نداشته باشد، شبیه سازی چکیده ی داده ها با فراهم آوری توابع برای روزآمد کردن و در دسترس قرار دادن همه ی بخش های داده امکان پذیر نیست.
۳- **استفاده از برنامه ی جست و جوی سیستم یا مولد اجرای متقابل برای پیدا کردن همه ی مراجع مربوط به داده ها**

به نظر این فرآیند زمان بر ولی آسان است. در نسخه های قدیمی تر زبان ها مانند FORTRAN که امکانات محدودی برای بازسازی داده ها دارد، برنامه نویسان استراتژی های مدیریت داده های پیچیده را طراحی می کنند.[۱]

۷,۳– پیاده سازی

در حین پیاده سازی برنامه طبقه بندی ها نباید نه زیاد بزرگ و نه زیاد کوچک باشند. یکی از اصول پیاده سازی این است که طبقه بندی باید به صورت تک مفهومه پیاده سازی شود. اگر طبقه بندی به صورت چند مفهومه پیاده سازی شود، احتمالا مقادیر همبستگی پایینی خواهد داشت. زیرا این مفاهیم باید به صورت جداگانه پیاده سازی شوند. اما اگر طبقه بندی به صورت تک مفهومه پیاده سازی نشود، به معنی این است که مفهوم بین چندین طبقه به کار رفته است و احتمالا به شدت به هم پیوسته اند [FAMOOS99].

مهندسی پیشرو باید طراحی سطح پایین را طوری طراحی کند که همبستگی و رابطه ی متقابل سیستم در یک سطح نگه داشته شود تا قادر به استفاده ی مجدد و ترمیم سریع باشد.[۲]

مقاله ی دوم ص ۱۱ و ۱۲
مقاله ی اول ص ۷

۷,۴– آزمایش

آزمایش برای پیدا کردن اشتباهاتی که در طول اجرای برنامه ممکن است رخ دهد، ایجاد شده است و همین طور چیزهایی که در سیستم کار می کنند، بهبود می بخشد (بهینه سازی). بعضی از قسمت های آزمون برای بخش هایی که بنیادی و اساسی هستند به صورت پیشرفته برنامه ریزی می شوند. چنین قسمت هایی از آزمون معمولا در طول مراحل طراحی مجدد یا مهندسی پیشرو پیاده سازی و اجرا می شوند.

آزمون اشخاص در آزمون برنامه نیز استفاده می شود. کسانی که برنامه را آزمون می کنند تجربه های متفاوتی از کار کردن با انواع نرم افزارها به دست می آورند و به نتایج متفاوتی می رسند. این نتایج در بهبود وجه اشتراک کاربر استفاده می شوند و اشتباهاتی که مهندسان در تست عمومی، آن ها را پیدا نکرده اند، آشکار می کنند.[۱]

---

۱ مقاله ی اول ص ۷

۸. تفاوت های بین الگوهای طراحی و الگوهای مهندسی مجدد

۸,۱– الگوهای طراحی

الگوی طراحی شامل ۴ بخش اساسی است:

- **نام:** نام کوتاه و توصیفی برای شرح دادن الگوی طراحی
- **مسئله:** وقتی الگویی کاربردی شد، مسائلی که باید پاسخ داده شوند، شرح می دهد.
- **پاسخ:** شرح همه ی عناصری است که طراحی پاسخ از آن ها شکل گرفته است که شامل ارتباط بین آن ها، مسئولیت پذیرشان نسبت به هم و همکاری با یکدیگر می باشد.
- **نتیجه ی منطقی:** نتایج و رابطه ی جایگزینی، هزینه ها، سودها و ...

یک الگوی طراحی نباید به خاطر جزئیات کوچک به اولین راه حل قانع شود و به اصطلاح ملاحظه ی راه حل را بکند. همان طور که در بالا اشاره شد بخش مسئله، الگویی که باید به کار رود، شرح می دهد. در راه حل ها باید به دنبال عناصر طراحی بگردیم و این که چه اثر متقابل و ارتباطی با هم دارند و هر کدام چه وظیفه ای در سیستم بر عهده خواهند گرفت. الگوی طراحی باید پیامدهای استفاده از آن عناصر را نیز نشان دهد. دانستن پیامدها، نتایج و رابطه ی جایگزینی در هنگام انتخاب الگوی مناسب بسیار مهم است. راه حل و پاسخی که مسئله ی دیگری را برانگیزاند، مناسب نیست.

۸,۲– الگوهای مهندسی مجدد

الگوهایی هستند که چگونگی تبدیل سیستم قدیمی به سیستم جدید را شرح می دهند. هدف اصلی الگوهای مهندسی مجدد پیشنهاد راه حل مناسب برای حل مسائل مهندسی مجدد است. توسعه دهنده باید مسئله را تشخیص دهد، همه ی گزینه های قابل دسترس را ببیند و روش ویژه ی کار را انتخاب کند.

قالب و فرمت الگوها توسط ویژگی های زیر تعریف شده است:

تمرکز بر فرآیند مهندسی مجدد

الگوی مهندسی مجدد باید بر فرآیند مهندسی مجدد متمرکز شود. برای مثال: چگونه می توان مسائلی که از سیستم معاصر برخاسته است، تعریف کرد؟ آیا زمانی که الگو در انتقال سیستم به کار رفت، دام و تله به وجود می آید؟[1]

[1] مقاله اول ص ۸ و ۹

ماهیت آسان

مهندسی مجدد باید قادر به تصمیم گیری در مورد الگوهای قابل اجرا باشد. بنابراین الگو باید اطلاعات شفاف، مختصر و جامع درباره ی نیت مصرف بدهد.

تفکیک کردن ابزارها و زبان وابسته به موضوع

تا زمانی که الگو در اکثر روش ها کاربردی نشده است نباید ابزار یا زبانی که مورد استفاده قرار گرفته است، نام برد. زیرا ابزارها و زبان هایی که اشخاص استفاده می کنند، موضوعی برای تغییر خواهد بود. در حالی که الگو به عنوان راه حل کلی باید یکنواخت باقی بماند.

واژگان و نمادسازی استاندارد

در تفکیک ابزارها و زبان وابسته استفاده از زبان بی طرف نمادسازی و واژگان الزامی است.

ساختار الگوی مهندسی مجدد شامل بعضی از عناصر و موارد ذیل برای بیان شباهت ها و تفاوت های بین الگوی مهندسی مجدد و الگوی طراحی به کار می روند:

- **نام الگو:** نام باید کوتاه، شفاف و توصیفی باشد. نام باید بر پایه ی عملیاتی باشد که الگو برای انجام آن انتخاب شده است.
- **نیت:** شرح فرآیند مهندسی مجدد، نتایج و چرایی خوشایند بودن آن.
- **اجرایی بودن:** شرح اجرایی بودن یا نبودن الگویی خاص
- **انگیزش:** در این قسمت به شرح سیستم قدیمی و ساختار آن، عامل یابی مجدد سیستم و ارتباط بین آن ها پرداخته می شود.
- **ساختار:** شرکا و همکاران در این قسمت شناخته می شوند. ساختار شامل زیان ها و منافع در عامل یابی مجدد سیستم از ساختار معاصر به سمت هدف جدید ساختار می باشد.
- **فرآیند:**
  - ✓ *بازرسی، ردیابی، اکتشاف، نمایان سازی:* توصیف روش ها و ابزارهایی که از اجازه دادن کدها به مسئله ی مشکوک جلوگیری می کند.
  - ✓ *دستورالعمل:* چگونه مهندسی مجدد و گزینه های ممکن اجرا شود.
- ✓ *مشکلات:* موقعیت های اختیاری که برای مهندسی مجدد امکان پذیر و موجه نیست.[1]
- **بحث و گفت و گو:** در این بخش در مورد سیستم قدیمی و عامل یابی مجدد سیستم بحث می شود. موضوعاتی مانند: مقدار قیمت، روابط جایگزینی، به دست آوردن سود، بزرگ بودن مشکل و ...

مقاله اول ص ۹

۸,۳– نتیجه گیری

اگرچه الگوهای طراحی و الگوهای مهندسی مجدد مشخصات مشترکی دارند ولی در کل متفاوت هستند.

شباهت ها عبارتند از:

- نام
- شرح مسئله
- شرح راه حل
- پیامدها و بحث و گفت و گو

تفاوت در انواع مسائل باعث شد تا الگوها برای پاسخ به این مسائل به وجود آیند. الگوهای طراحی چگونگی جواب دادن به یک مسئله ی کلی و معمول را در یک شیوه ی درست شرح می دهند ولی الگوهای مهندسی مجدد چگونگی عامل یابی مجدد سیستم را شرح می دهند. الگوهای مهندسی مجدد نسبت به الگوهای طراحی وسیع تر هستند. چون فاز طراحی فقط قسمتی از کل فرآیند مهندسی مجدد می باشد.[۱]

---

[۱] مقاله ی اول ص ۹ و ۱۰

۹. رویکرد منافع الگوی مهندسی مجدد

۹,۱– مقدمه

استفاده از الگوهای مهندسی مجدد چندین مزیت دارد. البته همه ی الگوها به طور یکسان خوب نیستند ولی تخصص در تعداد کمی از این ابعاد بستگی به مرحله ای که در آن طراحی شده اند، دارد. در این قسمت در مورد این که چه چیزهایی از ابعاد فرآیند مهندسی مجدد مهم هستند و چگونه الگوهای مهندسی مجدد این ابعاد را هدایت می کنند، بحث می شود.

۹,۲– درک پروژه

بعضی از الگوهای مهندسی مجدد به بخش های مهمی از فرآیند اشاره می کنند که بستگی به درک نرم افزار دارد. به علاوه در ابتدا برای ساخت بهتر سیستم باید درک کلی از سیستم داشت [FAMOOS99].

درک فرآیند در مرحله ی آغازین مهندسی مجدد ضروری است. زیرا معمولا اطلاعاتی توسط آن ها تهیه می شود که در مراحل بعدی بسیار تعیین کننده است. بعضی از این الگوها به آزمون رمز بستگی دارند. خواندن کد مبدا پروژه و شناخت قسمت هایی که در پروژه های جدید استفاده خواهند شد و همین طور قسمت هایی که باید در تولید نهایی جایگزین گردند. بقیه ی الگوها به مستندسازی پروژه اشاره می کنند. مستندسازی موجود باعث می شود نگاهی کلی از سیستم درباره ی ساختار و توسعه ی روش و همچنین رویه ی به کار برده شده در توسعه ی پروژه ی اصلی به دست آید.

در ذیل به سه گروهی که به مرحله ی آغازین درک پروژه بستگی دارند و الگوهای به هم پیوسته ای هستند، اشاره می شود. این گروه ها شامل فرآیند تجارت مهندسی مجدد (BPR)، اولین تماس و اقتباس خصوصیات است:

فرآیند تجارت مهندسی مجدد (BPR)

BPR دلیل واقعی همه ی فرآیندهای مهندسی مجدد است که تاکنون وجود داشته است. BPR به تنهایی شامل کل فرآیند مهندسی مجدد است و برای نرم افزار مهندسی مجدد حائز اهمیت است. زیرا BPR مهندسی مجدد کل تجارت است و یک تجارت جدید به معنی نیاز به نرم افزار جدید است [FAMOOS99] [BPSARLSS94].

بنابراین کارگروه مهندسی مجدد برای آزمون متغیرها نیازهای سیستم قدیمی را روزآمد می کند. چون سیستم اولیه توسعه داده شده است و به اصطلاح روزآمد شده است. استفاده از الگوها فقط درک اولیه ای از پروژه می دهد نه درکی که باعث طراحی مجدد پروژه شود.[۱]

---

۱ مقاله ی اول ص ۱۱

ابعاد تجارتی عبارتند از:

- ❖ قوانین تجارت معاصر چیست؟
- ❖ اهداف تجارت چیست؟
- ❖ آیا قوانین و / یا اهداف احتمال تغییر دارند؟

این ابعاد برای اجرا کردن موفقیت آمیز سیستم جدید برای کارگروه بسیار مهم است.

الگویی که به آزمون تغییرات تجارت و تجزیه و تحلیل قوانین بستگی دارد به کارگروه به مستعد کردن سیستم در گرفتن تقاضاهای جدید تجارت کمک می کند [BPSARLSS94].

اولین تماس

در ابتدا با نگاه کردن به پروژه تعدادی مشکل از قبیل تهدیدآمیز بودن مقیاس سهام پروژه دیده می شود که اگر فرآیند مهندسی مجدد صورت نپذیرد، کار با ریسک همراه خواهد بود. در ابتدا ممکن است پیشرفت، کم باشد که باعث حجم کاری زیاد در پروژه می شود. چنین حجم کاری باعث تحویل با تاخیر و حتی لغو تولید جدید می شود که باعث درگیری متخصص مهندسی مجدد با همکاران در باره ی عدم توانایی متخصص در مهندسی مجدد به خصوص در نیازهای واقعی مثل فرآیند می شود. اگر آن ها در موقعیت رهبری هستند و متخصص نتواند راه پیشرفت را به آن ها نشان دهد، دیگر به توانایی او اعتماد نخواهند کرد. به دست آوردن درک صحیحی از پروژه بر همکاران اثر می گذارد و آن ها را متقاعد به نیاز به تولید جدید خواهد کرد [FAMOOS99].

اقتباس خصوصیات

بعد از به دست آوردن درک صحیحی از پروژه باید از مستندات موجود، اطلاعات و منابع دیگر اطلاعات مانند پایگاه داده را استخراج کرد. با بررسی پایگاه داده به ارتباط پروژه و داده پی برده می شود و نیز با بررسی کد مبدا می توان اهداف و طبقاتی که در تولیدات جدید نقش دارند، حدس زد. این رویکردها تحت الگوهای " بررسی پایگاه داده " و " حدس زدن اهداف " قرار می گیرند. حدس زدن اهداف برای رویکرد توسعه ای اهداف گرایشی پروژه مفید هستند. همین طور ممکن است در اجرای اهداف غیر گرایشی نیز استفاده شوند.

۹,۳– کارآیی منبع

نظر به این که سیستم های قدیمی روش برنامه نویسی داشتند یا خیر، مقادیر زیادی از منابع شامل نیروی انسانی، ابزار و زمان را مصرف می کردند.[۱] از وقتی که منابع گران شدند استفاده از الگوهای مهندسی مجدد نیاز به منابع را

به طور چشمگیری کاهش داد.

---

۱ مقاله ی اول ص ۱۲

الگوهایی که با کارایی منابع سر و کار دارند به " نقاط بحرانی " یا " نواحی تمرکز" بستگی دارند. نواحی تمرکز برای آزمایش کردن و طراحی مجدد نسبت به بقیه ی قسمت های سیستم مهم تر هستند. چون منابع بسیار مهم هستند تمرکز منابع در این نقاط حیاتی باعث می شود پروژه در راه صحیح خود گام بردارد و پیشرفت کند. نواحی متمرکز به چندین مقوله تقسیم می شوند: ابعاد زمان، پیچیدگی و کارکرد آن نواحی.

استفاده از الگوها در تصمیم گیری در نقاط بحرانی یا نواحی متمرکز، در مدیریت محدودیت منابع، آزادی عمل بیشتری را سبب می شود.[1]

۹,۴– بهبود ساختار برنامه

نیاز به بهینه کردن حافظه و درک مهندسی نرم افزار توسط برنامه نویسان حاکی از این است که سیستم های قدیمی خوش ساختار نیستند. چون ساختار کنترلی، انشعاب های غیر شرطی و کنترل منطقی آن ها بسیار درهم ریخته است. تغییرات در برنامه ممکن است باعث عدم دسترسی به بعضی از کدها شود. البته این موضوع پس از تحلیل گسترده قابل کشف است. برنامه نویسان پشتیبان اغلب جرات حذف کد را ندارند. شکل ۵، کنترل منطقی پیچیده را نشان می دهد. برنامه ای که در ذیل نشان داده شده است، یک هدایت کننده ی سیستم گرمایشی است که توسط برنامه ی FORTRAN نوشته شده است. ممکن است صفحه کلید بر روی روشن، خاموش یا کنترل شده تنظیم باشد. اگر سیستم، کنترل شده باشد، روشن یا خاموش بودن به تنظیم تایمر و ترموستات بستگی دارد. اگر گرمایش روشن باشد، کلید گرمایش، آن را خاموش می کند و بر عکس.

```
Start:    Get (Time-on, Time-off, Time, Setting, Temp, Switch)
          If Switch = off goto off
          If Switch = on goto on
          goto Cntrld
off:      if Heating-status = on goto Sw-off
          goto loop
Cntrld:   if Time = Time-on goto on
          if Time = Time-off goto off
          if Time < Time-on goto Start
          if Time > Time-off goto Start
          if Temp > Setting then goto off
          if Temp < Setting then goto on
Sw-off:   Heating-status := off
          goto Switch
Sw-on:    Heating-status := on
Switch:   Switch-heating
Loop:     goto Start
```

شکل ۵ - برنامه ی کنترلی با منطق اسپاگتی[2]

مقاله ی اول ص ۱۲

مقاله دوم ص ۸

وقتی برنامه ها با محدودیت حافظه رو به رو هستند و برنامه نویسان در تلاشند که از کدهای تکراری بپرهیزند، ساختار پیچیده ی کد به وقوع می پیوندد.

شکل ۶، سیستم کنترلی مشابهی را نشان می دهد. برنامه ها به ترتیب از بالا به پایین خوانده می شوند. سه تا کلید موقعیت روشن، خاموش، و کنترل شده که به کد وابسته ی آن متصل هستند به خوبی مشخص شده اند.

```
Loop
    -- The Get statement finds values for the given variables from from the system's
    -- environment.
    Get (Time-on, Time-off, Time, Setting, Temp, Switch) ;
    case Switch of
        when On => if Heating-status = off then
                        Switch-heating ; Heating-status := on ;
                   end if ;
        when Off => if Heating-status = on then
                        Switch-heating ; Heating-status := off ;
                    end if;
        when Controlled =>
            if Time >= Time-on and <= Time-off then
                if Temp > Setting and Heating-status = on then
                        Switch-heating ; Heating-status = off ;
                else if Temp < Setting and Heating-status = off then
                        Switch-heating ; Heating-status := on ;
                 end if ;
            end if ;
    end case ;
end loop ;
```

شکل ۶ - برنامه ی کنترلی ساختارمند

همین طور کنترل بی ساخت موقعیت های پیچیده به عنوان بخشی از فرآیند بازسازی برنامه محسوب می شود. شکل ۷، جمله ی شرطی not را برای درک بیشتر نشان می دهد.

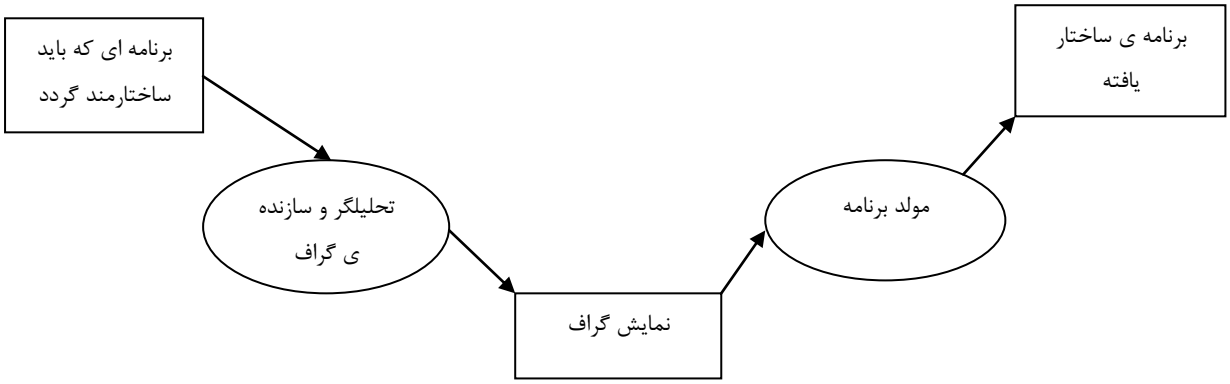
```
-- Complex condition
If not (A > B and (C < D or not (E > F)))...
-- Simplified condition
If A <= B and (C >= D or E > F)...
```

شکل ۷ - ساده سازی شرطی

بم و ژاکوپینی (Bohm and Jacopini,1966)، اذعان کردند که هر برنامه ای می تواند با عبارات ساده ی شرطی if، then و else دوباره نوشته شود. این قضیه اساس بازسازی برنامه ی خودکار است که نیازی به جملات غیر شرطی و loop ها ندارد.[1]

شکل ۸، مراحل بازسازی اتوماتیک برنامه را نشان می دهد. این اولین تغییر در گراف جهت دار است. سپس یک برنامه معادل ساخت یافته بدون دستورالعمل goto تولید می شود.

گراف جهت دار یک برنامه ی گراف گردشی است که چگونگی حرکت کنترل را در طی برنامه نشان می دهد. ساده سازی و تغییر شکل تکنیک ها بدون تغییر در معنا در این گراف به کار می روند. این عملیات قسمت های غیرقابل دسترس کد را حذف می کند.



شکل ۸ - خودکار کردن بازسازی برنامه

مشکلات بازسازی برنامه ی خودکار عبارتند از:

۱- ***کمبود نکات کمکی در برنامه برای راهنمایی کاربر***: اگر برنامه در راستای comment ها قرار گیرند، به عنوان قسمتی از بازسازی فرآیند مفقود هستند.
۲- ***فقدان مستندسازی***: بین مستندسازی برنامه ی خارجی و خود برنامه ارتباطی وجود ندارد. در بسیاری از مواقع، هم comment ها و هم مستندسازی منسوخ هستند.
۳- ***تقاضای محاسباتی سنگین***: الگوریتم های به کار برده شده در بازسازی ابزارها پیچیده هستند. حتی با سخت افزارهای سریع و پیشرفته برای بازسازی فرآیند برنامه های بزرگ باز هم زمان زیادی به طول می انجامد.

اگر برنامه به زبان غیر استاندارد نوشته شود ابزارهای بازسازی استاندارد کار خود را به خوبی انجام نمی دهند و نیازمند مداخله ی دستی هستند.

---

[1] مقاله ی دوم ص ۹

در بعضی مواقع ممکن است برای بازسازی همه ی برنامه های سیستم، تحلیل هزینه - سود وجود نداشته باشد. آرتور (Arthur,1988) بیان می کند که داده ها باید برای برنامه هایی که از بازسازی آن ها سود بیشتر کسب می شود، جمع آوری شود.[1]

معیارهای ذیل در شناسایی کاندیداهای بازسازی به کار می روند:[2]

- نرخ خرابی
- درصد تغییر کد مبدا در سال
- اجزای پیچیدگی

۹,۵– مهندسی مجدد اخیر

در مراحل بعدی فرآیند مهندسی مجدد تاکید بر این است که چه پروژه ای بدون تغییر در کد مبدا به پایان می رسد یا همچنان ادامه خواهد داشت. اگر کارگروه مهندسی مجدد به این نتیجه رسید که پژوهش به ادامه ی مهندسی مجدد نیاز دارد یا خیر، فرآیند در این مرحله به پایان می رسد.

به هر حال، اگر کارگروه تصمیم به اعمال تغییرات گرفت، الگوهای جدید برای سازماندهی و این که چگونه و چه موقع تغییرات اعمال شوند، به کار می روند. در حقیقت سیستم های قدیمی بسیار پیچیده هستند بنابراین درک بخش هایی از فرآیند مهندسی مجدد، موفقیت پروژه ی مهندسی مجدد را تضمین می کند. الگوی " تجدید نیرو برای درک کردن" به این نکته اشاره می کند که با تغییر نام و ساخت مجدد قسمت ها می توان با صرفه جویی در زمان به بهترین راه برای اجرای پروژه دست یافت [FAMOOS99].

بعد از مراحل بالا، آزمایش این تغییرات برای اتمام موفقیت آمیز پروژه ی مهندسی مجدد بسیار مهم است. بسیاری از الگوها علاوه بر آزمایش الگوهای مهندسی مجدد به الگوهای طراحی نیز نیاز دارند. شایان ذکر است که آزمایش سیستم قدیمی تجدید ساختار شده بسیار سخت تر از سیستمی است که به تازگی طراحی شده است. زیرا در سیستم های قدیمی بعضی از قسمت های سیستم از جمله عبارت ساختار و یا ارتباط بین بخش های دیگر قابل درک نیست [FAMOOS99]. طراحی خوب متن آزمون و برنامه ی آزمون مستلزم استفاده از استراتژی های خوش تعریف و متدولوژی ها در قالب الگوها است که شامل الگوهای مهندسی مجدد و الگوهای طراحی مهندسی پیشرو می باشد.[3]

---

[1] مقاله ی دوم ص ۱۰
[2] مقاله ی دوم ص ۱۰
[3] مقاله ی اول ص ۱۳

۹,۶– مهندسی مجدد داده ها

فرآیند تجزیه و تحلیل و سازماندهی مجدد ساختارهای داده، مهندسی مجدد داده نامیده می شود. ذخیره سازی، سازماندهی و قالب داده های پردازش شده توسط برنامه های قدیمی باید رشد و نمو کنند تا تغییرات نرم افزار یا برنامه را منعکس کنند. به طور کلی اگر عاملیت سیستم تغییرناپذیر باشد مهندسی مجدد داده نیاز نخواهد بود.[۱]

دلایل تغییر و اصلاح داده های برنامه ها در سیستم های قدیمی عبارتند از:

۱- **تنزل داده ها:** کیفیت داده ها رو به کاهش میل می کند. تغییرات در داده ها، خطاها را معرفی می کنند. ممکن است مقادیر تکراری ایجاد شوند و تغییرات در محیط خارجی منعکس نشوند که البته اجتناب ناپذیر است. زیرا طول عمر داده اغلب طولانی است. مانند: وقتی یک حساب باز می شود، داده های بانکداری خصوصی ایجاد می شوند.

۲- **محدودیت ذاتی که در برنامه ساخته شده اند:** برنامه نویسان خیلی از محدودیت های برنامه را روی بعضی از داده ها پردازش می کنند. ممکن است در قسمت هایی از برنامه محدودیت از بین برود.

۳- **فرضیه ی معماری:** اگر یک سیستم مرکزی به ساختار معماری توزیعی انتقال یابد، ضروری است که هسته ی معماری آن، مدیریت داده شود تا به مشتری از راه دور دستیابی داشته باشد. برای انتقال داده ها از فایل های جدا از هم به سیستم مدیریت پایگاه داده، مهندسی مجدد داده نقش به سزایی دارد. برنامه ی معماری توزیعی زمانی شروع می شود که سازمان تصمیم بگیرد از فایلی که بر پایه ی مدیریت داده است به سیستم مدیریت پایگاه داده حرکت کند.

رویکردهایی که نیاز به مهندسی مجدد داده را بیان می کنند:

رویکرد	شرح
پاک سازی داده ها	داده های ثبت شده و ارزش ها تجزیه و تحلیل می شوند تا کیفیت آن ها بهتر شود. اطلاعات تکراری و زائد حذف می شوند. این رویکرد معمولا نیازی به تغییرات برنامه ی به هم پیوسته ندارد.
تعمیم داده ها	در این حالت، داده ها و برنامه های به هم پیوسته مهندسی مجدد شده اند تا محدودیت ها را در فرآیند پردازش داده حذف کنند. این رویکرد برای افزایش درازای میدان و تعدیل محدودیت های بالارتبه به تغییرات نیاز دارد. ممکن است داده ها برای منعکس کردن تغییرات برنامه دوباره نوشته یا پاک سازی شوند.
انتقال داده ها	در این حالت، داده ها به سیستم مدیریت پایگاه داده ی پیشرفته و جدیدی انتقال می

[۱] مقاله ی دوم ص ۱۲

یابند. داده ها در فایل های جداگانه ذخیره می شوند.

شکل ۹ - رویکردهای مهندسی مجدد داده[1]

ریکت و همکاران (Rickts, Del Monaco et al.,1993) برخی مسائل داده های سیستم های قدیمی که از چندین برنامه ی مشترک ساخته شده اند مطرح کرده اند:

۱- **مسائل نام گذاری داده ها:** نام ها ممکن است رمزی باشند و یا قابل فهم نباشند. نام های مختلف (هم معنی ها) و نام های مشابه ممکن است در برنامه های مختلف برای معنی کردن چیزهای مختلف به کار روند.

۲- **مسائل درازای میدان:** مسئله ای است که صراحتا در ثبت های (رکوردهای) برنامه تعیین شده است. درازای میدان ممکن است خیلی کوتاه باشد و فقط داده های جاری را نمایش دهد.

۳- **مسائل ثبت سازمان:** ثبت ها هویت مشابه و یکسانی که در برنامه های مختلف باید سازماندهی شود، نمایش می دهند. این مشکل در زبان برنامه نویسی COBOL است که سازماندهی فیزیکی ثبت و ضبط ها توسط برنامه نویسان تنظیم و در فایل ها منعکس می شوند. البته این مشکل در زبان برنامه نویسی C++ یا Java نیست. زیرا در این برنامه ها سازماندهی فیزیکی ثبت وظیفه ی برنامه مترجم است.

۴- **الفاظ سخت رمزگذاری شده:** ارزش (مطلق) الفاظ مثل نرخ مالیات به طور مستقیم با بخش هایی از برنامه در ارتباط است.

۵- **وجود نداشتن فرهنگ لغت داده:** فرهنگ لغت داده ای که اسامی استفاده شده و کاربردشان را نشان دهد، وجود ندارد.

بعد از این که تعاریف داده ها مهندسی مجدد شد، مقدار داده ها برای این که با ساختار جدید مطابقت داشته باشند باید تغییر کنند. ریکت و همکارانش برخی مقدار داده های ناسازگار را همان طور که در ذیل آمده است، شرح داده اند:[2]

ناسازگاری داده ها	شرح
مقادیر پیش فرض ناسازگار	برنامه های مختلف، مقادیر پیش فرض متفاوتی را برای عناوین مشابه داده های منطقی ذکر می کنند. وقتی یک پیش فرض را برای مقدار گمشده در نظر می گیریم که داده ی گمشده را نمی توان پیدا کرد.
واحدهای ناسازگار	اطلاعات مشابه در واحدهای مختلف در برنامه های مختلف ارائه می شوند. برای مثال: در آمریکا یا انگلیس، داده های وزن دار در برنامه های قدیمی، پوند و در برنامه های جدید به صورت کیلوگرم ارائه می شوند. یکی از این مسائل از اروپا تحت عنوان واحد پول اروپا سرچشمه می گیرد. سیستم های

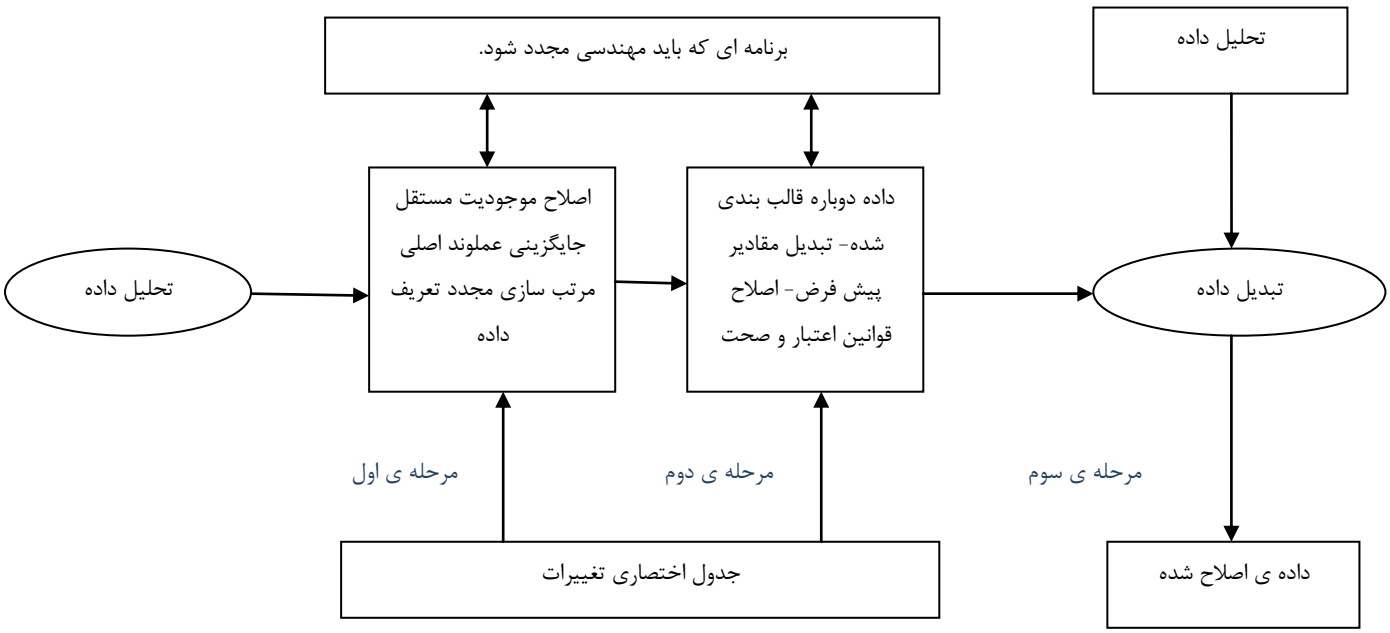
[1] مقاله دوم ص ۱۳

[2] مقاله ی دوم ص ۱۵

	قدیمی برای رسیدگی به واحدهای پولی ملی نوشته شده اند اما اکنون داده ها باید به واحد پول اروپا تغییر کنند.
قوانین اعتبار ناسازگار	برنامه های مختلف، قوانین اعتبار داده های گوناگون را به کار می گیرند. داده های نوشته شده توسط یک برنامه ممکن است توسط برنامه ی دیگری پذیرفته نشود. این مشکل خاص برای داده های آرشیو (بایگانی) است. داده هایی که همراه تغییرات روزآمد نمی شوند.
نمایش معنایی ناسازگار	برنامه ها برخی معانی را به روشی که عناوین ارائه شده اند، فرض می کنند. مثال: برخی برنامه ها حروف بزرگ متن را به معنی نشان دادن آدرس قلمداد می کنند. همچنین ممکن است برنامه ها از قراردادهای متفاوتی استفاده کنند و داده هایی را که به طور معنایی معتبر هستند، نپذیرند.
بررسی مقادیر منفی ناسازگار	**بعضی برنامه ها مقادیر منفی را به عنوان موجودیت مستقل نمی پذیرند. مقادیری که باید مثبت باشند. بعضی دیگر از برنامه ها این مقادیر منفی را فقط برای تشخیص و تبدیل آن ها به مقادیر مثبت می پذیرند.**

شکل ۱۰ - ناسازگاری های مقادیر داده

تجزیه و تحلیل جزئیات برنامه ها قبل از مهندسی مجدد داده ها ضروری است. این تجزیه و تحلیل باید در کشف تابع تعیین کننده ی هویت برنامه، پیدا کردن مقادیر لفظی که باید جایگزین شوند، کشف قوانین اعتبار داده های تعبیه شده و تغییر ارائه ی داده ها هدفگذاری شود. ابزارهایی مثل تحلیلگر سیستم مختصات، به چنین تجزیه و تحلیل هایی کمک می کند. شکل ۱۱، فرآیند مهندسی مجدد را به فرض این که تعاریف داده ها اصلاح، مقدارهای اصلی نام گذاری، فرمت های داده ها سازماندهی مجدد و مقادیر داده ها تبدیل شده اند، نشان می دهد. تغییرات جدول اختصاری، جزئیات همه ی تغییراتی که در مراحل و فرآیند مهندسی مجدد داده استفاده می شود، نگهداری می کند.



شکل ۱۱ - فرآیند مهندسی مجدد[1]

در مرحله ی اول از فرآیند بالا، تعاریف داده ها در برنامه اصلاح شدند. البته داده ها به تنهایی توسط این اصلاحات مؤثر نیستند بلکه برای خودکار کردن این فرآیند و برخی حوزه هایی که از الگوی تطبیق سیستم مانند awk (Aho, Kernighan et al.,1988) برای یافتن و جایگزین کردن تعاریف یا توسعه دادن XML شرح داده ها (St Laurent and Cerami,1999) استفاده می کنند از این ابزار برای تغییر داده استفاده می شود.

[1] مقاله دوم ص ۱۶

البته بعضی کارهای دستی نیز برای تکامل فرآیند لازم است. اگر هدف ساده و قابل درک باشد فرآیند مهندسی مجدد داده در همین مرحله متوقف می شود. چنان چه مشکلات مقادیر داده وجود داشته باشد، مرحله ی دوم از این فرآیند آغاز می شود. همچنین اگر سازمان تصمیم بگیرد که مرحله دوم را ادامه دهد، مرحله ی سوم که تغییر داده ها است، شروع می شود. این فرآیند پرهزینه است. زیرا برنامه ای باید نوشته شود که دانش سازمان قدیمی و جدید را در خود جای دهد و داده های قدیمی را برای خروجی و تبدیل اطلاعات پردازش کند که امکان دارد الگوی تطبیق سیستم ها، دوباره برای اجرای تغییرات به کار رود.[1]

---

[1] مقاله ی دوم ص ۱۵

۱۰. هدایت الگوها

تکنیک هایی که برای شناسایی الگوها به کار می روند، عبارتند از:

- **مطالعه ی پروژه و استفاده از چهار تکنیک زیر:**
  ۱- حضور در بحث های غیررسمی شرکت در طی زمان پیشرفت پروژه
  ۲- حضور در جلسات مربوط به پروژه
  ۳- سؤال کردن از طراحان ارشد (بالا رتبه) در باره ی استراتژی مورد استفاده در سیستم مهندسی مجدد و دلیل استفاده از آن استراتژی
  ۴- مصاحبه با تصمیم گیرندگان ارشد و مهندسان جوان در مراحل گوناگون پروژه

چون مشکل است که افراد را از محل کارشان برای مصاحبه بیرون برد، دو تکنیک اول بسیار مفید هستند.

- **در زمان مطالعه ی پروژه مسائلی مشاهده شود که کارگروه با به کار بردن استراتژی و تاکتیک های مربوطه در آن نواحی امکان انحراف وجود دارد.**
- **مصاحبه با مهندسانی که در زمینه ی مهندسی مجدد سال های متمادی تجربه و تبحر دارند و سؤال درباره ی طرز آشنایی با الگوی مربوطه**
- **مطالعه ی پروژه های مهندسی مجددی که به چاپ رسیده اند.**

هدایت الگوها بر پایه ی عواملی چون انعطاف پذیری، قابلیت درک و فهم، سعی و تلاش، مقیاس پذیری و اثر جهانی الگو قرار دارد.

انعطاف پذیری

تلاش برای تغییر و تبدیل رابطه ی قدیمی به رابطه ی جزیی. رابطه ی قدیمی به صورت ساکن و ایستا تغییر می کند ولی رابطه ی جزیی به صورت پویا و دینامیک تغییر می کند. رابطه ی جزیی، انعطاف پذیری سیستم را افزایش خواهد داد. روش دیگر افزایش انعطاف پذیری، کشف و حذف وابستگی های بین مجموعه برنامه های سیستم می باشد. زیرا اگر این وابستگی ها کشف و حذف نشوند، به بهره برداری از خصوصیات سیستم و نگهداری از آن ها لطمه وارد می کند.

قابلیت درک و فهم

اگر بتوان طبقات پیچیده را به سلسله مراتب کوچک تر اما ویژه تر تقسیم کرد، قابلیت فهم افزایش می یابد. به عبارتی اگر طبقات تفکیک گردند، درک این سلسله مراتب ها آسان تر می گردد.[1]

---

[1] مقاله ی اول ص ۱۴

۱۱. ابزارها

❖ **نام:** GOOSE

شرح: ابزاری برای پشتیبانی خودکار از کشف مسئله

سیستم های برنامه نویسی مدرن مهندسی مجدد به دلیل اندازه و پیچیدگی هایش، پرهزینه هستند. خواندن کدهای مبدا برای یادگیری سیستم با اهمیت است. بنابراین برای پی بردن به دانش سیستم، ابزارهای خودکار این زمینه را فراهم می کنند.

❖ **نام:** DUPLOC

شرح: ابزاری برای یافتن کدهای تکراری

تغییر نرم افزاری که در آن کدهای تکراری وجود دارد تقریبا سخت است و البته پیدا کردن کدهای تکراری برای سیستم و فرآیند بسیار مهم است.

❖ **نام:** کد خزنده (Code Crawler)

شرح: ابزاری که از برنامه نویسی پروژه های بزرگ مهندسی معکوس از طریق آمیختن مقادیر و پدیدارنمایی پشتیبانی می کند.

مرکز پشتیبانی تکنولوژی نرم افزاری نیرو هوایی آمریکا (STSC)، ابزارهای مختلفی برای مهندسی مجدد معرفی کرده است که عبارتند از:[1]

- فرآیند تجارت مهندسی مجدد
- عقلایی کردن نام داده
- داده های مهندسی مجدد
- مهندسی پیشرو
- ترمیم مدل هدف
- مستند سازی مجدد
- قالب بندی مجدد
- تجدید ساختار
- هدف یابی مجدد
- مهندسی معکوس

---

[1] مقاله ی اول ص ۱۵

- قطعه قطعه کردن
- منبع رمز برگردان

همه ی این موارد در فرآیند مهندسی مجدد به نوبه ی خود مهم هستند. در ادامه بعضی از موارد بالا شرح داده می شوند:

مهندسی معکوس

ابزارهای این مورد، در اولین مرحله از کل فرآیند مهندسی مجدد به کار می آیند. مانند: فلوچارت ها و انواع نمودارها.

مثال ها [STSC99]:

- **نام ابزار:** تحلیلگر خودکار

  فروشنده: شرکت نرم افزار پیشرفته ی خودکار

  شرح: تحلیلگر خودکار، شرایطی را برای محیط آزمایش فراهم می کند، نمودار ساختار تعاملی کد مبدا را می سازد، اثرات استفاده از متغیرهای کلی را پیگیری می کند، آزمایش تحلیل پوششی را اندازه گیری می کند و عملکرد اطلاعات را نشان می دهد.
- **نام ابزار:** طراحی C و مستندسازی زبان برنامه نویسی

  فروشنده: شرکت طراحی سیستم های نرم افزاری

  شرح: CDADL از طریق بهبود کیفیت طراحی و بهره برداری طراح به برنامه ریزی کمک می کند. CDADL، شبه کد و کد C را برای یافتن اشتباهات و گزارش بازدهی آن، تجزیه و تحلیل می کند.[1]

مهندسی پیشرو

ابزارهایی هستند که برخلاف مهندسی معکوس در مرحله آخر مهندسی مجدد به کار می روند.

مثال ها [STSC99]:

- **نام ابزار:** ARIS

  فروشنده: شرکت طراحی سیستم های نرم افزاری

  شرح: ARIS ابزاری است که کد Ada تولید می کند.
- **نام ابزار:** دست ابزار جعبه ی Auto-G

  فروشنده: بنگاه RJO

مقاله ی اول ص ۱۶

شرح: مترجم Auto-G، زبان طراحی G & T (G & TDL) را به Ada ترجمه می کند. این دست ابزار یک سیستم کمک مهندسی چرخه ی حیات کامپیوتر است.

[PBIGGS96] ReThree - C++

این یک مهندسی معکوس یکپارچه، مستندسازی مجدد و دست ابزار بازیافت است که توسط آقای پیت بیگز (Pete Biggs) در دانشگاه بریجهام یانگ (Brigham Young University) ساخته شده است. C++ اطلاعات را از کد مبدا استخراج می کند و مخزنی از طبقات C++ برای بازیابی های بعدی فراهم می سازد و به سه بخش مهم تفکیک می شود:

۱- مهندسی معکوس کد مبدا C++
۲- مستند کردن کد مبدا C++
۳- ساختن، نگهداری، جست و جوی مخزن قابل استفاده ی مجدد طبقات C++

ReThree – C++، کد مبدا را مستند و با فرمت rtf ذخیره می کند. کیفیت مستندات تولید شده به توسعه دهنده (کسی که کد مبدا را می نویسد)، بستگی دارد.[1]

---

[1] مقاله ی اول ص ۱۶

۱۲. روش شناسی (متدولوژی) استاندارد

این بخش یک دید کلی از روش شناسی که در پروژه های مهندسی مجدد (هم در مهندسی مجدد نرم افزار و هم در BPR) کاربرد دارند، ارائه می دهد. روش شناسی ها از کلاس بالایی برخوردارند ولی از چشم انداز جامع فرآیند مهندسی مجدد تبعیت می کنند. در ذیل به مزایا و زیان های روش شناسی می پردازیم:[۱]

روش ۱: بهبود مستمر

۱. توصیف پروژه
۲. خلق بصیرت، ارزش ها و اهداف
۳. طراحی مجدد فرآیندهای تجارت و ابزارها
۴. ارزیابی مفهوم و مزایا
۵. برنامه ریزی و اجرایی کردن
۶. کاربردی کردن پاسخ
۷. گذار به فرآیند بهبود مستمر

این متدولوژی بر استمرار پروژه بعد از اتمام آن تاکید می کند و روزآمد بودن پروژه در طی زمان و کاهش ریسک پذیری مهندسی مجدد را تضمین می کند.

روش ۲: تشخیص

۱. تعریف پروژه
۲. مستند کردن فرآیندها آن چنان که هستند (سیستم تشخیص)
۳. طراحی مجدد فرآیندهای تجارت
۴. توسعه ی تحلیل هزینه – سود
۵. برنامه ریزی و اجرایی کردن
۶. ارزیابی عملکرد

این متدولوژی، ساختار وقت گیری دارد.

مقاله ی اول ص ۱۸

روش ۳: طراحی همراه یادگیری

۱. تعریف پروژه
۲. یادگیری از دیگران (مشتری ها، شرکا، آزمایش، تکنولوژی)
۳. خلق بصیرت و طراحی مدل جدید
۴. توسعه در معماری و مدل های خوب
۵. اجرای تحلیل هزینه در مقابل سود
۶. تعریف فرآیندها، سیستم و قوانین آموزشی
۷. برنامه ریزی کاربردی
۸. توسعه ی راه حل ها
۹. کاربردی کردن راه حل ها و اندازه گیری عملکرد

این متدولوژی، از جمله روش های قوی محسوب می شود.

روش ۴: بهترین انطباق

۱. تعریف پروژه و شناخت منابع گروه
۲. فکر بکر فرآیندهای جدید و تکنولوژی ها
۳. تجزیه و تحلیل و اولویت بندی فرصت ها (تحلیل سود)
۴. انتخاب بهترین فرصت و طراحی راه حل
۵. توسعه ی استفاده از ابزار و دیگر فرآیندها
۶. تحول و انتقال برنامه (گذار برنامه)
۷. پیاده سازی راه حل ها
۸. اندازه گیری نتایج

طبق نظر نویسنده [RM]، این متدولوژی از سه روش دیگر سریع تر است ولی در تولید محصولات بادوام کاستی هایی دارد. در حقیقت کارگروه پروژه ارتباط کمی با مشتری و مصرف کننده نهایی دارد که همین امر ممکن است باعث شود پروژه از اهداف اصلی اش دور گردد. اگر چیزی که مصرف کننده به آن نیاز دارد تولید نشود، عقاید تولید کننده اهمیتی نخواهد داشت. شاید بهترین راه حل انطباق، انتخاب بهترین راه حل از میان مجموعه راه حل های بد باشد. در ضمن استفاده از طوفان مغزی بدون دانش مناسب در مورد پروژه منجر به هزینه های بالاتر سیستم خواهد شد.[۱]

خلاصه و نکته ها

این فصل کوتاه نتیجه گیری ما را بر مبنای گزارش نشان می دهد. اعتقاد ما بر این است که اثبات شده، مفهوم مهندسی مجدد مفید بوده و دارای اهمیت فزاینده ای خواهند شد زمانی که سیستم های نرم افزاری قدیمی، مقصود گرا و غیر مقصودگرا ، به دنبال افزایش گردش اطلاعات ، منسوخ خواهند شد. با مد نظر قرار دادن این تغییرات سریع، و میزان گسترده ای از منابع به منظور بروز نگهداشتن نرم افزار ، اعتقاد ما بر این است که ابزارا و روش های جدید رو به توسعه بوده، به صورتی که سیستم ها به جای منسوخ شدن رو به پیشرفت می باشند.

مقالات تخصصی انگلیسی همراه با ترجمه فارسی

عرضه شده به صورت رایگان و اختصاصی در ایران عرضه

توجه !

این فایل از سری محصولات رایگان (فرمت PDF) ایران عرضه میباشد، لیکن

شما عزیزان میتوانید جهت تهیه مقالات تخصصی ترجمه شده این رشته به

صورت کامل و با فرمت ورد (قابل ویرایش) همراه با نسخه انگلیسی

مقاله از نشریات معتبر خارجی ISI و Sciencedirect

(Elsevier ،IEEE ،Springer ،Wiley)

اینجا کلیک نموده و با قیمت مناسب خریداری نمائید(تحویل آنی).