فهرست

فصل اول

نگهداری نرم‌افزار

آشنایی با تعریف و مفهوم نگهداری

بررسی انواع نگهداری

هزینه‌های مربوط به نگهداری

آشنایی با روش‌های نگهداری

نگهداری نرم‌افزار

1-1) خلاصه بخش

دو فاز عمده در چرخه حیات نرم‌افزار، شامل فاز توسعه نرم‌افزار و فاز نگهداری نرم‌افزار هستند. مطالعه‌ای که توسط Yip and Lam (1994) انجام شد ، نشان داد که 66٪ از هزینه‌های چرخه ی حیات نرم‌افزار صرف فاز نگهداری نرم‌افزار می‌شود. بیشتر تحقیقات اینگونه گزارش داده‌اند که 50 تا70 درصد ازچرخه حیات توسعه نرم‌افزارصرف نگهداری نرم‌افزار می‌شود. بهبود قابلیت نگهداری نرم‌افزار، بخصوص در فاز اولیه، می‌تواند بازخورد اولیه برای کمک به بهبود کیفیت نرم‌افزار را فراهم آورد و هزینه بالای نگهداری نرم‌افزار را کاهش دهد.

2-1) ماهیت نگهداری

مطالعات قبلی نشان دادند که یکی از عوامل مشکلات نگهداری نرم‌افزار این است که در طی مراحل طراحی و توسعه نرم‌افزار، قابلیت نگهداری به طور جدی در نظر گرفته نمی‌شود. Lee در سال 1998 و Balci در سال 2003 کاهشی را در هزینه‌های نگهداری نرم‌افزار پیشنهاد کردند که می‌توانست با فرایند کنترل شده بهتر طراحی و پیاده سازی در مراحل اولیه چرخه حیات بدست آید. بنابراین شناسایی ودرک عوامل ایجاد مشکل، در فرایند توسعه نرم‌افزار واطمینان از قابلیت نگهداری نرم‌افزار مؤثر خواهد بود.

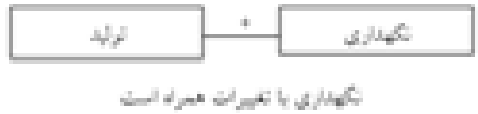
فرایندهای توسعه نرم‌افزار بطور گسترده‌ای پیچیده و غیر قابل پیش بینی هستند. هر روزه، پروژه‌های توسعه نرم‌افزار، نرخ بالایی از شکست را دارند.طبق گزارش Standish Group's CHAOS در سال 2006 بیش از 63٪ از حدود 10000 پروژه IT در سطح دنیا دیر تحویل داده شدند، بیش از 50٪ از این پروژه‌ها شامل ویژگیهای نیازمندیها نبودند وبیش از 15٪ پروژه‌ها، هزینه‌هایی بیش از مقدار بوجه تعیین شده داشتند.بهبود فرایندهای توسعه نرم‌افزار، کیفیت سیستم‌های نرم‌افزاری و کارایی کل پروژه نرم‌افزار و سازمان را بالا خواهد برد.

3-1) تعریف نگهداری

طبق تعاریف ارایه شده در استانداردها و مدلهای کیفیت، قابلیت نگهداری بصورتهای زیر تعریف شده است:

- کار لازم برای یافتن و تصحیح کردن خطاها در برنامه
- تلاش لازم برای تغییر دادن یا توسعه امکانات سیستم
- توانایی تغییر محصول نرم‌افزاری. تغییرات ممکن است شامل اصلاحات، بهبودها، وفق پذیری نرم‌افزار با تغییر در محیط و نیازمندی‌ها و خصوصیات عملیاتی باشد.
- میزان سهولتی که یک کاربرد یا مولفه بتواند نگهداری شود.
- نگهداری، رفع خطا و اشکال زدایی، پشتیبانی،توسعه، گسترش و ارتقاء می‌باشد.

طبق شکل(1-1)، یک سیستم در ابتدا تولید می‌شود و بعد از تولید و مورد استفاده قرار گرفتن در محیط عملیاتی نیاز به نگهداری دارد.



شکل 1-1 ایجاد سیستم کاربردی

1-4) انواع نگهداری

انواع نگهداری به صورت زیر تقسیم بندی می‌شود:

نگهداری اصلاحی[1]

نگهداری تطبیقی [2]

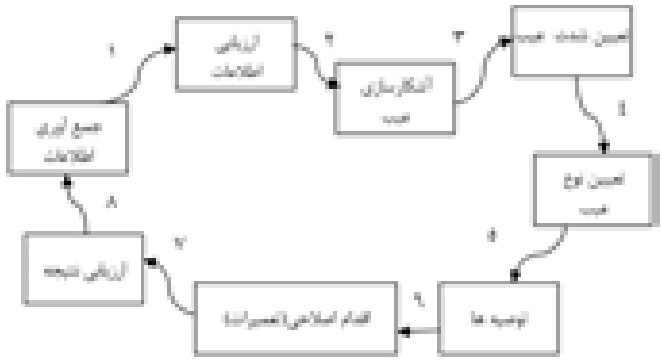
نگهداری تکمیلی [3]

نگهداری پیشگیرانه [4]

نگهداری اصلاحی

هنگامیکه سیستمی تولید می‌شود همه خطاهای آن در همان زمان قابل کشف نیستند، بلکه بعد از استفاده سیستم در محیط واقعی آن خطاها کشف می‌شوند. بنابراین نگهداری اصلاحی برای خطاهایی به کار می‌رود که قبلاً کشف نشده‌اند و نتیجه آن حذف یکسری فعالیت‌های سیستم می‌باشد که دارای خطا هستند. برای مثال می‌توان به تصحیح کدنویسی اشاره نمود.

روند چرخه نگهداری اصلاحی در شکل (1-2) نشان داده شده است.



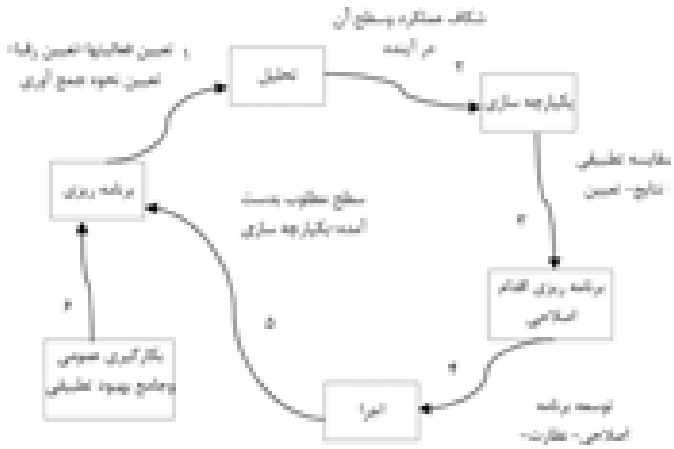
شکل 1-2 چرخه نگهداری اصلاحی

1 Corrective Maintenance
2 Adaptive Maintenance
3 Maintenance Perfective
4 Maintenance Preventative

نگهداری تطبیقی

نگهداری تطبیقی زمانی مورد نیاز است که در محیط سیستم تغییراتی حاصل می‌شود. بدین معنی که با تغییر محیط سیستم باید سیستم با محیط جدید وفق داده شود، این تغییرات در محیط ممکن است در همان موقع یا اینکه بعداً ایجاد شوند که در هر صورت سیستم تولید شده باید قابل استفاده باشد. برای مثال خطای در طراحی که در نهایت منجر به باز نویسی چند قطعه از کد می‌شود.

روند چرخه نگهداری تطبیقی در شکل 1-3 نشان داده شده است.

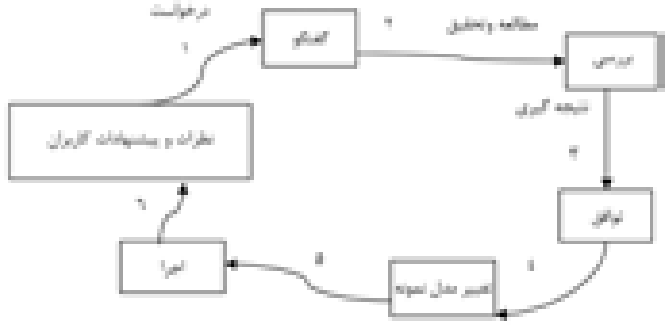


شکل 1-3 چرخه نگهداری تطبیقی

نگهداری تکمیلی

در این نوع نگهداری سیستم تولیدی به گونه‌ای تغییر پیدا می‌کند که وظایف تعریف شده برای آن تغییر نمی‌یابند. این نوع تغییرات از طرف کاربر صورت می‌گیرد. برای مثال اضافه کردن، حذف و اصلاح توابع.

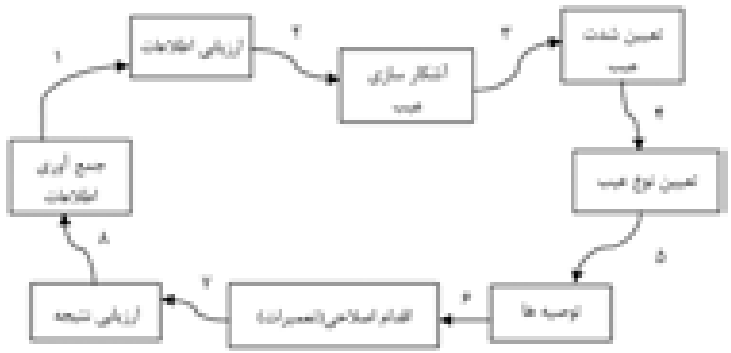
شکل 1-4 روند چرخه نگهداری تکمیلی را نشان می‌دهد.



شکل 1-4 چرخه نگهداری تکمیلی

نگهداری پیشگیرانه

شامل تمام اصلاحاتی است که منجر به تغییر یک قطعه نرم‌افزاری می‌شود. این نوع نگهداری درباره سخت‌افـزار فیزیکـی کامپیوتر است و توسط روش‌های مهندسی معکوس یا مهندسی مجدد قابل تشخیص مـی‌باشـد. نگهـداری پیشـگیرانه در واقع همان نگهداری اصلاحی زمانبندی نشده است که برای حفظ قابل استفاده بودن سیستم اجرا می‌شود.
شکل 1-5 روند چرخه نگهداری پیشگیرانه را نشان می‌دهد.

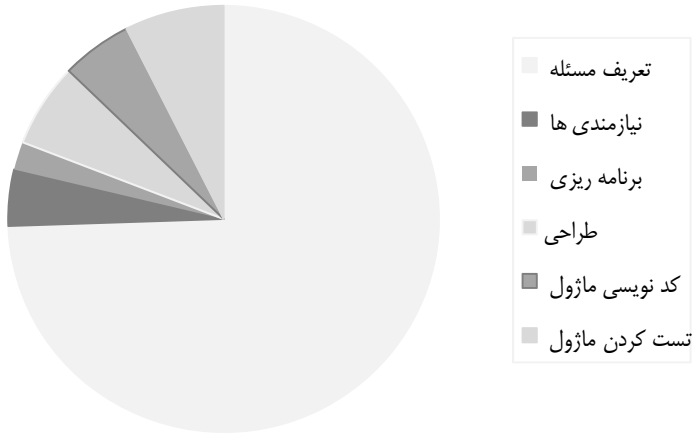


شکل 1-5 چرخه نگهداری پیشگیرانه

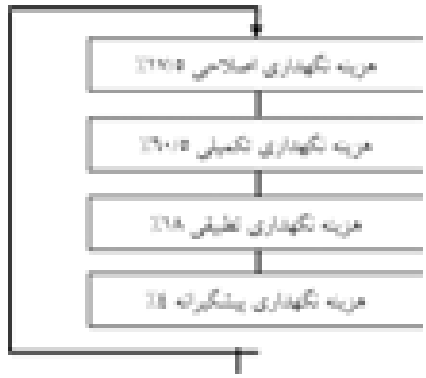
1-5) هزینه‌های نگهداری نرم‌افزار

نگهداری دارای هزینه بالایی می‌باشد این هزینه شامل موارد زیر است:
(1) تعریف مسئله 70٪
(2) نیازمندی‌ها 4٪
(3) برنامه ریزی 2٪
(4) طراحی 6٪
(5) کد نویسی ماژول 5٪
(6) تست کردن ماژول 7٪
(7) یکپارچگی 6٪

شکل ۱-۶ هزینه های نگهداری



هزینه هر یک از انواع نگهداری در شکل 1-7 نشان داده شده است.



شکل 1-7 هزینه‌های انواع نگهداری

1-6) ابزارهای نگهداری

تکنیک‌هایی که می‌توان در نگهداری نرم‌افزار از آن استفاده نمود، عبارتند از:

- مدیریت تغییرات، نگهداری و ردگیری تمامی تغییرات که در سیستم حاصل می‌شود به همراه تاریخ، ساعت و نوع تغییر
- مدیریت ساختار و حفظ ساختار اصلی برنامه
- آنالیز تاثیرگذاری
- استفاده از ابزارهای خودکار جهت تعمیر و نگهداری نرم‌افزار

استفاده از ابزارهای خودکار باعث شده متوسط زمان بین 2 تعمیر (MTTR[5]) نرم‌افزار پائین آید و همچنین باعث خواهد شد که گروه نرم‌افزاری به آسانی نرم‌افزار را نگهداری کند. ابزارهای مختلفی که برای نگهداری و تعمیرات سیستم‌های نرم‌افزاری وجود دارد به قرار زیر هستند:

1- ابزارهای شناسایی خودکار برای شناسایی تفاوت بین دو فایل

2- ابزارهای شناسایی اشکال و Debugging

3- ابزارهای نوشتن کدهای برنامه که به صورت خودکار تغییرات را نیز رکورد می‌کنند.

4- ابزارهای‌اندازه گیری کدها

1-7) هزینه و رقابت‌ها

اشکال گوناگون چندین بررسی تعمیر نرم‌افزار نشان داده که تعمیر و نگهداری سیستم حدود 50 تا 70 درصد از کل هزینه‌های دوره حیات سیستم را به خود اختصاص می‌دهد.

بزرگترین مشکلات تعمیر نرم‌افزار عبارتند از : درک برنامه و آنالیز ضربه‌ای و تست برگشت.

هنگامیکه یک تعمیر روی قطعات نرم‌افزار اتفاق می‌افتد خیلی مهم است که در نگهدارنده سیستم درک کاملی از ساختار، رفتار و توابع سیستم در شروع عملیات اصلاح به دست بیاورد. یکی از این شناخت‌ها تولید یک طرح پیشنهادی اصلاح برای انجام هدف تعمیر(قابل تعمیر بودن سیستم) می‌باشد.به عنوان نتیجه، نگهدارندگان، مقدار زیادی از وقتشان را صرف خواندن کد و مستندات زمینه آن برای درک هدف و ساختار سیستم می‌کنند.درک برنامه خیلی از اوقات ترکیبی از دو مسئله است زیرا اولا نگهدارنده سیستم خیلی به ندرت پیش می‌آید که صاحب (نویسنده) کد برنامه باشد و ثانیا مستندات کامل و به روز به ندرت در دسترس است. آنالیز ضربه‌ای عبارت است از عمل براورد تاثیر نیروی بالقوه برای تغییر با هدف کمینه کردن تاثیرات غیر منتظره. وظیفه آنالیز ضربه‌ای شامل برآورد تغییرات پیشنهادی و ارزیابی ریسک پذیری آن پیاده سازی است و تاثیر آن روی منابع، تلاش و زمانبندی می‌باشد.

فرایند تست سیستم بعد از اصلاح آن تست برگشت نامیده می‌شود. هدف از تست برگشت کسب اطمینان از دو چیز است: 1-تغییرات درست است. 2-قسمت‌های تغییر نیافته سیستم تحت تاثیر قرار نگرفته‌اند. مشکل اصلی این است که تعمیر نرم‌افزار می‌تواند بر روی کاستی‌های فرایند توسعه نرم‌افزار تاثیر بگذارد. Sneidewind اظهار کرده است که مشکل اصلی در انجام تعمیر و نگهداری این است که ما نمی‌توانیم عمل تعمیر و نگهداری را روی سیستمی که برای این منظور طراحی نشده است انجام دهیم.

1-8) وظایف تعمیر

وظایف تعمیر می‌تواند در 5 گروه دسته بندی گردد. تجزیه و تحلیل جداسازی، طراحی ، پیاده سازی، تست و مستند سازی. Besili et al (1996) وظیفه تجزیه و تحلیل / جداسازی شامل آنالیز ضربه ای، تجزیه و تحلیل هزینه موثر و جداسازی می‌باشد. آنالیز ضربه و تجزیه و تحلیل هزینه موثر شامل پیاده سازی‌های متفاوت تکراری و مقایسه تاثیر آن روی زمانبندی

5 Mean Time To Repair

هزینه و عملیات می‌باشد. طراحی شامل طراحی مجدد سیستم بر پایه ی درک تغییرات لازم می‌باشد. همچنین شامل مستندات نیمه رسمی شبیه به مرور و مستندات پخش شده از سیستم می‌باشد. پیاده سازی شامل کد نویسی و تست واحدها می‌باشد. کدنویسی و تست واحد مربوط به زمان کدنویسی وتغییرات می‌باشد. این مرحله بیشتر به برنامه نویس مربوط می‌گردد. تست واحد معمولا به صورت محلی، بر روی کاربر صورت می‌گیرد.عملیات تست شامل تست‌های مجتمع سازی، تست پذیرش و تست برگشت می‌باشد. تست مجتمع سازی به زمان صرف شده برای یکپارچه کردن اجزا نرم‌افزار مربوط می‌گردد.مستند سازی شامل مستندات سیستم، کاربر و دیگر مستندات می‌باشد. مستندسازی، فرایند بسیار مهمی است به این دلیل که تغییرات انجام شده به اسناد تغییرات یا اصلاحات گذشته متکی است.

ابزارهای تعمیر نرم‌افزار، یک محصول است. Taking and Grub استفاده از ابزارها برای ساده سازی وظایف و افزایش کارایی و سودمندی تعمیر نرم‌افزار می‌باشد. چندین معیار برای انتخاب درست ابزار برای وظایف وجود دارد. این معیارها شامل توانایی، ویژگی‌ها، هزینه سود، پلتفرم، زبان‌های برنامه نویس، سهولت برای استفاده، باز بودن معماری، پایداری تولیدکنندگان و فرهنگ سازمان می‌باشد.معیار توانایی، تصمیم می‌گیرد که آیا ابزار قادر به انجام وظیفه است یا نه و یک متد می‌تواند از آغاز مکانیزه شود یا خیر. سپس ابزار مورد نیاز برای رسیدگی به آن را فراهم می‌نماید.

مزیت‌های یک ابزار نماینده کیفیت، باروری، حساسیت و کاهش هزینه می‌باشد.محیطی که ابزار روی آن اجرا می‌گردد پلتفرم نام دارد. این مسئله مهم است که ابزاری که انتخاب می‌کنیم از یک زبان پشتیبانی کند که یک استاندارد صنعتی است. ابزار باید توانایی این را داشته باشند که با ابزارهای فروشندگان دیگر بتوانند یکپارچه گردند.

باز بودن معماری هنگامی که مشکلات تعمیر پیچیده می‌گردد یک وظیفه مهم را بر عهده دارد.بنابراین همیشه استفاده از یک ابزار کافی نیست و ممکن است نیاز باشد که چندین ابزار با هم کار کنند. تولیدکننده باید قادر به پشتیبانی از ابزار در آینده باشد. فاکتور مهم دیگر فرهنگ سازمان است. هر فرهنگ، الگوی کاری خودش را دارد و باید در انتخاب فاکتور به این مسئله توجه کرد.

ابزارهای انتخاب شده بایداز درک برنامه و مهندسی معکوس، عملیات تست ، مدیریت پیکربندی و مستندسازی پشتیبانی کنند. انتخاب یک ابزار مناسب می‌تواند باعث بهبود درک سیستم صرف شود. تقسیم برنامه به برنامه نویس کمک می‌کند که فقط قسمت‌هایی از برنامه که به وسیله تغییرات تاثیر پیدا کرده است را انتخاب کند یا ببیند.

عملیات تست زمان زیادی در فرایند تعمیر را به خود اختصاص می‌دهد و وظیفه طاقت فرسایی است. یک ابزار شبیه سازی تست، به نگهدارنده سیستم کمک می‌کند که تاثیر تغییرات را در محیط شبیه سازی شده قبل از پیاده سازی تغییرات روی محیط واقعی مشاهده نماید. یک تولید کننده تست‌های نمونه، داده‌ها تستی که برای تست توابع اصلاحی سیستم استفاده می‌گردد را تولید می‌نماید. مدیریت پیکربندی از ابزار مکانیزه شده بهره می‌برد. مدیریت پیکربندی و ابزار کنترل نسخه به نگهداری اشیا سیستم نرم‌افزاری کمک می‌کند. یک سیستم کنترل منبع می‌تواند برای نگهداری تاریخچه نسخه فایل‌هایی که پشت سر هم هستند استفاده گردد. و برنامه نویس می‌تواند از دنباله تغییرات فایل استفاده نماید.