

فهرست

11.....	فصل 1
12.....	1-1 خلاصه بخش
12.....	2-1 ماهیت نگهداری
12.....	3-1 تعریف نگهداری
13.....	4-1 انواع نگهداری
15.....	5-1 هزینه های نگهداری نرم افزار
16.....	6-1 ابزارهای نگهداری
17.....	7-1 هزینه و رقابت ها
17.....	8-1 وظایف تعمیر
20.....	10-1 مهندسی معکوس
20.....	11-1 مهندسی مجدد
21.....	12-1 نتیجه گیری
22.....	14-1 منابعی برای مطالعه بیشتر

23.....	فصل 2
24.....	1-2 خلاصه بخش
24.....	2-2 مقدمه
25.....	3-2 تعریف و ماهیت پیچیدگی نرم افزار
27.....	4-2 علل پیچیدگی نرم افزار
28.....	5-2 دستهبندی پیچیدگی نرم افزار
30.....	6-2 پیچیدگی در چرخه حیات نرم افزار
33.....	7-2 نتیجه گیری
34.....	8-2 سوالات متداول
34.....	9-2 منابعی برای مطالعه بیشتر

35.....	فصل 3
36.....	1-3 خلاصه بخش
36.....	2-3
36.....	3-3 مقدمه

36.....	3-3) اندازه‌گیری و اهداف اندازه‌گیری.....
37.....	4-3) کاربردهای اندازه‌گیری.....
38.....	5-3) تعریف متریک نرم‌افزاری.....
38.....	6-3) طبقه‌بندی متریک‌های نرم‌افزاری.....
38.....	1-6-3) متریک محصول.....
39.....	2-6-3) متریک فرآیند.....
39.....	3-6-3) متریک منابع.....
40.....	7-3) متریک‌های پیچیدگی نرم‌افزار.....
41.....	1-7-3) تعداد خطوط کد (LOC).....
42.....	2-7-3) متریک نقاط کارکرد (FP).....
43.....	3-7-3) متریک پیچیدگی دورانی مکیب.....
45.....	4-7-3) متریک‌هاستند.....
47.....	5-7-3) متریک جریان اطلاعات.....
47.....	6-7-3) متریک جریان اطلاعات هنری و کافورا (HK).....
48.....	7-7-3) متریک پیچیدگی مبتنی بر نیازمندیها.....
48.....	9-7-3) متریک‌های شیء‌گرا.....
49.....	10-7-3) سلسله متریک‌های C&K.....
51.....	8-3) سوالات متداول.....
52.....	9-3) منابعی برای مطالعه بیشتر.....
53.....	فصل 4.....
54.....	1-4) خلاصه بخش.....
54.....	2-4) مقدمه.....
54.....	3-4) میزان‌های تعیین‌کننده پیچیدگی در مهندسی نیازمندی‌ها.....
55.....	4-4) تأثیر نیازمندی‌ها در پیچیدگی.....
55.....	5-4) عوامل مؤثر بر پیچیدگی نرم‌افزار در فاز نیازمندی‌ها.....
56.....	1-5-4) سهامداران.....
58.....	2-5-4) تیم توسعه.....
59.....	3-5-4) نیازمندی‌ها.....
61.....	4-5-4) تعاملات و وابستگی.....
63.....	6-4) انتخاب تکنیک‌های مهندسی نیازمندی‌ها بر اساس میزان‌های پیچیدگی.....
66.....	7-4) سوالات متداول.....

66 8-4) منابعی برای مطالعه بیشتر.....

67 فصل 5.....

68 1-5) خلاصه بخش.....

68 2-5) مقدمه.....

71 3-5) نیازمندیها.....

72 1-3-5) تاریخچه مهندسی نیازمندیها.....

72 2-3-5) تعریف نیازمندی.....

73 4-5) طبقه‌بندی نیازمندیها.....

76 5-5) ذینفعان.....

77 6-5) مهندسی نیازمندیها.....

78 7-5) دلایل اهمیت مهندسی نیازمندیها.....

80 8-5) سوالات متداول.....

81 3-5) منابعی برای مطالعه بیشتر.....

83 فصل 6.....

84 1-6) خلاصه بخش.....

84 2-6) مهندسی نیازمندیها.....

85 3-6) مراحل مهندسی نیازمندیها.....

86 1-3-6) استخراج نیازمندیها.....

87 2-3-6) تحلیل و مذاکرات نیازمندیها.....

89 3-3-6) مستندسازی نیازمندیها.....

91 4-3-6) تایید اعتبار نیازمندیها.....

92 4-6) محصول مهندسی نیازمندیها.....

94 5-6) چگونگی انجام مهندسی نیازمندیها.....

95 6-6) مروری بر تکنیک‌های مهندسی نیازمندیها.....

96 7-6) اهمیت مهندسی نیازمندیها در پیچیدگی نرم‌افزار.....

97 8-6) سوالات متداول.....

98 9-6) منابعی برای مطالعه بیشتر.....

99	فصل 7
100	1-7 (خلاصه فصل
100	2-7 مقدمه
103	4-7 تخمین اندازه
103	1-4-7 تخمین اندازه براساس تعداد خطوط
104	2-4-7 تخمین اندازه براساس وظیفه
104	3-4-7 تخمین اندازه براساس اشیاء
104	5-7 تخمین تلاش
104	1-5-7 استاندارد COSMIC-FFP
106	6-7 تخمین زمان
106	7-7 تخمین هزینه
107	1-7-7 انتخاب روش تخمین
107	2-7-7 دلایل شکست تخمین
107	8-7 انواع دیگری از روشهای تخمین در مهندسی نرم افزار
108	1-8-7 COCOMO
112	9-7 نتیجه گیری
112	10-7 سوالات متداول
112	11-7 منابعی برای مطالعه بیشتر

113	فصل 8
114	1-8 خلاصه بخش
114	2-8 مقدمه
115	3-8 مدل های سازمانی
115	4-8 مدل فرایندی کسب و کار سازمان
116	5-8 مدل موجودیتهای کسب و کار سازمانی
116	6-8 یکپارچه سازی در سازمان
117	8-8 رهیافتهای مهم یکپارچگی
117	1-8-8 یکپارچگی دادهای
117	2-8-8 (برنامهای کاربردی نقطه به نقطه
118	3-8-8 پرتالها
118	4-8-8 یکپارچگی دلال
118	5-8-8 SOA with ESB

- 118 9-8) تعریف یکپارچگی کاربردی سازمانی
- 120 11-8) تعریف و قابلیت های یکپارچگی کاربردی سازمانی
- 122 13-8) نتیجه گیری
- 122 14-8) سوالات متداول
- 123 15-8) منابعی برای مطالعه بیشتر

- 125 فصل 9
- 126 1-9) خلاصه بخش
- 126 2-9) مقدمه
- 127 3-9) مهندسی نرم افزار
- 128 4-9) آسیب پذیری و حملات
- 129 5-9) علت آسیب پذیری ها
- 130 6-9) دارایی ها و اقدامات متقابل
- 130 7-9) صفت کیفیتی امنیت
- 131 8-9) تعریف واژه ها و اصطلاحات فنی و تخصصی حوزه امنیت
- 132 9-9) اهمیت مهندسی امنیت
- 133 10-9) مدل بلوغ قابلیت مهندسی امنیت سیستم ها (SSE-CMM)
- 134 1-10-9) معماری مدل SSE-CMM
- 135 2-10-9) مزایای استفاده از SSE-CMM
- 136 11-9) فرآیند PSSS
- 136 1-11-9) تطبیق فعالیت های فرآیند PSSS با چرخه حیات تولید نرم افزار
- 137 12-9) متدولوژی SQUARE
- 137 1-12-9) خصوصیات SQUARE
- 137 13-9) چرخه حیات توسعه امنیت محیط کامپیوتری قابل اطمینان میکروسافت
- 138 14-9) فرآیند نرم افزاری تیمی برای توسعه نرم افزار امن
- 139 15-9) نتیجه گیری
- 140 17-9) منابعی برای مطالعه بیشتر
- 143 ضمیمه

فصل اول

نگهداری نرم افزار

آشنایی با تعریف و مفهوم نگهداری

بررسی انواع نگهداری

هزینه‌های مربوط به نگهداری

آشنایی با روش‌های نگهداری

نگهداری نرم افزار

1-1) خلاصه بخش

دو فاز عمده در چرخه حیات نرم افزار، شامل فاز توسعه نرم افزار و فاز نگهداری نرم افزار هستند. مطالعه ای که توسط Yip and Lam (1994) انجام شد، نشان داد که 66٪ از هزینه های چرخه ی حیات نرم افزار صرف فاز نگهداری نرم افزار می شود. بیشتر تحقیقات اینگونه گزارش داده اند که 50 تا 70 درصد از چرخه حیات توسعه نرم افزار صرف نگهداری نرم افزار می شود. بهبود قابلیت نگهداری نرم افزار، بخصوص در فاز اولیه، می تواند بازخورد اولیه برای کمک به بهبود کیفیت نرم افزار را فراهم آورد و هزینه بالای نگهداری نرم افزار را کاهش دهد.

2-1) ماهیت نگهداری

مطالعات قبلی نشان دادند که یکی از عوامل مشکلات نگهداری نرم افزار این است که در طی مراحل طراحی و توسعه نرم افزار، قابلیت نگهداری به طور جدی در نظر گرفته نمی شود. Lee در سال 1998 و Balci در سال 2003 کاهشی را در هزینه های نگهداری نرم افزار پیشنهاد کردند که می توانست با فرایند کنترل شده بهتر طراحی و پیاده سازی در مراحل اولیه چرخه حیات بدست آید. بنابراین شناسایی و درک عوامل ایجاد مشکل، در فرایند توسعه نرم افزار و اطمینان از قابلیت نگهداری نرم افزار مؤثر خواهد بود.

فرایندهای توسعه نرم افزار بطور گسترده ای پیچیده و غیر قابل پیش بینی هستند. هر روزه، پروژه های توسعه نرم افزار، 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 پروژه ها شامل ویژگی های نیازمندیها نبودند و بیش از 15٪ پروژه ها، هزینه هایی بیش از مقدار بوجه تعیین شده داشتند. بهبود فرایندهای توسعه نرم افزار، کیفیت سیستم های نرم افزاری و کارایی کل پروژه نرم افزار و سازمان را بالا خواهد برد.

3-1) تعریف نگهداری

طبق تعاریف ارایه شده در استانداردها و مدل های کیفیت، قابلیت نگهداری بصورت های 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 نرم افزاری. تغییرات ممکن است شامل اصلاحات، بهبودها، وفق پذیری نرم افزار با تغییر در محیط و نیازمندی ها 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 می باشد.
- طبق شکل (1-1)، یک سیستم در ابتدا تولید می شود و بعد از تولید و مورد استفاده قرار گرفتن در محیط عملیاتی نیاز به نگهداری دارد.



نگهداری با تغییرات همراه است

شکل 1-1 ایجاد سیستم کاربردی

1-4 انواع نگهداری

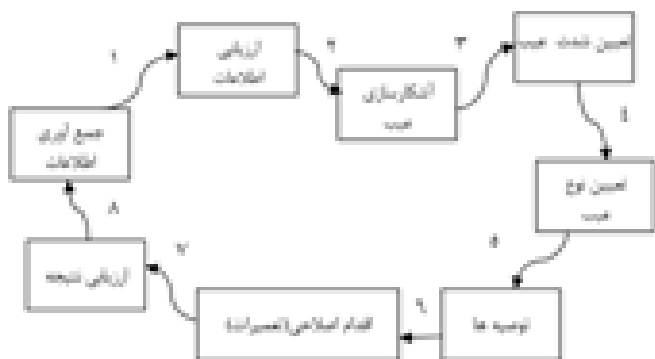
انواع نگهداری به صورت زیر تقسیم بندی می شود:

- ۱ نگهداری اصلاحی
- ۲ نگهداری تطبیقی
- ۳ نگهداری تکمیلی
- ۴ نگهداری پیشگیرانه

نگهداری اصلاحی

هنگامیکه سیستمی تولید می شود همه خطاهای آن در همان زمان قابل کشف نیستند، بلکه بعد از استفاده سیستم در محیط واقعی آن خطاها کشف می شوند. بنابراین نگهداری اصلاحی برای خطاهایی به کار می رود که قبلاً کشف نشده اند و نتیجه آن حذف یکسری فعالیت های سیستم می باشد که دارای خطا هستند. برای مثال می توان به تصحیح کدنویسی اشاره نمود.

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

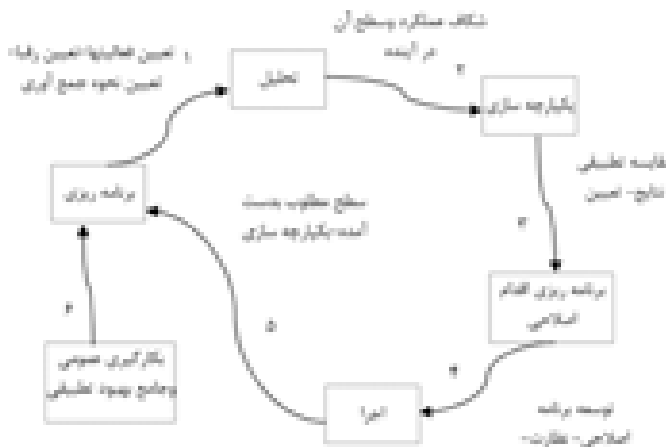


شکل 1-2 چرخه نگهداری اصلاحی

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

نگهداری تطبیقی

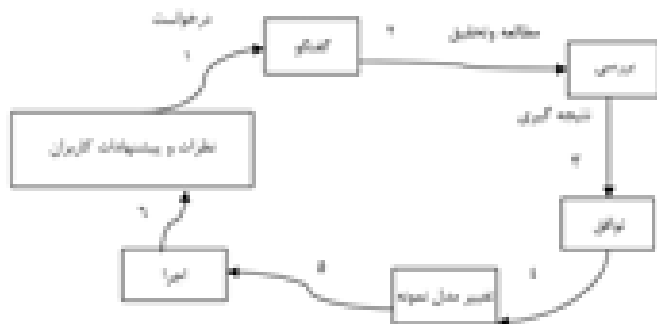
نگهداری تطبیقی زمانی مورد نیاز است که در محیط سیستم تغییراتی حاصل می شود. بدین معنی که با تغییر محیط سیستم باید سیستم با محیط جدید وفق داده شود، این تغییرات در محیط ممکن است در همان موقع یا اینکه بعداً ایجاد شوند که در هر صورت سیستم تولید شده باید قابل استفاده باشد. برای مثال خطای در طراحی که در نهایت منجر به باز نویسی چند قطعه از کد می شود. روند چرخه نگهداری تطبیقی در شکل 3-1 نشان داده شده است.



شکل 3-1 چرخه نگهداری تطبیقی

نگهداری تکمیلی

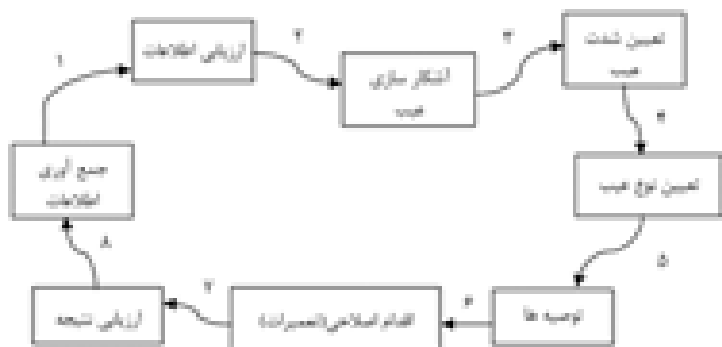
در این نوع نگهداری سیستم تولیدی به گونه ای تغییر پیدا می کند که وظایف تعریف شده برای آن تغییر نمی یابند. این نوع تغییرات از طرف کاربر صورت می گیرد. برای مثال اضافه کردن، حذف و اصلاح توابع. شکل 4-1 روند چرخه نگهداری تکمیلی را نشان می دهد.



شکل 4-1 چرخه نگهداری تکمیلی

نگهداری پیشگیرانه

شامل تمام اصلاحاتی است که منجر به تغییر یک قطعه نرم افزاری می شود. این نوع نگهداری درباره سخت افزار فیزیکی کامپیوتر است و توسط روش های مهندسی معکوس یا مهندسی مجدد قابل تشخیص می باشد. نگهداری پیشگیرانه در واقع همان نگهداری اصلاحی زمانبندی نشده است که برای حفظ قابل استفاده بودن سیستم اجرا می شود. شکل 1-5 روند چرخه نگهداری پیشگیرانه را نشان می دهد.



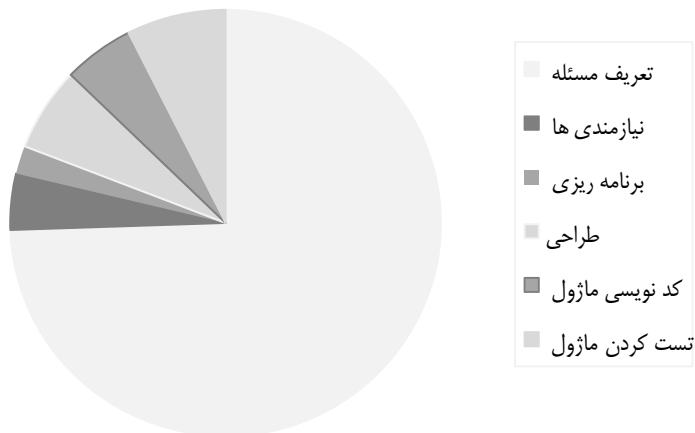
شکل 1-5 چرخه نگهداری پیشگیرانه

5-1) هزینه های نگهداری نرم افزار

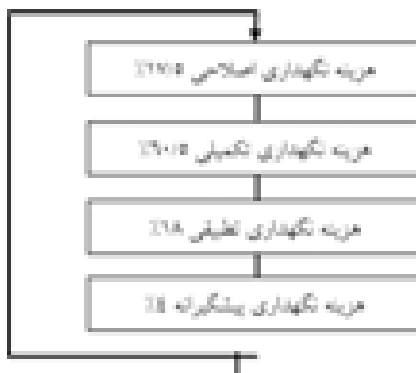
نگهداری دارای هزینه بالایی می باشد این هزینه شامل موارد زیر است:

- (1) تعریف مسئله 70%
- (2) نیازمندی ها 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 ۱-۷ نشان داده شده است.



شکل ۱-۷ هزینه های انواع نگهداری

۱-۶) 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 تاثیر گذاری
- استفاده از ابزارهای خودکار جهت تعمیر و نگهداری نرم‌افزار

استفاده از ابزارهای خودکار باعث شده متوسط زمان بین 2 تعمیر ($MTTR^5$) نرم افزار پائین آید و همچنین باعث خواهد شد که گروه نرم افزاری به آسانی نرم افزار را نگهداری کند. ابزارهای مختلفی که برای نگهداری و تعمیرات سیستم های نرم افزاری 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 می کنند.
- 4- ابزارهای اندازه گیری کدها

1-7) هزینه و رقابت ها

اشکال گوناگون چندین بررسی تعمیر نرم افزار نشان داده که تعمیر و نگهداری سیستم حدود 50 تا 70 درصد از کل هزینه های دوره حیات سیستم را به خود اختصاص می دهد.

بزرگترین مشکلات تعمیر نرم افزار عبارتند از : درک برنامه و آنالیز ضربه ای و تست برگشت. هنگامیکه یک تعمیر روی قطعات نرم افزار اتفاق می افتد 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 شناخت ها تولید یک طرح پیشنهادی اصلاح برای انجام هدف تعمیر (قابل تعمیر بودن سیستم) می باشد. به عنوان نتیجه، نگهدارندگان، مقدار زیادی از وقتشان را صرف خواندن کد و مستندات زمینه آن برای درک هدف و ساختار سیستم می کنند. درک برنامه خیلی از اوقات ترکیبی از دو مسئله است زیرا اولاً نگهدارنده سیستم خیلی به ندردت پیش می آید که صاحب (نویسنده) کد برنامه باشد و ثانیاً مستندات کامل و به روز به ندردت در دسترس است. آنالیز ضربه ای عبارت است از عمل برآورد تاثیر نیروی بالقوه برای تغییر با هدف کمینه کردن تاثیرات غیر منتظره. وظیفه آنالیز ضربه ای شامل برآورد تغییرات پیشنهادی و ارزیابی ریسک پذیری آن پیاده سازی است و تاثیر آن روی منابع، تلاش و زمانبندی می باشد. فرایند تست سیستم بعد از اصلاح آن تست برگشت نامیده می شود. هدف از تست برگشت کسب اطمینان از دو چیز است: 1- تغییرات درست است. 2- قسمت های تغییر نیافته سیستم تحت تاثیر قرار نگرفته اند. مشکل اصلی این است که تعمیر نرم افزار می تواند بر روی کاستی های فرایند توسعه نرم افزار تاثیر بگذارد. Sneidewind اظهار کرده است که مشکل اصلی در انجام تعمیر و نگهداری این است که ما نمی توانیم عمل تعمیر و نگهداری را روی سیستمی که برای این منظور طراحی نشده است انجام دهیم.

1-8) وظایف تعمیر

وظایف تعمیر می تواند 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 می باشد. آنالیز ضربه و تجزیه و تحلیل هزینه موثر شامل پیاده سازی های متفاوت تکراری و مقایسه تاثیر آن روی زمانبندی

5 Mean Time To Repair

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

ابزارهای تعمیر نرم افزار، یک محصول است. Taking and Grub استفاده از ابزارها برای ساده سازی وظایف و افزایش کارایی و سودمندی تعمیر نرم افزار می باشد. چندین معیار برای انتخاب درست ابزار برای وظایف وجود دارد. این معیارها شامل توانایی، ویژگی ها، هزینه سود، پلتفرم، زبان های برنامه نویسی، سهولت برای استفاده، باز بودن معماری، پایداری تولیدکنندگان و فرهنگ سازمان می باشد. معیار توانایی، تصمیم می گیرد که آیا ابزار قادر به انجام وظیفه است یا نه و یک متد می تواند از آغاز مکانیزه شود یا خیر. سپس ابزار مورد نیاز برای رسیدگی به آن را فراهم می نماید.

مزیت های یک ابزار نماینده کیفیت، باروری، حساسیت و کاهش هزینه می باشد. محیطی که ابزار روی آن اجرا می گردد پلتفرم نام دارد. این مسئله مهم است که ابزاری که انتخاب می کنیم 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 می گردد 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 باید از درک برنامه و مهندسی معکوس، عملیات تست، مدیریت پیکربندی و مستندسازی پشتیبانی کنند. انتخاب یک ابزار مناسب می تواند باعث بهبود درک سیستم صرف شود. تقسیم برنامه به برنامه نویسی کمک می کند که فقط قسمت هایی 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 می دهد و وظیفه طاقت فرسایی است. یک ابزار شبیه سازی تست، به نگهدارنده سیستم کمک می کند که تاثیر تغییرات را در محیط شبیه سازی شده قبل از پیاده سازی تغییرات روی محیط واقعی مشاهده نماید. یک تولید کننده تست های نمونه، داده ها تستی که برای تست توابع اصلاحی سیستم استفاده می گردد را تولید می نماید. مدیریت پیکربندی از ابزار مکانیزه شده بهره می برد. مدیریت پیکربندی و ابزار کنترل نسخه به نگهداری اشیا سیستم نرم افزاری کمک می کند. یک سیستم کنترل منبع می تواند برای نگهداری تاریخچه نسخه فایل هایی که پشت سر هم هستند استفاده گردد. و برنامه نویس می تواند از دنباله تغییرات فایل استفاده نماید.