فصل دهم

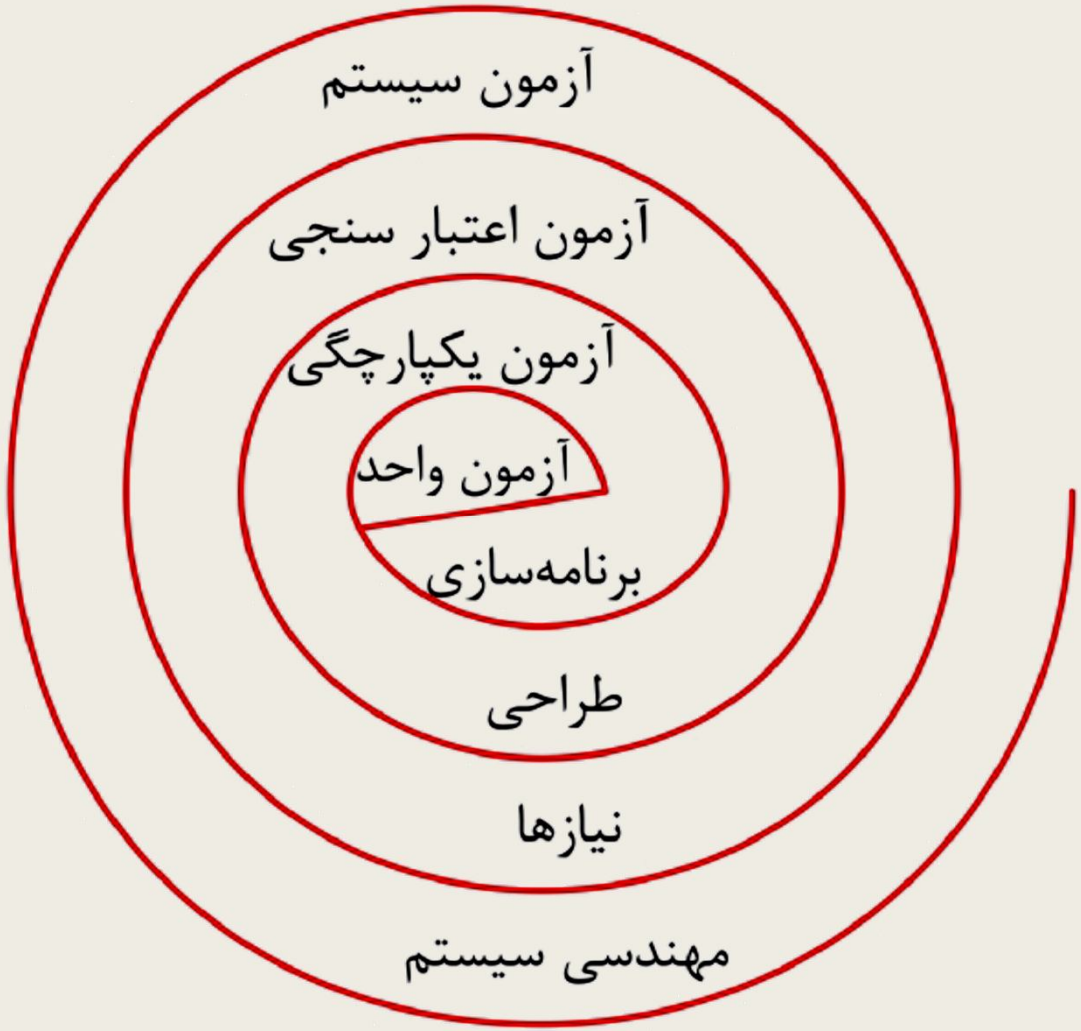
آزمون نرم افزار و راهبردها

مقدمه

آزمایش نرم افزار عنصری حیاتی از تضمین کیفیت نرم افزار است و بیانگر مرور و بازنگری نهایی مشخصهٔ سیستم، تحلیل، طراحی و برنامه های مبدأ است.

- **شیوهٔ راهبردی آزمایش نرم افزار**
  - **آزمایش از سطح مؤلفه شروع می شود، به سمت خارج در جهت یکپارچه سازی کل سیستم رایانه ای پیش می رود.**
  - **روش های متفاوت آزمایش، در نقاط زمانی مختلف مناسب هستند.**
  - **آزمایش توسط توسعه دهندهٔ نرم افزار و برای پروژه های بزرگ توسط گروه مستقل آزمایش، هدایت می شود.**
  - **آزمایش و اشکال زدایی فعالیت های متفاوتی هستند، اما اشکال زدایی باید با هر راهبرد آزمایش همراه باشد.**

راهبرد آزمایش نرم افزار



راهبرد آزمایش نرم افزار

- **آزمایش واحد**
  - **بر روی هر واحد (مؤلفه) نرم افزار متمرکز می شود که با برنامه های مبدأ پیاده سازی شده است.**
- **آزمایش یکپارچه سازی**
  - **بر طراحی و ساخت معماری نرم افزار تأکید دارد**
- **آزمایش اعتبارسنجی**
  - **نیازهایی که به عنوان بخشی از تحلیل نیازهای نرم افزار ایجاد شده اند، در مقابل نرم افزاری که ساخته شده اعتبارسنجی می شوند.**

راهبرد آزمایش نرم افزار

- **آزمایش سیستم**
  - **در آن نرم افزار و عناصر دیگر سیستم به صورت یک مجموعهٔ یکپارچه، آزمایش می شوند.**

برای آزمایش نرم افزار، در طول مارپیچ حرکت کرده و در هر دور، محدودهٔ آزمایش را گسترش می دهیم.

مبانی آزمایش نرم افزار

- **اهداف آزمایش**
  - **Myers چند قانون زیر را بیان می کند که اهداف مناسبی برای آزمایش هستند:**
    - **آزمایش فرایند اجرای برنامه با هدف یافتن خطاست.**
    - **یک نمونهٔ آزمایش خوب، نمونه ای است که با احتمال بالایی خطاها را بیابد.**
    - **آزمایش موفق، آزمایشی است که خطاهای تاکنون پیدا نشده را بیابد.**

مبانی آزمایش نرم افزار

- **اصول آزمایش**
  - **Davis مجموعه ای از اصول زیر را پیشنهاد می کند:**
    - **تمام آزمایش ها باید بر اساس نیازهای مشتری قابل پیگیری باشند.**
    - **آزمایش ها باید مدتی طولانی قبل از شروع آزمایش طراحی و برنامه ریزی شوند.**
    - **اصل Pareto برای آزمایش نرم افزار به کار گرفته شود (قانون ۸۰–۲۰)**
    - **آزمایش باید با توجه به "اجزا" شروع شود و به سمت آزمایش "کلی" پیش رود.**
    - **برای داشتن بیشترین تأثیر، آزمایش باید توسط تیم مستقلی هدایت شود.**
    - **آزمایش کامل و جامع امکان پذیر نیست.**

مبانی آزمایش نرم افزار

- **قابلیت آزمایش**
  - **در موارد ایده آل، مهندس نر مافزار اقدام به طراحی و توسعه برنامه رایانه ای، یک طرح سیستمی یا محصولی را با در نظر داشتن قابلیت آزمایش می کند:**
    - **عملیاتی بودن : "نرم افزار هرچه بهتر کار کند، با کارایی بالاتری آزمایش می شود."**
    - **قابلیت مشاهده : "آنچه می بینید همان است که آزمایش می کنید."**
    - **قابلیت کنترل: "هرچه نرم افزار بهتر کنترل شود، آزمایش بیشتر به طور خودکار و بهینه پذیر انجام پذیر است."**

مبانی آزمایش نرم افزار

- **قابلیت آزمایش**
  - **تجزیه پذیری: "با کنترل دامنهٔ کاربرد آزمایش، می توان مسایل را با سرعت بیشتری تجزیه کرد و آزمایش های مجدد را با هوشمندی انجام داد."**
  - **سادگی: "هرچه مورد برای آزمایش کمتر باشد، آزمایش با سرعت بیشتری انجام می گیرد."**
  - **پایداری: "هرچه تغییرات کمتر باشد، انحراف از آزمایش کمتر است."**
  - **قابلیت فهم: "هرچه اطلاعات بیشتری در اختیار داشته باشیم، آزمایش هوشمندانه تر انجام می شود."**

مبانی آزمایش نرم افزار

- **طراحی نمونه های آزمایش**
  - **طراحی نمونه آزمایش هایی برای نرم افزار و محصولات مهندسی دیگر می تواند به اندازهٔ طراحی اولیهٔ خود محصول متغیر باشد. هر محصول مهندسی (و اکثر چیزهای دیگر) می تواند به یکی از دو روش زیر آزمایش شود**
    - **آزمایش جعبه سیاه: با دانستن عملکرد تابع خاصی که یک محصول برای آن طراحی شده است، آزمایش هایی طراحی م یشوند که مشخص می کنند هر یک ازتوابع کاملا عملیاتی هستند و درعین حال در هر تابع برای یافتن خطاها جستجو صورت می گیرد.**
    - **آزمایش جعبهٔ سفید: با دانستن عملکرد داخلی محصول، آزمایش ها به گونه ای طراحی می شوند که تعیین کنند اعمال داخلی مطابق با مشخصه های تعریف شده انجام می شوند و تمام مؤلفه های داخلی به شکل مناسبی آزمایش می شوند.**

آزمایش جعبهٔ سفید

- **با این روش، مهندس نرم افزار می تواند نمونه های آزمایشی را طراحی کند که:**

1. **تضمین کنند تمام مسیرهای مستقل در پیمانه حداقل یک بار آزمایش می شوند.**
2. **تمام تصمیمات شرطی را در دو بخش درست و غلط بررسی کنند.**
3. **تمام حلقه ها را در شرایط مرزی و در محدوده های عملیاتی اجرا کنند.**
4. **ساختمان داده های داخلی را بررسی کنند تا از اعتبار آنها مطمئن شوند.**

آزمایش جعبهٔ سیاه

- **آزمایش جعبهٔ سیاه سعی در یافتن خطاهایی در دست هبندی های زیر دارد:**
  1. **توابع غلط یا حذف شده**
  2. **خطاهای واسط ها**
  3. **خطا در ساختمان داده ها یا دسترسی به بانک اطلاعاتی خارجی**
  4. **خطاهای رفتاری یا کارایی**
  5. **خطاهای آماده سازی و اختتامیه**

آزمایش جعبهٔ سیاه

- برخلاف آزمایش جعبهٔ سفید که در اوایل فرایند آزمایش انجام می شود، آزمایش جعبهٔ سیاه در مراحل آخر آزمایش به کار گرفته می شود؛ چون آزمایش جعبهٔ سیاه عمدا به ساختار کنترلی توجهی ندارد و بر دامنهٔ اطلاعات متمرکز است:
  - چگونه اعتبار عملکردی آزمایش می شود؟
  - چگونه رفتار و کارایی سیستم آزمایش می شود؟
  - چه رده هایی از ورودی، نمونه های آزمایش خوبی می سازند؟
  - آیا سیستم به مقادیر خاص ورودی حساس است؟
  - چگونه مرزهای یک رده از داده ها مجزا می شود؟
  - سیستم چه نواساناتی در مقابل تغییرات سرعت و حجم داده ها دارد؟
  - ترکیبات خاص داده ها چه اثری بر عملکرد سیستم دارند؟

آزمایش جعبهٔ سیاه

- با به کارگیری روش های آزمایش جعبه سیاه، مجموعه ای از نمون ههای آزمایشی به دست می آیند که صحت عملکرد سیستم را تأیید می کنند:
  - تحلیل مقادیر مرزی
  - آزمایش محیط ها، معماری ها
  - آزمایش مستندات و امکانات راهنما
  - آزمایش یکپارچه سازی
  - آزمایش رگرسیون
  - آزمایش دود
  - صحت و اعتبارسنجی

آزمایش جعبهٔ سیاه

- آزمایش اعتبارسنجی
- آزمایش های پذیرش
- آزمایش سیستم
- آزمایش احیا
- آزمایش امنیت
- آزمایش فشار
- آزمایش کارایی

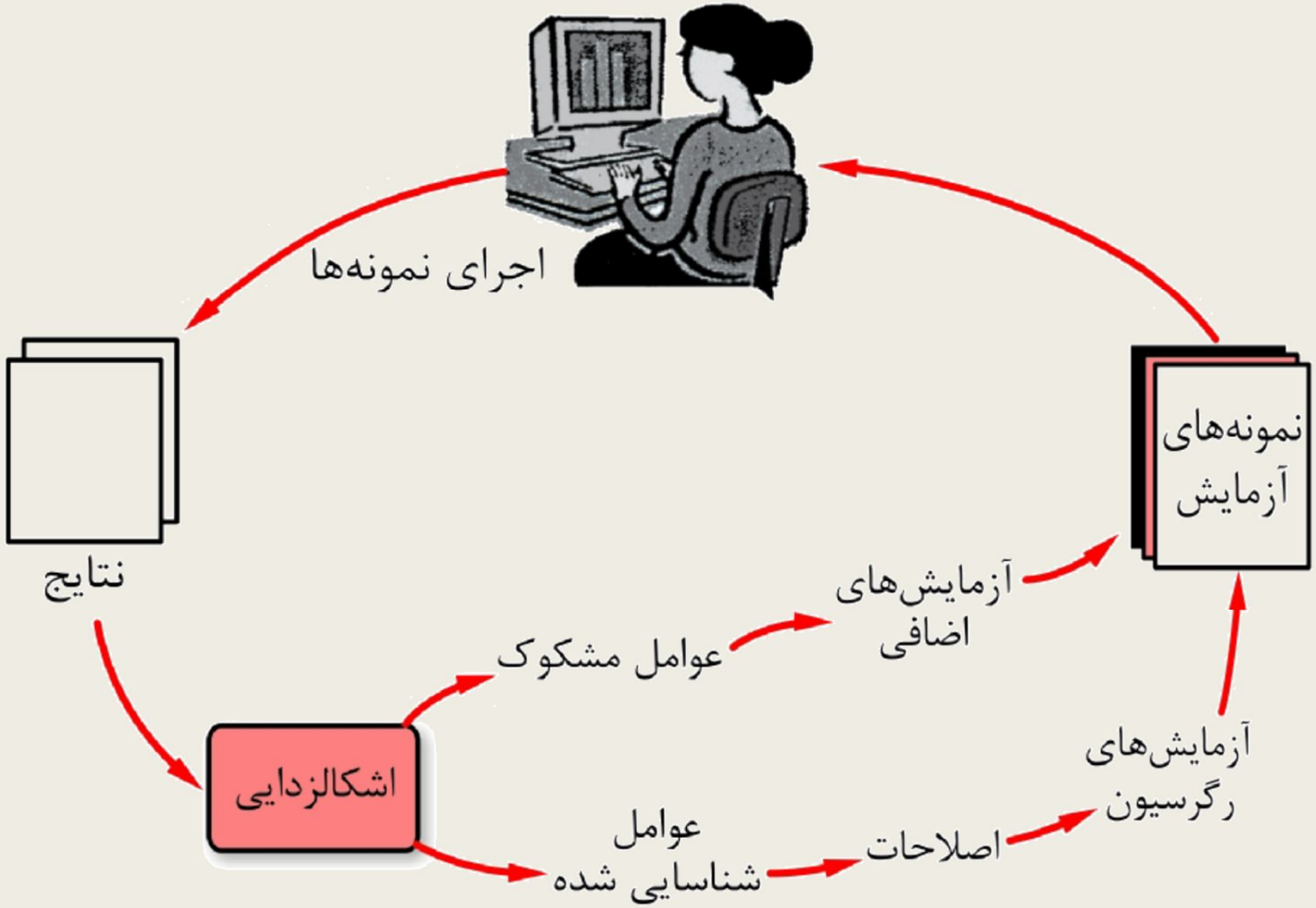
هنر اشکال زدایی(Debugging)

- **اشکال زدایی در نتیجهٔ آزمایش موفق انجام می شود؛ یعنی هنگامی که نمونه های آزمایش خطایی را کشف می کند، اشکال زدایی آغاز می شود. اشکال زدایی فرایندی است که باعث حذف خطا می شود. اگرچه اشکال زدایی فرایندی پوششی است، ولی تا حد زیادی هنری است.**

هنر اشکال زدایی(Debugging)

- **اشکال زدایی در نتیجهٔ آزمایش موفق انجام می شود؛ یعنی هنگامی که نمونه های آزمایش خطایی را کشف می کند، اشکال زدایی آغاز می شود. اشکال زدایی فرایندی است که باعث حذف خطا می شود. اگرچه اشکال زدایی فرایندی پوششی است، ولی تا حد زیادی هنری است.**

هنر اشکال زدایی



هنر اشکال زدایی

- شیوه های اشکال زدایی
  - اشکال زدایی یک هدف پوشش دهنده یعنی **یافتن و تصحیح علت خطای نرم افزار** را دارد. این هدف با ترکیب ارزیابی سیستماتیک، ادراک و شانس به دست می آید. در حالت کلی، سه دسته بندی برای روش های اشکال زدایی توسط Meyer پیشنهاد شده است:
    - نیروی مطلق
    - عقب گرد
    - حذف علت

هنر اشکال زدایی

- شیوه های اشکال زدایی
  - نیروی مطلق(Brute force)

احتمالاً متداول ترین و کم بازده ترین روش برای شناسایی علت خطای نرم افزار است. روش اشکال زدایی نیروی مطلق زمانی به کار گرفته می شوند که همه روش های دیگر با شکست روبه رو شده باشند. این کار با فلسفه "یافتن خطا توسط خود رایانه"، صورت می گیرد که محتویات حافظه روی صفحه نمایش نشان داده می شود. پیگیری های زمان اجرا انجام می شوند و احکام write در برنامه قرارمی گیرند. انتظار می رود در جایی از اطلاعات تولیدشده، کلیدی پیدا شود که بتواند باعث هدایت به سمت خطا شود. علی رغم حجم زیاد اطلاعات تولیدشده ممکن است این روش تا حدی به موفقیت منتهی شود ولی در اکثر موارد باعث اتلاف تلاش و زمان می شود.

هنر اشکال زدایی

- **شیوه های اشکال زدایی**
  - **عقب گرد (Backtracking)**

روشی نسبتاً متداول است که می تواند با موفقیت در برنامه های کوچک به کار گرفته شود. با شروع از محلی که علامت در آنجا ظاهر شده، برنامه مبدأ به سمت عقب (به صورت دستی) دنبال می شود تا زمانی که محل علت بروز خطا پیدا شود. بدبختانه با افزایش تعداد خطوط برنامه، تعداد مسیرهای برگشت بسیار زیاد خواهد بود و کارایی روش پایین می آید.

هنر اشکال زدایی

- **شیوه های اشکال زدایی**
  - **حذف علت**

روش سوم اشکال زدایی است که با بسط یا حذف انجام می شود و مفهوم تقسیم بندی دودویی را همراه دارد. داده های مرتبط با خطا سازماندهی می شوند تا علل بالقوه را جدا کنند. در یکی از حالات، لیستی از تمام علت های ممکن توسعه داده می شود و آزمایش های مربوط انجام می شوند تا هر یک را حذف کند. اگر آزمایش های اولیه نشان دهند که یک علت خاص، مربوط به علامت ظاهرشده است، داده ها باید پالایش شوند تا خطا از بین برود.

فصل یازدهم
مفاهیم مدیریت پروژه های نرم افزاری

مقدمه

مدیریت پروژه شامل برنامه ریزی، نظارت و کنترل افراد، فرایندها و رویه هایی می شود که به موازات توسعه و تکامل نرم افزار از مفهوم مقدماتی تا پیاده سازی عملی رخ می دهند.

در مورد روش های مدیریت پروژه دیدگاه های مختلفی تاکنون عرضه شده است، ولی معتبرترین دیدگاه توسط مؤسسه مدیریت پروژه(PMI) در سال ۱۹۹۹ ارائه شده است. در سال های اخیر نسخه های جدیدتری از استاندارد مدیریت پروژه توسط این مؤسسه انتشار وتوزیع شده است. این مؤسسه کتابی تحت عنوان پیکرهٔ دانش مدیریت پروژه(PMBook) برای مدیریت پروژه ارائه داده است. این کتاباستانداردی برای تمام کسانی است که مسئولیت مدیریت پروژه را بر عهده دارند.

حوزه های پیکره دانش مدیریت پروژه

- مدیریت حوزه عملکرد(Scope Management)
- مدیریت یکپارچگی(Integration Management)
- مدیریت زمان(Time Management)
- مدیریت هزینه(Cost Management)
- مدیریت منابع انسانی(Human Resource Management)
- مدیریت ریسک(Risk Management)
- مدیریت کیفیت(Quality Management)
- مدیریت ارتباطات(Communication Management)
- مدیریت تدارکات(Procurement Management)

طیف مدیریتی – افراد

- در مدیریت مؤثر پروژه های نرم افزاری بر چهار محور اساسی تأکید شده است:
- افراد(People)

عامل انسانی چنان اهمیتی دارد که مؤسسهٔ مهندسی نر مافزار، مدل بلوغ قابلیت های مدیریت افراد(PM-CMM) را توسعه داده است تاآمادگی سازمان های نرم افزاری برای به عهده گرفتن کاربردهای پیچیده را بهبود بخشد. مدل بلوغ قابلیت های مدیریت افراد، زمینه های کلیدی عملی زیر را برای نرم افزارنویسان تعیین می کند:

یافتن افراد جدید، انتخاب، مدیریت اجرا، آموزش، جبران خدمت، توسعه حرفه ای، سازماندهی و طراحی کار و توسعه تیمی/ فرهنگی.

طیف مدیریتی – محصول

- **محصول(Product)**

دامنهٔ کاربرد محصول و تجزیه مسئله و همچنین روش های مربوط از طریق ارتباط با مشتری تعیین می شود. انجام برآوردهای منطقی از پروژه، مستلزم یک تحلیل نیازها همراه با جزئیات است که متأسفانه این کار زمان زیادی می برد. بنابراین محصول، دامنه و هدف آن باید بررسی و مبنایی برای برآوردهای پروژه ایجاد شود.

طیف مدیریتی – محصول

- **دامنهٔ کاربرد محصول**

نخستین فعالیت مدیریت پروژه نرم فزار، تعیین دامنهٔ کاربرد محصول است. دامنهٔ کاربرد با مشخص کردن مؤلفه های زیر تعیین می شود:

- **محیط: چگونه نرم افزار ایجادشده با سیستم های موجود یا محیط حرفه منطبق می شود و محدودیت ها در این محیط چیست؟**
- **اهداف اطلاعاتی: موجودیت ها و اقلام داد های برای خروجی مطلوب و مورد نظر کدام هستند.**
- **عملکرد و کارایی: نحوهٔ تبدیل داده های ورودی به اطلاعات خروجی و کارایی مورد نظر چیست؟**

طیف مدیریتی – محصول

- **تجزیه مسئله**

تجزیه مسئله کمک بزرگی برای یافتن دامنه و حوزه عملکرد مسئله است. به همین منظور باید از راه های اصولی ارتباط با مشتری استفاده کرد. تجزیه مسئله در دو زمینهٔ اصلی صورت می پذیرد:

- **قابلیت عملیاتی قابل تحویل**
- **فرایندهای مورد نیاز برای تحویل**

طیف مدیریتی – فرآیند

- **فرآیند (Process)**

مدیر پروژه باید در مورد انواع مدل های فرایندی توسعهٔ نرم افزار تصمیم گیری کند به طوری که مدل فرایندی برای حالت های زیر مناسب باشد:

- **مشتری و افرادی که کار را انجام می دهند.**
- **ویژگی های محصول**
- **محیط پروژه که در آن تیم نرم افزاری کار می کند.**

پس از انتخاب مدل فرایندی توسعهٔ نرم افزار، مدیر پروژه باید برنامهٔ توسعهٔ نرم افزار را تهیه و ارائه کند. به همین منظور وی باید اقدام به "تلفیق فرایند و محصول" و "تجزیه فرایند" اجرای پروژه کند.

طیف مدیریتی – پروژه

- **پروژه (Project)**

برای مدیریت یک پروژه نرم افزاری موفق، باید بدانیم چه چیزهایی ممکن است اشتباه شود و چگونه م یتوان آن ها را به راحتی رفع کرد. John Reel علایم در خطر بودن یک پروژه را به شرح زیر اعلام کرده است:

- **عدم درک نیازهای مشتری**
- **عدم تعریف خوب از حوزه محصول**
- **عدم مدیریت مطلوبِ تغییرات**
- **تغییر فن آوری انتخاب شده**
- **تغییر نیازهای تجاری**
- **مهلت های غیر واقع بینانه**
- **مقاومت کاربران**
- **از بین رفتن حمایت مالی**
- **عدم مهارت افراد تیم پروژه**
- **عدم تلاش کافی برای استفاده از تجارب گذشته**

فصل دوازدهم
معیارهای اندازه گیری و سنجش نرم افزار

مقدمه

معیارهای سنجش نرم افزار به گسترهٔ وسیعی از اندازه گیری های مربوط به نرم افزارهای رایانه ای بازمی گردد. اندازه گیری را در مورد فرایند نرم افزار می توان

- **به قصد بهبود مستمر به کار برد.**
- **برای کمک به انجام برآورد، کنترل کیفیت، ارزیابی بهره وری و کنترل پروژه به کار گرفت.**
- **و نهایتاً می توان برای کمک به ارزیابی کیفیت محصولات فنی و کمک به تصمیم گیری های تاکتیکی در حین پیشرفت پروژه به کار بست.**

در حوزهٔ مدیریت پروژه های نرم افزاری، ابتدا با معیارهای بهره وری و سنجش کیفیت سر و کار داریم که شامل اندازه های خروجی فرایند توسعهٔنرم افزار به صورت تابعی از تلاش و زمان به کار رفته است و سپس با اندازه گیری های مربوط به تطابق با کاربرد محصولات تولیدشده، سر و کارداریم.

اندازه، معیار سنجش و شاخص

فرهنگ واژه های مؤسسه مهندسی نرم افزارIEEE، معیار سنجش را به عنوان اندازه مقداری از درجه ای که یک سیستم، مؤلفه یا یک فرایند درمورد یک صفت ویژه اعمال می کند، تعریف کرده است. مثل تعداد خطاهای حاصل از بازبینی یک پیمانهٔ مجزا.

معیار سنجش نرم افزار با تک تک اندازه ها در ارتباط است مانند تعداد میانگین خطاها در اثنای بازبینی های رسمی.

شاخص یک معیار سنجش یا تلفیقی از آن هاست که امکان نگرش به درون فرایند نرم افزار، پروژهٔ نرم افزاری و یا خود محصول را فراهم می کند.

اندازه، معیار سنجش و شاخص

ابزارهای سنجش، بینشی در اختیار مدیر قرار می دهد که منجر به تصمیم گیری های آگاهانه می شود. بنابراین:

- **اندازه(Measure):**

  نمایش مقداری از میزان، ابعاد، ظرفیت و و یژگی های نرم افزار است.

- **معیار(Metric):**

  اندازهٔ مقداری درجه یک سیستم، مؤلفه یا فرایند است.

- **شاخص(Indicator):**

  مجموعه ای از معیارهای سنجش برای تصحیح عملکرد پروژه است.

معیارهای سنجش فرایند

معیارهای سنجش فرایند در طول تمام پروژه ها و در طی یک دورهٔ زمانی طولانی گردآوری می شوند. هدف از تهیه معیارهای سنجش فرایند فراهم آوردن شاخص هایی برای بهبود فرایند نرم افزار در بلندمدت است.

شاخص های پروژه، یک مدیر پروژه نرم افزاری را قادر می کند که:

1. **وضعیت یک پروژه در حال جریان را ارزیابی کند.**
2. **ریسک های بالقوه را پیگیری کند.**
3. **زمینه های مشکل را پیش از آنکه « بحرانی » شوند، معلوم کند.**
4. **جریان کار یا فعالیت ها را تعدیل کند.**
5. **قابلیت گروه پروژه را در مورد کنترل کیفیت محصولات کاری مهندسی نرم افزار مورد ارزیابی قرار دهد.**

معیارهای سنجش فرایند

Grady استفاده های عمومی و خصوصی برای انواع متفاوت داده های فرایند را مطرح می کند.

- **معیارهای خصوصی**
  - **نرخ خرابی (توسط فرد)**
  - **نرخ خرابی (توسط پیمانه)**
  - **خطاهای پیداشده در طول توسعه نرم افزار**

هدف: یافتن شاخص های بهبود و افزایش کارایی فرد است

معیارهای سنجش فرایند

- **معیارهای عمومی**
  - **تعداد نقص های گزارش شده در مورد توابع و عملکردهای عمده نرم افزار**
  - **تعداد خطاهای مشاهده شده در حین بازبینی های فنی رسمی**
  - **تعداد خطوط برنامه (LOC) یا ارزش تابعی (FP)**

هدف: یافتن شاخص های بهبود و افزایش کارایی فرایند سازمانی است.

معیارهای سنجش پروژه

معیارهای سنجش فرایند نرم افزاری برای مقاصد راهبردی به کار می روند. اندازه های مقداری پروژه نرم افزاری جنبه فنی دارند.

یعنی معیارهای سنجش و شاخص های پروژه از آن ها حاصل شده و از سوی مدیر پروژه و گروه نرم افزاری برای تطبیق جریان کار پروژه و فعالیت های فنی مورد استفاده قرار می گیرند.

معیارهای سنجش پروژه

مدل اندازه گیری پروژه

مدل دیگری برای اندازه گیری پروژه موارد زیر را پیشنهاد می کند:

- **ورودی ها:** شاخص های مقداری منابع (مثلاً افراد و محیط) مورد نیاز برای انجام کار
- **خروجی ها:** شاخص های مقداری اهداف واسط یا محصولات کاری تولیدشده در طول فرایند مهندسی نرم افزار
- **نتایج:** شاخص های مقداری که بر ثمربخشی اهداف و محصولات واسط دلالت دارد

درحقیقت، این مدل را می توان هم برای فرایند و هم پروژه به کار برد.

اندازه گیری نرم افزاری

در دنیای واقعی اندازه گیری را می توان به دو دستهٔ عمده تقسیم کرد:

اندازه گیری مستقیم (نظیر طول یک پیچ) و اندازه گیری غیرمستقیم (نظیر سنجش «کیفیت» پیچ های تولیدشده (از طریق شمارش تعداد پیچ های ردشده).

اندازه گیری مستقیم فرایندهای مهندسی نرم افزار دربرگیرندهٔ سنجش هزینه و تلاش است. اندازه گیری مستقیم نرم افزار شامل تعداد خطوط برنامه(LOC) تولیدشده، سرعت اجرا، میزان حافظه و خرابی های گزارش شده در مدت یک دورهٔ زمانی است.

سنجش های غیرمستقیم نرم افزار شامل قابلیت تابعی بودن(Functionality)، کیفیت، پیچیدگی، کارایی، قابلیت اعتماد، قابلیت نگهداری و«قابلیت»های بسیار دیگر است.

اندازه گیری نرم افزاری

- **معیارهای سنجش مبتنی بر اندازه**

معیارهای سنجش مبتنی بر اندازهٔ نرم افزار از طریق نرمال سازی اندازه های مقداری کیفیت یا بهره وری با استفاده از « اندازهٔ » نرم افزار تولیدشده حاصل می شوند. مانند:

- **تعداد افراد مشارکت کننده در توسعه نرم افزار (People)**
- **تعداد صفحات مستندات (pp.Doc)**
- **میزان هزینه (Cost)**
- **تعداد خطوط برنامه (LOC)**
- **تعداد خطاهای پیداشده (Error)**
- **تعداد خرابی یا شکستها (Defects)**

برای به دست آوردن معیارهای سنجش که قابل مقایسه با معیارهای سنجش پروژهٔ دیگر باشد، شاخص تعداد خطوط برنامه (LOC) را به عنوان ارزش هنجارسازی در نظر گرفته ایم

اندازه گیری نرم افزاری – معیارهای سنجش مبتنی بر اندازه

- معیارهای نرمال شده
  - تعداد خطاها در هر KLOC (هزار خط برنامه)
  - تعداد خرابی یا شکست ها در هرKLOC
  - هزینه هرLOC
  - تعداد صفحات مستندات در هرKLOC

به علاوه، شاخص های جالب دیگری را نیز می توان به صورت زیر محاسبه کرد:

- نسبت خطاها به نفر – ماه
- نسبت LOC به نفر – ماه
- نسبت هزینه به حجم مستندات

اندازه گیری نرم افزاری

- **معیارهای سنجش مبتنی بر تابع (عملکرد)**

از یک معیار اندازه گیری تابعی برنامه به عنوان ارزش نرمال شده استفاده می کنند. چون تابعی بودن یک نرم افزار را نمی توان مستقیما اندازه گیری کرد پس باید با استفاده از سایر معیارهای مستقیم، به طور غیر مستقیم به دست آورد.

معیارهای سنجش مبتنی بر عملکرد توسط Albrecht با نام ارزش تابعی معرفی شده است. ارزش تابعی با استفاده از یک رابطهٔ تجربی مبتنی بر معیارهای قابل اندازه گیری (مستقیم) دامنه اطلاعات نرم افزار و ارزیابی های مربوط به پیچیدگی نرم افزار حاصل می شود. برای این کار پنج ویژگی از دامنهٔ اطلاعاتی تعیین و شمارش می شوند:

اندازه گیری نرم افزاری – معیارهای سنجش مبتنی بر تابع

- **تعداد ورودی های کاربران: شامل هر ورودی کاربر است که داده های مبتنی بر کاربردهای مجزا برای نرم افزار فراهم می آورند.**
- **تعداد خروجی های کاربر: هر خروجی مورد نظر کاربر است و شامل اطلاعات حاصل از برنامه محسوب می شود. در این زمینه خروجی ها به گزارش ها، صفحات نمایش، پیغا مهای خطا و ... اطلاق می شود. اقلام داده ای مجزا در یک گزارش به طور جداگانه محاسبه نمی شوند.**
- **تعداد پرس وجو (درخواست) های کاربران: یک درخواست به عنوان یک ورودی لحظه ای تعریف می شود که حاصل آن تولید پاسخ توسط نرم افزار به شکل خروجی های لحظه ای است.**
- **تعداد پرونده ها: هر پروندهٔ اصلی منطقی (نظیر گروه بندی منطقی داده ها به عنوان بخشی از یک پایگا ه دادهٔ بزرگ یا یک پرونده جداگانه) شمارش می شوند.**
- **تعداد واسط های خارجی: شمارش تمام واسط های قابل خواندن توسط دستگاه (نظیر پرونده های داده ای روی نوار یا دیسکت) که برای ارسال اطلاعات به یک سیستم دیگر مورد استفاده هستند.**

اندازه گیری نرم افزاری – معیارهای سنجش مبتنی بر تابع

- **سپس جدول زیر تکمیل و ارزش تابعی از رابطهٔ تجربی ارائه شدهٔ زیر به دست می آید:**

پارامترهای اندازه گیری	تعداد		ضرایب وزنی: ساده	متوسط	پیچیده	
تعداد ورودیهای کاربر		×	۳	۴	۶	=
تعداد خر وجیهای کاربر		×	۴	۵	۷	=
تعداد درخواستهای کاربر		×	۳	۴	۶	=
تعداد فایلها		×	۴	۱۰	۱۵	=
تعداد واسطهای خارجی		×	۵	۷	۱۰	=

شمارش کل

اندازه گیری نرم افزاری – معیارهای سنجش مبتنی بر تابع

وقتی داده های بالا گردآوری شد ارزش پیچیدگی هر یک از آن ها تعیین می شود. سازمان هایی که از روش ارزش تابعی استفاده می کنند معیارهایی را برای تعیین ساده، متوسط یا پیچیده بودن داده های ورودی، توسعه داده اند. برای محاسبه ارزش تابعی از رابطه زیر استفاده می شود:

$$FP = \text{شمارش کل} \times [0.65+0.01\Sigma F_i]$$

که در آن F_i میزان تعدیل پیچیدگی است و پاسخ ۱۴ سؤال است که به صورت زیر برای سؤالات تعیین می شوند:

۵	۴	۳	۲	۱	۰
اساسی	با اهمیت	متوسط	معقول	ضعیف	بی تاثیر

- **نرمال سازی ارزش تابعی**

وقتی ارزش تابعی محاسبه شد، به طریقی شبیه به روش LOC مقادیر حاصله نرمال سازی می شوند تا برای سنجش بهره وری، کیفیت و سایرمشخصه های نرم افزار به کار روند.

تلفیق راهکارهای متفاوت سنجش

جدول روبرو شامل برآوردهای خام مربوط به متوسط تعداد خطوط برنامهٔ مورد نیاز برای ساخت یک واحد ارزش تابعی توسط زبان های برنامه نویسی مختلف را نشان می دهد.

مروری بر این داده ها نشان می دهد که قابلیت تابعی حاصل از یک LOC زبان C++ به طور تقریبی ۱/۶ برابر LOC زبان فرترن است. همچنین یک LOC زبان های نسل چهارم سه تا پنج برابر زبا نهای برنامه نویسی قراردادی، قابلیت تابعی ایجاد می کنند.

زبان برنامه نویسی	متوسط LOC/FP
زبان اسمبلی	۳۲۰
زبان C	۱۲۸
زبان کوبول	۱۰۵
زبان فرترن	۱۰۵
زبان پاسکال	۹۰
زبان C++	۶۴
Ada 95	۵۳
زبان های شیء گرا مانند Visual basic	۳۰
زبان های نسل چهارم مانند smaltalk	۲۰
مولد کد	۱۶
SQL	۱۲
صفحه گسترها	۶
زبان های گرافیکی	۴

تلفیق راهکارهای متفاوت سنجش

- **اندازه گیری کیفیت**

اگرچه معیارهای بسیاری در مورد کیفیت، مانند صحت (درستی عملکرد)، قابلیت نگهداری، یکپارچگی و قابلیت استفادهٔ مجدد تعاریف و معیارهایی را برای هر یک از آنها Gilb نرم افزار وجود دارند که شاخص های مفیدی را برای گروه پروژه فراهم می آورند، با این وجود پیشنهاد می کند

- **صحت (درستی عملکرد):**

یک برنامه باید به طور درستی عمل کند وگرنه برای کاربر ارزشی نخواهد داشت. درستی عملکرد درجه ای است که نشان دهندهٔ صحت عملکرد نرم افزار است. از معمول ترین معیارهای درستی عملکرد می توان به تعداد خرابی در هر KLOC اشاره کرد که در آن خرابی (شکست) به عنوان عدم تطابق مشاهده شده با نیازها تعریف می شود.

تلفیق راهکارهای متفاوت سنجش–اندازه گیری کیفیت

- **قابلیت نگهداری:**

قابلیت نگهداری امکانی است که از طریق آن در صورت بروز خطا می توان یک برنامه را اصلاح کرد و یا در صورت تغییر محیط آن را با محیط تطبیق داد، یا درصورتی که مشتری خواهان تغییری در نیازهای نرم افزار باشد امکاناتی به آن اضافه کرد. هیچ راهی برای اندازه گیری مستقیم قابلیت نگهداری وجود ندارد، بنابراین باید از معیارهای غیر مستقیم استفاده کرد. معیار ساده مبتنی بر زمان میانگین زمان مورد نیاز برای تغییر (MTTC) است که مدت زمان لازم برای تحلیل درخواست تغییر، آزمایش و توزیع نسخه تغییریافته بین تمام کاربران را نشان می دهد. به طور متوسط برنامه هایی قابل نگهداری هستند که دارای MTTC پایین تری(برای انواع تغییرات یکسان) هستند.

تلفیق راهکارهای متفاوت سنجش–اندازه گیری کیفیت

- **یکپارچگی:**

یکپارچگی نرم افزار در عصر سرقت ها و ویروس های رایانه ای به طور فزاینده ای اهمیت یافته است. این مشخصه، قابلیت یک سیستم را در مقابله با دستکاری های تصادفی یا عمدی در راستای ایمنی آن اندازه گیری می کند. این دستکاری ها می توانند در مورد سه جزء نرم افزار (برنامه، داده و مستندات) انجام گیرند.

برای سنجش یکپارچگی نرم افزار دو مشخصه تهدید و ایمنی به صورت زیر تعریف می شود:

- **تهدید احتمال رخداد یک دستکاری از نوع خاص در یک زمان مشخص است (این مشخصه را می توان از شواهد تجربی بر آورد کرد).**
- **ایمنی احتمال دفع یک دستکاری خاص است (این مشخصه را نیز می توان از شواهد تجربی بر آورد کرد).**

در این صورت یکپارچگی سیستم به صورت زیر تعریف می شود:

$$\text{یکپارچگی} = \Sigma[\text{(ایمنی}-1)\text{تهدید}\text{-}1]$$

که در آن تهدید و ایمنی برای هر نوع دستکاری محاسبه می شوند.

تلفیق راهکارهای متفاوت سنجش–اندازه گیری کیفیت

- **قابلیت استفاده:**

بحث سازگاری کاربر با محصولات نرم افزاری و به عبارت دیگر کاربرپسندی نرم افزار جایگاه خاص خود را پیدا کرده است. اگر برنامه ای کاربرپسند نباشد علی رغم دارا بودن توابع و عملکردهای با ارزش، اغلب با شکست مواجه می شود. قابلیت استفاده تلاشی برای مقداری کردن کاربرپسندی نرم افزار است که از طریق چهار ویژگی زیر اندازه گیری می شود:

1. **مهارت فیزیکی یا ذهنی مورد نیاز برای یادگیری نرم افزار**
2. **مدت زمان مورد نیاز برای کسب کارایی بالا در استفاده از سیستم**
3. **خالص افزایش بهره وری در مقایسه با سیستم قبلی**
4. **ارزیابی شهودی کاربر از نرم افزار (گاهی این معیار از طریق پرسشنامه و نظرسنجی از کاربران نسبت به سیستم به دست می آید).**

بازدهی رفع نقص (شکست)

یکی دیگر از معیارهای کیفی که در هر دو سطح پروژه و فرایند سودمند است، کارایی یا بازدهی رفع نقص (DRE) است و به صورت زیر تعریف می شود:

$$DRE = E /(E + D)$$

که در آن:

E: تعداد خطاهای پیداشده قبل از تحویل نرم افزار به کاربر نهایی

D: تعداد شکست هایی که پس از تحویل نرم افزار به مشتری مشخص می شوند.

مقدار ایده آل DRE برابر با ۱ است که نشان دهنده عدم وجود خرابی در نرم افزار است.

اگر شاخص DRE به عنوان معیاری برای قابلیت فیلترکردن فعالیت های تضمین و کنترل کیفیت به کار رود، می تواند یک گروه پروژه نرم افزاری را به برقراری روش هایی برای یافتن خطاهای هرچه بیشتر قبل از تحویل نرم افزار به مشتری تشویق کند.

فصل سیزدهم
برنامه ریزی پروژه های نرم افزاری

مقدمه

فرایند مدیریت پروژه های نرم افزاری شامل مجموعه ای از فعالیت هاست که اولین فعالیت آن برنامه ریزی پروژه نامیده می شود. قبل از آغاز پروژه، مدیر و تیم نر مافزاری باید ضمن تعریف فعالیت های پروژه منابع مورد نیاز، زمان و هزینه پروژه را برآورد کنند.

هدف از برنامه ریزی پروژه، تهیه و تدوین چارچوبی برای مدیریت پروژه در جهت برآورد معقول و منطقی منابع، هزینه و زمان است که در مدت اجرای پروژه به هنگام می شوند.

برآوردها

برآورد منابع، هزینه و زمان بندی برای کارهای مهندسی نرم افزار نیاز به داشتن تجربه، دستیابی به اطلاعات تاریخی خوب و جرأت انجام پیش بینی های کمی و مقداری به هنگام وجود اطلاعات صرفاً کیفی است. برآورد، به طور ذاتی با ریسک همراه است و همین ریسک منجر به عدم قطعیت در فرایند مهندسی نرم افزار می شود.

بنابراین برآورد منبع، هزینه و زمان بندی توسعهٔ نرم افزار نیاز به دانستن عوامل تأثیرگذار بر آن ها به شرح زیر دارد:

- **تجربه**
- **دستیابی به اطلاعات سابقه ای و مشابه**
- **داشتن معیارهای اندازه گیری و**
- **عدم قطعیت ها**

برآوردها

برآورد منابع، هزینه و زمان بندی برای کارهای مهندسی نرم افزار نیاز به داشتن تجربه، دستیابی به اطلاعات تاریخی خوب و جرأت انجام پیش بینی های کمی و مقداری به هنگام وجود اطلاعات صرفاً کیفی است. برآورد، به طور ذاتی با ریسک همراه است و همین ریسک منجر به عدم قطعیت در فرایند مهندسی نرم افزار می شود.

بنابراین برآورد منبع، هزینه و زمان بندی توسعهٔ نرم افزار نیاز به دانستن عوامل تأثیرگذار بر آن ها به شرح زیر دارد:

- **تجربه**
- **دستیابی به اطلاعات سابقه ای و مشابه**
- **داشتن معیارهای اندازه گیری و**
- **عدم قطعیت ها**

عوامل تأثیرگذار بر برآوردها: پیچیدگی پروژه، اندازهٔ پروژه، درجه یا میزان عدم قطعیت ساختاری

منابع

دومین کار برنامه ریزی پروژهٔ نرم افزاری تخمین و برآورد منابع لازم برای انجام فعالیت توسعه نرم افزار است. شکل زیر منابع مورد نیاز برای توسعه نرم افزار را به صورت هرم نشان می دهد. محیط توسعه، شامل ابزارهای نرم افزاری و سخت افزاری، در بخش بنیادی هرم منابع قرارمی گیرد و فراهم کنندهٔ زیربنایی برای حمایت از تلاش های توسعه نرم افزار است. در سطح بالاتر، اجزای نرم افزاری قابل استفادهٔ مجدد در نظر گرفته می شوند که عبارت اند از اجزای تشکیل دهنده نرم افزار که می توانند هزینه های توسعه را کاهش داده و تحویل نرم افزار را شتاب بخشند. در بالای این هرم، منبع افراد قرار دارد. هر منبع با چهار ویژگی مشخص می شود: توصیف منبع، قابلیت دسترسی به منبع، زمان مورد نیاز و مدت زمان به کارگیری منبع.

منابع


افراد
مولفه‌های
نرم افزاری قابل استفاده مجدد
ابزارهای
سخت افزاری/نرم افزاری

منابع

- **منابع انسانی**

تعداد افراد لازم برای پروژه نرم افزاری می تواند فقط بعد از برآورد فعالیت توسعه (برای مثال، نفر– ماه) و تعیین مواردی مانند تعداد افراد و مهارت های مورد نیاز شامل شرح مهارت منبع، وضعیت دسترسی، توالی زمانی مورد استفاده از منبع و مدت زمان استفاده از آن انجام شود.

- **منابع نرم افزاری قابل استفادهٔ مجدد**

مهندسی نرم افزار مبتنی بر مؤلفه ها(CBSE) بر قابلیت اطمینان تأکید دارد، چنین اجزای تشکیل دهنده ای، اغلب مؤلفه نامیده می شود و باید برای ارجاع آسان به آن ها طبقه بندی شوند، برای به کارگیری آسان باید استاندارد شوند و برای ترکیب ساده باید اعتبارسنجی شوند بنابراین:

- **مؤلفه های آماده اعتبارسنجی شده**
- **مؤلفه های کاملاً آزمایش شده**
- **مؤلفه های آزمایش نشده**
- **مؤلفه های جدید**

منابع

- **منابع محیطی**

محیطی که پروژه نرم افزار را حمایت می کند، اغلب محیط مهندسی نرم افزار(SEE) نامیده می شود که نرم افزار و سخت افزار را به کار می گیرد. سخت افزار، مبنایی را ایجاد می کند که ابزارهای لازم برای تولید محصولات کاری مناسب مهندسی نرم افزار را پشتیبانی می کند. چون اکثر سازمان های نرم افزاری دارای چندین گروه تخصصی هستند که نیازمند دسترسی به محیط مهندسی نرم افراز(SEE) هستند، برنامه ریز پروژه باید دوره زمانی لازم برای استفاده از سخت افزار و نرم افزار را مشخص کرده و تعیین کند که آیا این منابع در آن دوره ها در دسترس هستند یاخیر؟ بنابراین منابع محیطی شامل نرم افراز، سخت افراز و ارتباطات داده ای است.

برآورد پروژه نرم افزاری

برآورد پروژهٔ نرم افزاری می تواند از یک هنر ناشناخته به یک سری مراحل سیستماتیک تبدیل شود که تخمین هایی را همراه با ریسک قابل قبول فراهم کند. روش های برآورد عبارت اند از:

1. تأخیر در برآورد حتی تا اواخر پروژه
2. استفاده از بهترین برآوردهای انجام شده روی پروژه هایی که قبلاً کامل شده اند (مشابه سازی)
3. استفاده از روش های تجزیه نسبتاً ساده برای برآوردهای هزینه ای، زمانی و تلاش پروژه
4. استفاده از مدل های تجربی برآورد هزینه و تلاش مورد نیاز

برآورد پروژه نرم افزاری

مدل های تجربی را می توان برای تکمیل روش های تجزیهٔ پروژه به کار برد و یک روش ارزشمند بالقوه ارائه داد. ابزارهای برآورد خودکار نیز یک یا چند روش تجزیه و مدل تجربی را پیاده سازی می کنند.

روش اول عملی نیست، روش دوم در صورت وجود پروژهٔ مشابه قابل قبول است. حالت های باقی مانده، روش های عملی برای برآورد پروژهٔ نرم افزار هستند.

روش های تجزیه از شیوهٔ تقسیم و تجزیه به اجزای کوچک برای برآورد پروژه های نرم افزاری استفاده می کنند. با تجزیهٔ پروژه به توابع و فعالیت های عمدهٔ مهندسی نرم افزار، برآورد هزینه و تلاش می تواند به روش مرحله ای انجام شود.

روش های تجزیهٔ برآورد پروژهٔ نرم افزار

روش تجزیه از دو دیدگاه متفاوت تجزیه مسئله و تجزیهٔ فرایند مد نظر هستند. در برآورد پروژه های نرم افزاری یکی یا هر دو، مورد استفاده قرار می گیرند.

- **تعیین اندازه نرم افزار**

دقت برآورد پروژهٔ نرم افزار به چند عامل بستگی دارد:

1. **درجه ای که برنامه ریز اندازهٔ محصول را تخمین زده است**
2. **توانایی ترجمهٔ برآورد اندازه نرم افزار به فعالیت انسانی، تقویم زمانی و هزینه (تابعی از در دسترس بودن یا قابل اطمینان بودن معیارهای نرم افزار از پروژه های قبلی)**
3. **درجه ای که طرح پروژه توانایی تیم نرم افزار را نشان می دهد،**
4. **پایداری نیازهای پروژه و محیطی که فعالیت مهندسی نرم افزار را حمایت می کند.**

روش های تجزیهٔ برآورد پروژهٔ نرم افزار

- چون برآورد پروژه منوط به برآورد اندازه نرم افزار است بنابراین اولین کوشش عمدهٔ برنامه ریز پروژه است.
- در برنامه ریزی پروژه، منظور از اندازه به نتیجهٔ کمیّت پذیر از پروژه نرم افزاری اشاره دارد.
- اگر روشی مستقیم به کار گرفته شود، اندازه می تواند برحسب LOC اندازه گیری شود.
- اگر روش غیر مستقیم به کار گرفته شود، اندازه برحسب FP نشان داده می شود.

روش های تجزیهٔ برآورد پروژهٔ نرم افزار

- **برآورد مبتنی بر مسئله**

داده های LOC و FP به دو روش در ضمن برآورد پروژه نرم افزار استفاده می شوند:

1. **به عنوان متغیر تخمین برای تعیین اندازهٔ هر عنصر نرم افزاری**
2. **به عنوان معیارهای بنیادی جمع آوری شده از پروژه های قبلی و استفاده به همراه متغیرهای تخمینی از هزینه و کار**

برآورد LOC و FP روش های برآورد متمایزی هستند. با این وجود هر دو ویژگی های مشترکی دارند. برنامه ریز پروژه اقدام به تجزیهٔ نرم افزار می کند که هر یک م یتوانند به صورت مجزا برآورد شوند. سپس LOC یا FP (متغیر برآوردی) برای هر تابع تخمین زده می شود.

روش های تجزیهٔ برآورد پروژهٔ نرم افزار – برآورد مبتنی بر مسئله

- برنامه ریز پروژه با برآورد محدوده مقادیری برای هر تابع یا مقداری در محدودهٔ اطلاعات، کار را شروع می کند. با استفاده از داده های سابقه ای یا(در صورت شکست موارد دیگر) با استفاده از ادراک، برنامه ریز مقدار اندازهٔ **خوشبینانه**، **محتمل** و **بدبینانه** را برای هر تابع یا هر مقدار موجود در محدودهٔ اطلاعات تخمین می زند. سپس یک مقدار مورد انتظار محاسبه می شود. مقدار مورد انتظار برای متغیر برآورد(اندازه)، S، به صورت متوسط وزنی مقدار خوش بینانه(S_{opt})، محتمل(S_m) و بد بینانه(S_{pess}) برآورد ها به صورت زیر محاسبه می شود:

$$S = (S_{opt} + 4S_m + S_{pess}) / 6$$

روش های تجزیهٔ برآورد پروژهٔ نرم افزار

- **برآورد مبتنی بر فرایند**

متداول ترین روش برآورد پروژه، استفاده از مبنای فرایندهاست؛ یعنی هر فرایند به مجموعه ای نسبتاً کوچک از کارها تقسیم می شود و تلاش لازم برای انجام هر کار برآورد می شود.