بسم تعالی

شاخص و معیارهای پیچیدگی نرم افزار

(مهندسی نرم افزار و دلایل بروز بحران نرم افزار و پیچیدگی ها و روشهای کنترل پیچیدگی)

Measurement Software Complexity

استاد راهنما : مهندس کـــــامفر

گرد آورنده : فــرهــاد کــریــمــی

دانشکده فنی امام صادق (ع) بابل

دی ماه سال 1391

فهرست

مقدمه

امروزه با رشد روز افزون فناوری ارتباطات،طبیعتا نیاز به گسترش نرم افزار های کاربردی در همه زمینه ها بیش از پیش احساس می شود.از طرفی همانطور که میدانید یک فعالیت مفید در راستای گسترش یک پروژه تا زمانی توجیه شده بوده و به سود دهی برای طراحان آن خواهد رسید که از هر نظر مورد برسی قرار گیرد و به قولی بهترین و بهینه ترین روشها برای به طراحی و بهره برداری از آن مورد استفاده قرار گیرد.

در همین راستا برسی "**شاخص ها معیارهای پیچیدگی نرم افزار**" این امکان را فراهم می سازد تا با نگاهی اجمالی و مفصل به مقوله گسترش نرم افزارها در هرچه بهتر و بهینه شدن پروژه از مواردی که موجب "**پیچیدگی بیهوده نرم افزار**" و یا بروز "**بحران های نرم افزار**" می باشد جلوگیری به عمل آید.در طی مطالعاتی که برای گرد آوری این مقاله انجام داده ام به اصول کلیدی پی بردم که بنده را بیش از پیش با بکار گیری راهبردهای مدون در طراحی نرم افزار ، همینطور ازریابی و اندازه گیر متریک های گسترش نرم افزار بیش از پیش آشنا کرد.

تلاش کرده ام تا این نوشته به صورت اصولی و پیوسته از مراحل اولیه به برسی **اصول اندازه گیری نرم افزار** تا تشریح **بحرانهای نرم افزاری** و پیچیدگی و معیار ها و شاخص ها آن بپردازم.این مقاله از تاریخ 4/11/91 تا 14/11/91 با استفاده از منابع متعدد و همینطور تحلیل های ناشی از مطالعات صورت گرفته گرد آوری شده است که امید است مورد قبول خوانندگان آن قرار گیرد و بتواند قدمی در راستای رسیدن به دیدگاهی اصولی و مدون و بهینه در طراحی نرم افزار در کشور عزیزمان باشد.

فرهاد کریمی – دانشجوی ترم آخر کاردانی نرم افزار

4/11/91

برخی از مفاهیم بنیادی

اصل اندازه گیری

ضرب المثلی میگوید :If you measure it , you can manage it یعنی اگر بتوانید اندازه گیریش کنید ، میتوانید آن را مدیریت کنید.

این اصل یکی از کلیدی ترین مباحث نه در این عرصه بلکه در سایر مسائل می باشد.یکی از مواردی که مدیریت را چه در عرصه نرم افزار و چه در موارد دیگر می تواند آسانتر و قابل کنترل نماید این است که ما از ابعاد آن باخبر باشیم.یعنی به یک جمع بندی کلی از پروژه خود رسیده و مقادیر کمی از آن در دست داشته باشیم.این مقادیر می تواند ما را یاری کند تا به جمع بندی نسبتا" دقیق درباره ماهیت کلی پروژه و روشهایی که میتواند در هرچه موفقت تر بودن آن به ما کمک کند تا به جمع بندی نسبتا دقیق درباره ماهیت کلی پروژه و روشهایی که می تواند در هرچه موفقتر بودن آن ما را یاری کند.

تعریف مؤلفه های اندازه گیری و پیچیدگی نرم افزار

اندازه(measure) : یک مقدار عددی برای نشان دادن مقادیر ، نظیر : وزن ، مقدار و...

اندازه گیر (measurement) : فرایند تشخیص یک اندازه را گویند.

معیار(metric) :یک اندازه کمی برای نشان دادن اندازه دار بودن یک مؤلفه می باشد.

شاخص(Indicator) : تلفیقی از اندازه ها و میعارها که فرد برای رسیدن به یک دیدگاه اصولی درباره یک محصول ، پروژه و یا یک سازمان کمک می کند.

میانگین وزنی : به مجموعه معیارهایی با ضرایب خاص گویند.

دلایل اندازه گیری از دیدگاه بزرگان امر مهندسی نرم فزار

"راجر پرس من" دلایل اندازه گیری و فرایند مربوط به آن (measurement) را به چهار دسته تقسیم می کند:

- برای مشخص سازی در پروژه
- برای ارزیابی پروژه
- برای پیش بینی موارد مختلف مربوط به پروژه
- برای بهبود و ارتقاء پروژه

تعریف معیارهای نرم افزاری (software metrics)

به مجموعه معیارهایی که ما را به نتیجه کلی از اطلاعات(شاخص ها) درباره نرم افزار و مراحل بهینه برای پیش برد اهداف کلی پروژه می رساند گویند.

کاربرد این معیار یا متریک ها:

یکی از کلیدیترین مواردی که این متریک ها به ما میدهد این است که برای ما امکان برآورد کردن نیازهای پروژه و همینطور هزینه های آن را فراهم می کند.از جمله نرم افزارهایی که مجموعه ای از این امکانات را به همراه محیطی برای محاسبه و جمع بندی از پروژه برای ما فراهم می کند ، میتوان به Cocomo اشاره کرد.در اینگونه نرم افزارهای که وظایف را به صورت اتوماتیک انجام میدهند باید به اصل دقیق بودن اطلاعات دقت داشت تا به برآوردی واقعیتر دست یابیم.

در همین راستا افرادی چون **گودمن** نظریاتی را درباره اصول اندازه گیری مطرح نموده اند ، گودمن 6 اصل برای اندازه گیری دقیق مطرح کرده است که شامل موارد زیر می باشد:

1- مصالح و مصلحت گرایی : تعیین محدوده دقیق اندازه گیری به گونه ای که نه کم و نه بیشتر باشد.
2- اندازه گیری نیرو انسانی : این امر هرگز نباید صورت بگیرد ، چون موجب کاهش بهره وری میشود.
3- مدلسازی
4- جمع آوری اطلاعات و تحلیل
5- دیدگاه کلی و همه جانبه
6- شکّ فرهنگ

تشریح بحران نرم افزار

امروزه با گسترش تولید محصولات سخت افزاری شاهد جا ماندن عرصه نرم افزار و بروز کاستی هایی در تولید نرم افزارهایی متناسب با سخت افزار ها شده است ،این کاستی ها در روشهای تولید نرم افزار موجب بروز پیچیدگی های بیهوده ای می شود که یکی از بزرگترین نواقص نرم افزارهای امروزی می باشد.

*****باید توجه داشت که به صورت کلی نمی توان پیچیدگی را به صورت کامل از بین برد اما باید کوشید و حدالمکان میزان آن را کاست و کنترلش کرد.**

بروز بحران در نرم افزارها امروزه به شکل های مختلفی در صنعت نرم افزار قابل مشاهده است.برخی از عوامل آن عبارتند از :

• **عدم بهره گیری نرم افزار از قدرت سخت افزار** : همانطور که اشاره شد امروزه صنعت سخت افزار روز به روز پیشرفت چشمیگری دارد به گونه ای که به یکی از نخستین صادرات جهانی مبدل شده است. یکی از عواملی که موجب ضعف نرم افزارها می باشد عدم تسلط و اشراف برنامه نویس به ابعاد سخت افزار می باشد. به عنوان مثال در یک سیستم ارسال اس ام اس عدم آشنایی با برخی کامند ها و یا دستورات توسط برنامه نویس موجب دوباره کاری ، پیچیدگی و از دسترس خارج شدن برنامه می شود.

• **ناتوانی روشهای تولید نرم افزار در پاسخگویی افزایش تقاضا:**

هزینه بالا در تولید نرم افزار:عدم رعایت اصول مهندسی سازی نرم افزار موجب افزایش هزینه های تولید می شود به گونه ای که از هر چه بهینه بودن آن کاسته و موجب صرف بیهوده منابع و زمان می شود.

• **عدم تحویل به موقع پروژه های نرم افزاری:**

پیچیدگی نرم افزار موجب شکل گیری رشته ای از کدها در اصطلاح "اسپاگتی کدها" می شود که انعطاف و ابتکار عمل را از تولید کننده گرفته و موجب به تعویق افتادن در ارائه آن می شود.

• **کیفیت پایین و نامطمئن:**

اصول مهندسی نرم افزار و به کارگیری(Pattern) یا در اصطلاح الگوهای مناسب در طراحی نرم افزار موجب بالا رفتن کیفیت و جلوگیری از بحران نرم افزاری می شود.

• **سختی نگه داری به علت کیفیت پایین در طراحی:**

یکی از عوامل دیگر که موجب بحران نرم افزار می شود سختی نگهدار می باشد. باید توجه داشته باشیم زمانی که ما نرم افزار خود را به صورت غیر اصولی گسترش می دهیم قطعا کیفیت آن پایین می آید ، پشتیبانی از آن غیر ممکن و یا ضعیف می شود.

مهندسی نرم‌افزار و اهمیت آن در کاستن از بحران نرم‌افزار

مهندسی نرم‌افزار را می‌توان مکانیز و الگویی برای ارائه هر چه بهتر نرم‌افزار دانست. درواقع می‌توان اینگونه بیان کرد که مهندسی نرم افرار اصولی را به ما ارائه میدهد تا بتوانیم بر طبق این اصول نرم‌افزار خود را هرچه بهتر و با درنظر گرفتن موارد مورد نیاز چه در حال و چه در آینده طراحی نماییم.

دانشنامه جهانی **ویکی پدیا** در باره مهندسی نرم‌افزار اینچنین نوشته است :

مهندسی نرم‌افزار طراحی، برنامه نویسی، توسعه، مستندسازی و نگهداری نرم‌افزار با بکارگرفتن روشهای فنی و عملی از علوم کامپیوتر، مدیریت پروژه، مهندسی، محدوده کاربرد، طراحی رابط، مدیریت تجهیزات دیجیتال و سایر زمینه‌ها است.کاربردهای مهندسی نرم‌افزار دارای ارزش‌های اجتماعی و اقتصادی هستند، زیرا بهره‌وری مردم را بالا برده، چند و چون زندگی آنان را بهتر می‌کنند. مردم با بهره‌گیری از نرم‌افزار، توانایی انجام کارهایی را دارند که قبل از آن برایشان شدنی نبود. نمونه‌های از این دست نرم‌افزارها عبارت‌اند از: سامانه‌های توکار، نرم‌افزار اداری، بازی‌های رایانه‌ای، و اینترنت.فناوری‌ها و خدمات مهندسی نرم‌افزار به کاربران برای بهبود بهره‌وری و کیفیت یاری میرساند.

ویکی پدیا درباره ضرورت مهندسی نرم افزار اینگونه نوشته است :

نرم‌افزار عموماً از محصولات و موقعیتهایی شناخته می‌شود که قابلیت اطمینان زیادی از آن انتظار می‌رود، حتی در شرایط طاقت فرسا، مانند نظارت و کنترل نیروگاه‌های انرژی هسته‌ای، یا هدایت یک هواپیمای مسافربری در هوا، چنین برنامه‌هایی شامل هزاران خط کد هستند، که از نظر پیچیدگی با پیچیده‌ترین ماشینهای مدرن قابل مقایسه‌اند. به‌عنوان مثال یک هواپیمای مسافربری چند میلیون قطعه فیزیکی دارد (و یک شاتل فضایی خدود ده میلیون بخش دارد)، در حالی که نرم‌افزار هدایت چنین هواپیمایی می‌تواند تا ۴ میلیون خط کد داشته باشد.

ماهیت مهندسی نرم افزار را اینگونه تشریح کرده اند:

دیوید پارناس گفته‌است که مهندسی نرم‌افزار یک شکل از مهندسی است. استیو مک‌کانل گفته‌است که هنوز اینطور نیست، ولی مهندسی نرم‌افزار باید یک شکل از مهندسی بشود.

دیوان فعالیتهای آماری آمریکا مهندسان نرم‌افزار را به عنوان زیرگروهی از «متخصصین کامپیوتر»، با فرصت‌های شغلی‌ای مانند «دانشمند کامپیوتر»، «برنامه نویس» و «مدیر شبکه» دسته بندی کرده‌است. BLS تمام مهندسین دیگر این شاخه علمی، که شامل مهندسین سخت افزار کامپیوتر نیز هست، را به‌عنوان «مهندسین» دسته بندی می‌کند.

نقش مهندسی نرم‌افزار و اهمیت آن

همانطور که مطرح شد نقش بسزای مهندسی نرم‌افزار زمانی روشن می‌شود که پروژه پیش رو گسترده باشد و قاعدتا" از پیچیدکی ذاتی نیز برخوردار باشد.حال اینجاست که نقش گسترده مهندسی نرم‌افزار و اصول آن بیش از پیش مشخص می شود.هموار در یک پروژه پیچیدگی وجود دارد از این رو نقش طرحها و پترنهای مهندسی نرم‌افزار اهمیت خود را نشان می دهد.

از مهمترین ویژگی‌هایی که اهمیت مهندسی نرم‌افزار را بیشتر میسازد می‌توان به موارد زیر اشاره داشت:

- افزایش کیفیت،قابلیت نگهداری،قابلیت اطمینان
- رضایت کاربران و سهامداران
- کاهش هزینه‌ها
- تحویل به‌موقع
- استفاده مجدد
- استفاده از مؤلفه های استاندارد

در شکل فوق تکنولوژی مهندسی نرم افزار و چهارچوب کلی آن به شیوه ای گویا نمایش داده شده است،از مطالب ذکر شده در سرفصلهای قبلی می توان به این نتیجه گیری رسید که عدم وجود راهکارها و روشهایی مناسب جهت تولید نرم افزار به بحران نرم افزار دامن میزند.

حال به نظر شما چاره چیست؟

طبق مطالعاتی که داشتم به این نتیجه گیری کلی رسیدم که در ادامه بیان خواهم نمود؛

ایجاد یک سیستم مبتنی بر مدل به صورتی که هر بحش از این سیستم قسمتی از عملکرد نرم افزار را توصیف و عملی نماید و به صورت بهینه با سایر مدلها و زیر مدلها در ارتباط مناسب باشد.اعمال مدلسازی در نرم افزار ما را به یک متدلوژی کلی و به عبارتی دید کلی از نرم افزار می رساند که چهارچوب کلی را در سراسر پروژه مشخص نموده و به ما کمک می کند تا از مسیر اصلی تولید نرم افزار و اهداف خود خارج نشده بلکه آن را به صورتی اصولی و مهندسی پیش ببرد و به ما کمک نماید تا در مسیر حساب شده ای قدم بگذاریم.

ویژگی های یک متدلوژی خوب را می توان اینگونه برسی کرد :

1- ارائه تعریف مناسب و مفهومی و کامل از مفاهیم به کار رفته در متدلوژی
2- ارائه مدلی کلی که شامل زیر مدل ها نیز باشد
3- داشتن مدل زیربنایی
4- ارائه شیوه استاندار برای علامتگذاری
5- پیاده سازی متدلوژی مناسب و اعمال تکنیک هایی برای کنترل پیچیدگی نرم افزار
6- ارزیابی نتایج حاحل از به کارگیری متدلوژی و کنترل مستمر آن برای گرفتن بازخورد مناسب و آنالیز متدلوژی اعمال شده
7- به کارگیری ابزارهای امتوماتیک برای کمک و اجرای مدلهای مبتنی بر متدلوژی

پیچیدگی نرم افزار و برسی انواع

زمانی که درمورد پیچیدگی نرم افزار بحث می کنیم باید توجه داشته باشیم که پیچیدگی مفهوم ثابتی نیست و باید در شرایط مختلف مورد برسی قرار گیرد.از این رو می گوییم پیچیدیگی نرم افزار اصطلاح غیر استانداری است زیرا از محدوده مشخصی برخوردار نیست، امکان دارد نرم افزار بزرگی با خطوط زیادی از کد و پیمانه های مختلف و مرتبط را پیچیده و همینطور نرم افزار کوچکی با الگوریتمی سخت را نیز پیچیده ارزیابی نماییم .برای همین است که نمی توان پیچیدگی را از یک بعد مورد برسی قرار داد.موسسه IEEE تعریفی از پیچیدگی ارائه داده است که به شرح زیر می باشد:

"میزان سختی در درک یک سیستم و یا مؤلفه که طراحی و پیاده سازی شده است."

از این رو باید عنوان کرد که پیچیدگی نرم افزار به فاکتورهایی از اندازه گیری تاثیر گذار در هزینه ، نگهداری و توسعه اطلاق می شود.البته افراردی چون "زئوس" این نظریه را زیر سوال برده و پیچیدگی را از دید روانشناسی نرم افزار مورد برسی قرار داده و دسته بندی نموده اند.

Zeus پیچیدگی روانشناسی نرم افزار را به سه بخش زیر تقسیم نموده:

1- پیچیدگی در مساله
2- پیچیدگی در طراحی سیستم
3- پیچیدگی رویه ای

مثلا" درمورد پیچیدگی در مساله باید عنوان داشت که این امر به پیچیدگی کلی الگوریتم مساله بر میگرد.یعنی ممکن است یک سیستم پیچیدگی ذاتی کم یا بیشتری داشته باشد که در میزان پیچیدگی کلی آن تاثیرگذار خواهد بود.همینطور می توان به پیچیدگی موجود در طراحی سیستم اشاره داشت که به دو قسم:**ساختاری** و **داده ای** تقسیم می شود.

پیچیدگی ساختاری به مفهوم کلی اتصال اشاره دارد و اتصال بین پیمانه های درونی در زبان برنامه نویسی را مورد برسی قرار می دهد.به عنوان مثال ؛ در یک زبام برنامه نویسی (فرض کنید C++) ارتباطاتی بین پیمانه های درونی زبان + کلاسها و کتابخانه ها وجود دارد که به خودی خود پیچیدگی هایی را به دنبال خواهد داشت.

پیچیدگی داده ای به مفهوم دیگری با عنوان "چسبندگی" اشاره دارد و چسبندگی های درون ساختاری را مورد برسی قرار می دهد.

پیچیدگی ساختاری و داده ای ارتباط مبتنی بر FAN-IN و FAN-OUT در پیمانه ها و متغییرهای خروجی/ورودی را مورد برسی قرار میدهند.

آنگونه که اشاره شد پیچیدگی های ساختاری و داده ای به روابط بین پیمانه های درون سیستمی نرم افزار اشاره دارند حال آنکه در زبانهای پیش گام و با به کارگیری OOP یا برنامه نویسی شیء گرا از میزان این پیچیدگی ها تا حد زیادی کاسته شده است.(در ادامه به ویژگی های شیء گرایی در کاستن پیچیدگی اشاره خواهیم کرد)

*****باید اشاره داشت که اغلب پیچیدگی های درونی مانند آنچه گفته شد درصد پایین و ضریب اندکی از پیچیدگی نرم افزار را شامل می شوند و در شکل گیری بحران نرم افزار آنچنان تاثیر گذار نیستند.**

پیچیدگی رویه ای:این پیچیدگی به ساختار کلی برنانه ، خطوط کد ، حلقه ها و شرط ها و طریقه به کار گیری آنها اشاره می کند.

به صورت کلی پیچیدگی بخش غیرقابل اجتناب در تولید نرم افزار است یعنی به هر نحو میزانی ازز پیچیدگی در نرم افزار وجود دارد اما مشکل بحران نرم افزار زمانی رخ می دهد که این پیچیدگی ها از یک حد بیشتر شوند.وظیفه توسعه دهندگان نرم افزار این است که این پیچیدگی ها را تا حد ممکن کاهش دهد و این یعنی ارائه راهکار و رویه ای مناسب تا یک الگوریتم هر چه ساده تر و مفید تر پیاده سازی شود .

نمونه کد

در ادامه به مثال زیر توجه کنید که در آن یک روش رویه ای در کنترل پیچیدگی اعمال شده است.برنامه زیر دو عدد را مقایسه کرده و عدد بزرگتر را درون متغییری با نام Max میریزد .

```
int _tmain(int argc, _TCHAR* argv[])
{
    int a,b,Max;
    if(a>b)
        Max=a;
    else
        Max=b;

}
```

الگوی اول : این الگو دو عدد را با استفاده از if statement یا همان دستور شرطی if بررسی می کند و به پاسخ میرسد،حالا به الگوی زیر توجه کنید:

```
int _tmain(int argc, _TCHAR* argv[])
{
    int a,b,Max;
    (a>b)?Max=a:Max=b;
}
```

در مثال فوق همان برنامه اول منتهی با رویه ای مناسبتر برای جلوگیری از پیچیدگی نوشته شده است و از دستور شرطی یک خطی استفاده شده است.

***** این مثال الگویی ساده برای پی بردن به این نکته است که در بحران نرم افزار ، کنترل پیچیدگی امکان پذیر و راهبردی است.**

*****فراموش نکنیدم که هرچه مدل ما حاوی اطلاعات بیشتری باشد، طبیعتا مدل ما پیچیده تر خواهد بود.**

ماهیت پیچیدگی نرم افزار

همانطور که در بحث های ابتدایی و الزام آور بودن اصول اندازه گیری نرم افزار (measurement) مطرح شد، شناخت ماهیت نرم افزار باعث می شود از عملیات کورکورانه و به قولی آزمون و خطا جلوگیری شود.عدم رعایت این اصل(شناخت) یکی از مهمترین عوامل پیچیدگی می باشد.زمانی که برنامه ساز در مورد برنامه پیش رو شناخت کافی نداشته باشد و بر ماهیت کلی آن اشراف نداشته باشد قطعا نرم افزار تولیدی او نیز کارایی بالایی نخواهد داشت و امکان بروز اختلالات غیرمرقبه در حین کار با نرم افزار بالا می رود به گونه ای که عدم رضایت مشتریان را به دنبال خواهد داشت.

اجزای کلیدی در پیچیدگی

پیچیدگی برچسبی است که ما به وجود متغییرهای وابسته ی زیاد موجود در یک سیستم می دهیم،هرچه متغییرها بیشتر باشد وابستگی بین آنها بیشتر می شود.اتصالات بین متغییرها ویژگی های زیادی برای آن واحد می طلبد که منجر به بروز پیچیدگی بیشتر خواهد شد.

پیچیدگی چند جزء کلیدی دارد که به شرح زیر می باشد:

1- مقیاس :مقدار اجزاء سیستم نرم افزاری
2- تنوع:گستره اجزاء متنوع تشکیل دهنده سیستم
3- اتصال : ارتباط بین مؤلفه ها را فراهم می کند

× **مقیاس** : مقیاس به تنهایی مساله نیست و در صورت زیاد بودن مقیاس با استفاده از داده های آماری می توان آن را تعیین و کنترل نمود

× **تنوع** :این جزء انواع اجزایی را که باید تحلیل شوند افزایش می دهد.هرچه تنوع بیشتر شود تلاش بیشتری برای درک سیستم لازم داریم.

× **اتصال** : با افزایش مقیاس قاعدتا" تعداد اتصالات و تعامل بین اجزاء بیشتر خواهد شد و این امر نیز به نحوی موجب پیچیدگی می شود.

بررسی معیارهای پیچیدگی

پیچیدگی به صورت کم و بیش وجود دارد و وجود خواهد داشت به صورت کلی پیچیدگی دارای رابطه مستقیم با گستردگی پروژه می باشد.پر واضح است که هر مقدار که تعداد خطوط کد پروژه بیشتر باشد قاعدتا" از مدلها ، کلاسهای زیادی برخوردار بوده و پیچیدگی ذاتی آن بیشتر

است.حالا کافیست مطالب قبلی را به یاد بیاوریم "پیچیدگی به صورت کامل قابل از بین بردن نیست ، از این رو باید آن را کنترل نمود."

پیچیدگی نرم افزار دارای ویژگی های انحصاری و بارزی به شرح زیر می باشد:

- با پیچیدگی سخت افزاری در وسایل فیزیکی و مکانیکی تفاوت دارد و از یک مفهوم منطقی در نرم افزار نشآت می گیرد.
- یک خاصیت ذاتی در نرم افزارهای بزرگ می باشد.
- پیچیدگی حوزه مساله
  1-نیازمندی های گوناگون و متضاد
  2-ارتباط بین کاربر و نرم افزار
  3-تغییر نیازها
- پیچیدگی در فرایند تولید
- عدم انعطاف پذیری سیستم و استاندارد نبودن اجزاء آن
- مشکل در توصیف رفتارهای سیستم های گسسته

پیچیدگی عارضی

پیچیدگی ذاتی عموما مشخصه هر مساله نرم افزاری و الگوریتم های مربوط به آن می باشد.یعنی اگر مساله ای ساده را نیز در نظر بگیریم شامل راه حلی می باشد ، همین راه حل هر چند ساده یک پیچیدگی ذاتی برای نرم افزار محسوب می شود.در مقابل پیچیدگی ذاتی ، پیچیدگی عارضی قرار گرفته است که از انتخاب صورت گرفته در ساخت سیستم ناشی می شود.که به دلایل بروز آن به صورت کامل اشاره شد. به صورت اصولی باید گفت عملی ترین و بهترین راه برای کنترل پیچیدگی عارضی این است که ابتدا مساله را درست تعریف کنیم سپس به دنبال راه حل(چگونگی) انجام مساله برویم.

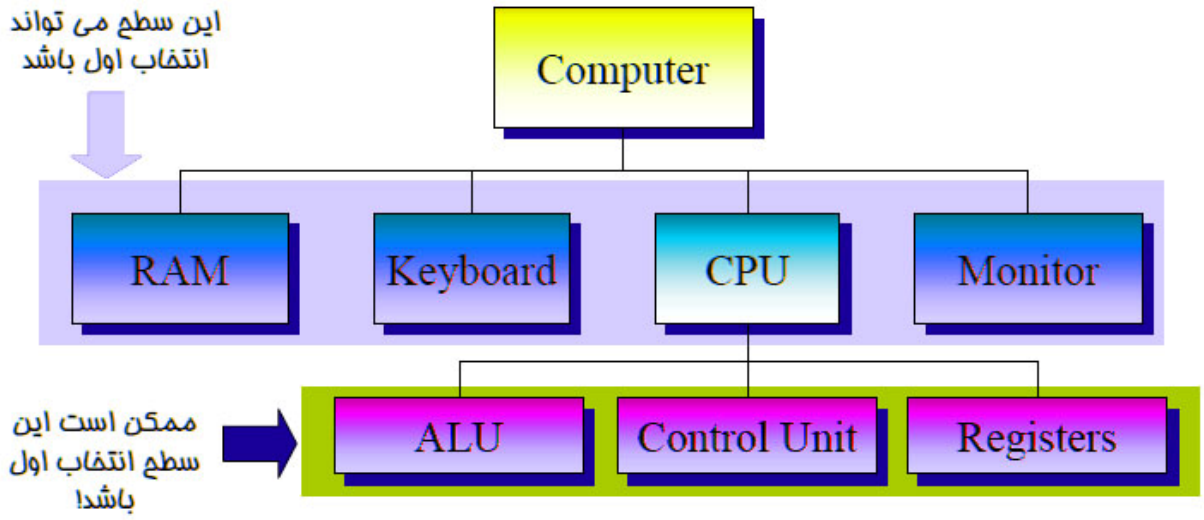
تمام مواردی که در سرفصلهای قبلی پیچیدگی مطرح شد به نوع عارضی اشاره دارند و با به کارگیری راهکارهای ذکر شده تا حدود زیادی می توان در مقابل بروز اینگونه پیچیدگی ها مصون بود.البته برخی راهکارها در مقابله با بروز پیچیدگی عارضی وجود دارد که در ادامه به آن اشاره خواهیم داشت.

*****در سرفصلهای بعدی به بررسی ابعاد پیچیدگی نرم افزار در قالب نمودارهای رابطه ای اشاره خواهم کرد تا بیشتر با نحوه بروز اینگونه موارد آشنا شوید.**

پیچیدگی سلسله مراتبی (Hierarchy)

در اغلب سیستم ها ، پیچیدگی به صورت سلسله مراتبی ظاهر می شود.در اینگونه سیستم ها ارتباط بین اجزای درونی سیستم قویتر از ارتباط بین خود زیر سیستم هاست و این یکی از ویژگی های پیچیدگی در ساختار سلسله مراتبی می باشد.

شکل زیر نمایانگر یک سیستم پیچیده سلسله مراتبی می باشد.

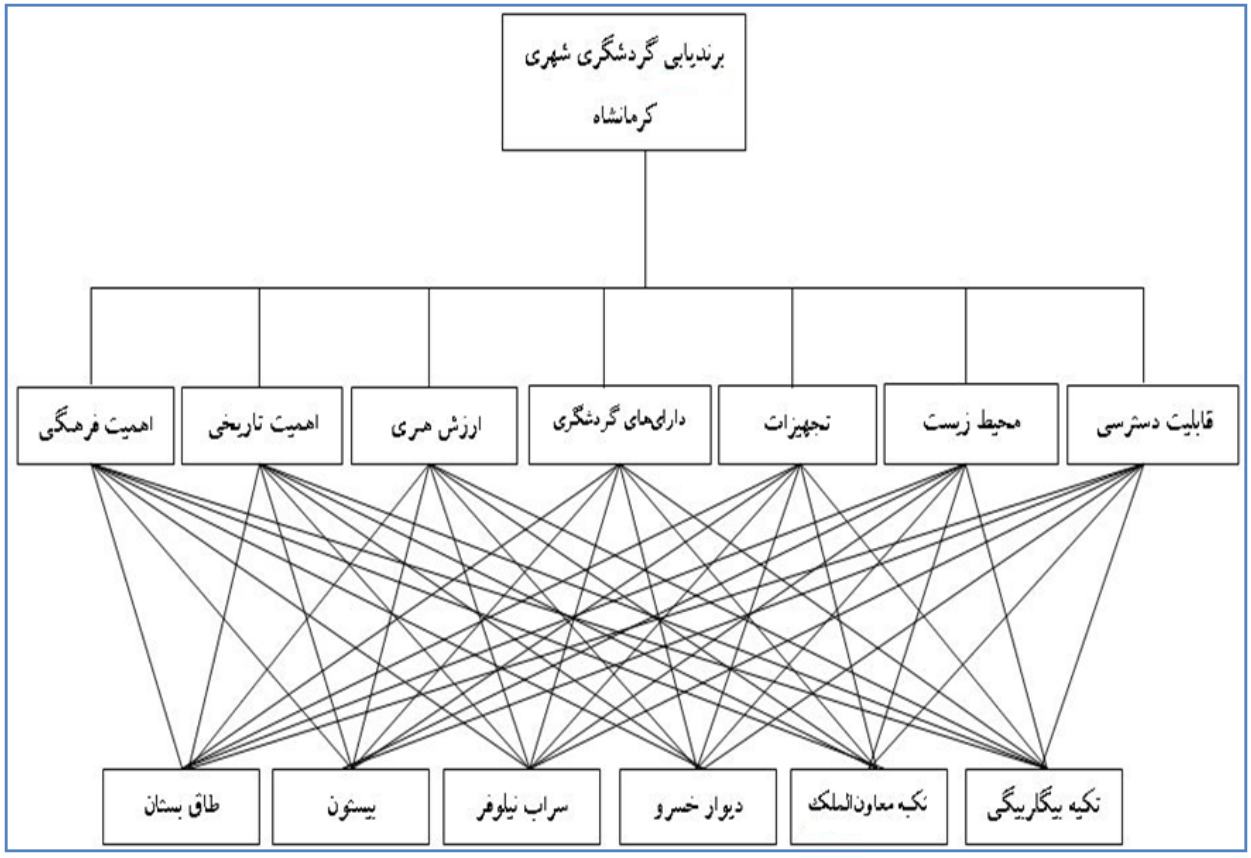


سیستم های سلسله مراتبی معمولا از تعداد کمی از زیر سیستم های مشخص و متفاوت تشکیل می شوند که این زیر سیستم ها به صورتهای گوناگون و ترتیب های مختلف ظاهر می شوند.

***باید توجه داشت که سیستم های پیچیده ای که حتی با وجود گستردگی به خوبی عمل می کنند دارای زیر سیستم های ساده و کار آمد و هدفمند می باشند که در تعامل با سایر زیر سیستم ها بوده و از پیچیدگی کنترل شده ای برخوردار می باشند.در غیر این صورت و اگر سیستم دارای زیر سیستم های اصولی و مهندسی شده نباشد ، قطعا ماهیت کلی سیستم به خطر افتاد و نرم افزار مورد نظر کار نخواهد کرد.

مثالی کاربردی از درخت سلسله مراتبی

مثال نمودار سلسله مراتبی زیر مربوط به بازاریابي گردشگري شهري در استان کرمانشاه بر اساس برندیابي با "مدل فرآیند تحلیل سلسله مراتبي" می باشد که تحلیل کوتاهی از آن را در ادامه ارائه خواهم داد:



نمودار سلسله مراتبی با استفاده از سنجه ها و وزن دادن به آنها ما را در اندازه گیری در بین گذینه های مختلف یاری میکند و در نهایت مجموعه ای مرتب شده با استفاده از روش سنجشی یاد شده به ما ارائه می دهد.

اولین قدم تعیین سلسله مراتب می باشد که با درج هدف نهایی در بالاترین سطح به پایین می آید،تا به جزیی ترین صفات برسیم.درنهایت باید هر سطح پایینی به سطخ بالاتر قبلی متصل شود تا نمودار پدید بیاید.بعد از رسم نمودار اقدام به مقایسه دو به دو معیارها می نماییم و با توجه به ارجحیت دو مقدار عملیات ارزشگذاری صورت می پذیرد.در ادامه باید مقادیر و معیارهای بدست آمده از نظر وزنی مقایسه شده و ضرایب و اهمیت آنها مورد برسی قرار گیرد.

نمودار نهایی پس از اعمال اصول اصلی نمودار سلسله مراتبی به شکل زیر در خواهد آمد:

شیء گرایی و نقش آن در کنترل پیچیدگی نرم افزار

یکی دیگر از مواردی که موجب کاسته شدن از پیچیدگی می شود به کار گیری روشهای شیء گرایی(Object Oriented) می باشد.

شیء گرایی دارای 4 اصل کلی می باشد که عبارتند از :

1– تجرید
2– محصور سازی
3– واحد بندی
4– سلسله مراتب

تجرید(Abstraction)

به صورت کلی به فرآیند تشخیص خصیصه های مهم یک پدیده گویند،در تجرید باید از پرداختن به ویژگی های موقت و غیر مهم جلوگیری کرد و خصیصه ها و ویژگی های مهم یک پدیده را مدنظر قرار دهیم.مثالا یک دانشجو را در نظر بگیرید(پدیده) ویژگی های مهم برای صدور کارنامه برای یک دانشجو شامل مواردی مانند : تعداد واحد ها ، نمرات ، شماره دانشجویی ، شماره شناسنامه می باشد که پس از اعمال اصل تجرید بدست خواهد آمد.

در صورتی که ما اصل تجرید را رعایت نکنیم امکان استفاده از ویژگی های موقت و غیرضروری مثل :شماره تلفن همراه ، وضعیت جسمانی و... را فراهم کرده ایم که این امر به خودی خود موجب بروز پیچیدگی بیشتر خواهد شد.

به زبان ساده اصل تجرید مؤلفه های غیرضروری که از اهمیت کمتری برخوردار هستند را حذف و به ویژگی های مؤثر در طراحی صحیح الگوریتم و رسیدن به روشی مناسب برای حل مساله می پردازد.

محصورسازی (Encapsulation)

محصورسازی موجب می شود تا پدیده از اثرات مخرب میحط خارجی به دور بماند و همینطور طریقه دسترسی سایر اشیاء به آن پدیده را نیز کنترل نماییم.

محصورسازی با کنترل راههای دسترسی به یک پدیده (شیء) باعث جلوگیری از تاثیرات منفی و احتمالی بر روی پدیده می شود و همینطور از گسترش خطاهای رخ داده در درون پدیده به سایر اشیاء جلوگیری می کند.

از جمله سایر نقش های محصورسازی در کنترل پیچیدگی این است که به دلیل ثبات در واسط ارتباطی(Interface) می توان تغییرات زیادی بر روی آن اعمال کرد و همینطور امکان استفاده مجدد را به پدیده می دهد.

در کپسوله سازی سا محصور سازی از دو اصطلاح **ماژول باز** و **ماژول بسته** استفاده می شود.درواقع به ماژولی که برای اعمال تغییرات آماده باشد باز و به ماژولی که امکان تغییر در آن وجود نداشته باشد ، ماژول بسته می گوییم.در عملیات Encapsulation هر دو ماژول باز و بسته کاربردی و لازم می باشند.

واحد بندی(Modularity)

یک سیستم را در نظر بگیرید که واحد های کوچکتر آن هر یک به صورت واحدهایی جداگانه دسته بندی شده باشند.به این دسته ها ماژول گفته می شود.سیستم هایی با دسته بندی(Module) و با حداقل ارتباط را سیستم های ماژولار یا واحد بندی شده گویند.

به عنوان مثال از فایلها در زبان زبان C++ ، یونیت ها در دلفی و پاسکال ،کامپوننتها در .NET و جاوا را می توان به عنوان ماژول نام برد.

واحد بندی دارای دو خصیصه اصلی می باشد 1 – انسجام :درجه ارتباط عملکردهای داخلی ماژول را گویند. 2- وابستگی : درجه ارتباط ماژولها با یکدیگر را مشخص می کند.3 – استقلال یا Independent 4 - Well-defined interfaces یا واسط های خوش تعریف

درواقع واحد بندی با شکستن مساله به زیر مساله ها از بروز پیچیدگی جلوگیری می کنند و موجب نظام مند شدن مساله برای کنترل پیچیدگی می شود.البته مواردی که باید در واحد بندی

مورد توجه قرار بگیرند این است که 1 – ماژول بندی نباید با تعداد خیلی زیاد و یا خیلی کم باشد
2-معیاری مشخص برای شکستن مساله باید لحاظ شود تا ماژول بندی مؤثر واقع شود.

سلسله مراتب (Hierarchy)

به مرتب ساختن Abstract ها در سطوح مختلف سلسله مراتب گفته می شود. باید توجه داشت که نمودار سلسله مراتبی دارای دو ویژگی زیر می باشد:

ساختار کلاس : سلسله مراتب IS-A

ساختار شیء :سلسله مراتب PART-OF

*****در کل باید گفت فرمول شکل اصلی یک سیستم پیچیده به این صورت می باشد :**

شکل اصلی سیستم پیچیده =خواص پنجگانه + سلسله مراتب IS-A + سلسله مراتب PART-OF

سلسله مراتب IS-A :این سلسله مراتب برای مشخص نمودن وراثت ها مورد استفاده قرار میگیرد از لحاظ گفتاری می توان مثال زیر را مطرح کرد : CAT **IS A** ANIMAL ، این مشخص کننده وراثتی می باشد که گربه از خانواده حیوانات میگیرد.

سلسله مراتب PART-OF : بیانگر قسمتی از چیزی می باشد ،از لحاظ گفتاری می توان به این صورت نوشت : HAND IS **PART OF** BODY ، این مثال مشخص کنند شیء یا قطعه ای از یک کلیت می باشد.

پیاده سازی در قالب سلسله مراتب ما را در دریافت بهتر مساله و درک بهتر آن کمک کرده لذا موجب کاستن پیچیدگی می شود.

در پایان می توان ویژگی های استفاده از شیء گرایی را در کنترل پیچیدگی چنین ارزیابی کرد:

1- منظم کردن فرآیند تولید نرم افزار
2- بالا رفتن درک مساله در سیستم های نرم افزاری
3- ارائه مدلی قدرتمندتر برای کنترل هرچه بیشتر پیچیدگی نرم افزاری
4- کاهش هزینه های تولید (زمانی – مالی)
5- استفاده مجدد و انعطاف پذیری بیشتر همینطور افزودن امکان پشتیبانی آسانتر

پایان

چکیده

در این مقاله تلاش شد در ابتدا به اصول اولیه و معانی تعاریف اصلی در مبحث اندازه گیری ، معیارهای نرم افزار ، بحران نرم افزار ، پیچیدگی و سایر مبانی بنیادین اشاره شده و در ادامه هر یک از مباحث به صورت مشروح مورد برسی قرار گیرد.تلاش بنده این بود که هر یک از مباحث مربوط به مهندسی نرم افزار که در ارتباط مستقیم با بروز بحران در نرم افزار بودند را به صورت مجزا مورد برسی قرار دهم و همینطور نقش هر یک از این مؤلفه ها در بروز پیچیدگی و بحران نرم افزار را به صورت نسبتا کامل شرح دهم.

در ادامه نیز به برسی راهکارهایی که می توان برای کنترل پیچیدگی نرم افزار به کار گرفت اشاره شد.این مقاله با استفاده از تحقیقات از مراجع معتبر و همینطور ترجمه متون زبان اصلی (انگلیسی) فراهم آمده است که امید است خواننده را به جمع بندی کلی از مفهوم اندازه گیر نرم افزار و معیارهای آن،بحران نرم افزار و دلایل آن و همینطور یکی از اصلی ترین دلایل بروز بحران نرم افزار یعنی پیچیدگی نرم افزار برساند.

و من الله التوفین – فرهاد کریمی 91/11/13

منابع

سایت رادمان ، مرجع مهندسی نرم افزار: http://weblog.radmanitd.com/archives/cat-4

اسلاید مهندسی نرم افزار پیشرفته فصلهای 1 و 2

وبلاگ: http://slgolavar.blogfa.com ،برسی افق مهندسی نرم افزار

وبلاگ : http://modida.persianblog.ir ، برسی صنعت نرم افزار ،پیچیدگی نرم افزار

با نگاه به جزوه مهندسی نرم افزار (پرسمن)

نگاه به کتاب :Controlling Software Complexity نوشته Ben Chelf & Andy Chou

با نگاهی به فصل سوم کتاب Software Estimation,Measurement, and Metrics

کتاب الکترونیکی بازاریابي گردشگري استان کرمانشاه بر اساس برندیابي با مدل فرآیند تحلیل سلسله مراتبي(AHP)