فهرست مطالب

فصل 1

فصل 2

SOFTWARE COMPLEXITY ENGINEERING

CHAPTER 1

پیچیدگی های نرم افزار

اهداف فصل

آشنائی با ماهیت پیچیدگی

آشنائی با تعریف پیچیدگی

مفاهیم پیچیدگی نرم افزار

فناوری ها و استانداردهای دسته بندی پیچیدگی

علل پیچیدگی

پروفسور کاوه پهلوان
متخصص و محقق در زمینه های مهندسی نرم افزار، تک پردازنده ها، ارتباطات دیجیتال، شبکه های اطلاعاتی بی سیم

1–1: خلاصه بخش

هزینه‌های رو به رشد نرم‌افزار سازمانهای نرم‌افزاری را بر آن داشته تا دنبال راه‌حلی برای کاهش این هزینه‌ها باشند. برای نیل به این مقصود محققان دنبال رابطه‌ای بین ویژگی‌های نرم‌افزار و مشکلات و دشواریهای کار توسعه نرم‌افزار بودند. دشواری کار زمان بیشتری برای انجام آن طلب می‌کند در این زمان منابع بیشتری بکار گرفته می‌شوند و منابع بیشتر یعنی تحمیل هزینه‌های بیشتر. یکی از دلایل پرداختن به موضوع پیچیدگی نرم‌افزار و اندازه‌گیری آن برای کنترل هزینه‌ها در چرخه حیات نرم‌افزار است زیرا پیچیدگی نرم‌افزار از عوامل اصلی در افزایش سریع هزینه‌های توسعه و نگهداری عنوان می‌شود . پیچیدگی نرم‌افزار فاکتوری است که در فعالیت برنامه‌نویسی کمتر شناخته شده و به راحتی قابل اندازه‌گیری یا توصیف نیست و غالباً در طول فرایند برنامه‌ریزی پروژه نادیده گرفته می‌شود . بنابراین به دنبال روشی کمی برای پیش‌بینی میزان دشواری نگهداری، تغییر و فهم نرم‌افزار هستیم. تا با اندازه‌گیری و کنترل آن هزینه‌ها را در چرخه حیات نرم‌افزار کاهش دهیم.

1–2: پیچیدگی نرم افزار

در چند دهه اخیر پیچیدگی نرم‌افزار دوره جدیدی در زمینه دانش کامپیوتری ایجاد کرده است. پیچیدگی نرم‌افزار را می‌توان بعنوان گرداننده اصلی هزینه، قابلیت اطمینان و عملکرد سیستم‌های نرم‌افزاری تعریف کرد. با هزینه‌های بالای نرم‌افزار و این حقیقت که حدود 75٪ چرخه حیات نرم‌افزار به تست و نگهداری نرم‌افزار سپری می‌شود، مسئله پیچیدگی نرم‌افزار اهمیت ویژه‌ای پیدا کرده است. پیچیدگی نرم‌افزار از عوامل اصلی افزایش سریع هزینه‌های توسعه و نگهداری می‌باشد و از طرف دیگر تاثیر بسزایی در کیفیت و بهره‌وری محصول دارد.

پیچیدگی نرم‌افزار فاکتوری است که براحتی قابل اندازه‌گیری یا توصیف نیست و غالباً در طول فرآیند برنامه‌ریزی پروژه نادیده گرفته می‌شود. بنابراین محققان بدنبال روشی کمّی برای پیش‌بینی میزان دشواری

فهم، تغییر و نگهداری نرم‌افزار هستند تا با اندازه‌گیری و کنترل آن، منجر به کاهش هزینه‌ها در چرخه حیات نرم‌افزار شوند.

اغلب تلاش‌های انجام شده در زمینه‌ی اندازه‌گیری پیچیدگی، به تاثیر پیچیدگی بر پروژه‌های نرم‌افزاری پرداخته‌اند که در راس آنها کیفیت و هزینه‌های محصول قرار می‌گیرد. در واقع توجه مدیران و مهندسان پروژه‌های نرم‌افزاری به پیچیدگی نرم‌افزار برای کنترل و پیش‌بینی کیفیت و بهره‌وری محصول است.

1-3: تعریف و ماهیت پیچیدگی نرم افزار

هنگامی که در مورد پیچیدگی نرم‌افزار صحبت می‌شود اولین سوالی که باید پاسخ داده شود این است که «پیچیدگی چیست؟». باید گفت که توافق عمومی روی چگونگی تعریف پیچیدگی وجود ندارد و باور عمومی بر این است که پیچیدگی نرم‌افزار را نمی‌توان فقط با استفاده از یک بعد تعریف کرد.

فرهنگ کامپیوتری استاندارد IEEE ، «پیچیدگی» را به این صورت تعریف می‌کند: «میزان سختی درک یا بازبینی یک سیستم یا مولفه که طراحی یا پیاده‌سازی شده است». عبارت «سختیِ درک» اشاره دارد که پیچیدگی لزوماً مقیاس مطلقی برای اندازه‌گیری نیست بلکه نسبی است. به این معنی که آنچه برای شخصی پیچیده به نظر می‌رسد می‌تواند برای دیگری براحتی قابل فهم باشد.

تعریف Zuse از پیچیدگی بدین شرح است، «پیچیدگی نرم‌افزار سختی تحلیل، نگهداری، تست، طراحی و تغییر نرم‌افزار است»، به عبارت دیگر پیچیدگی نرم‌افزار موضوعی است که در کل فرآیند توسعه نرم‌افزار و در هر مرحله از چرخه حیات محصول وجود دارد. از این‌رو می‌توان پیچیدگی را در دو بخش طبقه‌بندی کرد: پیچیدگی مسئله و پیچیدگی راه‌حل.

پیچیدگی مسئله: که می‌توان به‌عنوان پیچیدگی ذاتی اشاره کرد، که در طول فازِ نیازمندی‌ها ایجاد می‌شود.

پیچیدگی راه‌حل: که از آن به‌عنوان پیچیدگی افزوده نیز یاد می‌شود و بعد از پیچیدگی مسئله مطرح می‌شود. این نوع از پیچیدگی اساساً، در مراحل توسعه نرم‌افزار و بعد از فاز نیازمندی‌ها و خصوصاً در فازهای طراحی و تولید کد ایجاد می‌شود.

پیچیدگی نرم‌افزار را به اینصورت تعریف می‌کند: «میزان منابع صرف شده توسط سیستم (انسان یا هر چیز دیگری) که در حال تعامل با بخشی از نرم‌افزار، بمنظور انجام فعا لیتی معین است».

همچنین Henderson-Sellers با نقد نظریه Basili تعریف دیگری از پیچیدگی نرم‌افزار را ارئه داد: «پیچیدگی شناختی نرم‌افزار اشاره به خصوصیاتی از نرم‌افزار دارد که بر سطح منابع مورد استفاده توسط

یک فرد بمنظور انجام یک فعالیت بخصوص تاثیر دارد». این تعریف روی پیچیدگی ذاتی که از خصوصیات و کارکردهای نرم‌افزار ناشی می‌شود تاکید دارد.

Myers پیچیدگی را به اینصورت تعریف می‌کند:«پیچیدگی یک شی معیاری از تلاش ذهنی موردنیاز جهت فهم و ایجاد آن شی است».

این واقعیت که پیچیدگی نرم‌افزار در طول زمان ایجاد می‌شود ، تعریف و اندازه‌گیری آن را سخت می‌کند. ترکیب پیچیدگی حاصل از مسئله، طراحی و کدنویسی در یک عدد بسیار مشکل است. به همین دلیل محققان پیچیدگی را از جنبه‌های مختلفی مورد بحث قرار می‌دهند.

در مرکزِ موضوع پیچیدگی، مسئله شناخت قرار می‌گیرد. به اینصورت که، زمانیکه نرم‌افزار تأخیر دارد، ناقص است، بدرستی معین نشده یا بطور دقیق قیمت‌گذاری نمی‌شود می‌تواند به این دلیل باشد که توسعه‌دهندگان نرم‌افزار در یک حوزه ناشناخته و پویا کار می‌کنند و کورمال به دنبال شناخت دقیق و کامل موضوعاتی هستند که نرم‌افزار باید اداره کند. وقتی نرم‌افزار براحتی قابل استفاده مجدّد نیست یا نگهداری و عملیاتی شدن آن سخت است، بدلیل کنار هم قرار گرفتن پیچیده‌ی بخش‌هایی با تعامل بالاست که شناخت نرم‌افزار را مشکل می‌کند. علت بی دوامی نرم‌افزار یا بروز اتفاقات غافلگیرکننده و ناخوشایند، نبودن شناخت کامل است که به مصالحه و انعطاف ناپذیری منجر شده و تغییر و انحراف از استفاده معمول را خطرناک می‌کند. در هر حالت، پیامدِ پیچیدگی خسارتی است که بدلیل عدم شناخت می‌پردازیم. بنابراین پیچیدگی میزانی برای دشواری شناخت و درک یک چیز است.

1-4: علل پیچیدگی نرم افزار

دو کلاس مشخص از علل پیچیدگی یعنی داخلی[1](در نتیجه ماهیت اجزای سیستم) و خارجی[2](در نتیجه محیط سیستم) شناسایی شده‌اند.

پیچیدگی داخلی از مولفه‌های سازنده سیستم و تعامل بین آنها ناشی می‌شود. تحلیل «جامع» عوامل شرکت کننده در پیچیدگی نرم‌افزار نه عملی و نه مفید است. بجای آن تعدای از عوامل کلیدی تکنیکی شرکت کننده را عنوان می‌کنیم:

[1] Intrinsic Complexity
[2] Extrinsic Complexity

تعاملات: نرم‌افزارهـای مـدرن تعـاملات زیـادی در سـطوح مختلـف دارنـد. در سـطح توابـع، برنامـه‌هـا، زیرسیستم‌ها و ...

سیستمی از سیستم‌ها: سیستم‌ها بصورت سلسله مراتبی ایجاد می‌شوند. بعنوان مثال، سیستم‌های بانک روی نرم‌افزار اینترنت ایجاد می‌شود که روی نرم‌افزار ارتباطات ایجاد می‌شود که خود آن کامپیوترهای پیچیده و رسانه‌های ارتباطات را بکار می‌گیرند.

مولفه‌های Commercial off the shelf: تمایل به استفاده زیاد از نرم‌افـزار COTS وجـود دارد کـه مقـدار قابل توجهی عملکرد بلا استفاده را به سیستم معرفی می‌کند که ممکن است با توابع مطلوب تداخل داشـته باشند.

اکثر پروژه‌ها از صفر شروع نمی‌شوند. معمولاً مقداری از کار قبلاً انجام شده و می‌توانیم در پروژه جدید از آن استفاده کنیم. حالت دیگر این است که بخشی از کار را به پیمانکـار شـرکت‌هـای داخلـی یـا خـارجی واگذار کنیم.

زبانها و ابزارهای استفاده شده برای توسعه نرم‌افزار: همه ما تجربه کرده‌ایم که هنگام پیاده‌سازی سیستم بـا زبانهای نسل چهارم یا زبانهای ویژوال نسبت به استفاده از زبانهای سطح پایین یا به کمـک ویرایشـگرهای متنی به تلاش زیادی (مثلاً، نوشتن تعداد زیادی LOC) نیاز نداریم.

دو مورد اول اتصّالات و مورد آخر تنوع را افزایش می‌دهد. البته تعامل بین این موارد دشواری‌ها را تشدید می‌کنند.

پیچیدگی خارجی از محیطی که سیستم در آن استفاده می‌شود ناشی می‌شود عوامل خارجی کلیـدی مـوثر در پیچیدگی عبارتنداز:

موفقیت: سیستم‌های نرم‌افزاری، بسیار مورد استفاده و از نظر تجاری موفق هستند و همـین امـر منجـر بـه طرح نیازمندی‌های بیشتری می‌شود.

تعبیه در سازمان‌ها: نرم‌افزارها نیاز سازمان‌های پیچیده را برآورده می‌سازند و اغلب سیستم‌های کـامپیوتری سازمان‌ها را شکل می‌دهند. سیستم کامپیوتری باید روش کار سازمان‌ها را منعکس کرده و اهداف کسب‌و-کار آن را تأمین کنند همچنین با تغییر سازمان‌ها، سیستم‌ها باید تغییر کنند تا همچنان موثر باشند.

تعبیه در درون سیستم‌ها: در بسیاری از زمینه‌ها سیستم‌های کامپیوتری در داخل محصولاتِ مهندسـی سـازِ پیچیده قرار می‌گیرند و طراحان سیستم نیاز دارند که برای توسعه نرم‌افزار سیستم، محیط را درک کنند.

وابستگی و قابلیت اتّکا: وابستگی به سیستم‌های کامپیوتری روز به روز هم در زمینه کسب‌وکار و هم در زمینه بحرانی و حیاتی افزایش می‌یابد. همین مطلب نیاز به قابلیت اتّکا (حتی در وجود خرابی‌های داخلی و خارجی) را افزایش می‌دهد.

خودکاری: طراحی سیستم‌ها برای فعالیت‌های خودکار هر روز بیشتر می‌شود.

این عوامل، پیچیدگی را به اشکال مختلف افزایش می‌دهند. بعنوان مثال، نیازمندی‌های قابلیت اتّکا به افزایش مقیاس، اتّصالات و تنوّع پیکربندی گرایش دارد.

1-5: دسته بندی پیچیدگی نرم افزار

بر اساس نظریهFenton و Pfleeger پیچیدگی نرم‌افزار از بخشهای زیر تشکیل شده است:

پیچیدگی مسئله (یا پیچیدگی محاسباتی)، که پیچیدگی مسئله تحت بررسی را اندازه‌گیری می‌کند. این نوع پیچیدگی به فاز نیازمندی‌ها و در زمان تعریف مسئله برمی‌گردد.

این نوع از پیچیدگی مستقیماً توسط دامنه کاربرد یا فضای مسئله ایجاد می‌شود. اگر نرم‌افزاری برای پرتاب موشک نوشته شود، باید اصولی در مورد دانش موشک داشت. پیچیدگی دامنه، قابل اجتناب نیست و باعث مشکل ارتباطی اعضای تیم می‌شود که منجر به کاستی‌هایی در محصول، تجاوز هزینه، تأخیر در زمانبندی می‌شود. همچنین پیچیدگی دامنه، باعث درک بسیار پایین از مسئله، رخداد حالات ممکن متعدد برای برنامه و در نتیجه عدم قابلیت‌اطمینان است.

پیچیدگی الگوریتمی، که پیچیدگی الگوریتم پیاده‌سازی شده برای حل مسئله است. در اینجا مفهوم کارایی مطرح شده و با مقایسه تجربی الگوریتم‌های مختلف می‌توان الگوریتمی که کاراترین راه‌حل و در نتیجه کمترین میزان پیچیدگی افزوده را دارد تعیین کرد. این نوع پیچیدگی به محض ایجاد راه‌حل و معمولاً در فاز طراحی قابل اندازه‌گیری است.

پیچیدگی ساختاری، که ساختار نرم‌افزار مورد استفاده برای پیاده‌سازی الگوریتم را اندازه می‌گیرد. محققان در مدت زمان طولانی به این نتیجه رسیدند که ارتباطی بین ساختار محصولات و کیفیت آنها وجود دارد. به عبارتی، محصول مصنوعی که برای ساخت نرم‌افزار استفاده می‌شود پیچیدگی ساختاری را ایجاد می‌کند. معمولاً مهندسان نرم‌افزار با یادگیری جزئیات زبان‌های برنامه‌نویسی کار خود را شروع می‌کنند که اغلب با فضای مسئله هم‌ارزی ندارد (با پیچیدگی مسئله مرتبط نیست). مثالی از مشکلاتی که این نوع پیچیدگی را ایجاد می‌کند، زمانی اتفاق می‌افتد که برنامه‌نویس باید یک ویژگی کثیرالوقوع را با هدف بروز رسانی یا ایجاد نسخه‌ی جدیدی از برنامه تغییر دهد. اغلب پیدا کردن تمام نمونه‌های یک چنین ویژگی، در کد چند

میلیون خطی برای برنامه‌نویسان کار سختی است. ناتوانی در این کار می‌تواند خطاهای جدیدی، نظیر پیچیدگی وابسته به روان‌شناسی مطرح کند. پیچیدگی ساختاری توسط انسان قابل درک است. این نوع پیچیدگی، ویژگی‌های برنامه‌نویسان و پیچیدگی مسئله را در بر می‌گیرد.

پیچیدگی شناختی، میزان تلاش مورد نیاز برای درک نرم‌افزار است. که نرم‌افزار تحت بررسی یا تکمیل شده باید از جنبه روانشناسی یا دشواری نسبی بررسی گردد. بصورت یک متغیر توصیفی منتخب برای بررسی میزان کار مورد نیاز جهت توسعه‌ی تابع نرم‌افزار، که شامل تجزیه و تخصیص فرایندهای تابعی است و طراحی هر فرایند تابعی برای انجام نیازهای کاربران مطابق مشخصات نرم‌افزار است. ساده‌سازی تابع یک سیستم مهندسی همیشه ممکن نیست زیرا تابع هسته‌ی مطلوب یا مورد نیاز، خود بسیار پیچیده است. وقتی ماهیت اصلی یک کار پیچیده برای مدیریت از طریق فرایندهای مهندسی جامع مرسوم زیادی حجیم است، یک راه حل بهتر این است که فرایند تدریجی به کار بگیریم تا محیطی برای وقوع نوآوری‌های مداوم ایجاد شود.

پیچیدگی ساختاری و الگوریتمی به همراه پیچیدگی شناختی می‌تواند برای اندازه‌گیری پیچیدگی راه‌حل و مخصوصاً پیچیدگی افزوده استفاده شود. پیچیدگی مسئله ارتباط مستقیم با پیچیدگی نیازمندی‌های ابتدایی دارد. بنابراین واضح است که جنبه‌هایی از پیچیدگی که در طول فرایند توسعه نرم‌افزار قابل اندازه‌گیری است پیچیدگی ساختاری و الگوریتمی است. امّا زمانیکه مهندسان نرم‌افزار می‌خواهند تلاش یا منابع مورد نیاز برای یک پروژه خاص را پیش‌بینی کنند اندازه‌گیری پیچیدگی مسئله، مورد توجه است. با مقایسه مسائل جدید با مسائل قدیمی و در نظر گرفتن تأثیر راه‌حل‌های آنها می‌توان ویژگی‌های راه‌حل برای مسئله جدید، نظیر هزینه، چگالی نقص، اندازه و غیره را پیش‌بینی کنیم.

همچنین، سه نوع مشخص از پیچیدگی وابسته به روانشناسی که بر درک برنامه‌نویس از نرم‌افزار اثر می‌گذارد عبارتنداز: پیچیدگی مسئله، پیچیدگی طراحی سیستم و پیچیدگی رویه‌ای.

پیچیدگی مسئله، تابعی از دامنه مسئله است. به بیان ساده، فرض بر این است که درک مسائل پیچیده برای برنامه‌نویس مشکل‌تر از درک مسائل ساده است. از آنجا که این نوع از پیچیدگی غیر قابل کنترل است عموماً در مهندسی نرم‌افزار نادیده گرفته می‌شود. پیچیدگی سیستمی نگاشتِ فضای مسئله به یک نمایش مشخص است. پیچیدگی داده‌ای و پیچیدگی ساختاری دو نوع از پیچیدگی طراحی سیستم است که برای سیستمهای ساخت یافته تعریف می‌شود. پیچیدگی ساختاری به مفهوم اتّصال اشاره دارد. اتّصال، وابستگی بین پیمانه‌های کد منبع را اندازه‌گیری می‌کند، هر چه اتّصال بین پیمانه‌ها بیشتر باشد درک پیمانه

برای برنامه‌نویس مشکل‌تر خواهد بود. پیچیدگی داده‌ای مفهوم چسبندگی را تداعی می‌کند چسبندگی، وابستگی درون پیمانه را اندازه‌گیری می‌کند. در این حالت، هر چه پیمانه چسبنده‌تر باشد درک آن برای برنامه‌نویس آسان‌تر است. در فصل متریک‌های نرم‌افزاری به این مباحث بیشتر پرداخته شده و به بررسی متریک‌های مربوط به اندازگیری آن پرداخته می‌شود .

پیچیدگی سیستم بر مجموع پیچیدگی داده‌ای و ساختاری همه پیمانه‌های سیستم مبتنی است. این میزان‌ها، پیچیدگی اطلاعاتی سیستم در سطح پیمانه و سیستم را نشان می‌دهند. این روشِ چند سطحی در میزان‌های سنّتی نرم‌افزار که بر اتّصال و چسبندگی تأکید دارد، پایه‌ای برای مدلهای پیچیدگی پیشنهادی برای سیستمهای OO می‌باشد.

پیچیدگی رویه‌ای به ساختار منطقی برنامه مربوط می‌شود. این روش اندازه‌گیری پیچیدگی، طول برنامه (تعداد نشانه‌ها یا تعداد خط کد برنامه) یا تعداد ساختمانهای منطقی (ترتیب، تصمیم‌گیریها یا حلقه‌ها) را که برنامه شامل می‌شود بعنوان پیچیدگی برنامه در نظر می‌گیرد.

Brooks ، بر طبقه‌بندی غالب برای پیچیدگی، یعنی پیچیدگی ذاتی (ضروری[1]) و پیچیدگی ضمنی[2] اشاره می‌کند. پیچیدگی ضروری از نیازمندی‌ها ناشی می‌شود و در ارتباط با آنچه که سیستم انجام می‌دهد و نه چگونگی انجام آن می‌پردازد. پیچیدگی ضروری می‌تواند مثلاً از سخت‌افزار به نرم‌افزار منتقل شود ولی نمی‌تواند حذف شود مگر با حذف نیازمندی‌های غیر ضروری.

پیچیدگی ضمنی از برنامه‌های کامپیوتری یا از فرایند توسعه (برنامه‌نویسی کامپیوتری) یا از انتخاب‌های صورت گرفته درساخت یک سیستم ناشی می‌شود مثلاً از معماری نرم‌افزار، طراحی ساختمان‌داده، زبان برنامه‌نویسی و راهبردهای کد نویسی نشأت می‌گیرد. انتخاب‌های عاقلانه‌تر می‌توانند پیچیدگی ضمنی را کاهش دهند. پس پیچیدگی ضمنی در حقیقت نتیجه چگونگی ساخت نرم‌افزار است. بنابراین بهترین راه برای مقابله با پیچیدگی این است که ابتدا مسأله (چه چیز) را درست تعریف کنیم سپس دنبال ایجاد یا انتخاب راهکاری (چگونگی) مناسب برای حلّ مسأله باشیم. این نوع پیچیدگی با توسعه سیستم سر و کار دارد و شامل ویژگی‌هایی نظیر اندازه‌ی سیستم، تعداد پیمانه‌ها، تعداد توابع در درون سیستم و اتّصال بین پیمانه‌ها می‌باشد.

[1] Essential
[2] Incidental