

فهرست مطالب

فصل 1

- 1-1: خلاصه بخش 18
- 2-1: پیچیدگی نرم افزار 18
- 3-1: تعریف و ماهیت پیچیدگی نرم افزار 19
- 4-1: علل پیچیدگی نرم افزار 20
- 5-1: دسته بندی پیچیدگی نرم افزار 22
- 6-1: پیچیدگی در چرخه حیات نرم افزار 25
- 7-1: فازهای پیچیدگی نرم افزار 26
- 8-1: دلایل بروز پیچیدگی نرم افزار 26
- 9-1: تاثیر و نتایج پیچیدگی نرم افزار 27
- 1-9-1: کیفیت نرم افزار 27
- 2-9-1: نگهداری نرم افزار 27
- 10-1: بهره وری 28
- 11-1: پیچیدگی و مهندسی نیازمندها 28
- 12-1: مدل عوامل پیچیدگی نرم افزار 30
- 13-1: عوامل پیچیدگی ذاتی نرم افزار 33
- 1-13-1: نیازمندیهای کار کردی 33
- 2-13-1: نیازمندیهای غیر کار کردی 34
- 14-1: پیچیدگی عارضی نرم افزار 36
- 15-1: نتیجه گیری 37
- 16-1: سوالات متداول 37
- 17-1: منابعی برای مطالعه بیشتر 38

فصل 2

- 1-2: خلاصه بخش 40
- 2-2: ماهیت نگهداری 41
- 3-2: تعریف نگهداری 41
- 4-2: نرم افزار تعمیر و نگهداری چیست؟ 42

- 43 1-4-2: توانایی‌ها و ویژگی‌های یک نرم افزار نگهداری و تعمیرات (CMMS)
- 44 2-4-2: ویژگی‌های نرم افزار CMMS INTERNET BASE
- 45 5-2: دسته‌بندی‌های تعمیر و نگهداری نرم افزار
- 46 6-2: دسته بندی ISO در تعمیر نرم افزاری
- 47 7-2: هزینه و رقابت‌ها
- 48 1-7-2: نسبت هزینه های تعمیر و نگهداری نرم افزار
- 48 2-7-2: هزینه های تعمیر مطلق نرم افزار
- 48 3-7-2: انواع وظایف تعمیر
- 48 4-7-2: مقادیر کدهای موروثی
- 49 8-2: مدل‌ها، وظایف، استفاده مجدد و ابزارها
- 50 1-8-2: استفاده مجدد
- 51 2-8-2: وظایف
- 52 3-8-2: ابزار و محصولات در دسترس از لحاظ تجاری
- 53 9-2: فرایند
- 54 10-2: فرایند پیاده سازی
- 55 11-2: اصلاحات پیاده سازی
- 56 12-2: مدیریت تعمیر و نگهداری
- 56 1-12-2: طرح و برنامه ریزی
- 57 13-2: ساختار سازمانی متفاوت در سازمان و تعمیر نرم افزار
- 58 14-2: مهندسی معکوس
- 59 15-2: مهندسی مجدد
- 60 16-2: ارتباط پیچیدگی با نگهداری نرم افزار
- 61 17-2: فرایند نگهداری نرم افزار
- 61 18-2: انواع نگهداری
- 63 19-2: هزینه های نگهداری و برآورد آن
- 64 20-2: نتیجه گیری
- 65 21-2: سوالات متداول
- 66 22-2: منابعی برای مطالعه بیشتر

فصل 3

- 68 1-3: خلاصه بخش

68	2-3: اندازه‌گیری و اهداف اندازه‌گیری.....
70	3-3 کاربردهای اندازه‌گیری.....
70	4-3 تعریف متریک نرم‌افزاری.....
71	5-3: طبقه‌بندی متریک‌های نرم‌افزاری.....
74	6-3 متریک‌های پیچیدگی نرم‌افزار.....
75	1-6-3 خطوط متن.....
75	2-6-3 متریک فرارایت نیازمندی.....
76	3-6-3 تعداد خطوط کد.....
77	4-6-3 متریک نقاط کارکرد.....
80	7-3 متریک‌های جریان اطلاعات.....
80	1-7-3 متریک چسبندگی.....
80	2-7-3 متریک‌های پیوستگی.....
80	3-7-3 متریک جریان اطلاعات HENRY و KAFURA.....
81	8-3 متریک‌های شیء‌گرا.....
82	1-8-3 سلسله متریک‌های C&K.....
84	متریک پاسخ‌های کلاس (RFC).....
84	متریک فقدان انسجام بین متدها (LCOM).....
85	متریک پیوستگی بین کلاس‌های شیء (CBO).....
85	2-8-3 سلسله متریک‌های MOOD.....
87	9-3 نتیجه‌گیری.....
88	10-3 سوالات متداول.....
88	10-3 منابعی برای مطالعه بیشتر.....

فصل 4

90	1-4 خلاصه بخش.....
90	2-4 تعریف متریک‌های نرم‌افزاری.....
93	3-4 اندازه و میزانهای پیچیدگی.....
95	4-4 متریک‌های دیگر پیچیدگی.....
95	1-4-4 متریک جریان اطلاعات HENRY و KAFURA.....
96	2-4-4 استاندارد IEEE 982.2:.....
100	5-4 مدل پیچیدگی شیء‌گرا.....

- 102..... 6-4 دسته بندی معیارهای شی گرا(OO)
- 103..... 7-4 سوالات متداول
- 104..... 8-4 منابعی برای مطالعه بیشتر

فصل 5

- 106..... 1-5 خلاصه بخش
- 106..... 2-5 عدد دورانی MCCABE
- 107..... 3-5 محاسبه‌ی پیچیدگی سیکلوماتیک CC
- 107..... 4-5 مثالی از روش مک کیب
- 108..... 5-5 کاربرد های پیچیدگی دورانی مک کیب
- 109..... 6-5 سوالات متداول
- 110..... 7-5 منابعی برای مطالعه بیشتر

فصل 6

- 112..... 1-6: خلاصه بخش
- 112..... 2-6: روش HALSTEAD
- 113..... 3-6: مشکلات متریکهای هالستد
- 113..... 4-6: میزان های HALSTEAD
- 114..... 5-6: مثالی از روش HALSTEAD
- 116..... 6-6: سوالات متداول
- 116..... 7-6: منابعی برای مطالعه بیشتر
- 118..... 1-7 خلاصه بخش
- 118..... 2-7 مقدمه
- 119..... 3-7 تاریخچه مهندسی نیازمندیها
- 120..... 4-7 تعریف نیازمندی
- 121..... 5-7 طبقه‌بندی نیازمندیها
- 124..... 6-7 ذینفعان
- 126..... 7-7 مهندسی نیازمندیها
- 127..... 8-7 دلایل اهمیت مهندسی نیازمندیها
- 130..... 9-7 مراحل مهندسی نیازمندیها

132	7-9-1 استخراج نیازمندیها
-----	--------------------------

فصل 7

134	7-10 تحلیل و مذاکرات نیازمندیها
137	7-11 مستندسازی نیازمندیها
141	7-12 تایید اعتبار نیازمندیها
141	7-13 مدیریت نیازمندیها
142	7-14 محصول مهندسی نیازمندیها
146	7-15 چگونگی انجام مهندسی نیازمندیها
146	7-16 نتیجه گیری
147	7-17 سوالات متداول
147	7-18 منابعی برای مطالعه بیشتر

فصل 8

150	8-1 خلاصه بخش
150	8-2 مقدمه
152	8-3 معماری ها از کجا می آیند؟
153	8-4 انواع معماری
156	8-5 معماری نرم افزار
157	8-5-1 تعریف معماری نرم افزار
157	8-5-2 اجزای معماری نرم افزار
159	8-6 سایر دیدگاه ها
160	8-7 حلقه کاری معماری
161	8-8 مراحل فرآیند معماری نرم افزار
163	8-9 چرا معماری نرم افزار مهم است؟
171	8-10 درباره معماری نرم افزار
173	8-11 چه چیزی یک معماری خوب را می سازد؟
177	8-12 چه زمانی می توان طراحی را شروع کرد؟
178	8-13 معماری در چرخه حیات
178	8-14 الگوهای معماری، مدل های مرجع و معماری های مرجع

181	15-8 دیدها و ساختارهای معماری
182	16-8 ساختارهای نرم افزار
187	17-8 ساختارهای معماری یک سیستم
188	1-17-8 ساختارهای مرتبط با هم
189	2-17-8 کدام ساختارها انتخاب می شوند؟
190	18-8 الگوها و سبک های معماری
191	1-18-8 سبک های معماری نرم افزار
192	2-18-8 تعریف سبک معماری
192	3-18-8 انواع سبک های معماری
202	19-8 مدل های معماری نرم افزار
202	1-19-8 معماری MAIN FRAME
204	2-19-8 معماری FILE SERVER
205	3-19-8 معماری CLIENT/SERVER
211	4-19-8 معماری سرویس گرا
218	5-19-8 معماری مبتنی بر ERP
220	2.19.8 معماری نرم افزار شی گرا
228	20-8 خصوصیات معماری
229	21-8 وظیفه مندی و معماری
229	22-8 معماری و خصوصیات کیفی
230	1-22-8 مشخصه های کیفیتی نرم افزار
232	2-22-8 مدل های کیفیتی در نرم افزار
236	3-22-8 صفت کیفیتی امنیت در معماری نرم افزار
238	23-8 معماری نرم افزار در آینده
240	24-8 نتیجه گیری
240	25-8 سؤالات متداول
241	26-8 منابعی برای مطالعه بیشتر

فصل 9

244	1-9 خلاصه بخش
245	2-9 قوائد طراحی
245	3-9 انتزاع/تجرد/چکیدگی (ABSTRACTION)

245	4-9 وابستگی (COUPLING)
245	5-9 پیوستگی (COHESION)
246	6-9 تفکیک یا ماژول بندی (DECOMPOSITION/MODULARIZATION)
246	9-6-1 کپسوله سازی/پنهان سازی اطلاعات (HIDING INFORMATION/ENCAPSULATION)
246	9-7 فعالیتهای طراحی
247	9-8 روشهای کلی طراحی
248	9-9 خصوصیات کیفی طراحی (QUALITY ATTRIBUTES)
248	9-10 مراحل طراحی
249	9-10-1 فعالیت های طراحی سیستم های نرم افزاری بزرگ
249	9-11 متدهای طراحی DESIGN METHODS
250	9-12 توصیف طراحی (DESCRIBING DESIGN)
250	9-13 استراتژی های طراحی (DESIGN STRATEGIES)
251	9-14 کیفیت طراحی (DESIGN QUALITY)
252	9-15 سطح ذینفعان
254	9-16 سطح تیم پروژه
256	9-17 مدل نتایج پیچیدگی نرم افزار
257	9-18 خطا خیزی
258	9-19 متریک پیچیدگی مبتنی بر نیازمندیها
259	9-19-1 پیچیدگی ورودی خروجی (I/O COMPLEXITY)
259	9-19-2 پیچیدگی نیازمندیهای کارکردی (FUNCTIONAL REQUIREMENT COMPLEXITY)
261	9-19-3 پیچیدگی نیازمندیهای غیر کارکردی (NON-FUNCTIONAL REQUIREMENT COMPLEXITY)
266	9-20 سوالات متداول
266	9-21 منابعی برای مطالعه بیشتر

فصل 10

268	10-1 خلاصه بخش
268	10-2 الگوریتم
268	10-3 خصوصیات یک الگوریتم
269	10-3-1 منشاء واژهی الگوریتم
269	10-3-2 نقش الگوریتمها در علوم رایانه

269	4-10 تحلیل الگوریتم
270	1-4-10 جنبه حقوقی
270	5-10 ارزیابی کارائی الگوریتم‌ها
270	1-5-10 پیچیدگی زمانی
271	2-5-10 پیچیدگی حافظه
273	6-10 نمایش پیچیدگی الگوریتم
274	7-10 بهبود پیچیدگی یک برنامه
274	8-10 سوالات متداول
274	9-10 منابعی برای مطالعه بیشتر

فصل 11

276	1-11 خلاصه بخش
276	2-11 مقدمه
277	3-11 نیازمندی‌های امنیت
278	4-11 سیاست امنیتی
278	5-11 راهکارهای امنیتی
279	6-11 تضمین‌های امنیتی
279	7-11 مؤلفه‌های امنیت
281	8-11 مدل‌های امنیتی
281	1-8-11 مدل‌های کنترل دسترسی
285	2-8-11 مدل‌های جریان اطلاعات
285	9-11 معیارهای رسمی برای ترکیب مؤلفه‌ها
286	1-9-11 ترکیب آبادی-لمپسورت در چارچوب آلپرن-اشنایدر
287	10-11 جامعیت
288	11-11 رازداری (امنیت اطلاعات)
288	1-11-11 ارائه چارچوب
289	2-11-11 ترکیب
289	12-11 راهبردهای معماری برای امنیت نرم افزاری
290	1-12-11 تکنیک برچسب گذاری مبتنی بر شیء
292	2-12-11 مدل سازی ایمنی مبتنی بر UML
292	3-12-11 ASTER

293	4-12-11 مدل معماری سیستم
296	5-12-11 مقایسه رویکردهای معماری برای امنیت نرم افزار
297	13-11 رابطه پیچیدگی و امنیت
297	14-11 نتیجه گیری
298	15-11 سؤالات متداول
298	16-11 منابعی برای مطالعه بیشتر

فصل 12

300	1-12 خلاصه بخش
300	2-12 آشنایی با نرم افزار UNDERSTAND
301	3-12 ایجاد پروژه جدید
305	4-12 پنجره ی ENTITY FILTER
306	5-12 پنجره ی INFORMATION BROWSER
309	6-12 جست و جو در فایل ها
310	7-12 نمای گرافیکی
312	8-12 بروز رسانی تغییرات در فایل
314	9-12 پنجره ی ARCHITECTURE BROWSER (مرورگر معماری)
318	10-12 گراف های وابستگی معماری (DEPENDENCY GRAPHS)
322	11-12 گرفتن اطلاعات در INFORMATION BROWSER
324	12-12 مقایسه ی فایل ها و پوشه ها
325	13-12 متریک ها (METRICS)
329	14-12 گزارش ها (REPORTS)
333	15-12 نمایش متریک ها به صورت یک نقشه درختی
336	16-12 چک کردن کد ها (CHECK CODE)
347	واژه نامه

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

هزینه های رو به رشد نرم افزار سازمانهای نرم افزاری را بر آن داشته تا دنبال راه حلی برای کاهش این هزینه ها باشند. برای نیل به این مقصود محققان دنبال رابطه ای بین ویژگی های نرم افزار و مشکلات و دشواریهای کار توسعه نرم افزار بودند. دشواری کار زمان بیشتری برای انجام آن طلب می کند در این زمان منابع بیشتری بکار گرفته می شوند و منابع بیشتر یعنی تحمیل هزینه های بیشتر. یکی از دلایل پرداختن به موضوع پیچیدگی نرم افزار و اندازه گیری آن برای کنترل هزینه ها در چرخه حیات نرم افزار است زیرا پیچیدگی نرم افزار از عوامل اصلی در افزایش سریع هزینه های توسعه و نگهداری عنوان می شود. پیچیدگی نرم افزار فاکتوری است که در فعالیت برنامه نویسی کمتر شناخته شده و به راحتی قابل اندازه گیری یا توصیف نیست و غالباً در طول فرایند برنامه ریزی پروژه نادیده گرفته می شود. بنابراین به دنبال روشی کمی برای پیش بینی میزان دشواری نگهداری، تغییر و فهم نرم افزار هستیم. تا با اندازه گیری و کنترل آن هزینه ها را در چرخه حیات نرم افزار کاهش دهیم.

1-2: پیچیدگی نرم افزار

در چند دهه اخیر پیچیدگی نرم افزار دوره جدیدی در زمینه دانش کامپیوتری ایجاد کرده است. پیچیدگی نرم افزار را می توان بعنوان گرداننده اصلی هزینه، قابلیت اطمینان و عملکرد سیستم های نرم افزاری تعریف کرد. با هزینه های بالای نرم افزار و این حقیقت که حدود 75٪ چرخه حیات نرم افزار به تست و نگهداری نرم افزار سپری می شود، مسئله پیچیدگی نرم افزار اهمیت ویژه ای پیدا کرده است. پیچیدگی نرم افزار از عوامل اصلی افزایش سریع هزینه های توسعه و نگهداری می باشد و از طرف دیگر تاثیر بسزایی در کیفیت و بهره وری محصول دارد.

پیچیدگی نرم افزار فاکتوری است که براحتی قابل اندازه گیری یا توصیف نیست و غالباً در طول فرایند برنامه ریزی پروژه نادیده گرفته می شود. بنابراین محققان دنبال روشی کمی برای پیش بینی 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 بازیابی یک سیستم یا مولفه که طراحی یا پیاده‌سازی شده است». عبارت «سختی درک» اشاره دارد که پیچیدگی لزوماً مقیاس مطلق 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 این رو می‌توان پیچیدگی را در دو بخش طبقه‌بندی کرد: پیچیدگی مسئله و پیچیدگی راه‌حل.

پیچیدگی مسئله: که می‌توان به عنوان پیچیدگی ذاتی اشاره کرد، که در طول فاز نیازمندی‌ها ایجاد می‌شود. پیچیدگی راه‌حل: که از آن به عنوان 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 فعلی معین است».

همچنین Henderson-Sellers با نقد نظریه Basili تعریف دیگری از پیچیدگی نرم‌افزار را ارائه داد: «پیچیدگی شناختی نرم‌افزار اشاره به خصوصیات 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 نرم افزار ناشی می شود تاکید دارد.

Myers پیچیدگی را به اینصورت تعریف می کند: «پیچیدگی یک شی معیاری از تلاش ذهنی مورد نیاز جهت فهم و ایجاد آن شی است».

این واقعیت که پیچیدگی نرم افزار در طول زمان ایجاد می شود، تعریف و اندازه گیری آن را سخت می کند. ترکیب پیچیدگی حاصل از مسئله، طراحی و کدنویسی در یک عدد بسیار مشکل است. به همین دلیل محققان پیچیدگی را از جنبه های مختلفی مورد بحث قرار می دهند.

در مرکز موضوع پیچیدگی، مسئله شناخت قرار می گیرد. به اینصورت که، زمانی که نرم افزار تأخیر دارد، ناقص است، بدرستی معین نشده یا بطور دقیق قیمت گذاری نمی شود می تواند به این دلیل باشد که توسعه دهندگان نرم افزار در یک حوزه ناشناخته و پویا کار می کنند و کورمال به دنبال شناخت دقیق و کامل موضوعاتی هستند که نرم افزار باید اداره کند. وقتی نرم افزار براحتی قابل استفاده مجدد نیست یا نگهداری و عملیاتی شدن آن سخت است، بدلیل کنار هم قرار گرفتن پیچیده ی بخش هایی با تعامل بالاست که شناخت نرم افزار را مشکل می کند. علت بی دوامی نرم افزار 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 می کند. در هر حالت، پیامد پیچیدگی خسارتی است که بدلیل عدم شناخت می پردازیم. 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^۱ (در نتیجه ماهیت اجزای سیستم) و خارجی^۲ (در نتیجه محیط سیستم) شناسایی شده اند.

پیچیدگی داخلی از مولفه های سازنده سیستم و تعامل بین آنها ناشی می شود. تحلیل «جامع» عوامل شرکت کننده در پیچیدگی نرم افزار نه عملی و نه مفید است. بجای آن تعدادی از عوامل کلیدی تکنیکی شرکت کننده را عنوان می کنیم:

¹ Intrinsic Complexity

² Extrinsic Complexity

تعاملات: نرم‌افزارهای مدرن تعاملات زیادی در سطوح مختلف دارند. در سطح توابع، برنامه‌ها، 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 نرم‌افزار COTS وجود دارد که مقدار قابل توجهی عملکرد بلا استفاده را به سیستم معرفی می‌کند که ممکن است با توابع مطلوب تداخل داشته 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 پیمانکار شرکت‌های داخلی یا خارجی واگذار کنیم.

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

دو مورد اول اتصالات 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 کلیدی موثر در پیچیدگی عبارتند از:

موفقیت: سیستم‌های نرم‌افزاری، بسیار مورد استفاده و از نظر تجاری موفق هستند و همین امر منجر به طرح نیازمندی‌های بیشتری می‌شود.

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

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

وابستگی و قابلیت اتکا: 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 اتکا (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 اتکا به افزایش مقیاس، اتصالات و تنوع پیکربندی گرایش دارد.

1-5: دسته بندی پیچیدگی نرم افزار

بر اساس نظریه Fenton 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 قابلیت اطمینان 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. اما 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 عبارتند از: 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 نداشتن 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 اتصال اشاره دارد. اتصال، وابستگی بین پیمانه‌های کد منبع را اندازه‌گیری می‌کند، هر چه اتصال 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 اندازه‌گیری 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 روش چند سطحی در میزان‌های سنتی نرم‌افزار که بر اتصال 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 ضمنی² 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 در ساخت 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 حل 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 اتصال بین پیمانه‌ها می‌باشد.

¹ Essential

² Incidental