فهرست مطالب

CHAPTER 1

کلیات مهندسی نرم افزار

۱. کلیات مهندسی نرم افزار

Roger Pressman محقق و نویسنده شناخته شده کتاب های مراجع مهندسی نرم افزار، همانند:

Software engineering : a practitioner's approach (first ,1982 edition)

Making software engineering happen: a guide for ,1988 *instituting the technology*

.Software engineering: a beginner's guide,1988

Software shock : the danger & the opportunity,1991

Software engineering : a practitioner's approach (sixth ,2005 edition)

Web engineering : a practitioner's approach,2009

۱.۱ خلاصه فصل

امروزه برای هر شرکتی که با موضوع فناوری اطلاعات و صنعت نرم افزار سروکار دارد، توسعه و بهبود کیفیت مراحل تولید نرم افزار و افزایش کارایی و بهره‌وری افراد درگیر با آن به امر مهمی تبدیل شده است. همزمان با قدرتمند شدن کامپیوترها، تقاضا برای نرم افزارهای پایدارتر نیز افزایش یافته و به دلیل این که فناوری نقش بسیار حیاتی در پیشبرد کسب‌وکار ایفا می‌کند، مشکلات نرم افزار، مشکلات مهمی محسوب می‌شوند که بر روی عملکرد بسیاری از شرکت‌ها تاثیر گذارند. امروزه، بسیاری از شرکت‌ها دریافتند که اغلب مشکلات نرم افزاری، تکنیکی هستند و مهندسی نرم افزار متفاوت با سایر مهندسی‌ها است؛ زیرا محصولات نرم‌افزاری فکری بوده اما محصولات سایر مهندسی‌ها و دیگر علوم ملموس و فیزیکی هستند. در مرکزیت هر مهندسی، اندازه‌گیری وجود دارد که روشی بر پایه استانداردها یا قراردادهای شناخته شده است. اگر به اندازه‌گیری کارایی سیستم، میزان کارآمد بودن سازمان یا شرکت یا حتی داده‌هایی که مورد استفاده قرار می‌گیرند نپردازیم، مسلما امکان کنترل روند کار را نخواهیم داشت که در نتیجه، اعمالی که به منظور پیشرفت انجام می‌شوند تنها بر مبنای حدس و تخمین خواهند بود. چرا که تنها ثبت اطلاعات امروز است که امکان مقایسه را فراهم می‌آورد. همان‌طور که پاتریک هنری نیز با گفته‌ی خود، این مطلب را تایید می‌کند: " هیچ راهی را برای قضاوت در آینده نمی‌شناسم، مگر با استفاده از گذشته".

اندازه‌گیری را می‌توان در سرتاسر پروژه نرم‌افزاری با هدف بهبود بخشیدن فرآیند نرم‌افزاری، کنترل کیفیت، ارزیابی بهره‌وری و کنترل پروژه به کار برد. استفاده از اندازه‌گیری به مهندس نرم‌افزار در ارزیابی محصولات فنی یاری رسانده و وی را قادر به تصمیم‌گیری تاکتیکی به موازات پیشرفت پروژه می‌سازد. در این میان، متریک‌های نرم‌افزاری گستره وسیعی از اندازه‌گیری‌ها را برای نرم‌افزارهای کامپیوتری در بر می‌گیرند. می‌توان چنین بیان نمود که تلاش جهت بهبود فرآیندهای یک سازمان، بدون ابزار مناسب اندازه‌گیری، یعنی متریک‌های صحیح و استانداردهای عملکرد، کاری بیهوده است. به گفته فنتون و فلگ:" شما نمی‌توانید کنترل و پیش‌بینی کنید، آنچه را که نمی‌توانید اندازه‌گیری کنید". لذا ما برآنیم در این فصل با مفاهیم اولیه و پایه مهندسی نرم‌افزار آشنا شویم، بنابراین در ادامه یک مقدمه‌ای کوتاه بر تعریف و اهمیت آن داریم و در ادامه به بحث و بررسی کلی در خصوص نکات قابل توجه می‌پردازیم.

۲.۱ تعریف مهندسی

مهندسی یادگرفتن قوانین، اصول، استانداردها و نظم دادن به روشی است که به توان از آن استفاده کرد. مهندسی شامل متدها، سبک ها، روش‌ها، شناسایی مراحل، عوامل مختلف و چگونگی استفاده از آنها است. همچنین لغت مهندسی به این معنی است که می‌توان هر کاری را مجددا تکرار کرد و فرآیند می‌تواند تکرار شود به همین دلیل کیفیت برتر خواهد بود یا به عبارتی از قبل شروع، تضمین شده است.

۳.۱ تعریف مهندسی نرم افزار

به کارگیری روش سیستماتیک، اصولی و قابل بررسی برای توسعه، اجرا و پشتیبانی نرم‌افزار می‌باشد. مهندسی نرم‌افزار یک تکنیک و روش است، منطق و فلسفه‌ای در زمینه تولید، نظارت و کنترل فرآورده‌های نرم‌افزاری می‌باشد و دارای مراحل مختلف تحلیل، طراحی، پیاده سازی، تست، نگهداری و بهینه‌سازی است، که این مراحل دارای ترتیب خاصی می‌باشند و همچنین مهندسی نرم‌افزار وابسته به زمان، جغرافیا، سرمایه، زبان، سلیقه و ..... نیست، روش استانداردی است که از طریق مستندات مربوطه اعلام می‌شود و این امر باعث شده است که محصول به عوامل دیگر وابسته نشود و سلیقه فردی در آن به حداقل رسد.

۴.۱ اهمیت مهندسی نرم افزار

مهندسی در تولید نرم افزار حیاتی است زیرا کار به صورت گروهی انجام می شود، نظرها مختلف و متغیرها زیاد است و در نهایت همه باید به یک جواب برسند. در مهندسی نرم‌افزار فکرها مهم است و بر اساس تجربه های گوناگون به راه حل های مختلفی می رسند که به طور سیستماتیک به راه حل مورد نظر رسیدن، مهم می‌باشد و تنها روش قابل قبول مستندسازی مراحل کار است. مهندسی نرم افزار در مورد

ایجاد مستندات نمی باشد و مربوط به بالا بردن کیفیت است که کیفیت بهتر باعث تکرار کمتر می شود و تکرار کمتر در کار باعث زمان تحویل سریعتر می‌گردد.

اصل مهندسی نرم‌افزار می‌گوید که کار باید به طور مهندسی انجام شود و وابسته به تکنولوژی نباشد در حقیقت با چرخه حیات محصول وابسته به تکنولوژی نخواهید بود. دو عامل مهم در مهندسی نرم افزار هزینه و زمان می باشد و یک مهندس خوب باید برآورد دقیق از هزینه و زمان بدست آورد.

۵.۱ جایگاه مهندسی نرم افزار

مهندسی نرم‌افزار یک ساختار قانونمند و استاندارد در اختیار مهندسان قرار می‌دهد که بتوانند توسط آن کارهای گروهی را مخصوصا در ابعاد وسیع و برای موضوعات پیچیده و موضوعات حساس انجام دهند. در یک ساختار مهندسی نرم افزار می‌توان فهمید که دقیقا ورودی و خروجی به هر بخش کار چه چیزی است و نیز می‌توان مشخص کرد که در چه مقاطعی نباید فعالیت‌های دیگر روی کار تحویلی انجام پذیرد. هر چه مهندسی نرم‌افزار به صورت فنی‌تر و مهندسی‌تر اتفاق بیافتد مسلما می‌توان در مراحل مختلف کار خودتان با زمان‌بندی و بودجه‌های خاصی مقایسه کنید که برای پروژه در نظر گرفته شده است تا از آنها چه از نظر زمانی و بودجه تخطی نشود.

مهندسی نرم‌افزار جایگاه مهمی در موضوع کنترل پروژه های نرم‌افزاری دارد. اگر مهندسی نرم افزار به صورت جامع استفاده شود قابلیت نگهداری و اطمینان نرم افزار بیشتر خواهد شد و مهندسی نرم‌افزار در نهایت شما را به سمت و سوی می‌کشاند که وابستگی به زمان، تکنولوژی، نیروی انسانی، شرایط محیطی و ابزارها نخواهید داشت و ریسک های پروژه ها به شدت کاهش پیدا می‌کنند و همچنین نسبت به سرمایه-گذاری هایی که در این راستا انجام می شود مطمئن تر خواهید بود و برگشت سرمایه ها را بهتر می‌توانید تضمین کنید.

۶.۱ تعریف نرم افزار خوب مهندسی ساز و خواص آن

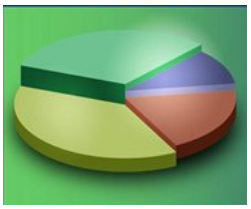
منظور از نرم‌افزار خوب مهندسی ساز تولید کردن محصول با روش و هزینه مناسب می باشد.

یک نرم افزار خوب مهندسی‌ساز باید دارای خواص زیر باشد:

- قابل نگهداری باشد، در طراحی باید نسبت به نگهداری دید داشت، زیرا بیشترین هزینه مربوط به نگهداری نرم افزار می باشد.
- قابل اطمینان باشد، به این معنی که کارایی آن با گذشت زمان کاهش پیدا نکند و بتوان به آن اطمینان داشت.
- کارا باشد، یعنی متناسب با نیازهای عملیاتی باشد.
- کاربری پسند باشد، یعنی قابل بهره برداری باشد.
- امنیت داشته باشد
- نیازهای عملکردی را برآورده کند
- با سیستم های قبلی سازگار باشد
- قابلیت تست داشته باشد
- صحت داشته باشد
- آموزش آن راحت باشد

۷.۱ دلایل شکست پروژه های نرم‌افزاری

گزارش CHAOS نظرات جالبی در مورد موفقیت برخی پروژه ها و شکست برخی دیگر در اختیار ما قرار می‌دهد. با توجه به بررسی انجام شده مشارکت مشتریان پروژه، پشتیبانی مدیریت اجرایی و بیان روشنی از نیازمندی‌ها، مهم‌ترین عوامل برای موفقیت یک پروژه فناوری اطلاعات بیان شده است از سوی دیگر به نظر می‌رسد که عدم مشارکت کاربر و کافی نبودن امکانات دو عامل اساسی برای شکست پروژه‌ها هستند.



شکل ۱-۱: گزارش Chaos در سال‌های ۲۰۰۴،۲۰۰۶،۲۰۰۹.

- درصد میانگین زمان افزون بر برنامه‌ریزی ۴۵٪
- درصد میانگین هزینه‌های افزون بر بودجه ۶۳٪
- درصد دارا بودن ویژگی‌های تعیین شده ۶۷٪

گزارش نشان می دهد:

- ۳۲ ٪ موفق (اتمام پروژه در زمان پیش‌بینی شده، با بودجه تعیین‌شده، دارای ویژگی‌ها و کارایی‌های پیش‌بینی شده)
- ۴۴٪ دچار چالش (اتمام بیش از زمان تعیین شده، هزینه‌ها افزون بر بودجه، نداشتن عملکردهای تعیین شده)
- ۲۴٪ معیوب (هرگز استفاده نشده)

منظور از پروژه‌های موفق آنهایی هستند که در موعد مقرر و با بودجه تعیین شده کامل شده‌اند، دچار چالش آنهایی که با بودجه و زمان بیشتر و ویژگی‌ها و کارایی کمتر به پایان رسیده‌اند و معیوب پروژه‌هایی که پیش از اتمام لغو شده‌اند.

دلایل اصلی

- فقدان ابزارهای یکپارچه نرم‌افزار
- جدایی کسب‌و کار از فناوری
- ارتباطات ضعیف تیمی
- نبود پیگیری و مدیریت
- نبود تعادل بین پیش‌بینی و بهره وری

۸.۱ فرآیند تولید نرم‌افزار

همان‌طور که در دنیای مهندسی برای تولید هر محصولی ترتیب مراحلی وجود دارد، مهندسی نرم‌افزار نیز از این قاعده مستثنی نیست، به عبارت دیگر جهت تولید محصولات نرم‌افزاری چرخه‌ای وجود دارد که روند تولید نرم‌افزار را به صورت قانونمند به مراحلی تقسیم کرده است. در چرخه حیات محصول هر مرحله‌ای دارای تعریف مشخص است و همچنین فعالیت‌هایی تعریف می‌شوند که باید در هر مرحله انجام شوند. در این چرخه حیات ورودی‌ها و خروجی‌هایی که به هر بخش داده و از آن گرفته شوند کاملا مشخص شده است و برای هر مرحله نقاط کنترلی تعریف می‌شوند که به واسطه آنها میزان پیشرفت کار، از لحاظ کمی و کیفیت مشخص خواهد شد. یک گام یا مرحله، یک واحد از چرخه حیات نرم‌افزار است. تولید نرم‌افزار نیز مانند هر محصول دیگری دارای یک سری مراحل است. این مراحل اگر بر طبق اصول مهندسی نرم‌افزار طراحی گردند، محصول تولید شده محصولی مهندسی خواهد بود. انتخاب مراحل تولید یک محصول بستگی به محصول دارد. اما مراحل زیر برای یک چرخه تولید پیشنهاد شده‌اند

(شکل ۱-۲). هرچند این مراحل همیشه ثابت و یکسان نیستند، یعنی با توجه به متدولوژی[1] که انتخاب می‌شود حجم این مراحل و و میزان فعالیت کمتر یا بیشتر می‌شود.

از نظر مدیریتی بهتر است تعداد مراحل کمتر باشد، هر مرحله خروجی مرحله قبل را به خوبی بشناسد و تا حد ممکن بین مراحل بازخورد[2] یا بازگشت به مرحله قبل نباشد.

شکل ۱-۲ : مراحل چرخه تولید محصول نرم‌افزاری

- فاز پیشنهاد[3]

در این فاز ایده اصلی تولید محصول که این ایده می‌تواند بر اساس نیاز موجود در بازار که از طرف یک کارفرما مطرح می‌شود و یا نیاز به یک محصول در آینده، باشد. مفاهیم و خصوصیات، همچنین فعالیت‌هایی که باعث کاهش هزینه محصول جدید می‌گردد همراه با بازار تجاری که برای آن محصول وجود خواهد داشت، مطرح می‌شود.

- فاز بررسی نیازمندی‌ها[4]

1 Methodology
2 Feedback
3 Proposal
4 Investigation

در این فاز امکان‌پذیر بودن تولید محصول با شرایط ذکر شده بررسی می‌شود و برنامه تولید محصول جهت سازگاری با بازار و استراتژی‌های فنی مورد اطمینان قرار می‌گیرد. خروجی این فاز مستند امکان‌سنجی[1] است که مورد توافق کارفرما و مدیریت چرخه تولید محصول قرار می‌گیرد.

- فاز طراحی[2]

در این فاز امکان‌سنجی به عمل آمده در مرحله قبل نیازمندی‌های عملیاتی را مشخص می‌کند و این نیازمندی‌ها به برنامه‌ها و طرح‌ها منجر می‌شود. این فاز خود از دو بخش دیگر تشکیل شده است: یک بخش طراحی سطح بالا معماری کل سیستم را مشخص می‌کند و به عبارتی در این بخش طراحی منطقی محصول انجام می‌شود، اینکه محصول به چه ماژول‌هایی نیاز دارد و این ماژول‌ها چه ارتباطی با هم دارند. پارامترهایی که بین این ماژول‌ها رد و بدل می‌شود نیز در این مرحله مشخص می‌شود. خروجی این مرحله، مستند مشخصات معماری[3] است. در بخش بعدی که طراحی جزئی نام دارد، هر ماژول تعریف شده در مرحله قبل جداگانه بررسی می‌شود و کلیه عملیاتی که باید در آن ماژول انجام شود، به تفصیل بیان می‌شود یعنی شرح عملکردی محصول به طور شفاف بیان می‌شود. خروجی این مرحله مستند، مشخصات‌طراحی، طراحی پایگاه‌داده و طراحی نرم‌افزار است.

- فاز توسعه[4]

در این مرحله کدهای کامپایل نشده محصول در یک محیط پیاده‌سازی می‌شود و پس از کامپایل، خطاهای آن گرفته می‌شود.

در این فاز عمل انسجام و تست محصولات بر اساس طرح و برنامه مورد توافق انجام می‌شود. در پایان این فاز یک نسخه اولیه از محصول که خطاهای اولیه آن گرفته شده است برای آزمایش آماده است.

- فاز ارزیابی داخلی[5]

در این فاز محصول نرم‌افزاری و کد نهایی تولید شده، کنترل کیفیت می‌شود. خطایی که در این مرحله کشف می‌شود ممکن است ریشه در عدم شناخت نیازمندی‌ها داشته باشد و یا در عدم آنالیز صحیح مدل یا ریشه در طراحی محصول داشته باشد. به هر صورت هرچه عمق خطا بیشتر باشد، هزینه‌های جبران آن بیشتر است. در این فاز با دو مساله اعتبارسنجی[6] و بازبینی[7] مواجهیم. بازبینی به معنی انجام درست عملیات خواسته شده است،

1 Feasibility Specification Document
2 Design
3 Architectural Specification Document
4 Development
5 Internal Qualification
6 Validation
7 Verification

یعنی اگر ورودی معتبر به بخش مربوطه بدهیم، خروجی‌اش مطابق با نیازمندی‌های خواسته باشد. بحث اعتبارسنجی درباره معتبر بودن ورودی‌هاست. اگر نیازمندی‌ها را درست نشناخته باشیم آنگاه ورودی‌های ما به محصول اشتباه خواهند شد و در نتیجه خروجی محصول مطابق نیازهای کاربر نخواهد بود.

- فاز جایگزینی کنترل شده[1]

در این فاز بر فرآیند تولید و بر فرآورده‌هایی که طی فرآیند توسعه تولید می‌کنیم، مانند مدل‌ها، نمودارها و دیاگرام‌ها نظارت می‌شود. این فرآیند موازی با فرآیند تولید اجرا می‌شود. زمانی که فرآیند، کیفیت لازم را داشته باشد احتمال اینکه در کنترل کیفیت محصول دچار خطا شود، کم می‌شود؛ به این مفهوم که اگر محصول به روشی استاندارد تولید شود احتمال رخداد خطا در پایان تولید آن کمتر است. فاز جایگزینی کنترل شده به کنترل استاندارد بودن فازهای تولید محصول نظارت دارد.

- فاز دسترسی عمومی[2]

فاز نهایی‌شدن و اتمام مراحل عملیاتی شدن است. در این فاز پشتیبانی از محصول، تصحیح و بهبود محصول در صورت نیاز انجام می‌شود.

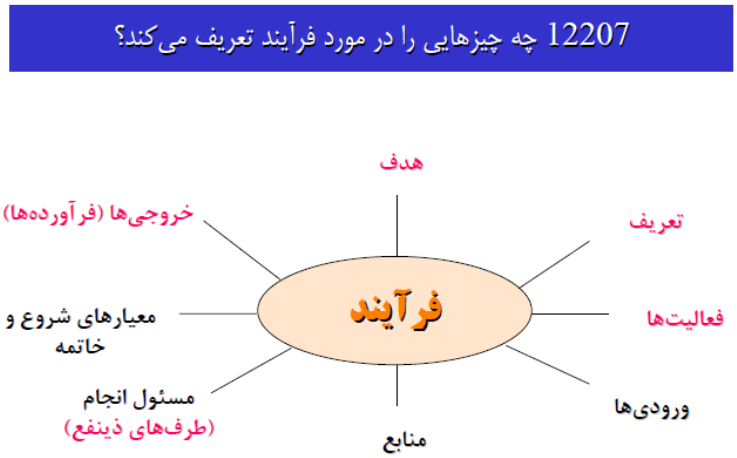
- فاز منسوخ کردن[3]

اگر وارد این فاز شوید، نیاز به محصول را کم می‌کنید و بر مرور زمان و بر حسب دوره‌های مختلف محصول اهمیت کمتر و نگهداری و پشتیبانی آن کاهش پیدا می‌کند.

۹.۱ استاندارد ایزو ۱۲۲۰۷

استاندارد ISO/IEC 12207:2008 یک استاندارد بین‌المللی برای مهندسی نرم‌افزار است که در آن فعالیت‌ها و وظایف مرتبط با چرخه حیات نرم‌افزار از ابتدا تا انتها مشخص شده است. در این استاندارد، فرآیندهای مهندسی‌نرم‌افزار بدین‌صورت تعریف می‌شوند: "مجموعه‌ای از فعالیت‌ها[4] که هر کدام مجموعه‌ای از وظایف[5] را شامل می‌شوند و هر کدام از این وظایف به صورت اعمالی تعریف می‌شوند که یک سری ورودی

1 Controlled Release
2 General Availability
3 Obsolescence
4 Activities
5 Tasks



شکل۱-۳: فرآیند ایزو ۱۲۲۰۷

را به یک سری خروجی تبدیل کنند". این استاندارد، فرآیندهای چرخه حیات نرم‌افزار را توصیف می‌کند اما هیچ مدل خاصی را برای چرخه حیات نرم‌افزار و همچنین هیچ متریکی را برای ارزیابی و اندازه‌گیری کارآیی پیاده‌سازی فرآیندها و وظایف پیشنهاد نمی‌کند. این استاندارد مستقل از فن‌آوری‌های روز مهندسی نرم‌افزار است و به عبارت دیگر فرآیندهای چرخه حیات در این استاندارد حول پرسش "چه چیزی باید انجام شود؟" هستند نه "چگونه باید انجام شود؟".

این استاندارد از دو فرآیند اصلی و هفت زیر فرآیند تشکیل شده است(شکل۱-۴):

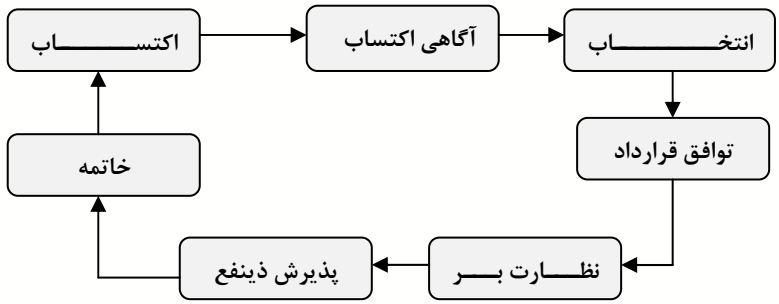
شکل۱-۴: فرآیند ها و زیر فرآیندهای ایزو ۱۲۲۰۷ سال ۲۰۰۸

فرآیندهای محتوایی سیستم

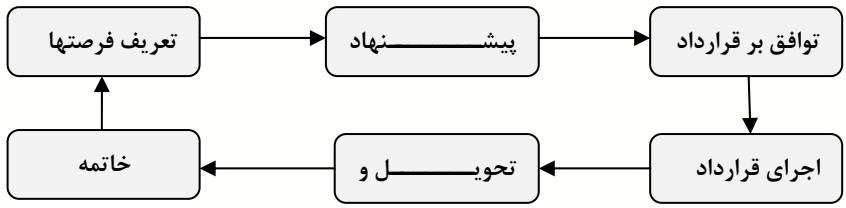
جدول۱-۱ : فرآیندهای توافق استاندارد ایزو۲۰۷ ۱۲

فرآیندهای توافق: این فرآیند از دو فعالیت اکتساب و عرضه تشکیل شده است.	
اکتساب	هدف: تعیین اینکه آیا محصول یا خدمات، نیاز و اهداف مشتری را برآورده می‌کنند
عرضه	هدف: تهیه محصولی یا خدماتی مطابق خواسته‌های مشتری

فعالیت‌های اکتساب و عرضه شامل وظایف زیر هستند که در شکل های (۱-۵) (۱-۶) :



شکل۱-۵: وظایف، فعالیت اکتساب



شکل۱-۶: وظایف، فعالیت عرضه

جدول ۱-۲ : فرآیندهای توانمند سازی- پروژه سازمانی استاندارد ایزو۲۰۷ ۱۲

فرآیندهای توانمند سازی- پروژه سازمانی	
مدیریت مدل چرخه حیات	هدف: تعریف نگهداری، تضمین دسترس‌پذیری سیاست‌ها، فرآیندهای چرخه حیات، مدل‌های چرخه حیات و رویه‌ها برای استفاده سازمان با رعایت استاندارد

	وظایف: پیاده سازی فرآیند، ارزیابی فرآیند، بهینه‌سازی فرآیند
مدیریت زیر ساخت	هدف: فراهم آوردن ساختاری جهت پشتیبانی سازمان و اهداف پروژه در کل چرخه حیات
	وظایف: پیاده‌سازی فرآیند، استقرار زیرساخت، نگهداری زیرساخت
مدیریت سبد پروژه	هدف: در نظر گرفتن تمام نیازمندی‌های پروژه مبتنی بر اهداف سازمانی
	وظایف: شروع پروژه(شناسایی- اولویت‌بندی- انتخاب و ایجاد فرصت‌های جدید)، ارزیابی سبد، اتمام پروژه
مدیریت منابع‌انسانی	هدف: تضمین فراهم آوردن اشخاص با تجربه و مهارت در اجرای چرخه حیات برای رسیدن به اهداف مشتری- پروژه- سازمان
	وظایف: شناسایی مهارت، توسعه مهارت، تدارک و اکتساب مهارت، مدیریت دانش
مدیریت کیفیت	هدف: رسیدن به رضایت مشتری
	وظایف: مدیریت کیفیت (ایزو 9001:2000، ISO/IEC 9003: 2004)، فعالیت اصلاحی مدیریت کیفیت

جدول۱-۳ : فرآیندهای پروژه استاندارد ایزو۲۰۷ ۱۲

فرآیند های پروژه	
برنامه‌ریزی پروژه	هدف: تعیین حوزه مدیریت پروژه و فعالیت‌های فنی، شناسایی خروجی فرآیند، وظایف پروژه، منابع مورد نیاز جهت انجام دادن وظایف پروژه
	وظایف: آغاز پروژه، طرح‌ریزی پروژه، فعالیت‌های پروژه
ارزیابی و کنترل پروژه	هدف: تضمین این که پروژه بر طبق برنامه‌ها، زمانبندی و بودجه اجرا می‌شود
	وظایف: نظارت بر پروژه، کنترل پروژه، ارزیابی پروژه، اتمام
مدیریت تصمیم	هدف: این فعالیت پاسخ گو در زمان مواجه با تصمیم‌گیری است
	وظایف: برنامه‌ریزی تصمیم، آنالیز تصمیم، پیگیری و رهگیری تصمیم
مدیریت ریسک	هدف: شناسایی، تحلیل تهدیدها(فرآیندی ادامه‌دار برای ریسک‌های سیستماتیکالی سراسر چرخه حیات سیستم یا نرم افزار)
	وظایف: برنامه‌ریزی مدیریت ریسک، مدیریت پروفایل ریسک، آنالیز ریسک، رفتارشناسی ریسک، پایش ریسک، ارزیابی فرآیند مدیریت ریسک

مدیریت پیکربندی	هدف: تعریف استراتژی‌های مدیریت پیکربندی و ایجاد خط‌مشی‌های پیکربندی
	وظایف: برنامه‌ریزی مدیریت پیکربندی، اجرای مدیریت پیکربندی
مدیریت اطلاعات	هدف: تولید، جمع‌آوری، انتقال، حفظ، بازیابی و انتشار اطلاعات
	وظایف: برنامه‌ریزی مدیریت اطلاعات، اجرای مدیریت اطلاعات
اندازه گیری	هدف: جمع‌آوری، تحلیل و ارائه گزارش از ارتباط اطلاعات در توسعه محصول و فرآیندهای پیاده سازی درون سازمان برای پشتیبانی موثر مدیریت و تولید با کیفیت
	وظایف: طرح‌ریزی اندازه‌گیری(تعریف معیارهایی برای جمع‌آوری،آنالیز و گزارش)، کارایی اندازه‌گیری، ارزیابی اندازه‌گیری

جدول ۱-۴: فرآیندهای تکنیکی(فنی) استاندارد ایزو۲۰۷ ۱۲

فرآیندهای تکنیکی(فنی)	
تعریف نیازمندی‌های ذینفعان	هدف: تعریف الزامات سیستم برای ارائه خدمات به کاربران در محیط تعریف شده ذینفعان
	وظایف: شناسایی ذینفعان، شناسایی نیازمندی‌ها، ارزیابی نیازمندی‌ها، توافق با نیازمندی‌ها، ثبت نیازمندی‌ها
آنالیز نیازمندی‌های سیستم	هدف: تبدیل نیازمندی‌های ذینفعان به مجموعه‌ای از نیازمندی‌های تکنیکی سیستم که راهنمایی برای طراحی سیستم خواهد بود
	وظایف: توصیف نیازمندی‌ها، ارزیابی نیازمندی‌ها
طراحی معماری سیستم	هدف: شناسایی نیازمندی‌های سیستمی که باید به عناصر سیستم تخصیص یابند
	وظایف: ایجاد معماری، ارزیابی معماری
پیاده سازی	
یکپارچگی سیستم	
تست وضعیت سیستم	هدف: تضمین کند پیاده سازی همه نیازمندی‌های سیستم جهت انطباق تست می‌شود
نصب نرم‌افزار	هدف: نصب محصول نرم‌افزاری در محیط هدف
	وظایف: نصب نرم‌افزار
پشتیبانی پذیرش نرم‌افزار	هدف: کمک به مشتری در رسیدن به اعتماد در برآورده شدن نیازهایش
	وظایف: پشتیبانی پذیرش نرم‌افزار

اجرای نگهداری	هدف: اجرای نرم‌افزار در محیط عملیاتی و حمایت از مشتری
	وظایف: آماده سازی برای اجرا، فعال سازی و وارسی اجـرا، اسـتفاده اجرایی(سیسـتم باید در محیط مورد نظر بر طبق مستند کـاربر اجـرا شـود)، پشـتیبانی مشـتری، حـل مشکلات اجرا
نگهداری نرم افزار	هدف: فراهم آوردن پشتیبانی مقرون به صرفه محصول نرم‌افزاری قابل تحویل
	وظایف: پیاده‌سازی، آنالیز تغییر و مشکل، پیاده‌سازی تغییر، بازرسـی/ قبـول تغییـرات، انتقال
مصرف نرم‌افزار	

فرآیندهای خاص نرم‌افزار

جدول۱-۵: فرآیندهای پیاده‌سازی نرم‌افزار استاندارد ایزو ۱۲۲۰۷

فرآیندهای پیاده‌سازی نرم‌افزار	
پیـــاده‌ســـازی نرم‌افزار	هدف: تحویل یک عنصر خاص از سیستم همانند نرم‌افزار یا ارائه خدمات
	وظایف: استراتژی پیاده‌سازی نرم‌افزار
آنالیز نیازمندی‌های نرم‌افزار	
طراحی معماری نرم‌افزار	
طراحی جزئیات نرم‌افزار	
ساخت نرم‌افزار	
یکپارچگی نرم‌افزار	
تست وضعیت نرم‌افزار	

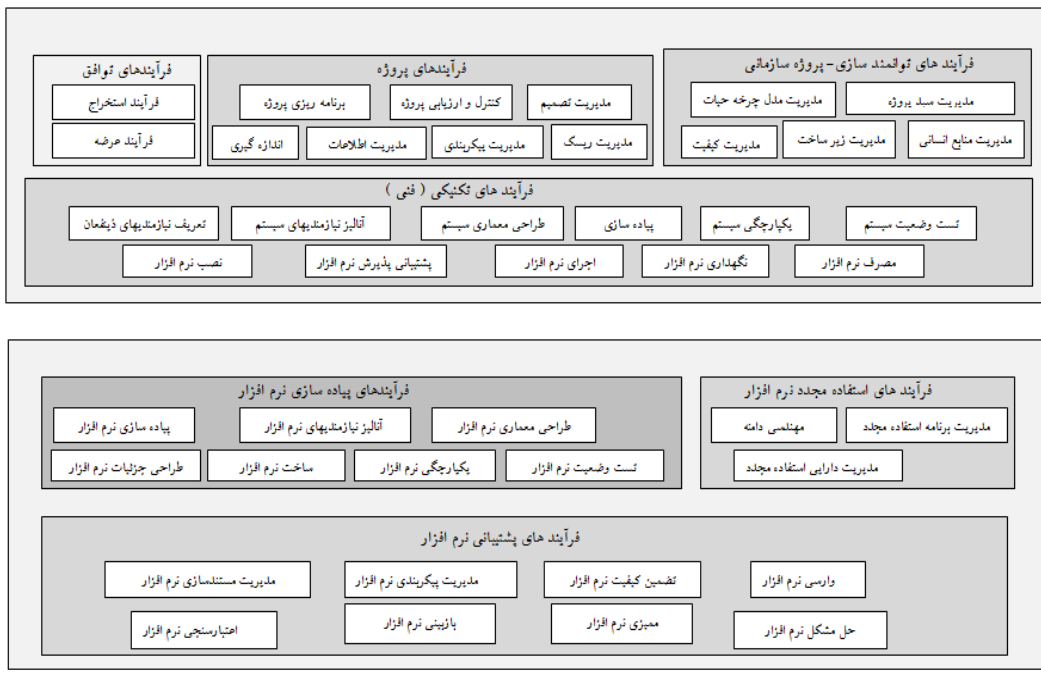
جدول ۱-۶: فرآیندهای پشتیبانی نرم‌افزار استاندارد ایزو ۱۲۲۰۷

فرآیندهای پشتیبانی نرم‌افزار	
مدیریت مستندسازی نرم‌افزار	هدف: توسعه و نگهداری اطلاعات ثبت شده نرم‌افزاری بدست آمده از هر فرآیند
	وظایف: پیاده‌سازی فرآیند، توسعه و طراحی، تولید، نگهداری
مدیریت پیکربندی	وظایف: شناسایی پیکربندی، کنترل پیکربندی، حسابداری وضعیت پیکربندی، ارزیابی پیکربندی، مدیریت محصول

تضمین کیفیت نرم افزار	وظایف: تضمین محصول، تضمین فرآیند، تضمین کیفیت سیستم‌ها
وارسی نرم‌افزار	
اعتبارسنجی نرم‌افزار	
بازبینی نرم‌افزار	
ممیزی نرم‌افزار	
حل مشکل نرم‌افزار	

جدول ۱-۷: فرآیندهای استفاده مجدد نرم‌افزار استاندارد ایزو ۱۲۲۰۷

فرآیندهای استفاده مجدد نرم‌افزار
مهندسی دامنه
مدیریت دارایی استفاده مجدد
مدیریت برنامه استفاده مجدد

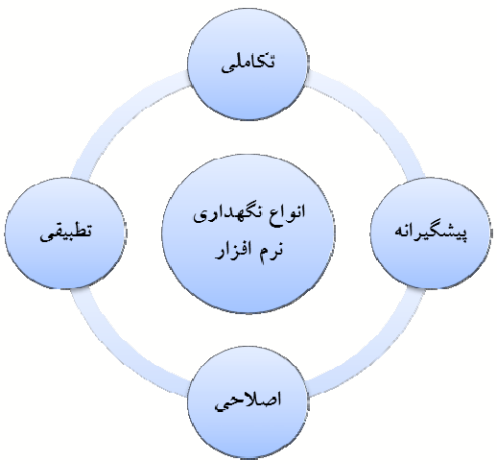


شکل۱-۷: ساختار استاندارد ایزو ۱۲۲۰۷:۲۰۰۸

۱۰.۱ اهمیت نگهداری نرم‌افزار

نگهداری نرم‌افزار یکی از فازهای چرخه حیات نرم‌افزار است. بنابر تعریف IEEE، نگهداری نرم‌افزار فرآیند تغییرات یک سیستم یا مولفه نرم‌افزاری پس از تحویل آن به منظور رفع خطا، بهبود کارایی یا سایر خصوصیات و تطبیق آن با تغییرات محیطی است. این تعریف اذهان دارد که نگهداری نرم‌افزار فعالیتی است که پس از تحویل سیستم به مشتری یا کاربر انجام می‌شود و شامل تمام فعالیت‌هایی است که سیستم را در حالت عملیاتی نگاه داشته و نیازمندی‌های کاربر را برآورده می‌سازد. درفاز نگهداری نرم‌افزار انواع مختلفی از نگهداری‌ها انجام می‌شود که عبارتند از:

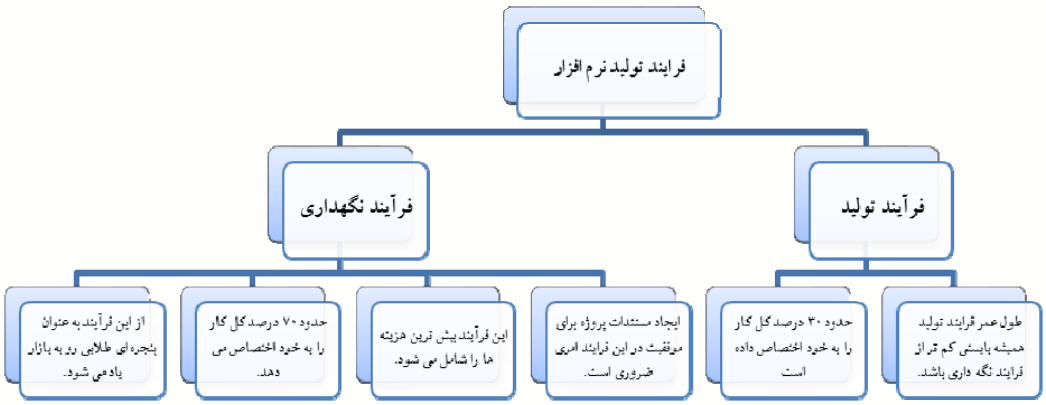
- نگهداری تطبیقی[1]: ویرایش نرم‌افزار جهت حفظ عملکرد آن در محیط جدید.
- نگهداری اصلاحی[2]: تعمیر اشتباهات موجود در نرم‌افزار بدون تغییر یا اضافه کردن کارکرد نرم‌افزاری جدید.
- نگهداری تکمیلی[3]: بهبود کارایی نرم‌افزار و توسعه عملکرد نرم‌افزار.
- نگهداری پیشگیرانه[4]: تغییرات انجام شده روی سیستم به منظور جلوگیری از رخداد اشتباهات و خطاهای آتی و بهبود ساختار و قابلیت نگهداری.



شکل ۱-۸: انواع نگهداری نرم افزار

1 Adaptive
2 Corrective
3 perfective
4 preventive

اما مهمترین چالش‌های موجود در نگهداری نرم‌افزار عبارتند از: فهم برنامه، تحلیل تاثیر و تست برگشتی. در زمان اعمال تغییر در بخشی از نرم‌افزار، درک کامل از عملکرد، ساختار و رفتار سیستم تحت تغییر، مورد نیاز است. در نتیجه فرد نگهدارنده‌ی سیستم زمان بسیار زیادی را صرف خواندن کد و مستندات مربوط به نرم‌افزار می‌کند. تخمین‌های انجام شده نشان می‌دهد که در حدود ۵۰٪ تا ۹۰٪ از مدت زمان صرف شده جهت نگهداری نرم‌افزار، جهت فهمیدن آن صرف می‌شود. همچنین بررسی و تحلیل تاثیر آن تغییر با هدف به حداقل رساندن میزان واکنش‌های ناخواسته انجام می‌شود. در ادامه تغییرات اعمال شده باید مورد تست قرار گیرد و تمام این موارد مدت زمان و منابع قابل توجهی را بخود اختصاص می‌دهد.



شکل ۱-۹: اهمیت فرآیند نگهداری در فرآیند تولید نرم افزار

۱۱.۱ تخمین

پروژه‌ی نرم‌افزاری موفق، پروژه‌ای است که در قالب هزینه و زمانی معین و از پیش تعیین شده به انجام برسد. نرم‌افزار، کاری تولیدی به شمار می‌رود که هزینه‌ی عمده‌ی آن نیروی کارآزموده ومتخصص است. بنابراین مهم‌ترین ابزار یک پروژه نرم‌افزاری و به طور تقریبی بخش اعظم هزینه‌های آن به نیروی کار متخصص درگیر در آن مرتبط است. سوال این است که چگونه می‌توان زمان و هزینه‌ی یک پروژه نرم‌افزاری را تخمین زد. ماهیت خلاق پروژه‌های نرم‌افزاری و انتزاعی بودن آن تخمین هزینه و زمان انجام آن‌ها را بی‌نهایت مشکل می‌کند. روش‌های متداول تخمین زمان و هزینه خود اساسا انتزاعی است با این همه هنوز هم تخمین پروژه امری لازم و ضروری محسوب‌می‌شود. روش‌های مختلفی در تخمین و برآورد حجم فعالیت‌های لازم در انجام یک پروژه نرم‌افزاری در جامعه نرم‌افزار ارائه شده است. فارغ از این‌که از چه روشی در تخمین زمان و هزینه یک پروژه نرم‌افزاری استفاده می‌شود، مهم آن است که بدون وجود

اطلاعات کافی در زمینه حوزه و دامنه سیستم و قابلیت‌ها و توانایی‌های آن و هم‌چنین شرایط محیطی و فرهنگی تیم تولید نرم‌افزار و پیچیدگی‌های تکنیکی آن، برآورد واقع‌بینانه پروژه کاری دور از دسترس می‌نماید.

تحقیقات زیادی نشان داده‌اند که یکی از مهمترین موضوعات در توسعه کیفیت نرم‌افزار تخمین درست است. به علاوه تخمین درست وابسته به درجه تودر تویی ذاتی نرم افزار یعنی پیچیدگی می‌باشد. تاکنون سنجش پیچیدگی مبتنی بر کد بوده که می توانست مشکلاتی را به همراه داشته باشد. اگر بتوانیم سنجش پیچیدگی را در همان فاز نیازمندی‌ها مبتنی بر مستند مهندسی نیازمندیهای IEEE که شامل مشخصات نیازمندیهای نرم‌افزار است بررسی کنیم می‌توانیم از اتلاف هزنیه و نیروی انسانی که از مهمترین مسائلند، جلوگیری کنیم.

۱.۱۱.۱ روش‌های تخمین نرم افزار

برآورد هزینه یکی از مهمترین مراحل مدیریت نرم‌افزاری می باشد بسیاری از پروژه ها در حین انجام با شکست مواجه می شوند و یا هزینه هایی که برای انجام پروژه صرف می شود بسیار بالا و گران است بنابراین علاوه بر کار تکینیکی برنامه نویسان یک مهندس نرم افزار در طول مدیریت نرم‌افزاری بایستی تخمین درستی از میزان هزینه ها داشته باشد. منظور از برآورد پروژه نرم افزاری تخمین هزینه و زمان می-باشد.بخش‌های اصلی هزینه پروژه عبارتند از: هزینه های سخت افزاری، هزینه های آموزشی و مسافرت، هزینه کار انجام شده.

تکنیک‌های برآورد هزینه

بر اساس نظریه بوهم هفت تکنیک برای برآورد نرم‌افزار ارائه شده است:

۱- مدلسازی الگوریتمی هزینه: در این روش از مدل هزینه ها در پروژه های گذشته استفاده می‌شود در این تکنیک درستی برآوردهای پروژه نرم افزاری به عوامل زیر بستگی دارد.

- درستی برآورد معیارها.
- توانایی بدست آوردن کار زمان و هزینه ها از برآورد معیار پروژه.
- انعکاس توانایی های سیستم نرم افزاری.
- پایداری خواسته های پروژه و محیط توسعه نرم افزاری.

۲- قضاوت خبرگان: در این روش از فرد یا افراد مخصوص و با تجربه در زمینه کاربردی مشابه پروژه، درخواست می شود برای هزینه پروژه برآوردی ارائه دهند.

۳- برآورد از طریق مقایسه و مشابه سازی: در این روش برآورد کننده پروژه شبیه به پروژه خود را پیدا کرده و با استفاده از تفاوت‌های دو پروژه و معیارهای آنها میزان هزینه را با توجه به این تفاوت‌ها بررسی و تعیین می‌کند.

۴- قیمت برای برنده شدن: در این روش قیمت بر اساس بودجه مشتری تعیین می‌شود برآورد کننده با توجه به مشتری مبلغی را حدس زده و بعد از کمی زیاد یا کم کردن، با مشتری به توافق می‌رسند.اما این روش باعث می شود که کار تکنیکی برنامه نویس به علت قیمت تعیین شده تحت تاثیر قرار بگیرد و نتواند بازدهی اصلی را داشته باشد.

۵- برآورد پایین به بالا: در این روش نرم‌افزار به وظایفی تقسیم بندی می‌شود که طی آن برآورد کننده مانند یک درخت تا پایین ترین سطح این وظایف را گسترش می‌دهد به طوری که در آخرین سطح، هر وظیفه توسط یک نفر قابل انجام خواهد بود بعد از ایجاد این درخت از بالا به پایین نتیجه تلاش‌های انجام شده از پایین به بالا محاسبه و با هم جمع بسته می‌شود تا برآورد کلی بدست آید.

۶- برآورد بالا به پایین و مدل‌های پارامتری : این مدل یک مدل ساده پارامتری می‌باشد میزان تلاش لازم برای پیاده سازی یک پروژه به عواملی که عمدتاً مربوط به مشخصات سیستم نهایی است بستگی دارد یک مدل پارامتری معمولاً دارای فرمول زیر می باشد.

$$\text{Effort} = \text{parameter valus} \times \text{constant value}$$

به عنوان مثال مقدار پارامتر می تواند از نوع KLOC و مقدار ثابت برابر ۲۵ روز باشد.

۷- قانون پارکینسون[1] : کار در طول معین توسعه می‌یابد بطوری که زمان داده شده را پر کند. این بدین معناست که منابع در دسترس، هزینه را تعیین می‌کند نه ارزیابی عینی. بعنوان مثال اگر قرار است نرم‌افزاری در عرض ۱۲ ماه تحویل داده شود و ۵ نفر در دسترسند، کار مورد نیاز، ۶۰ نفر – ماه برآورده می‌شود یعنی با فرض یک هدف سهل الوصول، کارکنان با دشواری کمتری کار می‌کنند.

1 Parkinson Rule

۱۲.۱ پیچیدگی

با پیشرفت نرم‌افزارهای کامپیوتری، افزایش نیاز کاربران به نرم‌افزارهای قدرتمندتر، کاربردی‌تر و با قابلیت‌های بیشتر مشهود است. از این‌رو نرم‌افزارها بزرگ‌تر و پیچیده‌تر شده و چنین تحولی فعالیت‌های توسعه و خصوصاً نگهداری نرم‌افزار را دشوارتر و توسعه‌دهندگان را با چالش‌های جدیدی مواجه می‌کند. دشواری کار، زمان بیشتری برای انجام آن می‌طلبد، در این زمان منابع بیشتری بکار گرفته می‌شود و منابع بیشتر به معنی تحمیل هزینه‌های بیشتر خواهد بود.

تاخیر در تحویل، سرریز در هزینه‌ها و بودجه‌های تعیین‌شده، کاهش کیفیت و قابلیت اطمینان و نارضایتی کاربر، از نتایج افزایش پیچیدگی[1] در نرم‌افزار است. بنابراین کاهش پیچیدگی، قلب توسعه‌ی نرم‌افزار بوده و یکی از عوامل موثر در موفقیت یا شکست پروژه محسوب می‌شود و بدلیل تاثیرات آن بر صفات مهم نرم‌افزار، از جایگاه ویژه‌ای در مباحث مهندسی نرم‌افزار برخوردار است. بنابراین در عصری که در آن قابلیت‌های نرم‌افزار پیوسته در حال افزایش است، واضح است که اگر بخواهیم هزینه‌ها بدون صدمه دیدن کیفیت نرم‌افزار کنترل شوند باید مهارت کافی در تشخیص منابع پیچیدگی، مدیریت آن و تا حد امکان حذف پیچیدگی را داشت.

برای نیل به این مقصود محققان به دنبال رابطه‌ای بین ویژگی‌های نرم‌افزار و مشکلات و دشواری‌های کار توسعه‌ی نرم‌افزار بوده‌اند. حاصل این تلاش‌ها ارائه معیارهایی برای اندازه‌گیری پیچیدگی نرم‌افزار بوده است. هر کدام از این معیارها ابعادی از پیچیدگی نرم‌افزار را در نظر داشته ولی عمدتاً بر روی کد متمرکز شده‌اند.

اما منظور از «پیچیدگی» چیست؟ پیچیدگی نرم‌افزار یک موضوع وسیع در مهندسی نرم‌افزار است و از سال ۱۹۷۶ محققان زیادی تلاش کرده‌اند تا تعریف دقیقی برای پیچیدگی نرم‌افزار پیدا کنند، اما توافقی عمومی بر روی چگونگی تعریف آن وجود ندارد با این‌حال بیشتر تعاریف مبتنی بر دیدگاه زیوز از پیچیدگی 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. ولی این واقعیت که پیچیدگی نرم‌افزار در طی فازهای مختلف ایجاد می‌شود، تعریف و اندازه-گیری آن را سخت می‌کند.

ترکیب پیچیدگی حاصل از تحلیل مسئله، طراحی و کدنویسی در یک عدد بسیار مشکل است. به همین دلیل محققان پیچیدگی را از جنبه‌های مختلفی مورد بحث قرار می‌دهند.

پیچیدگی نرم‌افزار را می‌توان به صورت سطوح پیچیدگی برای بخش‌های مختلف یک سیستم نظیر پیچیدگی الگوریتم، پیچیدگی کد، پیچیدگی زبان برنامه‌نویسی، پیچیدگی اتصال پیمانه‌ها، پیچیدگی

1 Complexity

نیازمندی‌ها یا پیچیدگی طراحی تعریف کرد. در واقع، پیچیدگی نرم‌افزار می‌تواند به عنوان شاخصی برای مراحل مختلف تولید و نگهداری سیستم استفاده شود.

در زمان توسعه نرم‌افزار، توسعه‌دهندگان نرم‌افزار نمی‌توانند چیزی را تولید کنند که به طور کامل مشخص نشده است. عدم شناسایی صحیح نیازمندی‌ها در فاز اول از چرخه حیات نرم‌افزار منجر به توسعه اشتباه و ناصحیح محصول و بروز پیچیدگی شده و این امر منجر به اتلاف منابع ارزشمندی خواهد شد. بنابراین بین پیچیدگی و تشریح نیازمندی‌های نرم‌افزار که خروجی فاز اول از چرخه حیات نرم‌افزار محسوب می‌شود، ارتباطی وجود دارد.

تشریح نیازمندی‌های نرم‌افزار حوزه توسعه سیستم را می‌سازد و مدلی از نیازهای ذینفعان[1] بوده و ویژگی‌های سیستمی که باید ایجاد شود را تشکیل می‌دهد. مشخصات این نیازمندی‌ها توسط مهندسان نیازمندی تهیه و به توسعه‌دهندگان نرم‌افزار تحویل داده می‌شود.

در این بین هدف فرآیند مهندسی نیازمندی‌های نرم‌افزار[2] تأمین بیشترین رضایت مشتری و کاربر با برآوردن انتظارات آن‌ها و تحویل نرم‌افزار در زمان و بودجه برآورد شده است. جهت دستیابی به رضایت مشتری و موفقیت پروژه در فرآیند مهندسی نرم‌افزار، می‌توان رابطه‌ی زیر را سرلوحه کار قرار داد:

محصول آماده + کیفیت خوب + تحویل در زمان‌بندی و بودجه‌بندی تعیین شده = رضایت مشتری

موفقیت هر پروژه نرم‌افزاری با میزان تحقق اهداف و نیازهای ذینفعان سنجیده می‌شود. مهندسی نیازمندی‌های نرم‌افزار فرآیند کشف این اهداف است. یک هدف[3] یک مقصود سطح بالاست که سیستم تحت بررسی باید به آن دست یابد. یک نیازمندی توصیف یک سرویس یا محدودیت سیستم است که یک کاربر برای رسیدن به هدف وضع می‌کند. بنابراین یک نیازمندی مشخص می‌کند که یک هدف در یک سیستم پیشنهادی چگونه باید حاصل شود.

۱۳.۱ مهندسی نیازمندی‌ها

کیفیت نرم‌افزار یک معیار بسیار مهم در موفقیت یا شکست پروژه است. یک محصول نرم‌افزاری موفق است اگر اهدافی را که به منظور آن توسعه یافته است را برآورده کند. اما سوال اینجاست که هدف از نرم‌افزار چگونه تعریف می‌شود، یک طراح چگونه بداند انتظار طراحی چه چیز را دارد؟ یک توسعه‌دهنده چگونه بداند انتظار توسعه چه چیز و آزمون‌گر چگونه بداند انتظار آزمودن چه چیز را دارد؟ مستند نیازمندی‌ها

1 Stakeholder

2 Software Requirement Engineering

3 Goal

پاسخی به تمام سوالات مطرح شده است. مستندات نیازمندی‌ها، هدف و انتظارات کاربر را بیان می‌کنند و می‌توان آن‌را حیاتی‌ترین مستند چرخه حیات نرم‌افزار دانست. بنابراین نیاز به فرآیندی جهت فراهم کردن و پالایش نیازمندی‌ها است و چنین فرآیندی، فرآیند مهندسی نیازمندی‌ها می‌باشد.

طبق نظریه زیوز «مهندسی نیازمندی‌ها شاخه‌ای از مهندسی نرم‌افزار است که در ارتباط با اهداف واقعی، توابع و محدودیت‌های سیستم‌های نرم‌افزاری است. همچنین درباره رابطه بین این فاکتورها با مشخصات دقیق رفتار نرم‌افزار و تکامل آن‌ها در طول زمان می‌باشد».

این تعریف ماهیت مهندسی نیازمندی‌ها را به عنوان یک نظام چندبعدی انعکاس می‌دهد و به دلایل متعددی جالب است. اولاً، اهمیت «اهداف واقعی» را که موجب توسعه سیستم نرم‌افزاری است را پررنگ می‌کند. این موارد، مسائل مربوط به «چرا» و «چگونگی» یک سیستم را نشان می‌دهد. دوماً این تعریف به «مشخصات دقیق» اشاره دارد. این موارد پایه تحلیل نیازمندی‌ها، تایید اعتبار اینکه آیا واقعاً نیازمندی‌ها آن چیزی است که ذینفع خواسته است، تعریف و تعیین آنچه که طراحان باید بسازند و تصدیق اینکه آن‌ها در زمان تحویل نیز همچنان به درستی عمل می‌کنند و در نهایت اشاره به مشخصه «تکامل در طول زمان»، بر وجود تغییر در جهان و نیز بر استفاده مجدد تاکید دارد. در واقع مهندسی نیازمندی‌ها تنها وابسته به مباحث و مسائل تکنیکی نیست بلکه مباحث مدیریتی، سازمانی، اقتصادی و اجتماعی را نیز بازگو می‌کند.

مهندسی نیازمندی‌ها شامل دو فرآیند اصلی توسعه نیازمندی‌ها و مدیریت نیازمندی‌هاست. توسعه نیازمندی‌ها شامل تمام فعالیت‌های درگیر در استخراج، تحلیل، مشخص‌سازی و تأیید اعتبار نیازمندی‌ها است. همچنین تعریف و یکپارچه‌سازی نیازمندی‌های کسب‌وکار، نیازمندی‌های کاربر و نیازمندی‌های سطح محصول نرم‌افزاری را نیز در بر می‌گیرد. بیشتر فعالیت‌های توسعه محصول در فازهای نیازمندی‌ها و فازهای اولیه چرخه حیات رخ می‌دهد.

از اوسط سال ۱۹۷۰، مهندسی نیازمندی‌ها بعنوان یک شاخه متمایز کاری و تحقیقی شناخته و تعاریفی برای آن ارائه شد. در ابتدا مهندسی نیازمندی‌ها بصورت مرحله اول مهندسی نرم‌افزار تعریف شد. در طول سال‌ها با تکامل تحقیقات مربوط به مهندسی نیازمندی‌ها، تعریف مهندسی نیازمندی‌ها وسعت یافت بطوری که کل چرخه‌ی حیات را در بر گرفت.

مهندسی نیازمندی‌ها می توانند به چند مرحله تقسیم شوند:

شکل۱-۱۰: فرآیند مهندسی نیازمندی‌ها

۱.۱۳.۱ دلایل اهمیت مهندسی نیازمندی‌ها

مهندسی نیازمندی‌ها فرآیندی است که در آن ذینفعان و نیازمندی‌هایشان تعیین می‌شود. پس از تعریف اهداف (کارهایی که باید انجام شود) سیستم شناسایی شده و پس از نهایی شدن این مراحل، تمام نیازمندی‌ها مستند شده و برای اعتبارسنجی، تصحیح و سپس پیاده‌سازی آماده هستند. ظاهراً مهندسی نیازمندی‌ها روند بسیار ساده‌ای است اما در واقع حساس‌ترین فرآیند چرخه حیات نرم‌افزار است. دلایل اهمیت مهندسی نیازمندی‌ها عبارتند از:

- شناسایی نیازهای ذینفعان: اولین ریسکی که پروژه با آن مواجه است این است که نیازهای مشتری و کاربر (ذینفعان) برآورده نشود. اگر نیازمندی‌ها ناکامل باشند، توسعه‌دهندگان محصولی ناقص را تحویل خواهند داد که نارضایتی مشتری، از دست دادن بازار، از دست دادن سود و سرمایه و اتلاف منابع را در پی دارد. از طرفی، اگر یک گروه از ذینفعان فراموش شوند یا کاربران در فرآیند نیازمندی‌ها درگیر نباشند، فاصله‌ای بین نیازمندی‌ها و محصول تولید شده خواهد بود و محصول نرم‌افزاری تولید شده رضایت مشتری را تأمین نمی‌کند.

- هزینه: نیازمندی‌ها در کنترل هزینه پروژه نرم‌افزاری نقش حساسی را ایفا می‌کنند. نیازمندی‌هایی که با تاخیر کشف می‌شوند منجر به افزایش چشمگیر هزینه‌های نرم‌افزار می‌شوند. بنابراین برای اجتناب از عدم‌اطمینان، شناسایی تمام نیازمندی‌های ممکن و معتبر، قبل از شروع مرحله پیاده‌سازی و طراحی، الزامی است. بر اساس تحقیقات انجام گرفته ۷۰ تا ۸۵ درصد از هزینه‌های دوباره‌کاری در پروژه‌های نرم‌افزاری ناشی از خطاهای نیازمندی است. مطالعات نشان داده‌اند که هزینه رفع نقص حاصل از نیازمندی درصورتی که تا بعد از تحویل پیدا نشود گاهی بیش از ۱۰۰ برابر هم خواهد شد و این یعنی هدر دادن منابع و حرکت به سمت شکست پروژه. هدف اصلی مهندسی نیازمندی‌ها اطمینان از تعریف نیازمندی‌های صحیح و واقعی است. مهندسی نیازمندی‌ها توافق ذینفعان روی نیازمندی‌ها در طی فرآیند توسعه را تضمین می‌کند.

- زمان: تغییرات مداوم روی نیازمندی‌ها بر روی زمان‌بندی پروژه نرم‌افزاری تاثیر می‌گذارد. نیازمندی‌هایی که با تاخیر و دیر شناسایی می‌شوند و تغییرات روی نیازمندی‌ها عمدتاً موجبات تاخیر در زمان‌بندی را به دنبال دارد و بنابراین جهت اجتناب از چنین تاخیراتی، باید نیازمندی‌ها در مراحل ابتدایی نهایی شوند. ناپایداری نیازمندی‌ها به تغییر نیازمندی‌ها بعد از توافق اولیه بر سر آن‌ها اشاره دارد. بعضی از این تغییرات به دلیل پالایش درک توسعه‌دهندگان نرم‌افزار و بعضی دیگر به دلیل تغییر محیط یا تغییر نیازهای مشتری در طول زمان اتفاق می‌افتد که در هر دوره از پروژه طبیعی است. اگر نیازمندی‌ها به دلیل نوشتن ضعیف یا مبهم مشخصات اولیه از قلم افتاده باشند ناپایداری در

نیازمندی‌ها اتفاق خواهد افتاد. این نوع از ناپایداری در نیازمندی‌ها با فرآیند مهندسی نیازمندی‌های خوب جلوگیری خواهند شد.

- معیار موفقیت و شکست: برای اندازه‌گیری موفقیت یا شکست پروژه باید معیارهایی موجود باشد؛ زیرا نمی‌توان نرم‌افزار را با روش‌های اندازه‌گیری دنیای واقعی (مانند اینچ، سانتی‌متر و غیره) اندازه-گیری کرد. بنابراین موفقیت و شکست پروژه تا حد زیادی به نیازمندی‌ها بستگی دارد. یک نرم‌افزار موفق است اگر تمام نیازمندی‌های ذینفعان را برآورده سازد و در غیر این صورت با شکست مواجه شده است. نیازمندی‌های ناقص، متضاد و مبهم هیچگاه نرم‌افزار را به سمت صحیح و معتبری هدایت نمی‌کنند.

هفت چالش در مهندسی نیازمندیها عبارتند از :

۱- چالش اول : درک دامنه مسئله

۲- چالش دوم : شناسایی پروسه کسب وکار

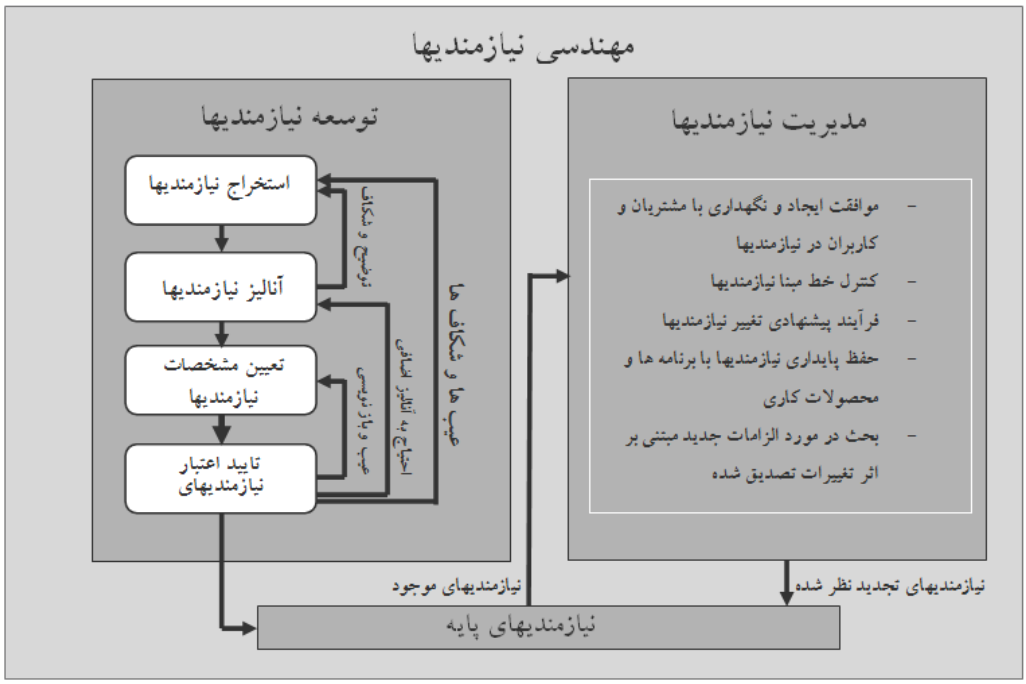
۳- چالش سوم : به وجود آمدن نیازمندی‌ها

۴- چالش چهارم : آنالیز و استنباط نیازمندی‌ها

۵- چالش پنجم : تایید اعتبار، تصدیق و پی گیری

۶- چالش ششم : مرور مجدد نیازمندی‌ها

۷- چالش هفتم : مدیریت و کنترل تغییر



شکل۱-۱۱: ساختار مهندسی نیازمندی‌ها

۱۴.۱ پیچیدگی و مهندسی نیازمندی‌ها

موفقیت هر پروژه نرم‌افزاری با میزان برآورده شدن اهداف مورد نظر سنجیده می‌شود. مهندسی نیازمندی‌های نرم‌افزار فرآیند کشف این اهداف است که با شناسایی ذینفعان و نیازهای آن‌ها و مستند کردن این نیازها صورت می‌گیرد.

پیچیدگی در فاز تولید نرم‌افزار تلاش مورد نیاز برای تحلیل و تشریح نیازمندی‌ها، طراحی، کد، آزمون و اشکال‌زدایی سیستم را به شدت تحت تاثیر قرار می‌دهد و در فاز نگهداری دشواری تصحیح خطا و میزان تلاش مورد نیاز برای تغییر بخش‌ها و پیمانه‌های مختلف نرم‌افزار را مشخص می‌کند.نیازمندی‌ها شالوده‌ی فرآیند توسعه نرم‌افزار را تشکیل می‌دهد. شالوده سست منجر به سقوط کل ساختمان می‌شود و ضعف در مستند نیازمندی‌ها (که خروجی فرآیند مهندسی نیازمندی‌هاست) منجر به سقوط و شکست پروژه نرم-افزاری می‌شود. در واقع کیفیت محصول نهایی وابستگی شدیدی به صحت توصیف نیازمندی‌ها دارد، از این‌رو باید بر بهبود مراحل اولیه توسعه تاکید داشت و بهبود به معنای شناسایی صحیح نیازمندی‌هاست و این اولین قدم در موفقیت پروژه است. تحقیقات اذهان دارند که ۴۴٪ تا ۸۰٪ از خطاها در فاز نیازمندی‌ها

وارد می‌شود. بنابراین اگر خطاها در فاز نیازمندی‌ها شناسایی نشود منجر به توسعه اشتباه و ناصحیح محصول و در نتیجه منجر به پیچیدگی و اتلاف منابع ارزشمندی خواهد شد.

باید در نظر داشت که با آنکه فرآیند مهندسی نیازمندی‌ها در یک فضای به‌هنجار و عادی صورت می‌گیرد، با این‌حال نتایج فرآیند مهندسی نیازمندی‌ها تحت تاثیر فاکتورهای مختلفی قرار می‌گیرد. این فاکتورها تاثیرات مثبت یا منفی بر فرآیند مهندسی نیازمندی‌ها و بالتبع بر پیچیدگی نرم‌افزار دانند. بعضی از مهم‌ترین و متداول‌ترین این فاکتورها عبارتند از:

- **روش‌ها و متدولوژی‌ها:** فرآیند مهندسی نیازمندی‌ها تحت تاثیر روش‌ها و متدولوژی‌های مورد استفاده جهت پیشبرد فرآیند قرار می‌گیرد. از آنجائیکه روش‌های مختلف بر موارد مختلفی تمرکز دارند، استفاده از روش‌های مختلف در طی فرآیند می‌تواند منجر به نتایج مختلفی شود. مثلاً تشریحِ رسمی نهایی سیستم، در صورت استفاده از تحلیل ساختیافته کاملاً متفاوت با تشریح رسمی همان سیستم با تحلیل شیءگرا خواهد بود.

- **ابزار:** تشریح نهایی وابسته به ابزار مورد استفاده درطی فرآیند است. به عنوان نمونه اگر از یک ابزار استدلال یا منطق برای نمایش رسمی استفاده شود، امکان شناسایی ناسازگاری‌ها وجود دارد و در غیر اینصورت این امکان فراهم نیست و ممکن است موارد ناسازگاری در تشریح الزامات باقی بماند.

- **جنبه‌های اجتماعی:** محیط اجتماعی تیم مهندسی نیازمندی‌ها بر نتایج کاری آن‌ها تاثیرگذار خواهد بود. مثلاً کشمکش و تعارضات بین افراد مختلف منجر به ارتباط غیرموثر خواهد شد. چنانچه افراد در محیط احساس خوبی داشته باشند، خروجی کار آن‌ها نیز بهتر خواهد بود.

- **مهارت‌های شناختی و ذهنی:** افراد مهارت‌های ذهنی مختلفی دارند. چناچه تیم فرآیند متشکل از افراد باهوش و کارآمد باشد نتیجه کار نیز بهتر است.

- **فشارهای مالی:** فشارهای مالی، منابع (افراد، پول، ابزار و غیره) را محدود می‌کند. این اصل که منابع بیشتر منجر به نتیجه بهتری خواهد شد همیشه صحیح نیست، اما چنانچه منابع در دسترس کمتر از حد مشخصی باشند، قطعاً کیفیت فرآیند را تحت تاثیر قرار می‌دهند. متاسفانه مدیران در خصوص تخصیص منابع مورد نیاز به فرآیند مهندسی نیازمندی‌ها، تصمیم‌گیری نمی‌کنند در حالیکه تخصیص زمان و منابع کافی در فاز نیازمندی‌ها، موفقیت یا شکست یک پروژه را تعیین می‌کند.

صرف زمان بیشتر روی فرآیند مهندسی نیازمندی‌ها منجر به صرف زمان کمتری جهت فعالیت‌های مجدد می‌شود، بنابراین بطور کلی زمان مورد نیاز برای فرآیند توسعه کاهش می‌یابد. مدیریت صحیح فرآیند

مهندسی نیازمندی‌ها، کل فرآیند پروژه را بهبود داده و آن‌را تسریع می‌کند و منجر به افزایش رضایت مشتری و کاهش پیچیدگی و هزینه توسعه سیستم خواهد شد.

۱۵.۱ اندازه‌گیری

اندازه‌گیری راهی را برای ارزیابی وضعیت برنامه یا پروژه در طول فرآیند توسعه فراهم می‌آورد تا بتوان با بهره‌گیری از آن، بر مشکلات موجود در پروژه غلبه کرده و در صورت نیاز، اقدامات اصلاحی را انجام داد. به منظور درک بهتر این مفهوم، ابتدا تعاریفی از اندازه‌گیری ارائه شده است.

اندازه‌گیری اساس تمام رشته‌های مهندسی است و مهندسی نرم‌افزار نیز از این قاعده مستثنی نیست. اندازه‌گیری به ما این امکان را می‌دهد تا از طریق فراهم‌آوردن مکانیزمی برای ارزیابی هدفمند، به بینش و تفکری راه پیدا کنیم. در صورت عدم اندازه‌گیری، قضاوت‌ها بصورت ذهنی انجام می‌شود و این کار یک روش علمی و قابل‌قبول نمی‌باشد. اندازه‌گیری به منظور بهبود و پیشرفت انجام می‌شود و هنگام جمع‌آوری اطلاعات کمیتی، موانع، عوامل ریشه‌ای و نواقص شناسایی شده و فرصت‌های بهبود کیفیت محصول و عملیات پردازشی فراهم می‌شود.

۱.۱۵.۱ اهداف اندازه‌گیری

باید هر فعالیت اندازه‌گیری در راه دستیابی به اهداف تعیین شده باشد. اهداف اندازه‌گیری نرم‌افزار را می‌توان به این صورت بیان کرد:

- **درک وفهم**[1]: اندازه‌گیری به درک آنچه که در طول فاز توسعه و نگهداری نرم‌افزار رخ داده است، کمک می‌کند. اندازه‌گیری باید به منظور استخراج مدل فرآیندها و بررسی ارتباط بین پارامترهای فرآیند انجام شود. این امر منجر به فهم و بهبود پروژه‌های نرم‌افزاری می‌شود.

1 Understanding

- **شناسایی زود هنگام مشکلات:** اندازه‌گیری امکان شناسایی مشکلات و تصحیح آن‌ها را با استفاده از استراتژی مدیریت ممکن می‌سازد و در واقع منتظر وقوع مشکلات نمی‌شود. چنانچه مشکلات هرچه زودتر شناسایی شوند در منابع سازمان صرفه جویی شده است.

- **تخمین و پیش‌بینی:** اندازه‌گیری منجر به برنامه‌ریزی و تخمین بهتری خواهد شد. حجم دوباره‌کاری‌ها مهمترین دلیل شکست در سرریز بودجه تخمین زده شده است. همچنانکه تعداد عیب‌ها[1] افزایش می‌یابد، هزینه تعمیر آن خطاها نیز افزایش می‌یابد.

- **کیفیت:** تعداد و فرکانس مشکلات و عیب‌ها با معکوس کیفیت نرم‌افزار متناسب است. این مورد یکی از اندازه‌گیری‌های مستقیم برای محصول نرم‌افزاری است.

- **برنامه‌ریزی[2]:** مقدار حجم کار، تعداد افراد و طریقه استفاده از فرآیندها، فاکتورهای اولیه‌ای هستند که در آماده‌سازی برنامه‌ریزی شرکت می‌کنند. اندازه‌گیری امکان کنترل آنچه که در پروژه رخ داده است را فراهم می‌کند و منجر به افزایش بهره‌وری تیم توسعه می‌شود. محققان ادعا دارند که چنانچه متریک‌ها، در اوایل توسعه نرم‌افزار بکار گرفته نشود منجر به افزایش خطاها در طی مراحل بعدی خواهد شد.

۱۶.۱ متریک‌های نرم‌افزاری

از آن‌جایی که وجه تمایز مهارت صنعتی و نظام مهندسی در آن است که صنعتگران از روش‌های کیفی استفاده می‌کنند ولی نظام مهندسی مبتنی بر روش‌های کمی است، بنابراین باید نتایج حاصل از اندازه-گیری را با مقادیر کمی بیان کنیم.

1 Defect

2 Schedule

مقادیر کمی همان متریک‌های نرم‌افزاری هستند. متریک‌های نرم‌افزاری، پارامترهایی برای اندازه‌گیری نرم‌افزار هستند که بدون آنها اندازه‌گیری معنا نخواهد داشت. همچنین متریک‌ها حلال تمام مشکلات نیستند؛ اما می‌توانند مدیران را در راستای بهبود فرآیندها، بهره‌وری و کیفیت یاری رسانند. به عبارتی، متریک‌های نرم‌افزاری، مهندسان و مدیران را به ادامه‌ی حیات سازمان در محیط کسب‌وکار امروز امیدوار می‌کنند، گرچه استفاده از متریک‌ها همیشه موفقیت آمیز نیست بلکه باید در قانون خود و با برنامه‌ریزی، بودجه‌بندی، منابع و تعهد مدیریت پیاده‌سازی شوند تا موفقیت را تضمین کنند.

متریک‌های نرم‌افزاری می‌توانند در طی یک پروژه نرم‌افزاری استفاده شوند تا به تخمین، کنترل کیفیت، ارزیابی قابلیت‌تولید و کنترل پروژه کمک کنند. همچنین می‌توانند به ارزیابی کیفیت محصولات تکنیکی و ارزیابی تصمیم‌گیری در طی پروژه کمک کنند. متریک‌ها، فعالیتی ادامه‌دار و قابل‌اجرا در کل پروژه هستند و در مدت زمانی طولانی جمع‌آوری شده و میزان پیشرفت کار را نشان می‌دهند. متریک‌ها دارای مکانیزمی حلقوی- افزایشی‌اند، چرا که با ارزش‌ترین اطلاعات زمانی حاصل می‌شوند که دنباله‌ای از داده‌ها را داشته باشیم، سپس داده‌های بدست آمده از متریک‌ها باید به صورت بازخورد در اختیار مدیر نرم‌افزار قرار گیرند تا وی با توجه به این اطلاعات، ایرادات موجود را پیدا کرده و راه‌حل‌هایی برای آن‌ها بیابد و نیز از بروز ایرادات جدید جلوگیری نماید. این امر موجب کشف نقایص قبل از ارائه به مشتری می‌شود.

متریک‌های نرم‌افزاری با اندازه‌گیری محصولات نرم‌افزاری و فرآیند توسعه‌ی آن‌ها در ارتباط هستند، از این روست که هدف اصلی متریک‌های نرم‌افزاری، شناسایی و اندازه‌گیری پارامترهای اساسی است که بر پیشرفت فرآیند نرم‌افزار اثر می‌گذارند. متریک‌ها، پروژه‌های پیچیده را قابل مشاهده و کنترل کرده و نیز با ارائه راهنمایی‌هایی در طول مراحل کار، ما را در رسیدن به اهداف سازمان یاری می‌نمایند.

در واقع متریک‌های نرم‌افزاری محدوده خاصی نداشته و می‌توانند برای هر محصول بنا به نوع و کیفیت آن، مقادیری کسب کنند. متریک‌های خوب نه تنها باید توصیف‌کننده‌ی پارامترها باشند، بلکه باید روند توسعه‌ی مدل‌های پیش‌بینی فرآیند و پارامترهای محصول را سهولت بخشند. البته مشکل اصلی این است که ما سعی در تخمین چیزی منطقی، نه فیزیکی داریم که تعریف کامل‌تر آن در فازهای بعدی چرخه حیات مشخص می‌شود. به علاوه پیچیدگی و چالش‌های تکنیکی موجود در پیاده‌سازی این خصوصیات نیز ناشناخته بوده یا در مراحل اولیه نادیده گرفته می‌شوند. در نتیجه می‌توان ادعا کرد که تخمین یک پروژه نرم‌افزاری مانند زدن هدفی متحرک است.

۱.۱۶.۱ متریک‌های شی گرایی[1]

ساختارهای مختلفی از مدل های شی گرایی را بررسی می کند. برای تعیین پیچیدگی این نوع کدها از روش CK که توسط Chidamber وKemerer بنا شده است استفاده می شود. در محاسبه این متریک از گزینه های زیر استفاده می شود.

WMC[2] : وزن کدها در هر کلاس را به روشهای مختلف مانند Loc، cc مشخص می‌کند.

DIP[3] : میزان عمق درخت وارثت در ساختار کلاسها را مشخص می کند.

NOC[4]: میزان و تعداد فرزندان یک کلاس را مشخص می‌کند تعداد بالای فرزندان نشان دهنده اهمیت والد و اهمیت تست کردن بیشتر والد را نشان می دهد.

CBO[5] : میزان استفاده کلاس‌ها از یکدیگر را اندازه گیری می‌کند بالا بودن این گزینه نشان دهنده پیچیدگی بیشتر، کاهش نگهداری و کاهش استفاده مجدد می‌باشد.

RFC[6] : اندازه مجموعه توابعی می باشد که به صورت بالقوه در هنگام دریافت پیام از اشیاء اجرا می‌شوند.

LCOM[7] : تعداد متدهای مختلفی در یک کلاس که با متغیرهایی همدیگر را مورد ارجاع قرار می‌دهند، محاسبه می‌کند. با افزایش LCOM پیچیدگی افزایش می یابد.

۱۷.۱ نتیجه گیری

با توجه به آنچه در این فصل خواندیم در می یابیم که هدف مهندسی نرم‌افزار تولید محصولات با کیفیت منطبق بر هزینه و زمان از پیش تعیین شده می‌باشد. لذا در جهت تحقق این هدف باید موارد زیادی را در نظر بگیریم. توجه به پیچیدگی زیرا تاخیر در تحویل، سرریز در هزینه‌ها و بودجه‌های تعیین شده، کاهش کیفیت و قابلیت اطمینان و نارضایتی کاربران از نتایج افزایش پیچیدگی در نرم‌افزار است بنابراین کاهش پیچیدگی قلب توسعه نرم‌افزار بوده و یکی از عوامل موفقیت یا شکست پروژه محسوب می‌شود. همچنین مهندسی نیازمندی‌ها فرآیندی است که در آن ذینفعان و نیازمندی‌هایشان تعیین می‌شود که حساس‌ترین فرآیند چرخه حیات نرم‌افزار محسوب می شود که باید در کل پروژه در نظر گرفته شود. باید به این مورد

---

1 Object Oriented Metrics

2 Weighted Methods per Class

3 Depth of inheritance tree

4 Number of children

5 Coupling between object classes

6 Response for class

7 Lack of Cohesion on Methods

هم توجه داشته باشیم که اندازه‌گیری راهی را برای ارزیابی وضعیت برنامه یا پروژه در طول فرآینـد توسـعه فراهم می‌آورد تا بتوان محصول با کیفیتی را تولید نمود. برای اندازه گیری ما نیاز به متریک‌های نرم‌افـزاری داریم که بدون آنها اندازه‌گیری معنا نخواهد داشت.

۱۸.۱ سوالات متداول

۱. تعریف مهندسی چیست؟

۲. تعریف مهندسی نرم‌افزار چیست؟

۳. تعریف و ویژگی‌های یک نرم‌افزار خوب و مهندسی ساز چیست؟

۴. دلایل شکست پروژه‌های نرم‌افزاری چیست؟

۵. فازهای تولید نرم‌افزار را نام برده و به طور مختصر توضیح دهید؟

۶. استاندارد ایزو ۱۲۲۰۷ دارای چند فرآیند و زیر فرآیند است؟

۷. انواع نگهداری نرم‌افزاری را نام ببرید؟

۸. تخمین چه تاثیری در موفقیت پروژه دارد و چه روش‌هایی برای آن به کار برده می‌شود؟

۹. تعریف پیچیدگی نرم‌افزار چیست؟

۱۰. مهندسی نیازمندی‌ها چه تاثیری بر کیفیت نرم‌افزار دارد؟

۱۱. دلایل اهمیت مهندسی نیازمندی‌ها چیست؟

۱۲. ۷ چالش مهندسی نیازمندی‌ها را نام ببرید؟

۱۳. ارتباط بین پیچیدگی و مهندسی نیازمندی‌ها در چیست؟

۱۴. اندازه‌گیری را تعریف و اهداف آن را نام ببرید؟

۱۵. متریک‌های نرم‌افزاری را تعریف نمایید؟

۱۹.۱ منابعی جهت مطالعه بیشتر

مجموعه مراجع زیر برای به دست آوردن جزئیات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.

- ناصر مدیری، فاطمه کشاورز کوهجردی، مفاهیم پیشرفته در مهندسی نرم‌افزار،۱۳۸۹
- ناصرمدیری، فاطمه‌دوامی، عصمت‌علی محمد ملایری، " متریک‌های نرم‌افزار،" شابک: ۰-۲-۶۱۶۹۳- ۶۰۰-۹۷۸، ۱۳۸۹.
- A Compilation of IEEE Standard Computer Glossaries, Institute of Electrical and Electronics Engineers, New York, 1990.
- P. Goodman “ Software Metrics: Best Practices for Successful IT Management,” Rothstein Associates Inc., 2004.
- Westfall, L., "Software Requirements Engineering: What, Why, Who, When, and How", Journal of Software Quality Professional, Vol. 7, No. 4, 2006, pp. 17-26
- Singh, Y., and Sabharwal, S., "A Systematic Approach to Measure the Problem Complexity of Software Requirement Specifications of an Information System", Journal of Information and Management Sciences, vol.15, issue1, 2004, pp.69-90
- ISO/IEC 12207:2008, International Standard, System and software engineering-Software life cycle processes.
- Raghu Singh, “An Introduction to International Standard ISO/IEC 12207 Software Life Cycle,” FAA, Washington DC, April 26, 1999.
- Day, G. “The Product Life cycle: Analysis and applications issues,” Journal of Marketing, Vol 45, autumn 1981.