مهندسی نرم‌افزار

تألیف: مهندس محسن کجباف

انتشارات پندار پارس

شابک : 978-600-8201-22-9 : ۲۵۰۰۰۰ ریال
شماره کتابشناسی ملی : ۴۴۳۵۲۹۵
عنوان و نام پدیدآور : مهندسی نرم‌افزار / تالیف محسن کجباف.
مشخصات نشر : تهران : پندار پارس، ۱۳۹۵.
مشخصات ظاهری : ۴۰۰ ص.: مصور، جدول، نمودار.
یادداشت : کتابنامه .
موضوع : نرم‌افزار -- مهندسی
موضوع : Software engineering
رده بندی دیویی : ۱/۰۰۵
رده بندی کنگره : QA۷۶/۱۳۹۵۷۵۸ ۹م۳ک/
سرشناسه : کجباف، محسن، ۱۳۶۵ -
وضعیت فهرست نویسی : فیپا

انتشارات پندارپارس
دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ info@pendarepars.com

نام کتاب : مهندسی نرم‌افزار
ناشر : انتشارات پندار پارس
تألیف : محسن کجباف
چاپ نخست : آبان ۹۵
شمارگان : ۵۰۰ نسخه
طرح جلد : رامین شکراللهی
چاپ، صحافی : روز
قیمت : ۲۵۰۰۰ تومان شابک : ۹-۲۲-۸۲۰۱-۶۰۰-۹۷۸

مقدمه

درس مهندسی نرم‌افزار، یکی از دروس مهم رشته‌های مهندسی کامپیوتر و فن‌آوری اطلاعات است. یکی از مشکلات اصلی این درس برای دانشجویان و علاقه‌مندان آن، گنگ بودن مفاهیم و سنگین بودن واژه‌های این درس می‌باشد. متاسفانه کتب مرجع این درس (کتاب مهندسی نرم‌افزار پرسمن و کتاب مهندسی نرم‌افزار سامرویل) تنها به بیان نظری مفاهیم اکتفا نموده‌اند و مثال عملی از مفاهیم این درس در فرایند تولید نرم‌افزار، بیان نکرده‌اند. برخی خوانندگان کتاب مهندسی نرم‌افزار، از غیرکاربردی بودن مفاهیم مهندسی نرم‌افزار سخن می‌گویند. حال آنکه این درس، عصاره تمام دروس رشته‌های مهندسی کامپیوتر و مهندسی فن‌آوری اطلاعات است: از درس پایگاه داده‌ها گرفته تا الگوریتم و فلوچارت.

مدتی بود، نوشتن یک کتاب مهندسی نرم‌افزار، برای رفع مشکل مذکور را مدنظر داشتم. با توجه به تدریس این درس در بیش از پنجاه سکشن درسی در دانشگاه‌های مختلف و مطالعه چند باره مراجع اصلی آن یعنی کتاب مهندسی نرم‌افزار پرسمن (ویراست هفتم و هشتم) و کتاب مهندسی نرم‌افزار سامرویل (ویراست نهم و دهم) و دیگر مراجع، بر آن شدم که اکنون زمان به عرصه ظهور نشاندن افکارم در زمینه مهندسی نرم‌افزار است. راه حل رفع این مشکل از دیدگاه اینجانب، درج مثال‌های عملی زیاد و متنوع است تا از این طریق خواننده بتواند درک کند که مفاهیم نظری در کتب، در عمل نیز کاربرد دارند. به باور نویسنده، درکنار هم قرار دادن مفاهیم نظری در کنار مثال‌های عملی در تولید نرم‌افزار، باعث درک و ملموس شدن مهندسی نرم‌افزار می‌شود.

اینجانب پیش از این اثر، کتاب تئوری و مسائل مهندسی نرم‌افزار سری شومز را ترجمه نموده‌ام. متاسفانه همه کتاب‌های مهندسی نرم‌افزار پیشین، مطالب درسی مهندسی نرم‌افزار را بدون مثال‌های عملی در تولید نرم‌افزار بیان نموده‌اند و باعث قابل لمس نبودن مطالب این درس برای دانشجویان و علاقمندان این درس شده است. برای نوشتن این اثر، از بیش از شصت مرجع لاتین استفاده شد. تلاش کردم برای هر مبحث مهندسی نرم‌افزار، از کتاب‌های تخصصی آن مبحث استفاده کنم. برای نمونه، برای فصل آزمون نرم‌افزار از دست‌کم ده کتاب معتبر رفرنس با عنوان آزمون نرم‌افزار استفاده کردم.

همچنین اینجانب بیشتر کتب مهندسی نرم‌افزار تالیفی و کنکوری (کتاب مهندسی نرم‌افزار آقای مهندس جهانی، کتاب مهندسی نرم‌افزار آقای دکتر رضوانی، کتاب مهندسی نرم‌افزار آقای مهندس زنجانی، کتاب مهندسی نرم افزار آقای دکتر خلیلی‌فر) را مطالعه کرده‌ام تا بتوانم با رویکردی متفاوت از آنها، اثر خود را به رشته تحریر دربیاورم. در چند مورد نیز از مثال‌های تالیفی این نویسندگان بزرگوار استفاده نموده‌ام که در اینجا بر خود لازم می‌دانم از زحمات این بزرگواران در تألیف اثرهای گران بهایشان، سپاسگزاری نمایم.

این کتاب سرفصل های وزارت علوم برای دروس "تحلیل و طراحی سیستم‌ها" و "مهندسی نرم‌افزار" دوره کارشناسی و "مبانی مهندسی نرم‌افزار" دوره کاردانی را پوشش داده است. همچنین با توجه به مطالب گفته شده

در زمینه تحلیل و طراحی شئ‌گرا و برنامه‌نویسی شئ‌گرا با زبان C++ ، می‌توان از کتاب حاضر برای دروس "طراحی سیستم‌های شئ‌گرا" و "برنامه‌نویسی شئ‌گرا" نیز استفاده نمود.

مبنای اصلی مطالب این اثر، دو مرجع اصلی این درس یعنی کتاب مهندسی نرم‌افزار پرسمن ویراست هشتم و کتاب مهندسی نرم‌افزار سامرویل ویراست دهم می‌باشد. همچنین بیشتر تمرین‌ها و مثال‌های کتاب پیش رو از کتاب‌های لاتین معتبر در زمینه مهندسی نرم‌افزار گردآوری شده‌اند.

در پایان از مدیریت محترم انتشارات پندارپارس، جناب آقای مهندس حسین یعسوبی که همواره برای ترجمه و تالیف کتاب، مشوق اینجانب بوده‌اند و زحمات زیادی در انتشار این اثر کشیده‌اند، کمال تشکر و سپاسگزاری را دارم. از همه مدرسان و دانشجویان محترم تقاضا دارم در صورت مشاهده ایراد در محتوای کتاب یا پیشنهاد درباره آن، نظر خود را از راه پست الکترونیکی M_kajbaf@yahoo.com با اینجانب در میان بگذارید.

محسن کجباف

پاییز ۱۳۹۵

درس تخصصی گرایش نرم‌افزار: تحلیل و طراحی سیستم‌ها

نام درس	تحلیل و طراحی سیستم‌ها		
نام درس به انگلیسی	Systems Analysis and Design		
نوع واحد	تخصصی	مهندسی کامپیوتر	۳ واحد
مقطع	کارشناسی		
هم‌نیازها			
پیش‌نیازها	برنامه‌سازی پیشرفته		
مطالب پیش‌نیاز	آشنائی کامل با یکی از زبان‌های برنامه‌نویسی C++ یا Java مفاهیم پایه شیءگرایی		
کتاب(های) مرجع	[1] L. D. Bentley and J. L. Whitten, *Systems Analysis and Design for the Global Enterprise*. 7th Edition, McGraw-Hill, 2007. [2] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Addison Wesley, 2004. [3] Roger S. Pressman, *Software Engineering: a Practitioner's Approach*. McGrawHill Inc., 7th Edition, 2011.		
اهداف درس	هدف از این درس آشنایی دانشجویان با مفاهیم تحلیل و طراحی سیستم‌های نرم‌افزاری است. در این درس دانشجویان با انواع سیستم‌های اطلاعاتی، چرخه حیات تولید و توسعه نرم‌افزار، روش‌های مختلف ایجاد نرم‌افزار، تحلیل و طراحی ساخت‌یافته و مفاهیم مدیریت پروژه آشنا می‌شوند.		
نتایج درس	دانشجویانی که این درس را با موفقیت پشت سر بگذارند بینش مناسبی در موارد زیر خواهند داشت: ۱- کاربرد فرآیند توسعه نرم‌افزار در تولید نرم‌افزار، ۲- انتخاب و به کارگیری ابزارهای توسعه نرم‌افزار، ۳- مستندسازی فرآورده‌های نرم‌افزاری به‌کمک زبان UML، ۴- تحلیل و طراحی سیستم‌ها به روش شیءگرا، ۵- اهمیت آزمون نرم‌افزار و طراحی آزمون.		
فهرست مباحث	۱- معرفی مهندسی نرم‌افزار و چالش‌های آن ۲- مدل‌های فرآیند توسعه نرم‌افزار و تفاوت آن‌ها ۳- روش‌های تحلیل و طراحی نرم‌افزار ۴- مهندسی نیازمندی‌ها و تحلیل سیستم ۵- طراحی سیستم و معماری نرم‌افزار ۶- ساخت نرم‌افزار ۷- مقدمه‌ای بر آزمون نرم‌افزار ۸- آشنایی مقدماتی با مدیریت پروژه و برنامه‌ریزی		
نرم‌افزارهای مورد نیاز	محیط برنامه‌نویسی در یک زبان شیءگرا (C++، Java، ...) – ابزار مدل‌سازی UML		
تکالیف پیشنهادی	۵ تکلیف دستی		
پروژه‌های پیشنهادی	یک پروژه کامل تحلیل و طراحی سیستم که تا حد امکان برگرفته از نیازهای واقعی است و طی چند مرحله در طول ترم کامل می‌شود.		
نمره‌دهی پیشنهادی	تکالیف	٪۱۰	
	پروژه	٪۳۰	
	آزمون‌ها	٪۶۰	
سایر مراجع	[1] David L. Olson, *Information System Project Management*. McGrawHill, 2004.		

درس تخصصی گرایش نرم‌افزار: مهندسی نرم‌افزار

نام درس	مهندسی نرم‌افزار		
نام درس به انگلیسی	Software Engineering		
نوع واحد	تخصصی	مهندسی کامپیوتر	۳ واحد
مقطع	کارشناسی		
هم‌نیازها			
پیش‌نیازها	تحلیل و طراحی سیستم‌ها		
مطالب پیش‌نیاز	مبانی تحلیل و طراحی سیستم‌ها – مدل‌سازی نرم‌افزار – برنامه‌نویسی شیءگرا		
کتاب(های) مرجع	[1] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*. McGrawHill, 7th Edition, 2011		
اهداف درس	هدف از این درس پرداختن به نکات مهندسی است که در کلیه مراحل تولید نرم‌افزار باید رعایت گردد. در این درس ابتدا تفاوت محصولی که به روش مهندسی تولید می‌گردد با محصولی که به روش هنری تولید می‌شود بیان می‌شود. سپس انتظاراتی که یک محصول مهندسی باید برآورده سازد تشریح می‌گردد. در ادامه درس با تاکید بر روش‌های مهندسی تولید از جمله مدل‌سازی، قابل اندازه‌گیری و ارزیابی بودن، درستی‌یابی و اعتبارسنجی محصولات بینابینی، مروری بر دست‌آوردهای علمی در این زمینه در کلیه مراحل تولید نرم‌افزار انجام می‌شود. با توجه به اینکه در درس‌های قبلی دانشجویان با مباحث توصیف صوری نیازها، اندازه‌گیری، تخمین و آزمون کمتر آشنا شده‌اند در این درس این فصول مورد تاکید بیشتر قرار می‌گیرد. در انتها فعالیت‌های حمایتی از جمله مدیریت پروژه، زمانبندی، مدیریت ریسک، مدیریت پیکربندی و تضمین کیفیت با تاکید بر تاثیر آن‌ها در تولید نرم‌افزار به صورت مهندسی مرور می‌شود.		
نتایج درس	دانشجویانی که این درس را با موفقیت پشت سر بگذارند بینش مناسبی در موارد زیر خواهند داشت: ۱- به‌کارگیری روش‌های مهندسی جهت ایجاد محصول با کیفیت ۲- دنبال کردن فرایندهای شناخته‌شده مهندسی نرم‌افزار ۳- به‌کارگیری روش‌های طراحی معماری نرم‌افزار ۴- آزمون نرم‌افزار در سطوح مختلف		
فهرست مباحث	۱- مقدمه‌ای بر مهندسی نرم‌افزار ۲- فرایندها و مدل‌های توسعه نرم‌افزار – توسعه مبتنی بر تکرار ۳- مروری بر تحلیل نرم‌افزار ۴- طراحی نرم‌افزار: اصول طراحی، الگوها، refactoring ۵- معماری نرم‌افزار: طراحی، مستندسازی و ارزیابی ۶- آزمون نرم‌افزار ۷- مدیریت کیفیت نرم‌افزار ۸- تخمین هزینه و زمان ۹- مدیریت پروژه – مدیریت نیروی انسانی – مدیریت ریسک ۱۰- مدیریت چرخه حیات – مدیریت تغییر – مدیریت پیکربندی ۱۱- روش‌های چابک		
نرم‌افزارهای مورد نیاز	ابزار مدل‌سازی UML – ابزار مدیریت پروژه		
تکالیف پیشنهادی	۵ تکلیف دستی		
پروژه‌های پیشنهادی	یک پروژه جهت بکارگیری اصول مهندسی نرم‌افزار در عمل در طول ترم		
نمره‌دهی پیشنهادی	تکالیف	۱۰٪	

	پروژه آزمون‌ها	۳۰٪ ۶۰٪
سایر مراجع		

فهرست

فصل نخست

نرم‌افزار و مهندسی نرم‌افزار

۱-۱ نرم‌افزار و مهندسی نرم‌افزار

در دنیای امروز، نرم‌افزار در همه‌ی جنبه‌های زندگی انسان از فرهنگ گرفته تا روابط اجتماعی رسوخ پیدا کرده است. بیشتر زیرساخت‌های زندگی بشر توسط سیستم‌های کامپیوتری کنترل می‌شوند. اما پرسش بسیار مهمی که مطرح می‌شود این است که نرم‌افزار چیست؟ پاسخ این است که نرم‌افزار از سه جزء تشکیل شده است، برنامه‌هایی که روی کامپیوتر اجرا می‌شوند، مستندات که شامل فرم‌ها می‌باشند و داده‌ها که شامل ارقام و حروف و اطلاعات تصویری می‌باشند. حدود پنجاه سال پیش بحران نرم‌افزاری بوجود آمد که نرم‌افزارهای زیادی موردنیاز بود، درحالیکه تولیدکنندگان نرم‌افزار بسیار کم بودند، بنابراین نسبت عرضه نرم‌افزار با تقاضای آن برابری نداشت و هیچ اصولی برای تولید یک نرم‌افزار وجود نداشت. مهندسی نرم‌افزار برای نظام‌مندکردن، اقتصادی بودن و مطابق اصول واحد بودن، برای تولید نرم‌افزار بوجود آمده است. گزارش‌هایی از شکست و عدم موفقیت نرم‌افزارها وجود دارد، بسیاری از این شکست‌ها به دو دلیل می‌باشند:

۱. تقاضاهای روزافزون: با توجه به افزایش نیازهای روزافزون کاربران نسبت به نیازهای گذشته، حجم سیستم‌های نرم‌افزاری بزرگتر و بنابراین پیچیده‌تر شده‌اند. افزون بر آن، این سیستم‌های پیچیده باید با سرعت بیشتر ایجاد شوند و خروجی را تحویل دهند.
۲. عدم استفاده از مهندسی نرم‌افزار: ایجاد برنامه‌های کامپیوتری بدون استفاده از اصول مهندسی نرم‌افزار ساده می‌باشد. بسیاری از شرکت‌های نرم‌افزاری در کار روزانه خود از روش‌های مهندسی نرم‌افزار استفاده نمی‌کنند، در نتیجه نرم‌افزار آنها گران‌تر است و قابلیت اعتماد کمتری دارد.

امروزه نرم‌افزار، نقشی دوگانه دارد. نرم‌افزار نوعی محصول است و درعین‌حال بستری است که می‌تواند یک محصول را تکمیل کند. نرم‌افزار به عنوان یک محصول می‌تواند در داخل یک تلفن همراه مورد استفاده قرار بگیرد "مانند نرم‌افزار ماشین حساب" و به عنوان بستر می‌تواند در سیستم‌های مخابراتی یا سیستم‌های توکار، نقش تکمیل کننده داشته باشد.

برای نمونه، یک نرم‌افزار کنترل دستگاه تولید یک کارخانه را درنظر بگیرید که باعث می‌شود دستگاه تولید، کار خود را مطابق دستورات داده شده به آن، انجام دهد، بنابراین در اینجا نرم‌افزار نقش مکمل دستگاه تولید را دارد. پس گاهی نرم‌افزار مستقیما به وسیله انسان‌ها و گاهی به‌صورت غیرمستقیم و درون سیستم‌ها به انسان یاری می‌رساند. مهم‌ترین محصولی که یک نرم‌افزار تحویل می‌دهد، اطلاعات است. نرم‌افزارها داده‌ها را با پردازش کردن تبدیل به اطلاعات می‌کنند.

پرسش دیگری که مطرح می‌شود این است که مهندسی نرم‌افزار چیست؟ چندین تعریف برای مهندسی نرم‌افزار مطرح شده است:

۱. تعریف مهندسی نرم‌افزار از دیدگاه پرفسور سامرویل: مهندسی نرم‌افزار یک نظام مهندسی است که با جنبه‌های تولید نرم‌افزار سر و کار دارد.
۲. تعریف مهندسی نرم‌افزار فرتیز باور: تصویب اصول مهندسی و استفاده از آنها برای به‌دست آوردن یک نرم‌افزار مقرون‌به‌صرفه که قابل اطمینان بوده و روی ماشین واقعی به‌طور کارآمدی اجرا شود.
۳. تعریف مهندسی نرم‌افزار از دیدگاه IEEE: کاربرد یک روش سیستماتیک عملی و کمیت‌پذیر در توسعه، راه‌اندازی و نگهداری نرم‌افزار، یعنی استفاده از مهندسی نرم‌افزار و مطالعه روش‌های مختلفی که این خصوصیات را داشته باشند.

امروزه محصولات نرم‌افزاری تولیدی به دو دسته تقسیم می‌شوند:

۱. محصولات نرم‌افزاری مورد استفاده عموم: این سیستم‌ها توسط شرکت‌های تولیدکننده نرم‌افزار تولید و در بازارها به مشتریان فروخته می‌شوند، مانند مجموعه نرم‌افزارهای Office
۲. محصولات سفارشی: سیستم‌هایی که توسط مشتری خاص سفارش داده می‌شوند و یک پیمانکار نرم‌افزار آن را برای مشتری تولید می‌کند. مانند نرم‌افزار حسابداری شرکت فولاد خوزستان

۱-۲ ویژگی‌های یک نرم‌افزار خوب

این ویژگی‌ها به موارد زیر تقسیم می‌شود:

۱. قابلیت نگهداری: نرم‌افزار باید به‌گونه‌ای نوشته شود که بتوان آن را ارتقاء داد. در دنیای امروزی که مدام درحال تغییر و تحول است، نیاز به محصولی داریم که بتواند ارتقاء پیدا کند تا نیازهای جدید کاربران را رفع کند.
۲. قابلیت اتکا و امنیت: قابلیت اتکا خود شامل قابلیت اعتماد، امنیت و ایمنی می‌باشد. نرم‌افزار قابل اعتماد نباید در اثر خرابی، منجر به ضررهای اقتصادی و فیزیکی شود. همچنین نرم‌افزار ایمن نباید به کاربران نفوذگر اجازه دهد تا به سیستم آسیب برسانند.

۳. کارایی: نرم‌افزار نمی‌بایست منابع سیستم مانند حافظه و پردازنده را هدر دهد. کارایی خود شامل زمان پردازش، تعداد پاسخگویی در واحد زمان و بهره‌وری از حافظه است.
۴. قابلیت پذیرش: نرم‌افزار می‌بایست برای کاربرانی که طراحی شده است، قابل درک، قابل استفاده و سازگار با سیستم‌های دیگر باشد.

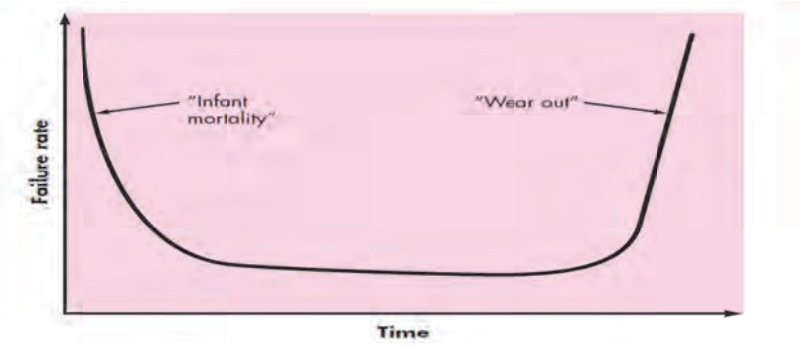
نرم‌افزار یک عنصر منطقی است نه یک عنصر فیزیکی، بنابراین دارای ویژگی‌هایی است که تفاوت چشم‌گیری با سخت‌افزار دارد:

۱. نرم‌افزار، مهندسی و توسعه داده می‌شود و تولید نمی‌شود.
۲. نرم‌افزار فرسوده نمی‌شود.
۳. بیشتر نرم‌افزارها بر خلاف محصولات صنعتی که مبتنی بر مولفه ساخته می‌شوند، به‌صورت سفارشی ساخته می‌شوند.

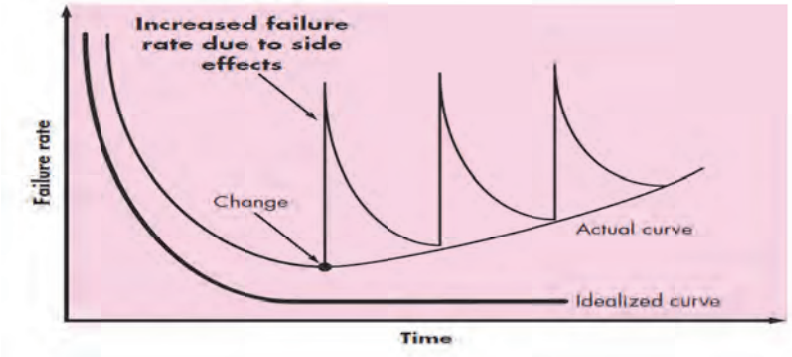
۱-۳ مقایسه سخت‌افزار و نرم‌افزار

شباهت‌هایی در توسعه نرم‌افزار و سخت‌افزار وجود دارد که در هر دو، کیفیت بالای محصول نهایی از طریق طراحی خوب به‌دست می‌آید، هر دو وابسته به انسان هستند و در هر دو لازم است یک محصول ساخته شود. سخت‌افزار و نرم‌افزار تفاوت‌هایی نیز با یکدیگر دارند.

به‌عنوان چند مثال از تفاوت‌های بین سخت‌افزار و نرم‌افزار، در فاز ساخت سخت‌افزار مشکلات کیفیتی به‌وجود می‌آید ولی در نرم‌افزار اینگونه نیست. رابطه میان کاری که قرار است انجام شود و تعداد افراد مورد نیاز، در سخت‌افزار متفاوت است، همچنین، روش‌های ساخت سخت‌افزار و نرم‌افزار باهم فرق دارند. در شکل زیر نمودار خرابی سخت‌افزار آورده شده است.



نمودار فوق نرخ خرابی سخت‌افزار را به‌صورت تابعی از زمان نشان می‌دهد. این منحنی در این شکل نشان می‌دهد که سخت‌افزار در ابتدای عمر خود نرخ خرابی نسبتا بالایی دارد، که خرابی‌ها غالبا ناشی از عیوب طراحی و تولید می‌باشند. پس‌از تصحیح این عیوب، نرخ خرابی برای یک دوره زمانی، مقدار ثابتی می‌شود. با گذشت زمان سخت‌افزار فرسوده می‌شود و دوباره نرخ خرابی بالا می‌رود. در شکل زیر نمودار خرابی نرم‌افزار آورده شده است.



نرم‌افزار نسبت به تغییرات محیط پیرامون که باعث فرسودگی می‌شود، نفوذپذیر نیست (مانند وجود یک عنصر خاص در یک منطقه). بنابراین در تئوری، منحنی خرابی نرم‌افزار باید به‌صورت ایده‌آل باشد. عیب‌های کشف نشده باعث نرخ خرابی بالا، در ابتدای عمر برنامه می‌شود. پس از برطرف شدن این عیوب منحنی صاف می‌شود. تناقض ظاهری بین منحنی ایده‌آل و منحنی واقعی را می‌توان چنین توضیح داد که نرم‌افزار در دوران حیات خود (از لحظه شروع توسعه نرم‌افزار تا پیش از تحویل آن) دستخوش تغییر می‌شود. با اعمال این تغییرات، احتمال دارد که برخی عیوب جدید وارد شوند و باعث رشد منحنی نرخ خرابی شوند. پیش از اینکه منحنی به نرخ خرابی یکنواخت پیشین خود برسد، تغییر دیگری درخواست می‌شود که باعث رشد دوباره منحنی می‌شود. این تغییرات آن‌قدر اعمال می‌شوند که نرم‌افزار فاسد می‌شود (یعنی آن‌قدر در نرم‌افزار ابتدایی تغییر اعمال می‌شود که ماهیتش به‌کل عوض می‌شود).

۱-۴ هفت گروه نرم‌افزارهای کامپیوتری

۱. نرم‌افزارهای سیستمی: شامل مجموعه‌ای از برنامه‌هاست که برای سرویس‌دهی به برنامه‌های دیگر نوشته می‌شود، مانند سیستم‌عامل.
۲. نرم‌افزارهای کاربردی: یک نیاز تجاری خاص را برآورده می‌کنند، مانند نرم‌افزار حسابداری شرکت فولاد خوزستان.
۳. نرم‌افزارهای مهندسی/ علمی: توسط الگوریتم‌هایی، اعداد و ارقام را پردازش می‌کنند، مانند نرم‌افزار Matlab .

۴. نرم‌افزارهای تعبیه شده: دارای حافظه فقط خواندنی هستند و برای کنترل محصولات و سیستم‌های صنعتی به‌کار می‌روند، مانند کنترل صفحه کلید فرهای مایکرویو.
۵. نرم‌افزارهای خط تولید: برای فراهم آوردن یک قابلیت خاص برای مشتریان طراحی می‌شود، مانند واژه‌پرداز Word.
۶. برنامه‌های کاربردی تحت وب: نرم‌افزارهای شبکه‌ای که شامل مجموعه گسترده‌ای از برنامه‌های کاربردی می‌شود، مانند سایت‌های اینترنتی.
۷. نرم‌افزارهای هوش مصنوعی: برای حل مسائل پیچیده‌ای که با روش‌های عددی قابل‌حل نیستند و از الگوریتم‌های غیرعددی استفاده می‌کنند، مانند بازی‌های کامپیوتری.

۱-۵ چالش‌های پیش روی مهندسی نرم‌افزار

۱. محاسبات در دنیای باز: در حال ایجاد نرم‌افزارهایی برای ماشین‌های با هراندازه برای برقراری ارتباط با یکدیگر در سراسر شبکه‌های گسترده هستیم.
۲. منابع شبکه: شبکه جهانی وب به سرعت در حال تبدیل به یک موتور کامپیوتری و نیز منبعی برای اطلاعات می‌باشد.
۳. کدمنبع باز: باز بودن کدمنبع مربوط به برنامه‌های کاربردی محاسباتی باعث می‌شود مشتریان بتوانند به راحتی و با قابلیت اعتماد، تغییرات خود را اعمال کنند.

۱-۶ دلایل اهمیت مهندسی نرم‌افزار

۱. با توجه و تأکید روزافزون افراد و جامعه به سیستم‌های نرم‌افزاری، باید بتوانیم سیستم‌های قابل اعتماد و امن را با قیمت مناسب و به‌طور سریع تولید کنیم.
۲. استفاده از روش‌ها و تکنیک‌های مهندسی نرم‌افزار برای سیستم‌ها در دراز مدت، ارزان‌تر تمام خواهد شد.

۱-۷ دلایل نیاز به تکامل سیستم‌های نرم‌افزاری قدیمی

۱. نرم‌افزار باید برای برآورده ساختن نیازهای محیط‌های جدید کامپیوتری یا فن‌آوری‌های جدید، اصلاح شود.
۲. نرم‌افزار باید بهبود یابد تا نیازهای جدید تجاری را برآورده سازد.
۳. نرم‌افزار باید توسعه داده شود تا با دیگر سیستم‌ها یا بانک‌های اطلاعاتی جدیدتر، قابلیت همکاری داشته باشد.
۴. نرم‌افزار باید معماری شود تا در یک محیط شبکه بتواند ادامه حیات دهد.

۱-۸ لایه‌های مهندسی نرم‌افزار

مهندسی نرم‌افزار، یک فن‌آوری لایه‌ای است. لایه‌های آن عبارتند از:

تمرکز بر کیفیت

فرآیند

روش‌ها

ابزارهای مهندسی نرم‌افزار

۱. تمرکز بر کیفیت: سنگ‌بنای مهندسی نرم‌افزار، توجه به کیفیت است. تمرکز بر کیفیت باعث ایجاد روش‌های کامل‌تری در مهندسی نرم‌افزار می‌شود، مانند استاندارد ISO/IEC15 .
۲. فرآیند: چارچوبی را تعریف می‌کند که باعث می‌شود محصول نرم‌افزاری به‌موقع و با کیفیت تحویل داده شود. به‌دیگر سخن، چارچوبی را تعریف می‌کند که اعمال مهندسی نرم‌افزار به‌صورت سازمان یافته، توسط سیستم نرم‌افزاری انجام شوند، مانند فعالیت‌های چهارچوبی تعریف شده توسط پرفسور پرسمن.
۳. روش‌ها: شیوه‌های فنی برای ساخت نرم‌افزار را فرآهم می‌آورند. روش‌ها شامل طیف وسیعی از وظایف مانند تحلیل نیازها، طراحی، ساخت برنامه‌ها، آزمایش و پشتیبانی می‌باشند. برای نمونه، برای مرحله تجزیه و تحلیل، می‌توان آن را به روش تجزیه و تحلیل ساخت‌یافته و تجزیه و تحلیل شئ‌گرا تقسیم کرد.
۴. ابزارهای مهندسی نرم‌افزار: به‌صورت خودکار یا نیمه خودکار از فرآیندها و روش‌ها پشتیبانی می‌کنند. مانند نرم‌افزار Rotional Rose برای مدل‌سازی نیازها به روش شئ‌گرا.

۱-۹ فرآیند نرم‌افزار

یک فرآیند نرم‌افزاری شامل سه عنصر زیر است:

۱. فعالیت‌ها: کوششی است برای رسیدن به یک هدف گسترده، مانند برقراری ارتباط با افراد ذینفع.
۲. اعمال: یک عمل شامل مجموعه‌ای از وظایف است که یک محصول کاری عمده تولید می‌کند، مانند شناسایی ذینفعان سیستم.

۳. وظیفه: کوچکترین واحد در یک فرآیند که باعث ایجاد نتیجه‌ای قابل لمس می‌شود، مانند تکمیل فرم محاسبه از کاربران نهایی.

هر فرآیند، شامل یک چارچوب است که این چهارچوب شامل تعدادی فعالیت می‌باشد. در کتاب آقای پرفسور پرسمن یک چارچوب برای فرآیند نرم‌افزاری که قابل استفاده در تمام پروژه‌های نرم‌افزاری است، معرفی شده که به شرح زیر است:

۱. ارتباط: پیش از آغاز هر کار تکنیکی، برقراری ارتباط و همکاری با مشتری و دیگر ذینفعان پروژه بسیار مهم است. هدف از برقراری ارتباط با مشتری، درک اهداف ذینفعان پروژه و جمع‌آوری نیازهایی است که مشخصه‌ها و قابلیت‌های عملیاتی نرم‌افزار را تعیین می‌کنند.
۲. برنامه‌ریزی: هر کار پیچیده‌ای نیاز به یک نقشه دارد. فعالیت برنامه‌ریزی، نقشه‌ای ایجاد می‌کند که به سیستم نرم‌افزاری کمک می‌کند. برنامه‌ریزی، یک برنامه کاربردی برای مهندسی نرم‌افزار فرآهم می‌کند و ریسک‌های فنی، فهرست منابع مورد نیاز، محصولات کاری تولید شده و تعیین زمان‌بندی کار را تعیین می‌کند.
۳. مدل‌سازی: برای درک بهتر نیازهای نرم‌افزار و ایجاد یک پیش‌زمینه برای طراحی نرم‌افزار، می‌بایست مدل‌سازی نیازها را انجام داد تا درک بهتری از نیازهای مشتریان داشته باشیم.
۴. ساخت: در این فعالیت، تولید کد و آزمون‌های لازم برای آشکار کردن خطاهای موجود در کد انجام می‌شود.
۵. استقرار: نرم‌افزار به مشتری تحویل داده می‌شود و محصول تحویل داده شده مورد ارزیابی مشتری قرار می‌گیرد و بر اساس این ارزیابی، بازخوردی به تولید کننده می‌دهد.

آقای پرفسور سامرویل چهار فعالیت اصلی مورداستفاده در بین تمام فرآیندهای نرم‌افزار را به‌صورت زیر تعریف کرده است:

۱. تعیین مشخصه‌های نرم‌افزار: مشتریان و مهندسان، قابلیت‌هایی را که نرم‌افزار باید داشته باشد تعریف می‌کنند.
۲. توسعه نرم‌افزار: نرم‌افزار طراحی و برنامه‌نویسی می‌شود.
۳. اعتبارسنجی نرم‌افزار: نرم‌افزار مورد بررسی قرار می‌گیرد تا تضمین شود نیازهای مشتری را برآورده می‌کند.
۴. تکامل نرم‌افزار: نرم‌افزار تغییر می‌کند تا نیازهای جدید مشتریان و نیازهای بازار را فراهم کند.

فرآیندهای مهندسی نرم‌افزار به دو دسته کلی تقسیم می‌شوند:

۱. فرآیندهای مبتنی بر برنامه‌ریزی: تمام فعالیت‌ها از پیش برنامه‌ریزی شده هستند و پیشرفت‌کار، بر اساس برنامه‌ریزی اندازه‌گیری می‌شود.
۲. فرآیندهای چابک: در فرآیندهای چابک، برنامه‌ریزی به‌صورت افزایشی انجام می‌گیرد و تغییر فرآیند برای منعکس ساختن نیازهای مشتری، آسان‌تر است.

فعالیت‌های چارچوبی فرآیند مهندسی نرم‌افزار با تعدادی از فعالیت‌های چتری تکمیل می‌شود. به‌طورکلی فعالیت-های چتری در سراسر یک پروژه نرم‌افزاری مورد استفاده قرار می‌گیرند و به تیم پروژه نرم‌افزاری کمک می‌کند تا پیشرفت، کیفیت، تغییر و ریسک را کنترل کند. فعالیت‌های چتری متداول شامل موارد زیر می‌باشند:

۱. کنترل و ردیابی پروژه‌های نرم‌افزاری: اجازه می‌دهد که تیم توسعه نرم‌افزار، میزان پیشرفت پروژه را ارزیابی کند و اقدامات اصلاحی را برای ادامه دادن بر طبق برنامه‌زمانی به‌عمل آورد.
۲. مدیریت ریسک: ریسک‌هایی را که ممکن است نتایج یا کیفیت پروژه را تحت تأثیر قرار دهد، ارزیابی می-کند.
۳. تضمین کیفیت نرم‌افزار: اعمال موردنیاز برای حفظ کیفیت را تعیین می‌کند.
۴. مرورهای تکنیکی: محصولات کاری مهندسی نرم‌افزار را برای آشکار کردن یا حذف خطاها پیش از آنکه خطاها به فعالیت‌های بعدی انتشار یابند، بررسی می‌کند.
۵. اندازه‌گیری: تعریف کمی فرآیند و پروژه و اندازه‌گیری محصول برای تعیین میزان برآورده شدن نیازهای مشتری، برای کمک به تیمی که در حال تحویل نرم‌افزار است، می باشد.
۶. مدیریت پیکربندی: مدیریت اثرات تغییرات (تغییرات در سراسر فرآیند مهندسی نرم‌افزار وجود دارند) را مدیریت می‌کند.
۷. مدیریت قابلیت استفاده مجدد: معیارهایی برای محصولات کاری قابل استفاده مجدد، تعریف می‌کند و سازوکارهایی برای دستیابی به اجزای قابل استفاده دوباره، تعریف می‌کند.
۸. تهیه و تولید محصول کاری: فعالیت‌هایی برای ایجاد مدل‌ها، مستندات، گزارش‌ها، فرم‌ها، لیست‌ها و غیره.

مدل فرآیندی که برای یک پروژه تعیین می‌شود، ممکن است با فرآیند تعیین شده برای پروژه‌ی دیگر تفاوت چشم‌گیری داشته باشد. برخی از این تفاوت‌ها می‌توانند شامل موارد زیر باشند:

۱. جریان کلی فعالیت‌ها، اعمال و وظایف و سطح وابستگی میان وظایف.
۲. درجه تعریف اعمال و وظایف کاری درون هر فعالیت چارچوبی.
۳. درجه شناسایی محصولات کاری و نیاز به آن‌ها.
۴. روشی که فعالیت‌های تضمین کیفیت اعمال می‌شوند.
۵. روشی که پروژه ردیابی و فعالیت‌ها کنترل می‌شوند.
۶. درجه کلی جزئیات به‌کار رفته در توصیف فرآیند.
۷. درجه درگیری ذینفعان در پروژه.
۸. سطح استقلال داده‌شده به تیم پروژه نرم‌افزاری.
۹. درجه توصیف نقش‌ها و سازماندهی تیم.

۱-۱۰ پندارهای باطل نرم‌افزاری

بسیاری از عواملی که باعث ایجاد مشکل در نرم‌افزار می‌شوند به دلیل باورهای نادرستی است که به نرم‌افزارها و فرآیند مورد استفاده در ساخت نرم‌افزار نسبت داده‌اند. پندارهای باطل نرم‌افزاری به‌دلیل ظاهر منطقی که دارند باعث انتشار اطلاعات غلط و گیج‌کننده‌ای برای افراد غیر حرفه‌ای شده‌اند. امروزه بیشتر مهندسان نرم‌افزار حرفه‌ای، این پندارهای غلط را می‌شناسند و می‌دانند که این پندارها ممکن است باعث گمراهی مدیران و دست‌اندرکاران پروژه نرم‌افزاری شود:

۱. پندارهای باطل مدیران: مدیرانی که در پروژه نرم‌افزاری مسئولیتی دارند، همانند بیشتر مدیران، زیر فشار هستند تا بودجه را حفظ کنند (هزینه‌ها را کاهش دهند) و هرگونه خللی در برنامه را از میان بردارند و کیفیت را بهبود دهند.

- پندار باطل: پیش از شروع پروژه، کتابی در اختیار داریم که حاوی استانداردها و رویه‌هایی برای ایجاد نرم‌افزار است. آیا این کتاب تمام آنچه را که افراد تحت مدیریت من، باید بدانند را در اختیارشان قرار می‌دهد؟
- واقعیت: کتاب استانداردها می‌تواند وجود داشته باشد، اما آیا این کتاب کامل است؟ آیا منعکس‌کننده مهندسی نرم‌افزار مدرن است؟
- پندار باطل: اگر از برنامه‌زمانی عقب باشیم، می‌توانیم از برنامه‌نویسان بیشتری استفاده کنیم و عقب‌ماندگی را جبران کنیم.
- واقعیت: توسعه نرم‌افزار یک فرایند مکانیکی مانند ساخت ماشین‌آلات نیست، برطبق نظر پرفسور بروکز "افزودن افراد در پروژه‌ای که از برنامه‌زمانی عقب است، باعث عقب‌افتادگی بیشتر می‌شود" زیرا افراد جدیدی که به تیم پروژه افزوده می‌شوند همگی نیاز به آموزش دارند. این آموزش توسط افرادی که در پروژه مشغول به‌کار بوده‌اند، باید انجام شود.
- پندار باطل: اگر تصمیم به برون‌سپاری یک پروژه نرم‌افزاری به شرکت دیگر بگیریم، می‌توانیم با آسودگی خاطر ایجاد نرم‌افزار را به آن شرکت بسپاریم.
- واقعیت این است که اگر یک سازمان از روش‌های مدیریت کنترل پروژه نرم‌افزاری آگاهی نداشته باشد، هنگامی که پروژه نرم‌افزاری را برون‌سپاری کند، باید منتظر تنش‌های زیادی باشد.

۲. پندارهای باطل مشتریان: مشتری که درخواست یک نرم‌افزار کامپیوتری دارد ممکن است یک کارمند، یک گروه کاری، واحد بازاریابی و یا بخش فروش یک شرکت باشد که قراردادی برای یک نرم‌افزار منعقد نموده است. در بسیاری از موارد مشتری پندارهای باطلی را درمورد نرم‌افزار در سر دارد که باعث انتظارات نادرست از سوی مشتری و در نهایت عدم رضایت سازنده می‌شود.

- پندار باطل: بیان کلی اهداف برای شروع برنامه‌نویسی کافی است. جزئیات را بعدا می‌توانیم تکمیل کنیم.

- واقعیت: بیان همراه با جزئیات و پایدار از نیازمندی‌ها، همواره امکان‌پذیر نیست. از سویی، یک تعریف ضعیف از نیازمندی‌ها، عامل عمده شکست فعالیت‌های نرم‌افزاری است. نیازهای غیرمبهم تنها از طریق برقراری ارتباط پیوسته و اثربخش میان مشتری و توسعه‌دهنده شکل می‌گیرد.
- پندار باطل: نیازهای پروژه پیوسته درحال تغییر است، اما با این تغییرات می‌توان به‌راحتی سازگار شد، زیرا نرم‌افزار انعطاف‌پذیر است.
- واقعیت: درست است که نیازمندی‌های نرم‌افزار تغییر می‌کند، ولی تأثیر تغییر به زمان اعمال تغییر بستگی دارد. تغییر در مراحل اولیه مهندسی نرم‌افزار کم هزینه‌تر از تغییر در مراحل پایانی مهندسی نرم‌افزار می‌باشد.

۳. پندارهای باطل توسعه‌دهندگان: پندارهای باطلی که برنامه‌نویسان به آن باور دارند، نتیجه پنجاه سال برنامه‌نویسی است.

- پندار باطل: هنگامیکه برنامه را نوشتیم و برنامه کار کرد، دیگر کار تمام است. یک عقیده براین است که "هرچه زودتر کدنویسی را شروع کنید مدت طولانی‌تری برای اتمام کار لازم است". داده‌های به‌دست آمده از پروژه‌ها نشان می‌دهند بین ۶۰ تا ۸۰ درصد تمام کوشش‌های صرف‌شده روی نرم‌افزار پس از نخستین تحویل به مشتری، صورت می‌گیرد.
- پندار باطل: تا هنگامی‌که برنامه را اجرا نکرده‌ایم، راهی برای ارزیابی کیفیت آن نداریم.
- واقعیت: یکی از مهم‌ترین مکانیزم‌ها در این زمینه، تضمین کیفیت نرم‌افزاری می‌تواند باشد که از شروع پروژه به‌کار گرفته می‌شود و شامل تکنیک‌های مرور می‌باشد.
- پندار باطل: تنها شئ قابل تحویل برای یک پروژه موفق، برنامه‌ی در حال کار است.
- واقعیت: یک برنامه در حال کار، تنها یک بخش از ساختار نرم‌افزار است که شامل عناصر فراوان می‌شود. همانطور که گفته شد یک نرم‌افزار شامل داده‌ها و مستندات نیز می‌شود.
- پندار باطل: استفاده از مهندسی نرم‌افزار باعث می‌شود که مستندات حجیم و بیهوده تهیه کنیم و این کار، سرعت ما را در توسعه نرم‌افزار کاهش می‌دهد.
- واقعیت: تهیه مستندات درجهت بالا بردن کیفیت محصول می‌باشد. وجود مستندات برای نرم‌افزار، تغییرات را آسان می‌کند.

۱-۱۱ شروع یک پروژه نرم‌افزاری

یک پروژه نرم‌افزاری می‌تواند به دلایل زیادی شروع شود:

- ✓ نیاز تجاری
- ✓ نیاز به تصحیح یک نقص در نرم‌افزار موجود

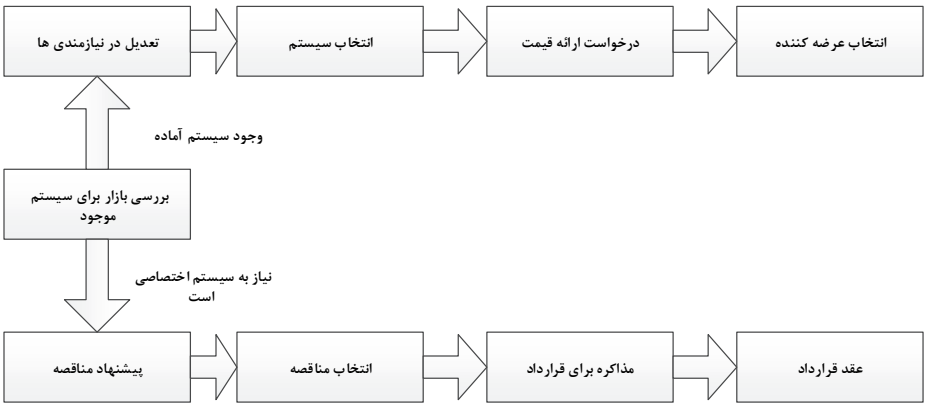
- ✓ ایجاد یک سرویس جدید برای نرم‌افزار
- ✓ نیاز به توسعه قابلیت‌های عملیاتی سیستم موجود

جورج پولیا در کتاب خود، چگونگی حل مسئله را شرح داده است. نظر وی مربوط به دوران پیش از وجود کامپیوترهای مدرن است، ولی این نظر می‌تواند به‌عنوان مبنای حل مسئله در مهندسی نرم‌افزار نیز به‌کار گرفته شود:

۱. درک مسئله (ارتباط و تحلیل سیستم)
۲. طرح یک راه‌حل (مدل‌سازی و طراحی نرم‌افزار)
۳. انجام طرح (تولید کد)
۴. بررسی نتایج برای صحت (آزمون و تضمین کیفیت)

۱-۱۲ فرآیند تهیه سیستم در سازمان

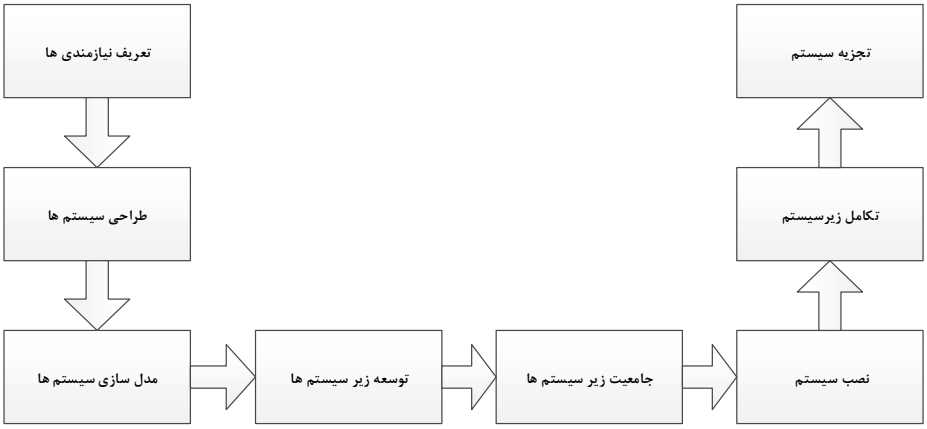
سیستم های پیچیده معمولا توسط سازمان‌هایی، غیر از سازمان تهیه کننده سیستم توسعه می‌یابد. علت این است که شغل تهیه کننده به ندرت توسعه سیستم است و در نتیجه، کارکنان و پرسنل آن فاقد مهارت‌های لازم برای توسعه سیستم‌های پیچیده هستند. درواقع سازمان‌ها، دارای قابلیت‌های طراحی، ساخت و آزمون تمام قطعات سیستم‌های پیچیده نیستند.



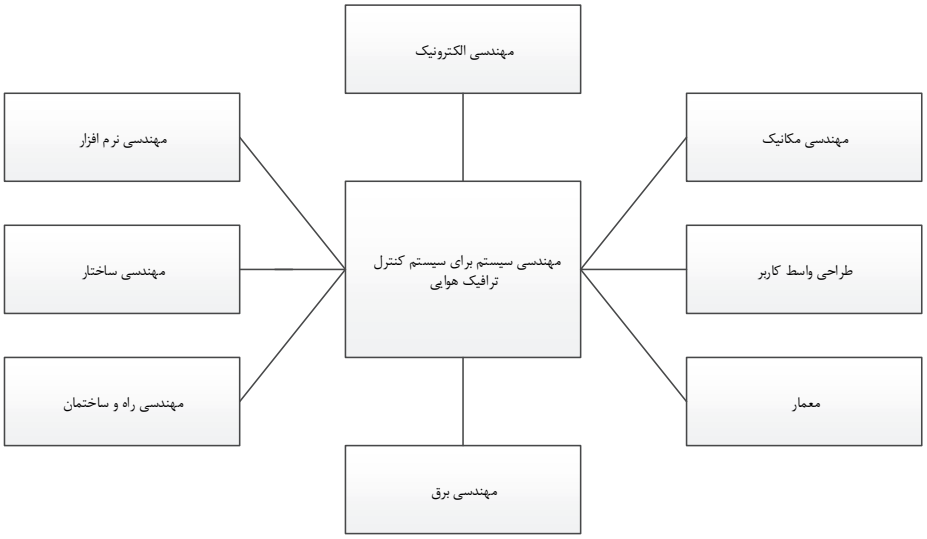
۱-۱۳مهندسی سیستم‌ها

مهندسی سیستم‌ها، فعالیت تعیین مشخصات، طراحی، پیاده‌سازی، اعتبارسنجی، استقرار و نگهداری سیستم‌ها است. مهندسان سیستم‌ها تنها با نرم‌افزار سروکار ندارند، بلکه با سخت‌افزار و تعامل‌های سیستم و نیز، با کاربران

و محیطش سروکار دارند. مهندسان نرم‌افزار باید مهندسی سیستم را درک کنند، زیرا مشکلات نرم‌افزار اغلب نتیجه تصمیمات مهندسی سیستم است.



مثال: مهندسی سیستم کنترل ترافیک هوایی

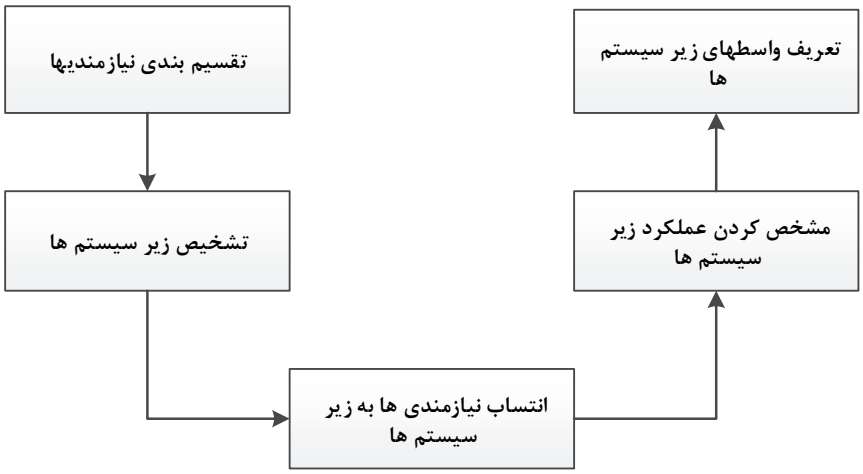


۱-۱۳-۱ تعریف نیازمندی‌های سیستم

فعالیت تعریف نیازمندی‌های سیستم، عملکرد و ویژگی‌های مطلوب سیستم را مشخص می‌کند. بخش مهمی از مرحله تعریف نیازمندی‌ها، تعیین مجموعه‌ای از اهداف کلی است که سیستم باید برآورده کند. مشکل اساسی در تعیین نیازمندی‌های سیستم این است که مسائلی که معمولا سیستم‌های پیچیده برای حل آنها ساخته می‌شوند، مسائل دشوار (سخت) نامیده می‌شوند. مسئله دشوار، مسئله‌ای است که بسیار پیچیده است و موجودیت‌های زیادی به آن وابسته است و مشخصات کامل مسئله معلوم نیست، مانند برنامه‌ریزی زلزله.

۱-۱۳-۲ طراحی سیستم‌ها

طراحی سیستم مشخص می‌کند که عملکرد سیستم چگونه باید توسط قطعات مختلف سیستم انجام شود. فرآیند طراحی سیستم به صورت زیر است:



الف) تقسیم بندی نیازمندی‌ها: نیازمندی‌ها تحلیل می‌شوند و در گروه‌های مرتبط به هم تفکیک می‌شوند.

ب) شناسایی زیرسیستم‌ها: زیرسیستم‌ها که هرکدام قابلیتی را برای سیستم فراهم می‌کنند، شناسایی می‌شوند.

ج) انتساب نیازمندی‌ها به زیرسیستم‌ها: هرکدام از نیازمندی‌ها به زیرسیستم‌ها انتساب داده می‌شوند.

د) مشخص کردن عملکرد زیرسیستم‌ها: عملکرد هریک از زیرسیستم‌ها تعیین می‌شود. روابط بین زیرسیستم‌ها باید در این مرحله مشخص شود.

ه) تعریف واسطهای زیرسیستم: در این مرحله واسطهای مورد نیاز هر یک از زیرسیستم‌ها تعریف می‌شود.

۱-۱۳-۳ مدل‌سازی سیستم

در حین طراحی و تعریف نیازمندی‌های سیستم، سیستم باید به صورت مجموعه‌ای از مولفه‌ها و روابط بین آن‌ها مدل‌سازی شود که معماری سیستم را نشان می‌دهند و به صورت یک نمودار بلوکی نمایش داده می‌شود. هر زیرسیستم در نمودار بلوکی توسط یک مستطیل نمایش داده می‌شود و روابط بین آن‌ها توسط فلش نمایش داده می‌شود.

۱-۱۳-۴ توسعه زیرسیستم

زیرسیستم‌هایی که در حین طراحی سیستم شناسایی شدند، پیاده‌سازی می‌شوند. زیرسیستم‌های مختلف معمولا به طور موازی توسعه می‌یابند.

۱-۱۳-۵ جامعیت زیرسیستم

در جامعیت زیرسیستم، زیرسیستم‌هایی که به طور مستقل از هم توسعه داده شده‌اند، در کنار هم قرار می‌گیرند تا سیستم کامل را ایجاد کنند. جامعیت سیستم را می‌توان با رهیافت "big bang" انجام داد. در این رهیافت، تمام زیرسیستم‌ها همزمان یکپارچه می‌شوند.

۱-۱۳-۶ نصب زیرسیستم

در این مرحله، سیستم یکپارچه شده، به طور اجرایی مورد استفاده قرار می‌گیرد. یعنی سیستم با نصب شدن، شروع به کار می‌کند.

۱-۱۳-۷ تکامل سیستم

طول عمر سیستم‌های بزرگ و پیچیده بسیار زیاد است. در حین حیات این سیستم‌ها، سیستم باید تکامل یابد تا خطاهای موجود در نیازمندی‌های اصلی سیستم را اصلاح کند و نیازمندی‌های جدید را برآورده کند. سازمان‌هایی که از سیستم استفاده می‌کنند، ممکن است درون خودشان تغییرات سازمانی ایجاد کنند و سیستم را به روش دیگری به کار گیرند. افزون بر این، محیط خارجی سیستم ممکن است تغییر کند و در نتیجه موجب تغییر سیستم شود.