موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

مهندسی نرم‌افزار

(مدل‌های فرآیند تولید نرم‌افزار)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی IT

براساس کتاب مرجع

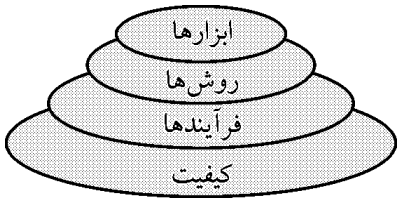
راجر اس. پرسمن هشتم ۲۰۱۴

ارسطو خلیلی‌فر

۲ مدل‌های فرآیند تولید نرم‌افزار

لایه‌های مهندسی نرم‌افزار

مهندسی نرم‌افزار یک فرآیند لایه‌ای است. به بیان دیگر مهندسی نرم‌افزار از چهار لایه تشکیل شده است. در شکل زیر، چهار لایه نشان داده شده است:



کیفیت

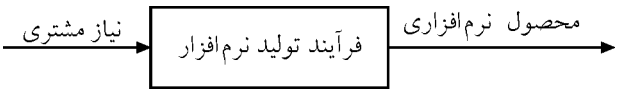
مهندسی نرم‌افزار یک کوشش لایه‌ای برای تولید یک محصول نرم‌افزاری با کیفیت که نیازمندی‌های مورد انتظار مشتری را برآورده می‌سازد می‌باشد. در صورتی که ابزارها، روش‌ها و فرآیندها به گونه‌ای درست و مطابق با کاربرد انتخاب و استفاده شوند می‌توان این‌گونه بیان نمود که کیفیت که همان برآورده ساختن نیازمندی‌های مورد انتظار مشتری می‌باشد برآورده شده است.

برای مثال کیفیت خانه برای برآورده ساختن آرامش و نیازهای مشتری، مهم است. اما شیوه‌های رسیدن به این مهم در شهرهای شمالی و جنوبی شباهت‌ها و تفاوت‌هایی دارد. به طور مثال در شهرهای شمالی بارندگی به وفور وجود دارد، پس روش‌ها و ابزارهای ساختمانی باید به گونه‌ای انتخاب شوند که در برابر بارندگی مقاوم باشند. مثل روش سقف شیروانی با استفاده از ابزار سفال!

فرآیندها (فرآیند تولید نرم‌افزار)

هر پروژه‌ی نرم‌افزاری، چه بزرگ و چه کوچک مراحلی را طی می‌نماید که در طی آن

مجموعه‌ای از نیازمندی‌های مشتری به یک محصول نرم‌افزاری تبدیل می‌گردد. الگو و قالبی که چگونگی طی مراحل مختلف یک پروژه را تعریف می‌نماید، اصطلاحاً **فرآیند تولید نرم‌افزار** نامیده می‌شود. شکل زیر فرآیند تولید نرم‌افزار و ورودی و خروجی آن را نشان می‌دهد:



همانطور که ملاحظه می‌نمایید، ورودی این فرآیند، نیاز یا خواسته‌های مشتری و خروجی آن، یک محصول نرم‌افزاری است. یک فرآیند تولید در یک پروژه به ما می‌گوید که برای دستیابی به هدف مطلوب که همان تولید یک فرآورده‌ی نرم‌افزاری با کیفیت مطلوب است، چه کسی[1]، چه کاری را[2]، چه موقع[3] و چگونه[4] باید انجام دهد، در واقع، بدون داشتن تعریف مشترکی از فرآیند تولید نرم‌افزار، هماهنگی انجام کار تیمی در یک پروژه‌ی نرم‌افزاری، امکان‌پذیر نخواهد بود.

توجه نمایید که مدل فرآیندی که انتخاب می‌کنیم دقیقاً به نرم‌افزاری که تولید می‌کنیم بستگی دارد. ممکن است که یک مدل فرآیند برای تولید سیستم الکترونیکی هواپیما مناسب باشد، در حالی که مدل فرآیندی دیگر، برای ایجاد یک وب‌سایت استفاده شود. پس انتخاب مدل فرآیند براساس ماهیت نرم‌افزار صورت می‌گیرد. در ادامه‌ی این فصل به طور مفصل در مورد انواع مدل‌های فرآیند تولید نرم‌افزار صحبت خواهیم کرد.

روش‌ها

شیوه‌های فنی برای ساخت نرم‌افزار را «**روش**» می‌گوییم و به دو شکل روش ساخت‌یافته و روش شیءگرا می‌باشد. در روش ساخت‌یافته می‌گوییم چه عملکردهایی داریم و این عملکردها به چه داده‌هایی نیاز دارند و داده و عملکرد را به طور جداگانه و به روش ساخت‌یافته تحلیل، طراحی و پیاده‌سازی می‌کنیم. اما در روش شیءگرا می‌گوییم چه داده‌هایی داریم و این داده‌ها چه عملکردهایی دارند. در واقع داده و عملکرد را در قالب یک بسته (کلاس) در کنار هم قرار می‌دهیم و به روش شیءگرا (مانند مفاهیم کلاس، وراثت و چندریختی) تحلیل، طراحی و پیاده‌سازی می‌کنیم. آنچه مهندسی‌نرم‌افزار به عنوان یک هدف و اصل سودآور برای سازنده نرم‌افزار و به‌تبع مقرون به صرفه‌شدن برای مشتری دنبال می‌کند، اصل قابلیت استفاده مجدد قطعات پروژه‌های نرم‌افزاری فعلی در پروژه‌های آتی است. زمان و هزینه‌ی فرآیند تولید نرم‌افزار

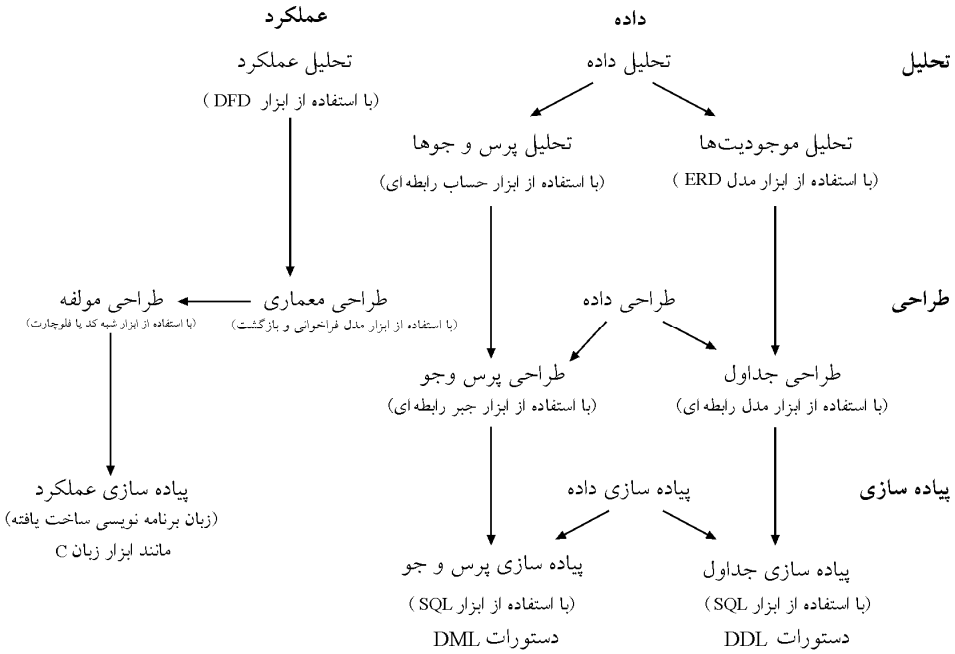
---

[1] Who
[2] What
[3] When
[4] How

به دلیل استفاده از قطعات آماده (قابلیت استفاده مجدد) کاهش چشمگیری دارد. در واقع یک قطعه با قابلیت استفاده‌ی مجدد یک بار ساخته می‌شود و بارها و بارها استفاده می‌شود و این یعنی سودآوری! بنابراین واضح است که شرط لازم برای ساخت قطعات قابل استفاده مجدد، قطعه قطعه کردن نرم‌افزار یا به عبارتی پیمانه‌ای‌کردن یا بُرش نرم‌افزار است. بنابراین روش یعنی چگونگی و نحوه بُرش نرم‌افزار، در متدولوژی ساخت‌یافته این بُرش یا به عبارتی پیمانه‌ای‌کردن نرم‌افزار براساس تابع تابع کردن نرم‌افزار انجام می‌گردد و در متدولوژی شیءگرا این بُرش یا به عبارتی پیمانه‌ای‌کردن نرم‌افزار براساس کلاس کلاس کردن نرم‌افزار انجام می‌گردد. همانطور که گفتیم شرط لازم برای ساخت قطعات قابل استفاده مجدد، قطعه قطعه کردن نرم‌افزار یا به عبارتی پیمانه‌ای‌کردن یا بُرش نرم‌افزار است. در مورد شرط کافی برای ساخت قطعات قابل استفاده مجدد که با مفاهیمی همچون اصل پنهان‌سازی اطلاعات، استقلال عملیاتی، انسجام بالا و اتصال پایین مرتبط است در فصل مفاهیم طراحی ساخت‌یافته و مفاهیم شیءگرایی به تفصیل صحبت خواهیم کرد.

ابزارها

ابزارهای فنی برای ساخت نرم‌افزار را «**ابزار**» می‌گوییم و به دو شکل ابزار ساخت‌یافته و ابزار شیءگرا می‌باشند. شکل زیر ابزارهای ساخت‌یافته را در روش ساخت‌یافته نمایش می‌دهد:



توجه: ابزارهای شیءگرا، در فصل مربوط به مباحث شیءگرایی معرفی خواهند شد. مانند ابزار UML که سرواژه عبارت Unified Modeling Language است و به معنی زبان مدل‌سازی یکپارچه می‌باشد که جهت مدل‌سازی متدولوژی شیءگراء مورد استفاده قرار می‌گیرد.

ابزارهای کاغذی (Tool) : به معنی استفاده از ابزار به صورت دستی و نقشی بر روی کاغذ بدون استفاده از کامپیوتر مانند رسم نمودار ER بدون کامپیوتر یا رسم نمودارهای UML بدون کامپیوتر یا نوشتن کدهای برنامه‌نویسی بر روی کاغذ بدون کامپیوتر.

ابزارهای کامپیوتری (CASE TOOL) : (Computer-Aided Software Engineering Tool)

به معنی استفاده از ابزار به کمک کامپیوتر، مانند رسم نمودار ER به کمک کامپیوتر توسط نرم‌افزار ویزیو یا رسم نمودارهای UML به کمک کامپیوتر توسط نرم‌افزار رشنال رُز یا نوشتن کدهای برنامه‌نویسی به کمک کامپیوتر توسط کامپایلرها.

توجه: ابزارهای کامپیوتری (CASE TOOL) می‌توانند، در کلیه مراحل فرآیند تولید نرم‌افزار (ارتباطات، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست) و استقرار) مورد استفاده قرار گیرند.

تکنیک‌های نسل چهارم (4GT)[1]

تکنیک‌های نسل چهارم، دسته‌ای از ابزارهای CASE هستند که در فعالیت **پیاده‌سازی**، کد برنامه را به صورت خودکار تولید می‌نمایند. البته شرط لازم برای این کار، توصیف مشخصات نرم‌افزار در یک سطح انتزاعی بالا در فعالیت مدل‌سازی (نمایش نرم‌افزار توسط اشکال و علائم گرافیکی) است. مانند تولید خودکار کدهای HTML توسط نرم‌افزار Dreamweaver یا FrontPage، تولید کدهای برنامه توسط نرم‌افزار رشنال رُز که نمودارهای آن در سطح انتزاعی بالا توسط نمودارهای UML ایجاد شده است.

متدولوژی

تمامی پروژه‌های نرم‌افزاری از چهار لایه‌ی مذکور تشکیل شده‌اند. به عبارت دیگر هر پروژه‌ی نرم‌افزاری، مدل فرآیند تولیدی برای انجام پروژه‌اش دارد. همچنین روش‌های مختلفی را در قسمت‌های مختلف پروژه برای انجام فعالیت‌های تعریف شده در فرآیند تولید نرم‌افزار به کار می‌برند و ممکن است برای هر یک از این روش‌ها از ابزار خاصی استفاده کنند. افزون بر این، هر پروژه نرم‌افزاری راهکاری برای تضمین کیفیت یک نرم‌افزار دارد زیرا هدف اصلی مهندسی نرم‌افزار تولید نرم‌افزار با کیفیت بالا می‌باشد. متدولوژی در واقع نحوه‌ی ارتباط این چهار لایه را با یکدیگر مشخص می‌کند. به بیان دیگر، متدولوژی مجموعه‌ای از فرآیندها، روش‌ها و ابزارهای

[1] Fourth Generation Techniques

مرتبط با هم و همه از یک متدولوژی خاص همچون ساخت‌یافته یا شیءگرا برای ایجاد یک محصول نرم‌افزاری، مطابق با استانداردهای مهندسی نرم‌افزار می‌باشد و بر دو طبقه‌ی ساخت‌یافته و شیءگرا می‌باشد.

توجه: اساس به وجود آمدن متدولوژی شیءگرا به وجود آمدن نیازهای جدید بوده که توسط متدولوژی ساخت‌یافته قابل پوشش دادن نبوده‌اند. متدولوژی شیءگرا برخی از نیازمندی‌های جدید را پوشش داده است و این طور نبوده‌است که استفاده از متدولوژی ساخت‌یافته را منسوخ کند.

متدولوژی ساخت‌یافته (مهندسی نرم‌افزار ساخت‌یافته)

متدولوژی ساخت‌یافته یا مهندسی نرم‌افزار ساخت‌یافته نظامی است یکپارچه شامل مدل فرآیندهای ساخت‌یافته (سنتی)، روش ساخت‌یافته و ابزارهای ساخت‌یافته که منجر به ایجاد نرم‌افزاری در بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و دقیقاً مطابق با نیازمندی‌های واقعی مشتری می‌گردد.

در متدولوژی ساخت‌یافته، در اولین مرحله مدل‌سازی (مدل تحلیل)، سیستم به دو وجه «داده» و «عملکرد» تفکیک می‌شود. سپس طی روندی سلسله مراتبی و مطابق با روش بالا به پایین، هر یک از این وجوه خود به مؤلفه‌های فرعی تجزیه می‌شوند. این روند تا به جایی ادامه می‌یابد که جزئیات توابع برنامه جهت پیاده‌سازی مشخص شوند.

توجه: متدولوژی SSADM متداول‌ترین نمونه از متدولوژی ساخت‌یافته براساس روش ساخت‌یافته، مدل فرآیند تولید آبشاری و ابزارهای ساخت‌یافته می‌باشد. و بر دو نوع داده‌گرا (جدولی) مانند نرم‌افزار حقوق و دستمزد که داده‌ها درون جداول ذخیره می‌شوند و تابع‌گرا (متغیری) مانند نرم‌افزار ماشین‌حساب که داده‌ها درون متغیرها ذخیره می‌شوند، می‌باشد.

توجه: نسبت نمونه متدولوژی ساخت‌یافته SSADM به متدولوژی ساخت‌یافته، مثل نسبت سیستم‌عامل ویندوز سنتی به مفاهیم سنتی سیستم‌عامل است.

متدولوژی شیءگرا (مهندسی نرم‌افزار شیءگرا)

متدولوژی شیءگرا یا مهندسی نرم‌افزار شیءگرا نظامی است یکپارچه شامل مدل فرآیند شیءگرا (مدرن)، روش شیءگرا (مبتنی بر مفاهیم کلاس، وراثت و چندریختی) و ابزارهای شیءگرا که منجر به ایجاد نرم‌افزاری در بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و دقیقاً مطابق با نیازمندی‌های واقعی مشتری می‌گردد.

در متدولوژی شیءگرا، در اولین مرحله‌ی مدل‌سازی (مدل تحلیل) سیستم در قالب کلاس‌ها (شامل داده (صفت) و عملکرد (متد)) نشان داده می‌شود. سپس طی روندی سلسله مراتبی و

مطابق با روش بالا به پایین، کلاس‌ها با جزئیات بیشتری مشخص می‌شوند. این روند تا بـه جـایی ادامه می‌یابد که جزئیات کلاس‌های برنامه جهت پیاده‌سازی مشخص شوند.

توجه: متدولوژی RUP متداول‌تـرین نمونـه از متـدولوژی شـیءگرا براسـاس روش شـیءگرا (مبتنی بر مفاهیم کلاس، وراثت و چندریختی)، مدل فرآیند مبتنی بر مؤلفه‌ی شـیءگرا بـا رویکـرد تکرار و تکامل و ابزارهای شـیءگرا (مثـل ابـزار مـدل‌سـازی UML و زبـان برنامـه‌نویسـی C++) می‌باشد.

توجه: نسبت نمونه متدولوژی شیءگرای RUP به متدولوژی شـیءگـرا، مثـل نسـبت سیسـتم عامل ویندوز مدرن به مفاهیم مدرن سیستم عامل است.

توجه: متدولوژی شیءگرا و متدولوژی RUP به تفصیل در فصـل مربوطـه شـرح داده خواهـد شد. در ادامه به تشریح مفاهیم مربوط به متدولوژی ساخت‌یافته خواهیم پرداخت:

فعالیت‌های چارچوبی فرآیند تولید نرم‌افزار

به طور کلی فعالیت‌های مرتبط با فرآیند تولید نرم‌افزار صرف‌نظر از اندازه، پیچیـدگی پـروژه و زمینه‌ی کـاربردی آن و مسـتقل از متـدولوژی سـاخت‌یافتـه و شـیءگرا بـه پـنج فعالیـت ارتبـاط، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست) و استقرار تقسیم می‌شـود، به بیان دیگر فعالیت‌ها در هر دو دسته‌ی متدولوژی ساخت‌یافته و شیءگرا همین‌هـا خواهنـد بـود، اما کارهایی که در هر فعالیت در متدولوژی ساخت‌یافته و شیءگرا انجـام مـی‌شـود شـباهت‌هـا و تفاوت‌هایی خواهد داشت.

در ادامه فعالیت‌های چارچوبی فرآیند تولید نرم‌افـزار براسـاس متـدولوژی سـاخت‌یافتـه بیـان خواهد شد:

۱- ارتباطات (مهندسی نیازمندی‌های مشتری یا مهندسی خواسته‌های مشتری)

فعالیت ارتباطات یا مهندسی نیازمندی‌های مشـتری نظـامی اسـت یکپارچـه، شـامل فرآینـدها، روش‌ها و ابزارها که منجر به تهیه لیست نیازمندی‌های مشتری می‌گردد.

فرآیند تهیه لیست نیازمندی‌های مشتری

هر پروژه‌ی نرم‌افزاری، چه بزرگ و چه کوچک مراحلی را طی می‌نماید که در طی آن لیسـت نیازمندی‌های مشتری تهیه می‌گردد. الگو و قالبی که چگونگی طـی مراحـل مختلـف تهیـه لیسـت نیازمندی‌های مشتری را تعریف می‌نماید، اصطلاحاً **فرآیند تهیه لیسـت نیازمنـدی‌هـای مشـتری** نامیده می‌شود.

روش‌های تهیه لیست نیازمندی‌های مشتری

روش‌های تشخیص برای تهیه لیست نیازمندی‌های مشتری را «روش‌های تهیه لیست نیازمندی‌های مشتری» می‌گوییم، یکی از روش‌های پرکاربرد روشی موسوم به «**روش QFD**» می‌باشد.

توجه: QFD سرواژه عبارت Quality Function Deployment و به معنی استقرار عملکرد کیفیت می‌باشد.

توجه: روش QFD جلوتر شرح داده خواهد شد.

ابزارهای تهیه لیست نیازمندی‌های مشتری

ابزارهای تشخیص برای تهیه لیست نیازمندی‌های مشتری را «ابزارهای تهیه لیست نیازمندی‌های مشتری» می‌گوییم و به پنج شکل «**گفتگو**»، «**مشاهده**»، «**پرسش‌نامه**»، «**مکانیزم نمونه‌سازی دورانداختنی**» و «**مکانیزم نمونه‌سازی تکاملی**» می‌باشد.

توجه: مکانیزم نمونه‌سازی دورانداختنی و تکاملی جلوتر شرح داده می‌شود.

توجه: فعالیت ارتباطات از طریق ارتباط با مشتری توسط ارتباط‌گر و ابزارها و روش‌های مطرح شده انجام می‌گردد.

مراحل فرآیند تهیه لیست نیازمندی‌های مشتری

به طور کلی مراحل مرتبط با فرآیند تهیه لیست نیازمندی‌های مشتری صرف‌نظر از اندازه، پیچیدگی پروژه و زمینه‌ی کاربردی آن و مستقل از متدولوژی ساخت‌یافته و شیءگرا به هفت مرحله **شناخت اولیه نیازمندی‌ها**، **شناخت بیشتر نیازمندی‌ها**، **تشریح نیازمندی‌های شناخته‌شده**، **مذاکره**، **تعیین مشخصات**، **اعتبارسنجی نیازمندی‌ها** و **مدیریت نیازمندی‌ها** تقسیم می‌شود، به بیان دیگر مراحل در هر دو دسته‌ی متدولوژی ساخت‌یافته و شیءگرا همین‌ها خواهند بود، اما کارهایی که در هر فعالیت در متدولوژی ساخت‌یافته و شیءگرا انجام می‌شود شباهت‌ها و تفاوت‌هایی خواهد داشت.

در ادامه مراحل فرآیند تهیه لیست نیازمندی‌های مشتری بیان خواهد شد:

۱- درک (شناخت اولیه نیازمندی‌ها)

در مرحله شناخت اولیه نیازمندی‌ها، شناختی پایه‌ای از نیازمندی‌های مشتری انجام می‌گردد، که این شناخت اولیه مستلزم ارتباط و گپ و گفت اولیه سازنده و مشتری است.

۲- استخراج (شناخت بیشتر نیازمندی‌ها)

در مرحله شناخت بیشتر نیازمندی‌ها، شناختی بیشتر از نیازمندی‌های مشتری انجام می‌گردد، که این شناخت بیشتر مستلزم ارتباط و گپ و گفت بیشتر سازنده و مشتری است.

توجه: توسط استفاده از روش QFD می‌توان به شناخت خواسته‌هایی از مشتری بپردازیم که برای مشتری ارزشمندتر است.QFD سه نوع خواسته را مشخص می‌کند:

خواسته‌های عادی: به خواسته‌هایی که مشتری بیان می‌کند و انتظار هم دارد که سازنده این خواسته‌ها را برآورده سازد، خواسته‌های عادی گفته می‌شود که در صورت برآورده‌سازی این خواسته‌ها توسط سازنده، مشتری راضی خواهد بود.

خواسته‌های مورد انتظار: به خواسته‌هایی که مشتری بیان نمی‌کند ولی به صورت پیش فرض انتظار هم دارد که سازنده این خواسته‌ها را برآورده سازد، خواسته‌های مورد انتظار گفته می‌شود که در صورت عدم برآورده‌سازی این خواسته‌ها توسط سازنده، مشتری ناراضی خواهد بود. مانند زمان پاسخ کوتاه در تعامل با نرم‌افزار توسط مشتری، یا سهولت در نصب نرم‌افزار.

خواسته‌های هیجان‌انگیز: به خواسته‌هایی که مشتری بیان نمی‌کند و انتظار هم ندارد که سازنده این خواسته‌ها را برآورده سازد، خواسته‌های هیجان‌انگیز گفته می‌شود که در صورت برآورده‌سازی این خواسته‌ها توسط سازنده، مشتری بسیار بسیار هیجان‌زده و راضی خواهد بود. مانند قابلیت چیدمان و آرایش صفحات برنامه به صورت دلخواه.

توجه: برآورده‌سازی «**خواسته‌های عادی**» و «**خواسته‌های مورد انتظار**» از سوی سازنده اجباری و معیار سنجش مشتری است و برآورده‌سازی «**خواسته‌های هیجان‌انگیز**» از سوی سازنده، اختیاری و معیار سنجش مشتری نیست ولی اگر از سوی سازنده این دسته از خواسته‌ها برآورده گردد، مشتری هیجان زده خواهد شد. راز موفقیت «استیو جابز[1]» رهبر فقید کمپانی اپل برآورده‌سازی «خواسته‌های هیجان‌انگیز» علاوه بر برآورده‌سازی «خواسته‌های عادی» و «خواسته‌های مورد انتظار» بود. موفقیت یعنی توجه کردن به جزئیات. «استیو جابز»

۳- تشریح نیازمندی‌های شناخته شده

در مرحله تشریح نیازمندی‌های شناخته‌شده، نیازمندی‌هایی که در دو مرحله شناخت اولیه نیازمندی‌ها و شناخت بیشتر نیازمندی‌ها کشف شده‌اند، با بیان ذکر جزئیات بیشتر، تشریح می‌شوند.

۴- مذاکره

در مرحله مذاکره، پس از آنکه نیازمندی‌های شناخته‌شده، تشریح شدند، نوبت به مذاکره مجدد مابین سازنده و مشتری می‌رسد، تا توافقات لازم را بر سر نهایی‌شدن نیازمندی‌های شناخته‌شده انجام دهند.

---

[1] Steve Jobs

۵- تعیین مشخصات

پس از توافقات لازم بر سر نهایی‌شدن نیازمندی‌های شناخته‌شده در مرحله مذاکره، در ادامه و در مرحله تعیین مشخصات، مشخصات سیستمی که باید ایجاد گردد، تحت عنوان لیست نیازمندی‌های مشتری نوشته می‌شود. همچنین این لیست می‌تواند به صورت گرافیکی توسط نمودار مورد کاربرد یا use case diagram مدل‌سازی شود.

توجه: نمودار مورد کاربرد یا use case diagram در فصل شیءگرایی تشریح می‌گردد.

۶- اعتبارسنجی نیازمندی‌ها

در اعتبارسنجی نیازمندی‌ها، آنچه در مرحله تعیین مشخصات حاصل گردید، جهت کنترل نهایی، توسط سازنده و مشتری مورد بررسی نهایی قرار می‌گیرد و در صورت وجود اشکالات مربوط به ناسازگاری در نظرات سازنده و مشتری، سازگاری لازم صورت می‌گیرد.

۷- مدیریت نیازمندی‌ها

نیازمندی‌های مشتری، در طول چرخه حیات نرم‌افزار مدام تغییر می‌کند، شاید تنها چیزی که در دنیا ثابت است، تغییر باشد. بنابراین در مرحله مدیریت نیازمندی‌ها، تغییراتی که در چرخه حیات نرم‌افزار حاصل می‌گردد تحت کنترل و مدیریت قرار می‌گیرد.

انواع نیازمندی‌ها

۱- وظیفه‌مندی[1] (کارکردی، عملکردی)

نیازمندی‌های وظیفه‌مندی، کمّی و قابل اندازه‌گیری هستند و در قالب قابلیت‌ها، کارکردها، ویژگی‌ها و سرویس‌های سیستم در حال توسعه یا تولید محقق می‌گردند. به عبارت دیگر، نیازمندی‌های کارکردی به بیان سرویس‌هایی که سیستم باید فراهم نماید، می‌پردازد. چگونگی واکنش سیستم در برابر ورودی‌های خاص و چگونگی رفتار سیستم در شرایط خاص نیز توسط این نیازمندی‌ها تعریف می‌شود.

مانند نیازمندی‌های مربوط به محاسبه‌ی فاکتوریل یک عدد و یا اجرای تابع فیبوناچی در یک نرم‌افزار و یا نیازمندی‌های مربوط به یک نرم‌افزار حقوق و دستمزد.

توجه: نیازمندی‌های وظیفه‌مندی، موسوم به نیازها یا «**خواسته‌های عادی**» است.

۲- غیروظیفه‌مندی[2] (غیرکارکردی، غیر عملکردی)

نیازمندی‌های غیروظیفه‌مندی، نیازمندی‌های کیفی و نه الزاماً قابل اندازه‌گیری هستند که به بیان کیفیت مورد انتظار از نیازمندی‌های وظیفه‌مندی و همچنین محدودیت‌هایی نظیر محدودیت‌های

---

[1] Functional
[2] Non-functional

زمانی، مالی و استانداردها می‌پردازند. برخی از انواع نیازمندی‌های غیروظیفه‌مندی عبارتند از:

قابلیت استفاده، سهولت یادگیری، قابلیت اعتماد، کارایی، زمان پاسخ، قابلیت پشتیبانی، قابلیت نگهداری، قابلیت حمل، بهره‌وری، محدودیت‌های طراحی، پیاده‌سازی و فیزیکی و همچنین نیازمندی‌های واسط کاربردی.

مانند محاسبه تابع فاکتوریل توسط الگوریتم حلقه با مرتبه اجرایی O(n) و یا توسط الگوریتم بازگشتی با مرتبه اجرایی O(n) و مصرف زیاد حافظه به دلیل استفاده از استک در پی هر فراخوانی.

و یا مانند محاسبه تابع فیبوناچی توسط الگوریتم حلقه با مرتبه اجرایی O(n) و یا توسط الگوریتم بازگشتی با مرتبه اجرایی نمایی زیاد $O(۲^{\frac{n}{۲}})$ و چاق و به تبع، کند و هم مصرف زیاد حافظه به دلیل استفاده از استک، در پی هر فراخوانی.

توجه: محصول نرم‌افزاری باید برآورده کننده نیازمندی‌های هم وظیفه‌مندی و هم غیروظیفه‌مندی مشتری باشد. محصول نرم‌افزاری که نیازمندی‌های وظیفه‌مندی را برآورده می‌کند، ولی برآورنده نیازمندی‌های غیروظیفه‌مندی نباشد، معمولاً با نارضایتی مشتریان همراه می‌شود.

توجه : انتخاب نوع الگوریتم براساس شرایط، حائز اهمیت می‌باشد.

توجه: نیازمندی‌های غیروظیفه‌مندی، موسوم به نیازمندی‌ها یا «**خواسته‌های مورد انتظار**» است.

۲- برنامه‌ریزی

برنامه‌ریزی یعنی هنر حرکت از مبدأ موجود به مقصد مطلوب برای رسیدن به نتیجه‌ای مطلوب براساس خواسته‌های مورد نیاز در یک زمان مشخص.

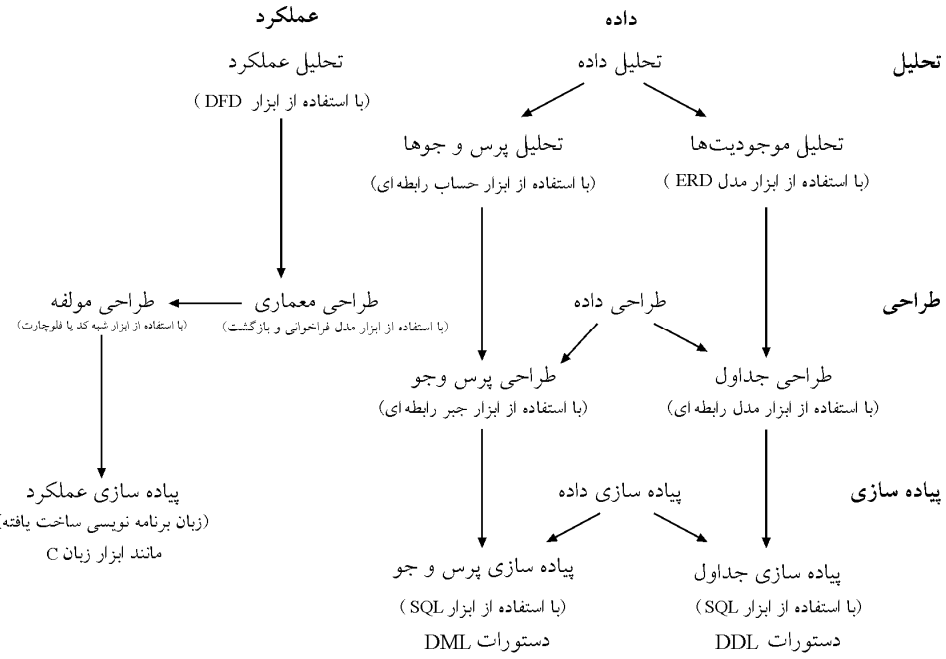
«هر تلاشی منجر به نتیجه‌ای مطلوب نمی‌گردد، بلکه این تلاشی مطلوب است که منجر به نتیجه‌ای مطلوب می‌گردد.»

لازمه‌ی تلاش مطلوب، برنامه‌ریزی است. برنامه‌ریزی می‌تواند اجرای هر کار پیچیده‌ای را ساده‌تر سازد. هر کار مهندسی مستلزم برنامه‌ریزی می‌باشد. مهندسی نرم‌افزار نیز مانند هر فعالیت مهندسی دیگری، نیازمند برنامه‌ریزی است. فعالیت برنامه‌ریزی، برنامه‌ای را برای فعالیت‌های مختلف بخش‌های مختلف فرآیند تولید نرم‌افزار پایه‌ریزی می‌کند. این فعالیت، وظیفه‌های فنی که باید هدایت شوند، ریسک‌هایی که محتمل می‌شوند (مانند عدم شناسایی برخی نیازمندی‌ها، از دست دادن داده‌ها و مدیران)، منابع مورد نیاز، واحدهای کاری که باید ایجاد شوند و برنامه زمان‌بندی برای کارها را تشریح می‌کند. مدیریت، برنامه‌ریزی را به مرحله‌ی اجرا می‌برد.

۳- مدل‌سازی (تحلیل و طراحی)

یک مدل، ساده شده یک واقعیت است. ایجاد یک مدل برای سیستم‌های نرم‌افزاری قبل از ساخت یا بازساخت آن، به اندازه داشتن نقشه برای ساختن یک ساختمان ضروری وحیاتی است.

بسیاری از شاخه‌های مهندسی، توصیف چگونگی محصولاتی که باید ساخته شوند را ترسیم می‌کنند و همچنین دقت زیادی می‌کنند که محصولاتشان طبق این مدل‌ها و توصیف‌ها ساخته شوند. مدل‌های خوب و دقیق در برقراری یک ارتباط کامل بین افراد پروژه، نقش زیادی می‌توانند داشته باشند. علت اصلی مدل کردن سیستم‌های پیچیده این است که نمی‌توان به یکباره کل سیستم را تجسم کرد و ممکن است سیستم دارای ابهامات بسیاری باشد. لذا برای رفع این ابهامات و نیز برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمت‌های مختلف آن، از مدل‌سازی استفاده می‌شود. فعالیت مدل‌سازی خود شامل دو مرحله‌ی مدل تحلیل و مدل طراحی می‌باشد. مدل تحلیل پس از فعالیت ارتباطات (جمع‌آوری نیازمندی‌ها) و قبل از مدل طراحی انجام می‌شود، در واقع خروجی مدل تحلیل، ورودی مدل طراحی می‌باشد. شکل زیر گویای این مطلب می‌باشد:



مدل تحلیل

پس از جمع‌آوری لیست نیازمندی‌های مشتری در فعالیت ارتباطات نوبت به مدل تحلیل (مدل‌سازی لیست نیازمندی‌های مشتری) می‌رسد. مدل‌سازی که فعالیتی فنی به شمار می‌رود نیازمندی‌ها را باید به گونه‌ای مدل نماید که برای سازنده و مشتری قابل فهم باشد.

در مدل تحلیل به روش ساخت‌یافته دو وجه مدل تحلیل داده و مدل تحلیل عملکرد وجود دارد. مدل تحلیل داده شامل تحلیل موجودیت‌ها و تحلیل پرس و جوها می‌باشد. تحلیل

موجودیت‌ها توسط ابزار مدل ER و تحلیل پرس و جوها توسط ابزار حساب رابطه‌ای مدل می‌شوند. مدل تحلیل عملکرد توسط ابزار DFD مدل می‌شود.

مدل طراحی

پس از مدل تحلیل، نوبت به مدل طراحی می‌رسد، مدل طراحی به روش ساخت‌یافته شامل چهار بخش **طراحی‌داده**، **طراحی‌معماری**، **طراحی‌مؤلفه** و **طراحی‌واسط** می‌باشد. طراحی داده بر دو بخش طراحی جدول و طراحی پرس و جو می‌باشد. طراحی جدول از بخش طراحی داده، تحلیل موجودیت (ERD) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط مدل رابطه‌ای، طراحی جدول را انجام می‌دهد. طراحی پرس و جو از بخش طراحی داده، تحلیل پرس و جو از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط جبر رابطه‌ای، طراحی پرس و جو را انجام می‌دهد.

طراحی معماری، تحلیل عملکرد (DFD) از مدل تحلیل را به عنوان ورودی دریافت کرده و توسط یکی از سبک‌های معماری (مانند فراخوانی و بازگشت)، طراحی معماری را انجام می‌دهد.

طراحی معماری یا معماری نرم‌افزار، ساختار کلی نرم‌افزار و شیوه‌های یکپارچگی یک سیستم را بیان می‌کند. به عبارت دیگر، ساختار سلسله مراتبی **مولفه‌های برنامه (توابع یا پیمانه‌ها)**، شیوه تعامل مولفه‌ها با یکدیگر و **ساختمان داده‌های** مورد نیاز مولفه‌ها را نشان می‌دهد. معماری نرم‌افزار یک مدل قابل درک از چگونگی سازمان‌دهی سیستم است. در واقع نشان‌گر ساختمان داده‌ها و مؤلفه‌های برنامه‌ای است که برای ساختن یک سیستم کامپیوتری لازم است. به طور دقیق‌تر معماری نرم‌افزار شامل دو سطح از طراحی می‌باشد یعنی طراحی داده و طراحی معماری. در واقع این ساختار مانند یک نقشه ساختمان، مبنای ساخت نرم‌افزار قرار می‌گیرد.

توجه: در طراحی معماری، اسکلت، ساختار و چیدمان کلی مولفه‌های (توابع) برنامه به این معنی که چه مولفه‌ای (تابعی) چه مولفه‌ای (تابعی) دیگر را صدا می‌زند، بدون ذکر جزئیات داخلی مولفه‌ها (توابع) مشخص می‌گردد (ساختار درختی برنامه بدون ذکر جزئیات مولفه‌ها (توابع)). مانند اسکلت یک ساختمان که گویای جایگاه مؤلفه‌های ساختمان است اما هنوز آجرچینی نشده است. (اسکلت یک ساختمان بدون آجرچینی).

توجه: به طراحی معماری، طراحی کلی نیز گفته می‌شود.

توجه: طراحی معماری ساخت‌یافته در فصل مفاهیم طراحی ساخت‌یافته، بیشتر توضیح داده خواهد شد.

توجه: سبک معماری فراخوانی و بازگشت جلوتر شرح داده خواهد شد.

طراحی مؤلفه، طراحی معماری از همان فعالیت مدل طراحی را به عنوان ورودی دریافت کرده و طراحی مؤلفه را توسط ابزارهایی همچون **شبه کد** یا **فلوچارت** ایجاد می‌کند.

طراحی مولفه، فعالیت تبدیل طراحی معماری به نرم‌افزار است. در این مرحله، سطح انتزاع

طراحی معماری به سطح انتزاع نرم‌افزار کاربردی نزدیک می‌گردد. طراحی در سطح مؤلفه‌ها، نرم‌افزار را در سطحی از انتزاع تصویر می‌کند که به کد نزدیک است. طراحی مولفه، به عنوان نقشه راهی دقیق، و نزدیک به زبان پیاده‌سازی، در فعالیت پیاده‌سازی نرم‌افزار، منجر به صرفه جویی در زمان و هزینه‌های تولید می‌گردد. در طراحی مولفه، مهندس نرم‌افزار باید ساختمان داده‌ها، واسط‌ها و الگوریتم‌ها را با جزییات کافی به نمایش در آورد تا راهنمای تولید کد منبع زبان برنامه‌نویسی باشد.

توجه: در طراحی مؤلفه، اسکلت، ساختار و چیدمان کلی مولفه‌های (توابع) برنامه به این معنی که چه مولفه‌ای (تابعی) چه مولفه‌ای (تابعی) دیگر را صدا می‌زند، با ذکر جزئیات داخلی مولفه‌ها (توابع) مشخص می‌گردد (ساختار درختی برنامه با ذکر جزئیات مولفه‌ها (توابع)). مانند اسکلت یک ساختمان که گویای جایگاه مؤلفه‌های ساختمان است و آجرچینی هم شده است. (اسکلت یک ساختمان به همراه آجرچینی).

توجه: به طراحی مولفه، طراحی جزئی، طراحی تفصیلی و طراحی رویه‌ای نیز گفته می‌شود.

طراحی واسط یا همان واسط کاربر، براساس ورودی‌ها و خروجی‌های مورد نیاز کاربران نهایی به شکل نقشی بر روی کاغذ یا طرحی بر روی کامپیوتر ایجاد می‌گردد. مانند نحوه چیدمان منوها و فرم‌ها.

۴- ساخت (پیاده‌سازی و تست)

پس از مدل طراحی نوبت به پیاده‌سازی و تست می‌رسد. پیاده‌سازی جداول از بخش پیاده‌سازی داده، طراحی جدول از مدل طراحی را به عنوان ورودی دریافت کرده و توسط دستورات DDL در SQL، پیاده‌سازی جداول را انجام می‌دهد.

پیاده‌سازی پرس و جو از بخش پیاده‌سازی داده، طراحی پرس و جو از مدل طراحی را به عنوان ورودی دریافت کرده و توسط دستورات DML در SQL پیاده‌سازی پرس و جو را انجام می‌دهد. پیاده‌سازی عملکرد، طراحی مؤلفه از مدل طراحی را به عنوان ورودی دریافت کرده و توسط یک زبان برنامه‌نویسی (ساخت‌یافته یا شیءگرا) پیاده‌سازی عملکرد را انجام می‌دهد.

توجه: دقت‌کنید که می‌توان مدل تحلیل و طراحی را به روش و ابزارهای ساخت‌یافته انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان برنامه‌نویسی شیءگرا انجام داد و از امکانات شیءگرایی زبان استفاده نکرد، اما عکس این مطلب امکان‌پذیر نیست، یعنی نمی‌توان مدل تحلیل و طراحی را به روش شیءگرا انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان ساخت‌یافته انجام داد زیرا زبان ساخت‌یافته امکانات شیءگرایی (همچون کلاس، وراثت و چندریختی) را پشتیبانی نمی‌کند.

پس از پیاده‌سازی نوبت به تست می‌رسد، در این مرحله کلیه‌ی موارد پیاده‌سازی شده از نظر خطاهای نحوی و خطاهای معنایی براساس لیست نیازمندی‌های مشتری (چک لیست) که در

فعالیت ارتباطات تهیه شده بود مورد وارسی قرار می‌گیرد تا مشخص شود نرم‌افزار براساس ورودی‌های مورد نظر مشتری، خروجی‌های مورد انتظار مشتری را برآورده می‌سازد یا خیر.

۵- استقرار

پس از فعالیت تست نوبت به فعالیت استقرار می‌رسد. در این مرحله، نرم‌افزار به مشتری تحویل داده می‌شود و مشتری با بررسی محصول دریافتی، بازخوردهای به دست آمده براساس همین ارزیابی‌ها را به تیم نرم‌افزاری ارائه می‌دهد. این بازخوردها می‌توانند مبنایی برای ارتقاء و یا تصحیح نسخه‌ی بعدی نرم‌افزار باشد.

این پنج فعالیت چارچوبی را می‌توان طی تولید برنامه‌های کوچک و ساده، در ایجاد برنامه‌های تحت وب و برای مهندسی سیستم‌های کامپیوتری پیچیده و عظیم به کار برد. جزئیات مدل‌های فرآیند تولید نرم‌افزار در هر مورد کاملاً متفاوت خواهد بود، ولی فعالیت‌های چارچوبی همین‌ها خواهد بود.

برای بسیاری از پروژه‌های نرم‌افزاری، فعالیت‌های چارچوبی به موازات پیشرفت پروژه به صورت تکراری به کار برده می‌شوند. یعنی فعالیت‌های ارتباطات، برنامه‌ریزی، مدل‌سازی، ساخت و استقرار به طور مکرر در چند دور تکرار پروژه به کار برده می‌شوند. در هر دور تکرار پروژه، یک نسخه (افزایش)[1] از نرم‌افزار ایجاد می‌شود که زیرمجموعه‌ای از قابلیت‌های عملیاتی و ویژگی‌های نرم‌افزار کامل را در اختیار مشتری قرار می‌دهد. با تولید هر افزایش، نرم‌افزار کامل و کامل‌تر می‌شود.

توجه: فعالیت‌های چارچوبی فرآیند تولید نرم‌افزار براساس متدولوژی شیءگرا در فصل شیءگرایی به تفصیل شرح داده خواهد شد.

فعالیت‌های چتری فرآیند تولید نرم‌افزار

انجام درست فعالیت‌های چارچوبی تا حدود بسیار زیادی تحت کنترل انسان می‌باشد ولی برای تولید موفق یک پروژه‌ی نرم‌افزاری به تنهایی کافی نیستند. زیرا در وادی زندگی علاوه بر قوانین انسانی، قوانین طبیعی نیز وجود دارند، اگر فعالیت‌های چارچوبی درست بودند، اما اطلاعات موجود در هارد دیسک به دلیل عوامل طبیعی از بین رفتند، چه کار کنیم، اگر فعالیت‌های چارچوبی درست بودند، اما به دلیل ماهیت انسانی بودن یک انسان و عوامل طبیعی همچون مرگ و میر و زلزله، مدیران و همکاران خود را از دست دادیم، چه کار کنیم و واقعاً اگر همه‌ی موارد تحت کنترل انسان برای رسیدن به موفقیت درست بودند، عوامل طبیعی که خارج از کنترل انسان هستند را چه کار کنیم ...

[1] Increment

به قول حضرت حافظ

در بیابان‌گر به شوق کعبه خواهی زد قدم سرزنش‌هاگر کند خار مغیلان غم مخور

در وادی مهندسی نرم‌افزار، توسط فعالیت‌های چتری، عوامل طبیعی خارج از کنترل انسان و هر آنچه مربوط به موفقیت پروژه باشد، نظارت و تحت کنترل دقیق قرار می‌گیرد، در واقع فعالیت‌های چارچوبی فرآیند تولید نرم‌افزار توسط تعدادی از فعالیت‌های چتری تکمیل می‌شود. به طور کلی، فعالیت‌های چتری در سرتاسر یک پروژه‌ی نرم‌افزاری به کار برده می‌شوند و به تیم نرم‌افزاری کمک می‌کنند تا پیشرفت، کیفیت، تغییر و ریسک را کنترل کنند. به بیان دیگر فعالیت‌های چتری بر محقق شدن خصوصیات پروژه‌های موفق نرم‌افزاری (بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و با صرف کمترین هزینه و دقیقاً مطابق با نیازمندی‌های واقعی کاربران) در حین انجام فعالیت‌های چارچوبی، تأکید می‌کند.

انواع فعالیت‌های چتری

۱- کنترل و پیگیری پروژه‌ی نرم‌افزاری

برای تحویل به موقع محصول، نیاز به یک زمان‌بندی اولیه است، اما از آن مهم‌تر کنترل و پیگیری این زمان‌بندی در طول پروژه است. بنابراین کنترل و پیگیری پروژه‌ی نرم‌افزاری به تیم پروژه امکان می‌دهد تا از پیشرفت واقعی پروژه با توجه به برنامه زمان‌بندی شده آگاه شده و در صورت لزوم اقدامات لازم را برای حفظ برنامه‌ی زمان‌بندی انجام دهند. در واقع در این فعالیت مقایسه می‌شود تا در صورت لزوم و در مواردی که از زمان زمان‌بندی عقب هستیم، کارهایی برای جبران این عقب ماندگی انجام پذیرد.

۲- مدیریت ریسک

گام اول مدیریت ریسک، شناسایی ریسک (تحلیل ریسک) است و گام دوم مقابله با ریسک، برای مقابله با ریسک باید هزینه کرد. مثلاً ریسک از دست دادن اطلاعات را می‌توان با هزینه و خریداری یک سیستم پشتیبان‌گیری از اطلاعات مرتفع نمود یا ریسک از دست دادن مدیران را می‌توان با هزینه و استخدام یک نیروی انسانی پشتیبان در کنار مدیران دارای درجه اهمیت بالا مرتفع نمود. در واقع در این فعالیت، ریسک‌های محتملی که ممکن است بر روی خروجی‌های پروژه و یا کیفیت محصول نهایی پروژه یا به عبارت دقیق‌تر بر روی خصوصیات پروژه‌های موفق نرم‌افزاری (نیاز، زمان و هزینه) تأثیر ناگواری بگذارند شناسایی، مدیریت و در صورت امکان تقلیل می‌یابند. دقت‌کنید که احتمال وقوع ریسک، تابعی از زمان است، بنابراین مدیریت ریسک می بایست در سراسر چرخه‌ی حیات نرم‌افزار، حضوری فعال و پررنگ داشته باشد.

۳- تضمین کیفیت نرم‌افزار (SQA : Software Quality Assurance)

فعالیت‌های لازم، کافی و دقیق در فعالیت‌های مختلف چارچوب فرآیند تولید نرم‌افزار

(ارتباطات، برنامه‌ریزی، مدل‌سازی، ساخت و استقرار) برای حصول اطمینان از کیفیت نرم‌افزار (نرم‌افزاری مطابق با خواسته‌های مشتری) را معین می‌کند. اگر فعالیت‌های مربوط به فازهای مختلف چارچوب فرآیند تولید نرم‌افزار درست باشند، نتایج خودشان درست خواهند بود.

۴- بازبینی‌های فنی (FTR : Formal Technical Review)

تمامی فرآورده‌های تولید شده در فعالیت‌های مختلف چارچوب فرآیند تولید نرم‌افزار برای آشکار کردن خطاها قبل از انتشار آنها در فعالیت بعدی و برطرف کردن آنها مورد ارزیابی و بازبینی قرار می‌گیرند. به بیان دیگر بعد از انجام هر مرحله از توسعه‌ی نرم‌افزار، نتایج حاصله مورد ارزیابی قرار می‌گیرند تا خطاهای مستقر در این مرحله به مراحل بعد انتشار نیابند. این ارزیابی توسط یک گروه از افراد پروژه انجام شده و باعث افزایش کیفیت نرم‌افزار می‌شود.

۵- اندازه‌گیری

در این فعالیت، اندازه‌ها، معیارها و شاخص‌های محصول، پروژه و فرآیند، تعریف و جمع‌آوری می‌شوند که با استفاده از این اطلاعات، مدیر و تیم پروژه قادر به تحویل نرم‌افزار با کیفیت بالا و مطابق با استانداردهای از پیش تعیین شده می‌باشند. در واقع این اطلاعات سبب مدیریت بهتر پروژه‌های نرم‌افزاری و تطابق دادن آن با استانداردها و به دست آمدن محصولی با کیفیت بالاتر می‌شود. به بیان دیگر با استفاده از فعالیت اندازه‌گیری، **معیارهایی کمّی**، برای ارزیابی و پیشرفت **فرآیند، محصول (پروژه)** ارائه می‌شود.

۶- مدیریت پیکربندی نرم‌افزار (Software Confiquration Management)

اثرات هرگونه تغییرات را در سرتاسر فرآیند تولید نرم‌افزار مدیریت می‌کند. مدیریت پیکربندی نرم‌افزار را می‌توان معادل مدیریت و کنترل تغییرات در نظر گرفت. بنابراین مدیریت پیکربندی نرم‌افزار همانند مدیریت تغییرات، هم روی تغییراتی که پس از تحویل محصول به مشتری رخ می‌دهند، اعمال می‌شود و هم تغییراتی که قبل از تحویل به مشتری رخ داده‌اند را کنترل می‌کند.

۷- مدیریت قابلیت استفاده مجدد

ضوابطی برای محصول کاری که قابلیت استفاده مجدد دارد (که شامل تمام مؤلفه‌ها می‌شود) تعریف شده و نیز مکانیزمی برای تولید مؤلفه‌هایی با قابلیت استفاده‌ی مجدد پایه‌ریزی می‌شود. در این دیدگاه، مؤلفه‌ی در حال تولید، باید هم نیازهای پروژه‌ی فعلی را برطرف سازد و هم نیازهای احتمالی پروژه‌های مشابه که در آینده تولید خواهند شد را پوشش دهد.

۸- تولید و پیش تولید محصولات کاری

شامل فعالیت‌های لازم برای تولید محصولات کاری مانند مدل‌ها، مستندات، فرم‌ها و لیست‌ها می‌شود.

مدل‌های فرآیند تولید نرم‌افزار

فرآیند تولید نرم‌افزار، مجموعه‌ای از فعالیت‌های چارچوبی (ارتباطات، برنامه‌ریزی، مدل‌سازی، ساخت و استقرار) است که هدفشان تولید نرم‌افزاری با کیفیت و مطابق با خواسته‌های مورد انتظار مشتری می‌باشد. مدل فرآیند تولید نرم‌افزار براساس ماهیت نرم‌افزاری که قرار است تولید شود انتخاب می‌گردد. همه‌ی مدل‌های فرآیند تولید نرم‌افزار (ساخت‌یافته و شیءگرا) از فعالیت‌های چارچوبی پیروی می‌کنند اما در جریان کار شباهت‌ها و تفاوت‌هایی دارند. مدل‌های فرآیند تولید نرم‌افزار بر دو طبقه‌ی سنتی و مدرن هستند.

مدل‌های فرآیند تولید نرم‌افزار سنتی (ساخت‌یافته)

مدل‌های فرآیند تولید نرم‌افزار سنتی بر دو دسته‌ی کلی غیرتکاملی سنتی و تکاملی سنتی هستند:

مدل‌های غیرتکاملی سنتی

مدل‌های فرآیند غیرتکاملی سنتی یا تجویزی[1] در ابتدا برای نظم بخشیدن به فرآیند تولید نرم‌افزار پیشنهاد شدند. این مدل‌های سنتی به میزان نسبتاً قابل قبولی به کار مهندسی نرم‌افزار ساختار بخشیده‌اند و راهنمای اثربخشی برای تیم‌های نرم‌افزاری بوده‌اند. این مدل‌ها را تجویزی می‌نامند، زیرا مجموعه‌ای از فعالیت‌های چارچوبی و چتری را برای هر پروژه تجویز می‌کنند. در این مدل‌ها، تولید نرم‌افزار مطابق فعالیت‌های چارچوبی، مراحل مختلفی دارد که هر مرحله دارای ورودی، فعالیت و خروجی خاص خود می‌باشد. خروجی هر مرحله در این مدل‌ها، ورودی مرحله بعدی است و از فعالیت ارتباطات تا استقرار ادامه می‌یابند.

۱- مدل آبشاری[2]

مدل آبشاری، که گاه از آن به عنوان چرخه‌ی حیات کلاسیک یاد می‌شود، روشی ترتیبی برای تولید نرم‌افزار پیشنهاد می‌کند. این مدل با فعالیت ارتباطات شروع می‌شود و با فعالیت‌های برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی) و ساخت (پیاده‌سازی و تست) پیش می‌رود و با فعالیت استقرار پایان می‌یابد، تحویل پروژه انجام می‌گیرد و کار تمام می‌شود. و دیگر فرصت و تکرار دیگری در کار نخواهد بود تا خواسته‌های فراموش شده یا جدید به پروژه اضافه گردد.

خصوصیت اصلی این مدل این است که هیچ‌گونه بازخوردی بین مراحل این مدل وجود ندارد. مانند آب که نمی‌تواند در آبشار به عقب برگردد، در این مدل نیز بعد از ورود به یک فعالیت به فعالیت‌های قبلی نمی‌توان بازگشت. این مدل زمانی کاربرد دارد که کلیه‌ی نیازمندی‌های مشتری

---

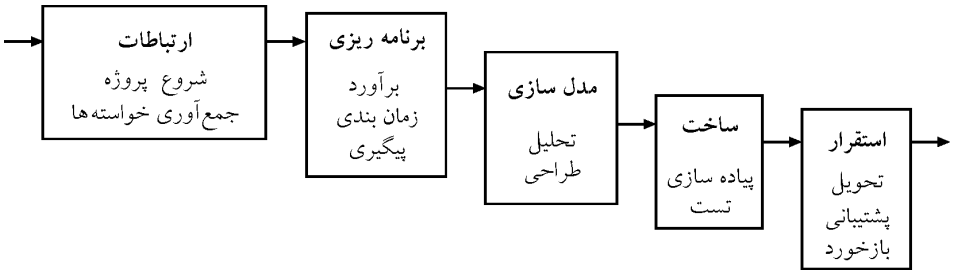
[1] Prescriptive
[2] Waterfall model

در همان ابتدای پروژه مشخص، ثابت و بدون تغییر باشد. بنابراین این مدل در تولید پروژه‌های کوچک ساده (مانند سیستم حقوق و دستمزد واحد حسابداری که نیازمندی‌های مشخص و ثابتی دارد.) و یا تولید مجدد پروژه‌های بزرگ و حتی پیچیده‌ی از قبل از ساخته شده (به دلیل مشخص بودن لیست نیازمندی‌ها در تولید مجدد) کارایی بالایی دارد و کارآمد خواهد بود.

اما در پروژه‌های بزرگ و پیچیده به دلیل عدم مشخص بودن تمام نیازمندی‌ها در ابتدای پروژه و حتی تغییرات نیازمندی‌ها در حین انجام پروژه کارایی پایینی خواهد داشت. مدل آبشاری در شرایطی که خواسته‌ها به طور کامل و جامع مشخص، ثابت و پایدار است و قرار است که کار تا پایان به شیوه‌ای خطی پیش برود، می‌تواند به عنوان مدلی مفید، مورد استفاده قرار گیرد.

برای مثال، در بسیاری از پروژه‌های ساختمان‌سازی، نیازمندی‌ها از قبل مشخص هستند، بنابراین فرآیند تولید پروژه را می‌توان براساس مدل آبشاری بنا نهاد. اما در دنیای نرم‌افزار چنین پروژه‌هایی به ندرت وجود دارد. به گونه‌ای که بسیاری از متخصصان نرم‌افزار بر این عقیده‌اند که تمام پروژه‌هایی که نیازمندی‌هایش به طور جامع و کامل مشخص باشند، قبلاً توسط دیگران انجام شده‌اند!

توجه: به مدل آبشاری، مدل خطی (Linear Model) و مدل ترتیبی (Sequential Model) نیز گفته می‌شود.



مدل آبشاری

معایب مدل آبشاری

۱- پروژه‌های واقعی به ندرت جریان ترتیبی پیشنهاد شده توسط این مدل را دنبال می‌کنند. ترتیبی بودن روال فعالیت‌های ارتباطات تا استقرار در این مدل نیازمند مشخص بودن تمامی نیازمندی‌های پروژه در ابتدای کار می‌باشد اما ماهیت اغلب پروژه‌های نرم‌افزاری بدین گونه نیست که تمامی نیازمندی‌ها در ابتدای پروژه مشخص باشند. بنابراین این مدل در مواردی که همه‌ی نیازمندی‌ها در ابتدای پروژه مشخص نباشد به دلیل ماهیت ترتیبی بودن مدل و عدم بازگشت به عقب کارایی لازم را نخواهد داشت. زیرا لازم است همه چیز از ابتدا مشخص باشند، که اغلب محال است! در یک بیان ساده اینطور می‌توان بیان کرد که ماهیت اغلب پروژه‌های نرم‌افزاری بدین

شکل است که به ندرت پیش می‌آید که یک مرحله را به طور کامل تمام کنند و وارد مرحله بعدی شوند. از آنجا که در اغلب پروژه‌های نرم‌افزاری نیازمندی‌ها ذره ذره شناسایی می‌شوند و منجر به تکامل نرم‌افزار در طی فعالیت‌های چارچوبی بعدی می‌شوند، مدل آبشاری در این موارد کارا نخواهد بود.

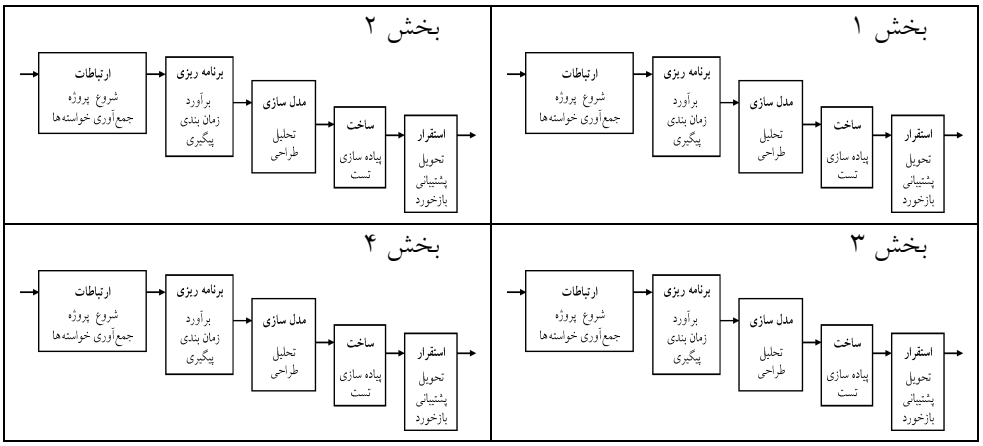
۲- اغلب برای مشتری دشوار است که همه‌ی نیازهای خود را در همان ابتدای کار بیان کند، او دوست دارد برخی از نیازمندی‌ها را در ابتدای کار بیان کند و برخی دیگر را در حین انجام کار. این علاقه و دوست داشتن مشتری با مدل آبشاری برآورده نمی‌شود. اغلب، شناسایی نیازمندی‌های واقعی سیستم، یک فرآیند مستمر است که این مورد با ماهیت مدل آبشاری سازگاری ندارد. به جز در پروژه‌های تولید سیستم‌های خاص که بارها و بارها نمونه‌ی مشابه‌شان تولید شده است، نیازمندی‌های یک سیستم را اغلب نمی‌توان قبل از شروع فرآیند تولید و پیش از اقدام به پیاده‌سازی آن، به طور کامل و جامع شناسایی نمود.

۳- مشتری باید حوصله داشته باشد. زیرا نسخه‌ی عملیاتی را دیر می‌بیند. به عبارت دیگر تا پایان تولید نهایی نرم‌افزار امکان دیدن آن توسط مشتری وجود ندارد. بنابراین، مشتری باید تا پایان کار پروژه صبور باشد، در این میان، ایجاد یک نقص در برنامه‌ی تولید شده و یا تولید چیزی غیر از خواسته‌های مشتری ممکن است فاجعه بار باشد، زیرا مشتری پس از فعالیت ارتباطات و بیان خواسته‌ها، دیگر در حین فعالیت‌های بعدی پروژه حضور نداشته است. بنابراین مهمترین مشکل رویکرد آبشاری، ضعف ذاتی آن در غلبه بر ریسک است. در اینجا، منظور از ریسک، کلیه‌ی شرایط، عوامل و نگرانی‌هایی است که می‌توانند مانع از دستیابی به موفقیت (شناسایی نیازهای دقیق مشتری، مقرون به صرفه بودن و در زمان مورد انتظار بودن) شوند. بسیاری از ریسک‌ها تنها در زمان پیاده‌سازی و تست آشکار می‌شوند. از آنجایی که در مدل آبشاری، پیاده‌سازی و تست سیستم به انتهای پروژه موکول می‌شود، در صورت آشکار شدن یک ریسک، فرصت کمی برای مدیریت آن وجود خواهد داشت و اغلب هزینه‌های زیادی برای مقابله با آن باید صرف شود. برای مثال، وجود نقص در مدل طراحی ممکن است ناشی از وجود نقص در مدل تحلیل و شناسایی نیازمندی‌ها باشد. این مشکل تنها در زمان پیاده‌سازی و تست، آشکار می‌گردد، یعنی زمانی که تصحیح آن باعث افزایش هزینه‌ها و طولانی‌تر شدن پروژه و یا حتی بسته شدن و شکست آن می‌شود. ضعف مدل آبشاری در مواجه با ریسک‌های پیچیده در پروژه‌های امروزی، مهمترین عامل انزوای این مدل در دنیای مهندسی نرم‌افزار مدرن است. فرض کنید مشتری سفارش ساخت یک میز را به نجار می‌دهد و نجار به سرعت آن را می‌سازد اما بعداً، از در اتاق مورد نظر خانه‌ی مشتری عبور نمی‌کند. زیرا مشتری و سازنده یادشان رفت بر سر اندازه‌های میز توافق کنند. این یعنی فاجعه، البته در این مورد خاص کمی فاجعه! این است که می‌گوییم اغلب مشتری در همان ابتدای کار نمی‌داند چه می‌خواهد یا یادش می‌رود دقیقاً چه می‌خواهد، انسان است و جایزالخطا بودنش، بنابراین باید بپذیریم که انسان فراموش می‌کند دقیقاً چه می‌خواهد، اما دوست دارد به او

فرصت دهیم تا در حین کار و با دیدن نمونه‌هایی از پروژه بقیه‌ی خواسته‌هایش را بگوید. او اینطور خوشحال می‌شود. مدل‌های بعدی سعی در خوشحال نمودن مشتری دارند.

مدل توسعه سریع RAD[1]

مدل RAD، شکل پُرسرعت مدل آبشاری می‌باشد، با این تفاوت که پروژه به بخش‌های مختلف تقسیم شده و هر بخش، توسط یک تیم، مطابق مدل آبشاری ایجاد می‌گردد و در پایان نتیجه‌ی تیم‌ها، برای خلق محصول نهایی ترکیب می‌گردد. مدل RAD، سرعت خود را مدیون بهره‌گیری از تکنیک بخش‌بندی و موازی‌سازی بخش‌های مختلف پروژه است. چنانچه نیازمندی‌ها به خوبی شناسایی شده و دامنه پروژه کوچک باشد این مدل قادر است یک سیستم کاملاً عملیاتی را در مدت زمان بسیار کوتاه (مثلاً بین ۶۰ تا ۹۰ روز) تولید نماید. در این مدل نرم‌افزار به قسمت‌های مختلف تقسیم شده و همواره سعی می‌شود که نرم‌افزار موردنظر سریع‌تر تولید شود. نکته قابل توجه این است که نرم‌افزار موردنظر باید خاصیت تفکیک‌پذیری داشته باشد تا بتوان این مدل را پیاده‌سازی کرد.



مدل RAD

ویژگی‌های مدل RAD

۱- شرط لازم برای انجام پروژه‌های نرم‌افزاری توسط مدل RAD، قابلیت بخش‌بندی پروژه است.

۲- از آنجا که در مدل RAD، هر بخش از مدل آبشاری استفاده می‌کند، پس بنا بر ویژگی‌های مدل آبشاری، باید تمامی نیازمندی‌های پروژه (لیست نیازمندی‌ها) در ابتدای پروژه مشخص باشد.

[1] Rapid Application Development

در این صورت این مدل می‌تواند ظرف مدت بسیار کوتاهی (۶۰ تا ۹۰ روز) محصول نهایی را ایجاد نماید.

۳- در پروژه‌های بزرگ، تعداد بخش‌های مختلف پروژه زیاد می‌شود، به همین دلیل به تیم‌های نرم‌افزاری بیشتری نیاز خواهد بود. بنابراین با افزایش حجم پروژه باید نیروی انسانی کافی برای پیمانه‌ها وجود داشته باشد.

۴- این مدل در پروژه‌هایی با ریسک‌های فنی بالا، به دلیل عدم امکان شناسایی نیازمندی‌های مشتری در ابتدای پروژه کارآمد نخواهد بود.

۵- موفقیت مدل RAD، وابسته به تعامل مناسب سازنده و مشتری است و هر دو باید برای انجام سریع فعالیت‌ها با یکدیگر هماهنگ باشند تا بتوانند در موعد مقرر تولید نهایی را تحویل مشتری دهند. چون اگر یک قسمت از این پروژه انجام نشده باشد. تحویل پروژه میسر نیست، بنابراین مدیریت این مدل اهمیت فراوانی دارد.

مکانیزم نمونه‌سازی دورانداختنی

همانطور که پیش از این نیز گفتیم، علاوه بر ابزارهای مشاهده، مصاحبه و گفتگو، مکانیزم نمونه‌سازی دورانداختنی نیز برای شناسایی نیازمندی‌های مشتری در فعالیت ارتباط مورد استفاده قرار می‌گیرد. در مکانیزم نمونه‌سازی دورانداختنی، یک پیاده‌سازی عملیاتی از سیستم با ابزارهای ارزان‌قیمت، فقط و فقط به منظور شناسایی نیازمندی‌های مشتری، ایجاد و سپس دور انداخته می‌شود، سپس سیستم نهایی براساس فرآیند تولید مجزایی تولید می‌گردد، توجه کنید که سیستم نهایی براساس فرآیند تولید مجزای دیگری تولید می‌گردد، مثلاً پس از شناسایی تمامی نیازمندی‌های مشتری توسط مکانیزم نمونه‌سازی دورانداختنی، لیست تمامی نیازمندی‌های مشتری آماده است، بنابراین در ادامه برای تولید نرم‌افزار از مدل آبشاری می‌توان استفاده نمود.

هدف از مکانیزم دورانداختنی، استخراج نیازمندی‌های مشتری است. شروع مکانیزم نمونه‌سازی دورانداختنی براساس نیازمندی‌هایی است که کمتر درک شده‌اند. یکی از مزایای این روش، کاهش ریسک نیازمندی‌ها است.

توجه: به مکانیزم‌نمونه‌سازی دورانداختنی، **الگوسازی بسته** نیز گفته می‌شود.

توجه: در ادامه و در سراسر این کتاب مکانیزم نمونه‌سازی دورانداختنی را، به اختصار «نمونه‌سازی» در نظر خواهیم گرفت.

فرض کنید مشتری سفارش ساخت یک میز با چوب گران قیمت را به یک نجار می‌دهد و نجار به سرعت آن را می‌سازد. اما بعداً از در اتاق مورد نظر مشتری عبور نمی‌کند و با وجود صرف وقت و هزینه، نیاز مشتری برای داشتن یک میز در اتاقش برآورده نمی‌شود. زیرا مشتری و سازنده یادشان رفت بر سر اندازه‌های میز توافق کنند، این یعنی فاجعه. البته در این مورد خاص کمی فاجعه. جملاتی که بیان کردیم در ذهن خود تجسم کنید. خب یادشان رفت، چه کار می‌توان

کرد، انسان است و جایزالخطا بودنش.

اما آیا نمی‌توان به آنها اشکال گرفت که چرا وقتی مطمئن نبودید که دقیقاً چه می‌خواهید و چه می‌خواهید بسازید با هم بر سر ساخت توافق کردید؟!

این یک مثال خیلی ساده بود، اغلب، انجام پروژه‌های نرم‌افزاری بزرگ صرف وقت و هزینه‌های سرسام‌آوری دارد و اگر نرم‌افزاری ساخته شود که در زمان مورد انتظار به کار مشتری نیاید می‌توان قاطع گفت که اینجا واقعاً فاجعه رخ داده است. فرض کنید نرم‌افزاری می‌بایست برای انتخابات ریاست جمهوری که در یک تاریخ از قبل مشخص شده برگزار می‌گردد، می‌رسید، اما نمی‌رسد!

زیرا براساس خواسته‌های دقیق وزارت کشور به عنوان مجری انتخابات ساخته نشده است و حتی نیازهایی فراموش شده است. روز انتخابات فرا می‌رسد، اما نرم‌افزار آماده نیست، فرصت برای جبران هم نیست، این یعنی فاجعه، به معنای واقعی کلمه.

اما چه می‌توان کرد؟ راهکار چیست؟ در یک جمله، راهکار نمونه‌سازی است. نمونه‌سازی راهکاری برای تشخیص دقیق خواسته‌های مشتری است.

در مثال مشتری و نجار آیا بهتر نبود، ابتدا میز با چوب ارزان قیمت ساخته می‌شد و پس از مشاهده مشتری و میزان رضایتمندی او و حتی ایجاد فرصت برای بیان خواسته‌های جدیدش، و بعد از مشخص شدن دقیق و کامل لیست نیازمندی‌های مشتری، ساخت میز با چوب گران قیمت آغاز می‌گردید؟

در مثال نرم‌افزار انتخابات ریاست جمهوری آیا بهتر نبود ابتدا توسط ابزارهای ارزان و برنامه‌نویسان ارزان و به تبع هزینه پایین، نمونه‌هایی سریع از بخش‌های مختلف نرم‌افزار آماده می‌شد و پس از مشاهده وزارت کشور و میزان رضایت‌مندی و حتی بیان خواسته‌های جدید و مشخص شدن دقیق لیست نیازمندی‌های وزارت کشور، ساخت نرم‌افزار با ابزارهای گران قیمت و برنامه‌نویسان گران قیمت براساس لیست دقیق نیازمندی‌های وظیفه‌مندی و غیروظیفه‌مندی وزارت کشور آغاز می‌گردید؟ پاسخ مثبت است، بله، واضح است که بهتر بود.

هرگاه برای شناسایی خواسته‌های مشتری و تهیه‌ی لیست نیازی‌ها ابهام داشتید، نمونه‌سازی کنید، نمونه‌ای ارزان قیمت از بخش‌های مختلف پروژه توسط ابزارهای ارزان و برنامه‌نویسان ارزان بسازید نشان مشتری دهید. کم‌کم، کم‌کم بقیه خواسته‌هایش را هم می‌گوید.

یادتان باشد در حال ساخت نمونه برای مشاهده مشتری و شناسایی مابقی نیازمندی‌های او هستید، لیست نیازمندی‌های مشتری که دقیقاً مشخص شد و بعد به توافق نهایی هم رسیدید، نمونه‌های ساخت شده را دور بریزید، زیرا به هدفی که می‌خواستید رسیدید، نیازها مشخص شدند و دیگر به نمونه‌ها نیازی ندارید!

نگران نباشید، دور بریزید، شما با این راهکار موفق شدید به طور دقیق بدانید مشتری واقعاً چه

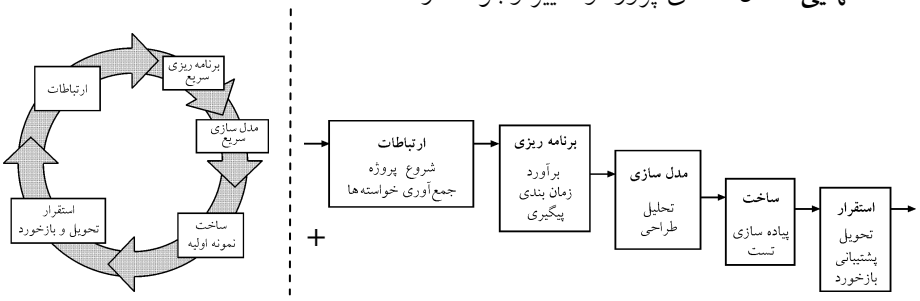
می‌خواهد. **تا همین جا موفقیت بزرگی نصیب شما شده است!**

حال ساخت محصول نهایی را به برنامه‌نویسان گران قیمت خود بسپارید، با خیالی آسوده، زیرا این بار براساس لیست دقیق نیازمندی‌های وظیفه‌مندی و غیروظیفه‌مندی، دقیقاً همان چیزی در حال تولید است که مشتری می‌خواهد.

مدل نمونه‌سازی

شرح مطالب مربوط به مکانیزم نمونه‌سازی دورانداختنی را به یاد آورید، در آنجا گفتیم که در مکانیزم نمونه‌سازی دورانداختنی، یک پیاده‌سازی عملیاتی از سیستم با ابزارهای ارزان قیمت فقط و فقط به منظور شناسایی نیازمندی‌های مشتری، ایجاد و سپس دور انداخته می‌شود، سپس سیستم نهایی براساس فرآیند تولید مجزایی تولید می‌گردد، همچنین تأکید کردیم که سیستم نهایی براساس فرآیند تولید مجزای دیگری تولید می‌گردد. اگر آن فرآیند مجزای دیگر، مدل آبشاری باشد، به حاصل جمع مکانیزم نمونه‌سازی دورانداختنی و مدل آبشاری «مدل نمونه‌سازی» گفته می‌شود.

در واقع مدل نمونه‌سازی از ترکیب مکانیزم نمونه‌سازی دورانداختنی و مدل آبشاری ایجاد شده است. در ابتدای کار توسط مکانیزم نمونه‌سازی دورانداختنی تمامی لیست نیازمندی‌های مشتری تکمیل می‌گردد، سپس توسط مدل آبشاری به شکل خطی نرم‌افزار ایجاد می‌گردد و کار تمام می‌شود. در واقع نیازمندی‌های مشتری طی تکرارهای مکانیزم نمونه‌سازی دورانداختنی شناخته می‌شوند، سپس طی یک روال خطی، توسط مدل آبشاری، نرم‌افزار تولید می‌گردد و کار تمام می‌شود و دیگر تکامل نمی‌یابد زیرا مدل نمونه‌سازی یک مدل غیر تکاملی است و **پس از ساخت نهایی** امکان ادامه‌ی پروژه و تغییر وجود ندارد.



مدل نمونه سازی

اغلب، مشتری مجموعه‌ای از اهداف کلی را برای نرم‌افزار تعریف می‌کند، اما جزئیات ورودی، پردازش یا شرایط مورد نیاز خروجی را تعریف نمی‌کند. در حالت‌های دیگر، سازنده ممکن است از کارایی الگوریتمی خاص، یا توانایی یک سیستم عامل، یا نحوه‌ی تعامل کاربران نهایی و نرم‌افزار اطمینان نداشته باشد، اگر نرم‌افزار بر مبنای همین نیازمندی‌های مبهم ایجاد گردد ممکن است

نتواند همه‌ی نیازهای مشتری را برآورده سازد و بدین ترتیب پروژه با شکست مواجه می‌شود. در این موارد و بسیاری از موار دیگر، مکانیزم نمونه‌سازی دورانداختنی که در ابتدای مدل نمونه‌سازی قرار دارد روش مناسبی برای به دست آوردن نیازمندی‌های دقیق مشتری است.

توجه کنید که مکانیزم نمونه‌سازی دورانداختنی راهکاری برای تشخیص لیست نیازمندی‌های مشتری است که در ابتدای مدل نمونه‌سازی قرار دارد. بنابر خاصیت مکانیزم نمونه‌سازی دورانداختنی سازنده قادر است نمونه‌ای از نرم‌افزاری را که می‌خواهد، تولید کند، هر چند به طور مختصر و مفید به طوری که مشتری ارتباط میان خود و نرم‌افزار را احساس کرده و متوجه عملکرد نسبی نرم‌افزار شود.

مدل نمونه‌سازی با جمع آوری خواسته‌های مشتری به کمک مکانیزم نمونه‌سازی دورانداختنی آغاز می‌گردد، سازنده در ابتدا با مشتری به گفتگو می‌پردازد و نظرات مشتری را در رابطه با نیازهایی که توقع دارد تا نرم‌افزار مورد نظرش در آینده برآورده سازد جویا می‌شود و آنها را جمع‌آوری می‌کند. سپس یک مدل‌سازی سریع (تحلیل و طراحی) اتفاق می‌افتد تا آن دسته از ویژگی‌هایی که به چشم مشتری می‌آیند (مثل شکل و شمایل صفحات ورودی) به طور سریع مدل‌سازی گردد. به دنبال این مدل‌سازی، یک نمونه‌ی اولیه از نرم‌افزار ساخته می‌شود و توسط مشتری مورد ارزیابی قرار می‌گیرد. نتایج ارزیابی برای پالایش و تکمیل نیازمندی‌های نرم‌افزاری که قرار است ساخته شود، استفاده می‌گردد. مجدداً براساس نیازمندی‌های پالایش شده، نمونه‌ی دیگری که نسبت به نمونه‌ی قبل‌تر کامل‌تر است ساخته شده و در اختیار مشتری قرار داده می‌شود تا ارزیابی گردد. این چرخه، تا زمانی که لیست نیازمندی‌های مشتری تکمیل گردد، تکرار می‌شود. در کل ایده‌ی مدل نمونه‌سازی، استفاده از مکانیزم نمونه‌سازی دورانداختنی به عنوان راهکاری برای تشخیص لیست نیازمندی‌های مشتری است. توجه کنید که نمونه‌های ساخته شده، دورانداخته می‌شوند.

معایب مدل نمونه‌سازی

۱- چون نمونه‌ای از نرم‌افزار بدون رعایت مسائل کیفی در اختیار مشتری قرار می‌گیرد و مشتری در ابتدا نمی‌تواند نرم‌افزار کامل را مشاهده کند، ممکن است تصور غلطی از نرم‌افزار نهایی پیدا کند، زیرا مشتری ظاهراً یک نسخه‌ی کاری از نرم‌افزار را می‌بیند. ولی نمی‌داند که این نمونه‌ی اولیه فقط یک «**ماکت**» است که با «**موم**» سرهم‌بندی شده است. خبر ندارد که کیفیت قربانی سرعت ساخت نمونه‌ی اولیه شده است. زمانی که مشتری اطلاع می‌یابد در آینده این محصول باید به طور مجدد به گونه‌ای ایجاد شود که کیفیت مطلوب به دست آید، ممکن است ناراحت شده و تقاضا کند همان محصول نمونه‌ی اولیه با کمی تغییر به محصول نهایی تبدیل گردد و یا کلاً پروژه را لغو کند!

۲- سازنده اغلب برای دستیابی سریع‌تر به مدل نمونه‌ی اولیه به مسائل کیفی توجه نمی‌کند.

برای‌مثال زبان برنامه‌نویسی نامناسب یا برنامه نویس ارزان قیمت ناآشنا به ایجاد الگوریتم‌های بهینه برای نوشتن مدل نمونه انتخاب می‌نماید آن هم فقط به دلیل سهولت کار یا این که صرفاً چون این برنامه فقط نمونه‌ای بیش نیست، اقدام به این انتخاب‌ها کند و پس از نوشتن برنامه‌ی نمونه، سازنده این برنامه را کنار نگذاشته و به تکمیل همان نمونه‌ی اولیه بدون کیفیت بپردازد و همان را به مشتری تحویل دهد که در آینده مشکلاتی را برای سازنده ایجاد خواهد کرد.

نتیجه اینکه، این مدل زمانی سرانجام خوشی برای سازنده و مشتری خواهد داشت که در ابتدای کار بر سر مسائل مختلف پروژه توافق کنند. مثلاً مشتری باید بداند که ساخت نمونه‌ی اولیه فقط به عنوان راهکاری جهت شناسایی نیازهای نرم‌افزار بوده و نرم‌افزار نهایی حتماً دارای معیارهای کیفیتی خواهد بود.

مدل‌های تکاملی سنتی

در طول سال‌های گذشته، بسیاری از افراد در دانشگاه‌ها و نیز شرکت‌های پیشروی صنعت نرم‌افزار، تلاش‌های زیادی برای ارائه و معرفی مدل‌ها و رویکردهای دیگری که جایگزین رویکرد مدل آبشاری شود، انجام داده‌اند. حاصل این تلاش‌ها، ارائه‌ی ده‌ها مدل فرآیند دیگر بوده است. یکی از راهکارها و تجارب موفق، رویکردی است مبتنی بر تکرار و تکامل که در مقابل رویکرد مدل آبشاری قرار دارد. در رویکرد مبتنی بر تکرار و تکامل، فرصت یادگیری و بهبود تدریجی در سرتاسر چرخه‌ی تولید فراهم است. بدین ترتیب، در طول پروژه، امکان تصحیح به موقع اشتباهات وجود خواهد داشت. در صورت بروز اشتباه در یک تکرار، امکان جبران آن در تکرار بعدی وجود دارد. در حالی که همانطور که پیش از این نیز بیان شد، در مدل آبشاری بسیاری از اشتباهات در انتهای پروژه آشکار می‌شود و در نتیجه فرصت کمی برای تصحیح آنها وجود خواهد داشت. رویکرد مبتنی بر تکرار و تکامل به برنامه‌ریزی مستمر و پویا در طول پروژه نیازمند است، در حالی که در مدل آبشاری، برنامه‌ریزی، یک بار و آن هم در ابتدای پروژه صورت می‌پذیرد. مدل‌های افزایشی و حلزونی براساس رویکرد مبتنی بر تکرار و تکامل ایجاد شده‌اند، اما مدل حلزونی به دلیل تأکید بر مدیریت ریسک در هر تکرار توسط تحلیل ریسک، توانایی قابل توجهی در مدیریت ریسک دارد. در اینجا منظور از ریسک، کلیه‌ی شرایط، عوامل و نگرانی‌هایی است که می‌تواند مانع از دستیابی به موفقیت (شناسایی نیازهای دقیق مشتری، مقرون به صرفه بودن و در زمان مورد انتظار بودن) شوند. در واقع، یک تکرار، عبارت است از انجام متوالی فعالیت‌های لازم در یک بازه‌ی زمانی کوتاه و انجام چندین باره‌ی آن در طول بازه‌ی زمانی یک پروژه. بنابراین، به جای انجام فعالیت‌ها، به صورت یکباره و متوالی، چندین بار و در بازه‌های کوچک‌تری این مجموعه فعالیت‌ها را تکرار می‌نماییم. این مدل‌ها با رویکردی مبتنی بر تکرار و تکامل در هر تکرار نسخه‌هایی از نرم‌افزار را ارائه می‌دهند که هریک از قبلی کامل‌تر است. بنابراین برخلاف سایر مدل‌های فرآیند غیرتکاملی که با تحویل نرم‌افزار پایان می‌یابند، مدل‌های تکاملی را می‌توان طوری

تطبیق داد که در سرتاسر عمر نرم‌افزار کامپیوتری قابل به کارگیری باشد.

توجه: در مدل‌های تکاملی در هر دور از تکرار نسخه‌ی کامل‌تری از نرم‌افزار تولید می‌شود.

مکانیزم نمونه‌سازی تکاملی

در نمونه‌سازی تکاملی، یک نمونه‌ی اولیه تولید می‌شود. در ادامه با اعمال اصلاحات بر روی نمونه‌ی اولیه، طی چند مرحله سیستم نهایی تولید می‌گردد. هدف از نمونه‌سازی تکاملی، تحویل یک سیستم عملیاتی به کاربران نهایی و شروع فرآیند تولید براساس نیازمندی‌هایی است که بهتر و بیشتر درک شده‌اند. کاربرد آن در مواردی است که نیازمندی‌های مشتری به طور کامل در ابتدای کار مشخص نباشد و یا نیاز به توسعه‌ی مبتنی بر تکرار و تکامل داشته باشیم. از مزایای نمونه‌سازی تکاملی می‌توان به درگیر کردن مشتری با فرآیند تولید سیستم که منجر به شناسایی بهتر نیازمندی‌ها می‌گردد، اشاره نمود.

توجه: به مکانیزم نمونه‌سازی تکاملی، **الگوسازی باز** نیز گفته می‌شود.

مدل افزایشی[1]

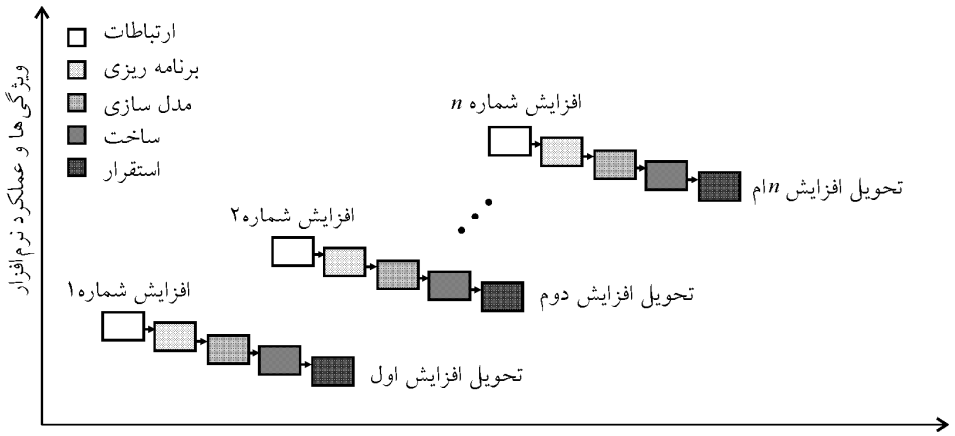
مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. بنابراین مدل افزایشی مبتنی بر مدل آبشاری و مکانیزم نمونه‌سازی تکاملی است.

نمونه‌سازی تکاملی به فرآیند تولید نرم‌افزار روح، حرکت و تکرار می‌دهد و مدل آبشاری فعالیت‌های چارچوبی هر تکرار (افزایش) را مشخص می‌کند. همچنین همانطور که پیش از این نیز گفتیم، هرگاه نیاز به شناسایی خواسته‌های مبهم مشتری وجود داشت می‌توان از مکانیزم نمونه‌سازی دورانداختنی استفاده نمود. بنابراین قبل از هر تکرار (افزایش) جهت شناسایی نیازمندی‌ها می‌توان از مکانیزم نمونه‌سازی دورانداختنی نیز استفاده نمود و نمونه را دور انداخت. اما حرکت، تکرار و تکامل همچنان می‌تواند توسط نمونه‌سازی تکاملی ادامه یابد تا محصول نهایی آماده گردد.

این مدل از یک سری فعالیت‌های چارچوبی تکراری تشکیل شده است که هر تکرار (افزایش) شبیه به مدل آبشاری است. با این تفاوت که روی قسمتی از نرم‌افزار انجام می‌شود. هر کدام از این قسمت‌ها یک «قطعه» قابل تحویل را ایجاد می‌کند. شکل زیر مدل افزایشی و مراحل هر افزایش را نشان می‌دهد:

---

[1] Incremental Model



مدل افزایشی

در این مدل، با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در افزایش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد.

مثال: نرم‌افزار واژه‌پرداز که با استفاده از مدل افزایشی توسعه یافته است، به صورت زیر تحویل داده می‌شود:

افزایش اول: عملیاتی از قبیل مدیریت فایل، تولید و ویرایش مستندات

افزایش دوم: قابلیت‌های پیچیده‌تر در ویرایش و تولید مستندات

افزایش سوم: چک کردن املاء و دستور زبان

افزایش چهارم: قابلیت‌های پیشرفته‌ی صفحه‌بندی

توجه: مشاهده می‌شود که محصول هر افزایش در مثال فوق، قطعه‌ای قابل تحویل است.

هنگامی که از یک مدل افزایشی استفاده می‌شود، افزایش نخست غالباً هسته‌ی اصلی محصول است، یعنی نیازهای اولیه رفع می‌شوند، اما بسیاری از ویژگی‌های مکمل (که برخی معلوم و برخی نامعلوم هستند) تحویل داده نمی‌شوند. هسته‌ی اصلی محصول، توسط مشتری مورد استفاده و ارزیابی قرار می‌گیرد. سپس در افزایش بعدی، قابلیت‌ها و ویژگی‌های دیگر مورد انتظار مشتری به هسته‌ی اصلی محصول اضافه می‌گردد. این فرآیند به دنبال تحویل هر قطعه تکرار می‌شود تا اینکه محصول کامل تولید شود.

توجه: مدل افزایشی، مانند مدل نمونه‌سازی و مدل‌های تکاملی دیگر، ماهیتی تکراری دارد. اما برخلاف مدل نمونه‌سازی، مدل افزایشی بر تحویل قطعه‌ای عملیاتی در هر افزایش تأکید دارد. قطعات اولیه، بخش‌هایی اولیه از محصول نهایی هستند، اما قابلیت ارائه خدمات به کاربر را دارند.

توجه: در مدل نمونه‌سازی، به واسطه‌ی مکانیزم نمونه‌سازی دورانداختنی فرآیند تکرار تنها در

مرحله‌ی جمع‌آوری نیازمندی‌ها انجام می‌شود. اما در مدل افزایشی فرآیند تکرار در تمام طول فرآیند تولید نرم‌افزار انجام می‌شود.

توجه: مدل افزایشی به واسطه‌ی استفاده از مکانیزم نمونه‌سازی تکاملی بر تولید تکاملی نرم‌افزار تأکید دارد. اما مدل نمونه‌سازی به واسطه‌ی استفاده از مکانیزم نمونه‌سازی دورانداختنی بر شناسایی تکاملی نیازمندی‌ها تأکید دارد و نه تولید تکاملی نرم‌افزار. در واقع در مدل نمونه‌سازی پس از شناسایی نیازمندی‌ها توسط مکانیزم نمونه‌سازی دورانداختنی در ادامه نرم‌افزار به شیوه‌ی مدل آبشاری در قالب یک قطعه و یکجا ایجاد می‌گردد. تولید تکاملی بدین معنی است که نرم‌افزار قطعه، قطعه، و ذره ذره، و کم کم، کم کم تکامل می‌یابد و نه در قالب یک قطعه و یکجا. بنابراین در مواقعی که تغییرات زیاد است و نیاز است نسخه‌هایی از نرم‌افزار ارائه گردد و در نسخه‌های بعدی نرم‌افزار تکامل یابد مدل نمونه‌سازی کارایی نخواهد داشت و مدل‌های تکاملی (افزایشی و پیچشی) مناسب خواهند بود.

مدل افزایشی، مواقعی مفید است که منابع مالی و انسانی لازم برای تکمیل پیاده‌سازی پروژه در مهلت کاری مقرر، در دسترس نباشد. بنابراین افزایش‌های اولیه را می‌توان با افراد کمتری پیاده‌سازی کرد. اگر هسته‌ی اصلی محصول به خوبی به دست آید، افراد دیگری را (در صورت لزوم) می‌توان اضافه کرد و افزایش بعدی را پیاده‌سازی کرد.

به علاوه، افزایش‌ها می‌توانند به گونه‌ای برنامه‌ریزی شوند که **ریسک‌های تکنیکی** را مدیریت کنند. برای مثال، یک سیستم جامع و کلی ممکن است نیاز به سخت‌افزار جدیدی داشته باشد که فعلاً در حال تولید است و تاریخ تحویل آن قطعی نیست. در نتیجه می‌توان افزایش‌های اولیه را به نحوی برنامه‌ریزی کرد که به این سخت‌افزار نیازی نداشته باشد. بدین ترتیب این امکان به وجود می‌آید که بخشی از عملکردها بدون تأخیر غیر عادی به کاربر نهایی تحویل داده شود.

توجه: مدل افزایشی در مواقعی که نیازمندی‌های اولیه‌ی مشتری به خوبی تعریف شده‌اند ولی نیازمندی‌های دیگری باقی مانده‌اند و نیاز به تکرار و تکامل می‌باشد مناسب است.

توجه: مدل افزایشی علاوه بر مدیریت ریسک‌های تکنیکی، از مکانیزم نمونه‌سازی دورانداختنی به عنوان راهکاری برای کاهش ریسک مربوط به شناسایی نیازمندی‌ها در هر افزایش استفاده می‌کند و به ریسک‌های دیگر پروژه مانند از دست دادن مدیران و اطلاعات به دلیل عدم مدیریت ریسک (تحلیل ریسک) توجه ندارد.

مدل پیچشی[1] (مارپیچی یا حلزونی)

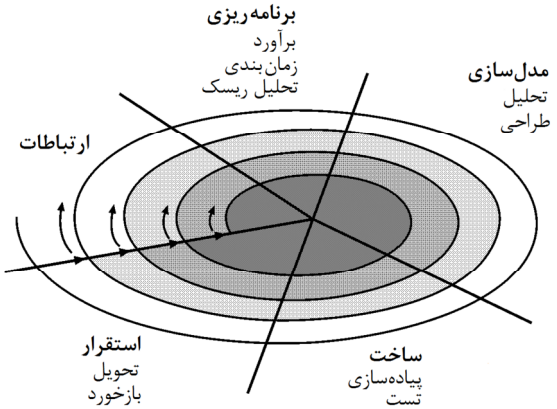
رویکرد مدل پیچشی را همانند مدل افزایشی در نظر بگیرید با این تفاوت اساسی که در مدل پیچشی مدیریت ریسک (تحلیل ریسک) در فعالیت برنامه‌ریزی به طور جدی و در تمام جوانب

[1] Spiral Model

پروژه انجام می‌گیرد. اما در مدل افزایشی فقط **ریسک مربوط به شناسایی درست نیازمندی‌ها** در هر تکرار توسط مکانیزم نمونه‌سازی دورانداختنی و **ریسک تکنیکی** مدیریت می‌گردد. مدل پیچشی را به مانند یک کلاف به دور خود پیچیده شده در نظر بگیرید. سپس این کلاف را باز کنید و به صورت یک خط راست در ذهن خود تجسم کنید، حال آن را به قطعاتی شامل فعالیت‌های چارچوبی (ارتباطات، برنامه‌ریزی، مدل‌سازی، ساخت و استقرار) تقسیم کنید، آن چه مشاهده خواهید کرد قطعاتی است مانند، قطعات (افزایش‌های) موجود در هر تکرار مدل افزایشی. حال قطعات تقسیم شده را مجدداً به هم بچسبانید. سپس به آن به شکل یک کلاف، پیچش دهید تا مجدداً شکل مدل پیچشی را به خود بگیرد!

مدل پیچشی نیز همانند مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. بنابراین مدل پیچشی مبتنی بر مدل آبشاری و مکانیزم نمونه‌سازی تکاملی است. نمونه‌سازی تکاملی به فرآیند تولید نرم‌افزار روح، حرکت و تکرار می‌دهد و مدل آبشاری فعالیت‌های چارچوبی هر تکرار (پیچش) را مشخص می‌کند. همچنین همانطور که پیش از این نیز گفتیم، هرگاه نیاز به شناسایی خواسته‌های مبهم مشتری وجود داشت می‌توان از مکانیزم نمونه‌سازی دورانداختنی استفاده نمود. بنابراین قبل از هر تکرار (پیچش) جهت شناسایی نیازمندی‌ها می‌توان از مکانیزم نمونه‌سازی دورانداختنی استفاده نمود و نمونه را دور انداخت. اما حرکت، تکرار و تکامل همچنان می‌تواند توسط نمونه‌سازی تکاملی ادامه یابد تا محصولی نهایی آماده گردد.

این مدل از یک سری فعالیت‌های چارچوبی تکراری تشکیل شده است که هر تکرار (پیچش) شبیه به مدل آبشاری است. با این تفاوت که روی قسمتی از نرم‌افزار انجام می‌شود. هر کدام از این قسمت‌ها یک «قطعه» قابل تحویل را ایجاد می‌کند. شکل زیر مدل پیچشی و مراحل هر پیچش را نشان می‌دهد:



مدل پیچشی

در این مدل با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود، یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در پیچش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد. هنگامی که از یک مدل پیچشی استفاده می‌شود، پیچش‌های نخست غالباً هسته‌ی اصلی محصول است، یعنی نیازهای اولیه رفع می‌شوند، اما بسیاری از ویژگی‌های مکمل (که برخی معلوم و برخی نامعلوم هستند) تحویل داده نمی‌شوند.

هسته‌ی اصلی محصول، توسط مشتری مورد استفاده و ارزیابی قرار می‌گیرد. سپس در پیچش بعدی، قابلیت‌ها و ویژگی‌های دیگر مورد انتظار مشتری به هسته‌ی اصلی محصول اضافه می‌گردد. این فرآیند به دنبال تحویل هر قطعه تکرار می‌شود تا اینکه محصول کامل تولید شود.

توجه: مدل پیچشی، مانند مدل نمونه‌سازی و مدل‌های تکاملی دیگر، ماهیتی تکراری دارد، اما برخلاف مدل نمونه‌سازی، مدل پیچشی بر تحویل قطعه‌ای عملیاتی، در هر افزایش تأکید دارد. قطعات اولیه، بخش‌هایی اولیه از محصول نهایی هستند، اما قابلیت ارائه‌ی خدمات به کاربر را دارند.

توجه: در موارد خاص در صورتی که حتی نیازمندی‌های اولیه نیز مشخص نباشند، پیچش‌های نخست می‌تواند مدلی کاغذی باشد و در تکرارهای بعدی هسته‌ی اصلی محصول ایجاد گردد.

توجه: در مدل نمونه‌سازی، به واسطه‌ی مکانیزم نمونه‌سازی دورانداختنی فرآیند تکرار تنها در مرحله‌ی جمع‌آوری نیازمندی‌ها انجام می‌شود اما در مدل پیچشی فرآیند تکرار در تمام طول فرآیند تولید نرم‌افزار انجام می‌شود.

توجه: مدل پیچشی به واسطه‌ی استفاده از مکانیزم نمونه‌سازی تکاملی بر تولید تکاملی نرم‌افزار تأکید دارد. اما مدل نمونه‌سازی به واسطه‌ی استفاده از مکانیزم نمونه‌سازی دورانداختنی بر شناسایی تکاملی نیازمندی‌ها تأکید دارد و نه تولید تکاملی نرم‌افزار. در واقع در مدل نمونه‌سازی پس از شناسایی نیازمندی‌ها توسط مکانیزم نمونه‌سازی دورانداختنی در ادامه نرم‌افزار به شیوه‌ی مدل آبشاری در قالب یک قطعه و یکجا ایجاد می‌گردد. تولید تکاملی یعنی نرم‌افزار قطعه قطعه، ذره ذره و کم کم، کم کم تکامل می‌یابد و نه در قالب یک قطعه و یکجا. بنابراین در مواقعی که تغییرات زیاد است و نیاز است نسخه‌هایی از نرم‌افزار ارائه گردد و در نسخه‌های بعدی نرم‌افزار تکامل یابد مدل نمونه‌سازی کارایی نخواهد داشت. و مدل‌های تکاملی (افزایشی و پیچشی) مناسب خواهند بود.

مدل پیچشی، مدلی امن، مطمئن و قابل اعتماد است. زیرا در هر تکرار یا پیچش در فعالیت مربوط به برنامه‌ریزی توسط عمل تحلیل ریسک، تمامی ریسک‌های مربوط به پروژه را به دقت در نظر می‌گیرد، شناسایی و در ادامه برطرف می‌نماید. از آنجا که تحلیل ریسک در ابتدای هر چرخش انجام می‌شود، بنابراین امکان وقوع ریسک بسیار پایین خواهد آمد. در نتیجه کیفیت نرم‌افزار تولیدی بالا خواهد رفت. همچنین مدیر پروژه بر امور جاری کنترل دقیق دارد و همه‌ی مسایل و

هزینه‌ها با توافق طرفین (مشتری و سازنده) صورت می‌گیرد. به این کنترل دقیق بر امور جاری پروژه در هر تکرار اصطلاحاً «**نقاط عطف لنگرگاهی**[1]» نیز گفته می‌شود.

توجه: مدل پیچشی در مواقعی که نیازمندی‌های اولیه به خوبی تعریف شده‌اند ولی نیازمندی‌های دیگری باقی مانده‌اند و نیاز به تکرار و تکامل می‌باشد مناسب است و یا حتی در مواقعی که نیازمندی‌های اولیه بخوبی تعریف نشده‌اند و مسائل ناشناخته‌ی بسیار زیادی وجود دارد مناسب است. اگر قرار باشد پروژه در امنیت بالایی از هر لحاظ ادامه یابد نیاز به مدیریت ریسک به طور مستمر می‌باشد که این خاصیت در مدل پیچشی به واسطه‌ی تحلیل ریسک وجود دارد.

توجه: مدل پیچشی از مکانیزم نمونه‌سازی دورانداختنی به عنوان راهکاری برای کاهش ریسک مربوط به شناسایی نیازمندی‌ها در هر پیچش، استفاده می‌کند و همچنین به دلیل تحلیل ریسک به واسطه‌ی مدیریت ریسک تمامی ریسک‌های مربوط به کلیت پروژه (شناسایی نیازمندی‌های دقیق مشتری، مقرون به صرفه بودن و در زمان مورد انتظار بودن) را کنترل می‌کند. در بیان ساده، مدل پیچشی به واسطه‌ی مدیریت ریسک (تحلیل ریسک) بستری امن، برای تولید نرم‌افزار ایده‌آل مشتری فراهم می‌سازد.

توجه: در صورتی که علاوه بر تحلیل ریسک، تحلیل برنده برنده، به معنی انجام توافقات برد برد بین مشتری و سازنده در تمامی مراحل پروژه به فعالیت برنامه‌ریزی اضافه گردد، مدل پیچشی، **مدل پیچشی برنده برنده** خواهد بود.

معایب مدل پیچشی

۱- قانع کردن مشتری به ویژه در این مورد که تکامل پروژه قابل کنترل می‌باشد، کار دشواری است. در واقع ما برای انجام مراحل مختلف پروژه نیازمند تأمین بودجه از سوی مشتری هستیم و ممکن است خروجی‌های بخش‌های مختلف تکامل پروژه از نظر مشتری چندان مطلوب نباشد و در تأمین بودجه‌ی مورد نیاز کوتاهی کند.

۲- این مدل براساس ارزیابی، در نظر گرفتن و تعیین کردن ریسک در هر چرخش بنا شده است که خود این ارزیابی و مدیریت ریسک نیازمند تخصص مورد نیاز خودش می‌باشد، بنابراین موفقیت این مدل فرآیند، کاملاً وابسته به تصمیمات و سیاست‌هایی است که فرد و یا گروه مدیریت ریسک اتخاذ می‌کنند.

۳- اگر یک ریسک بزرگ شناسایی و مدیریت نشود، ممکن است مشکلات غیرقابل پیش بینی را در نرم‌افزار به وجود آورد.

[1] Anchor Point Milestones

معایب مدل‌های تکاملی سنتی

مشخصه‌ی اصلی نرم‌افزارهای مدرن امروزی، تغییر پیوسته، در فواصل زمانی بسیار به هم فشرده و با تأکید بسیار بر رضایت مشتری است. **شاید تنها چیزی که در دنیا ثابت است، تغییر باشد.** در بسیاری موارد، زمان رساندن محصول به بازار، مهمترین خواسته مدیریتی است. اگر زمان مقرر برای ارائه به بازار از دست برود، خود پروژه‌ی نرم‌افزاری ممکن است دیگر بی‌معنا شود.

تصور می‌شد مدل‌های فرایند تکاملی سنتی این مشکلات را برطرف سازند و با این وجود، آنها نیز به عنوان طبقه‌ای عمومی از مدل‌های فرآیند تولید نرم‌افزار، نقطه ضعف‌هایی دارند.

به رغم مزایای غیرقابل تردید مدل‌های تکاملی سنتی، دغدغه‌های نیز وجود دارد:

۱- مدل‌های تکاملی سنتی به دلیل قطعی نبودن تعداد چرخه‌های (تکرارهای) لازم برای ساخته شدن محصول، برای برنامه‌ریزی پروژه ایجاد مشکل می‌کند. اکثر تکنیک‌های برآورد و مدیریت پروژه بر پایه چیدمان خطی فعالیت‌ها (یعنی مشخص بودن تعداد گام‌های لازم برای ساخته شدن محصول) استوارند، لذا مدل‌های تکاملی سنتی به خوبی در مدل‌های مدیریتی نمی‌گنجند.

توجه: متدولوژی RUP به دلیل قطعی بودن تعداد چرخه‌های (تکرارهای) لازم برای ساخته شدن محصول (تعداد تکرار برابر چهار گام است) برای برنامه‌ریزی بسیار مناسب است. بنابراین متدولوژی RUP به خوبی در مدل‌های مدیریتی می‌گنجد.

۲- مدل‌های تکاملی سنتی چندان سریع نیستند. زیرا اگر تکامل آنها به سرعت انجام شود، فرآیند دچار بی‌نظمی می‌شود (مثلاً نیازمندی‌ها به خوبی شناسایی نمی‌شوند) و اگر خیلی کند باشد، ممکن است روی بهره‌وری تأثیر منفی بگذارد (مثلاً پروژه در زمان مورد انتظار نرسد).

۳- در مدل‌های تکاملی سنتی، تمرکز بیشتر بر روی انعطاف‌پذیری و قابلیت گسترش‌پذیری می‌باشد تا بر روی کیفیت. زیرا اگر بخواهیم پروژه‌ای با کیفیت بالا را گسترش دهیم لازم است زمان زیادی را صرف کنیم (به خصوص در چرخه‌های اولیه) که این خود ممکن است فرصتی را که در بازار برای این نرم‌افزار وجود دارد از بین ببرد.

توجه: امروزه، چالش پیش روی مهندسان نرم‌افزار، استفاده از مدل فرآیندی است که بتواند بین دو فاکتور مهم سرعت (تولید به موقع) و کیفیت (رضایت مشتری) موازنه برقرار کند.

مدل توسعه‌ی هم‌روند[1]

اغلب سازمان‌های نرم‌افزاری، در یک بازه‌ی زمانی، احتمالاً چندین پروژه را در دست تولید دارند. بسیاری از مهندسان نرم‌افزار معتقدند مدل‌های قبلی نمی‌توانند تصویر دقیقی از وضعیت یک پروژه در اختیار مدیران قرار دهند. به عنوان مثال ممکن است، پنج پروژه از پروژه‌های سازمان در مرحله‌ی ایجاد باشند، اما احتمالاً وضعیت آن‌ها با یکدیگر متفاوت است (مثلاً یکی در مرحله‌ی

[1] Concurrent Development Model

تولید کد است در حالی که دیگری در حال تست قرار دارد). مدل توسعه‌ی هم‌روند که با نام مهندسی هم‌روند نیز شناخته می‌شود، جهت کنترل اجرای چند پروژه‌ی همزمان مورد استفاده قرار می‌گیرد که هر یک از فعالیت‌های چارچوبی فرآیند تولید نرم‌افزار مربوط به هر پروژه می‌تواند در وضعیت‌های مختلفی قرار داشته باشد، وضعیت‌های مختلف مربوط به هر یک از فعالیت‌های چارچوبی توسط یک گراف نشان داده می‌شود.

وضعیت‌های مختلف فعالیت‌های چارچوبی

۱- **انجام نشده:** مانند زمانی که فعالیت ارتباط با مشتری آغاز شده است، اما فعالیت طراحی هنوز شروع نشده است.

۲- **در حال توسعه:** فعالیت چارچوبی که در حال انجام است، در این وضعیت قرار دارد.

۳- **در حال مرور و معیارها:** کارهای انجام شده براساس لیست نیازمندی‌ها و استانداردها، شاخص‌ها و معیارهای مشتری مرور و اعتبارسنجی می‌شوند.

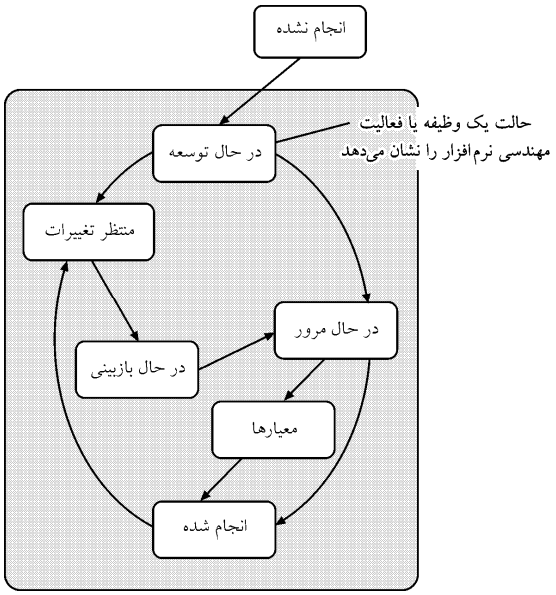
۴- **منتظر تغییرات:** اگر فعالیتی در وضیعت در حال توسعه یا انجام شده باشد و تغییراتی در روند انجام پروژه ایجاد گردد، فعالیت موجود به وضعیت منتظر تغییرات جهت انجام تغییرات می‌رود.

۵- **در حال بازبینی:** اگر فعالیتی در وضعیت منتظر تغییرات قرار گرفت، پس از اعمال تغییرات، تغییرات انجام شده در وضعیت بازبینی باید مورد اعتبار سنجی قرار گیرد و در ادامه براساس معیارها مرور گردد.

۶- **انجام شده:** هر مرحله از کار بعد از اینکه با توجه به معیارها و شاخص‌ها بازبینی شد و مورد تأیید قرار گرفت، به این مرحله وارد می‌شود و تا بروز تغییرات جدید در این مرحله می‌ماند.

همان طور که گفتیم، هر فعالیت چارچوبی از یک مجموعه وضعیت‌ها تشکیل شده است و بسته به اینکه فعالیت در چه حالتی قرار دارد، وضعیت‌های متفاوتی خواهد داشت. در این مدل، برای هر یک از فعالیت‌های چارچوبی (ارتباط، برنامه‌ریزی، مدل‌سازی، ساخت و استقرار) که جهت تکمیل هر پروژه و رسیدن به نتیجه‌ی نهایی انجام می‌گیرد. یک گراف به صورت شبکه‌ای از وضعیت‌های مختلف یک فعالیت چارچوبی، رسم می‌شود، بدین معنی که هر یک از فعالیت‌های چارچوبی موجود در یک پروژه، گراف مخصوص به خود را دارند، به عنوان مثال فعالیت مدل‌سازی، گراف مختص به خود را دارد که وضعیت‌های مختلف آن را نمایش می‌دهد. فعالیت‌های دیگر نیز به همین منوال هستند.

هر کدام از این فعالیت‌ها در هر لحظه می‌تواند در یکی از وضعیت‌های نشان داده شده در شکل زیر قرار گیرد:



مدل توسعه‌ی هم‌روند

به عنوان مثال وقتی فعالیت ارتباط با مشتری مربوط به یک پروژه (تعیین خواسته‌ها و نیازمندی‌ها)، یک تکرار را به پایان می‌برد و در وضعیت «انجام شده و سپس منتظر تغییرات» قرار می‌گیرد، در ادامه فعالیت تحلیل که هنگام کامل شدن ارتباط اولیه با مشتری در حالت «انجام نشده» قرار داشت به حالت «در حال توسعه» وارد می‌شود. حال اگر مشتری تغییراتی را به اطلاع برساند، وضعیت فعالیت تحلیل به «منتظر تغییرات» تبدیل می‌شود، در واقع از وضعیت «در حال توسعه» به وضعیت «منتظر تغییرات» انتقال می‌یابد. نتیجه اینکه در هر پروژه چندین فعالیت چارچوبی انجام می‌شود که هر کدام در وضعیت‌های متفاوتی قرار دارند. در این مدل، یکسری رویداد تعریف می‌شود که وقوع آنها باعث انتقال از وضعیتی به وضعیت دیگر برای هر یک از فعالیت‌های چارچوبی می‌گردد. برای مثال، وقوع یک ناسازگاری در فعالیت طراحی در طول مراحل اولیه آن که ناشی از وجود اشکال در فعالیت تحلیل بوده است، باعث انتقال فعالیت تحلیل از وضعیت «انجام شده» به حالت «منتظر تغییرات» و سپس در «حال بازبینی» می‌شود.

خصوصیات مدل توسعه‌ی هم‌روند

۱- در دنیای واقعی، مدل توسعه‌ی هم‌روند را در تولید تمامی انواع نرم‌افزارها می‌توان به کار برد و این مدل، تصویری درست از وضعیت جاری یک پروژه را نمایش می‌دهد.

۲- در مدل توسعه‌ی هم‌روند، به جای تعریف ترتیبی برای فعالیت‌های چارچوبی، شبکه‌ای از فعالیت‌های چارچوبی برای هر پروژه خواهیم داشت. به نحوی که رخدادهای یک فعالیت روی

وضعیت آن فعالیت و سایر فعالیت‌های دیگر تأثیر می‌گذارد.

مدل‌های فرآیند خاص

مدل‌های فرآیند خاص، بسیاری از خصوصیات مدل‌های متداول ارائه شده در بخش‌های قبل را دارند. اما فرق آنها در این است که در شرایط خاص محیطی قابل استفاده هستند. در واقع شرایط خاصی وجود دارند که استفاده از این مدل‌های فرآیندی خاص می‌تواند بهترین نتیجه را در آنها به همراه داشته باشد.

مدل توسعه‌ی مبتنی بر مؤلفه ساخت‌یافته

مؤلفه یک قطعه‌ی آماده، کاربردی و پیاده‌سازی شده است که دارای واسط لازم جهت اتصال با سایر قطعات و استقرار در بخشی از یک سیستم عملیاتی می‌باشد.

در دیدگاه ساخت‌یافته به مؤلفه، پیمانه نیز گفته می‌شود. در حیطه‌ی مهندسی نرم‌افزار ساخت‌یافته، **واحد مولفه**، یک قطعه عملیاتی مبتنی بر تابع است که بر دو نوع می‌باشد:

۱) تابع کاربردی: مانند تابع جمع که برنامه‌نویس آن را می‌نویسد. اگر تابع کاربردی، شرایط قابل حمل بودن (مانند تعریف متغیر درون تابع به صورت‌محلی و صریح و عدم استفاده از متغیرهای سراسری) را داشته باشد می‌تواند به عنوان یک قطعه‌ی آماده‌ی قابل استفاده‌ی مجدد در پروژه‌های بعدی مورد استفاده مجدد قرار گیرد.

۲) تابع سیستمی: مانند تابع سینوس که کامپایلر تعاریف آن را فراهم می‌کند و می‌تواند به عنوان یک قطعه‌ی آماده و قابل استفاده‌ی مجدد، در پروژه‌ها مورد استفاده مجدد قرار گیرد.

مدل توسعه‌ی مبتنی بر مؤلفه‌ی ساخت‌یافته از بسیاری از خصوصات مدل پیچشی استفاده می‌کند. این مدل از نظر ماهیت، مدلی تکاملی است و ساختاری تکرارشونده را برای توسعه‌ی نرم‌افزار در پیش می‌گیرد. اما در تولید برنامه‌های کاربردی از مؤلفه‌های آماده استفاده می‌کند، در واقع این مدل برنامه را با استفاده از ترکیب و سرهم کردن قطعات نرم‌افزاری از پیش ساخته شده می‌سازد.

روال استقرار تابع در معماری نرم‌افزار (ساختار برنامه یا اسکلت برنامه) براساس مدل توسعه‌ی مبتنی بر مؤلفه‌ی ساخت‌یافته به صورت زیر است:

۱- شناسایی توابع مورد نیاز برنامه

۲- جستجوی توابع در کتابخانه‌ی توابع سیستمی یا کاتالوگ توابع کاربردی

توجه: توابع کاربردی ایجاد شده در پروژه‌های قبلی در کاتالوگ توابع کاربردی، نگهداری می شوند.

۳- در صورت وجود توابع مورد نیاز در کتابخانه‌ی توابع سیستمی یا کاتالوگ توابع کاربردی، تابع استخراج و دوباره استفاده می‌شود. در غیراینصورت تابع مورد نیاز ایجاد می‌گردد.

۴- تابع در معماری نرم‌افزار (اسکلت برنامه) استقرار می‌یابد.

۵- انجام تست برای اطمینان از صحت کار انجام می‌شود.

توجه: دقت کنید که مراحل فوق مبتنی بر تکرار و تکامل صورت می‌گیرد، یعنی پروژه به تدریج و براساس تکرار، تکامل می‌یابد.

توجه: زمان و هزینه‌ی فرآیند تولید نرم‌افزار براساس مدل توسعه‌ی مبتنی بر مؤلفه ساخت‌یافته به دلیل استفاده از قطعات آماده (قابلیت استفاده مجدد) کاهش چشمگیری دارد. در واقع یک قطعه با قابلیت استفاده‌ی مجدد یک بار ساخته می‌شود و بارها و بارها استفاده می‌شود و این یعنی سودآوری!

مدل روش‌های رسمی[1]

مدل روش‌های رسمی (فرمال، قراردادی و صوری) شامل مجموعه‌ای از فعالیت‌ها است که سعی دارد پروژه‌ی نرم‌افزاری را در قالب روابطی رسمی و ریاضی، سیستم‌های کامپیوتری را تعریف، توسعه، پیاده‌سازی و ارزیابی نماید. در این مدل، با استفاده از تحلیل‌های ریاضی، بسیاری از ابهامات، نواقص و عدم سازگاری نرم‌افزار را تا حد زیادی می‌توان به سادگی کشف و تصحیح نمود. گونه‌ای از روش‌های رسمی وجود دارد که به مهندسی نرم‌افزار **اتاق تمیز**[2] معروف است.

توجه: این مدل، امکان کشف و تصحیح خطاهای زیادی را که تا زمان اجرا غیرقابل تشخیص هستند را در طول مراحل اولیه‌ی تولید نرم‌افزار، فراهم می‌کند.

توجه: یکی از مهمترین کاربردهای مدل روش‌های رسمی، ساخت سیستم‌های حساس و امن مانند ساخت نرم‌افزارهای کنترل هواپیما و موشک است.

معایب مدل‌های روش‌های رسمی

۱- استفاده از مدل روش‌های رسمی بسیار وقتگیر و گران است.

۲- آموزش گسترده‌ی سازندگان نرم‌افزار به علت اینکه تعداد اندکی از تولیدکنندکان نرم‌افزار پیش‌زمینه‌ی لازم برای به کار بردن روش‌های فرمال (رسمی) را دارند.

۳- از این مدل نمی‌توان برای برقراری ارتباط با مشتریان معمولی که دید فنی ندارند، استفاده نمود.

مدل فرآیند تولید نرم‌افزار مدرن (شیءگرا)

مدل فرآیند تولید نرم‌افزار مدرن، به صورت تکاملی مدرن می‌باشد.

مدل تکاملی مدرن

مدل تکاملی مدرن، مبتنی بر مؤلفه شیءگرا نام دارد و براساس رویکرد تکرار و تکامل

---

[1] Formal Method
[2] Clean Room

می‌باشد.

مدل توسعه‌ی مبتنی بر مؤلفه شیءگرا

مؤلفه یک قطعه‌ی آماده، کاربردی و پیاده‌سازی شده است که دارای واسط لازم جهت اتصال با سایر قطعات و استقرار در بخشی از یک سیستم عملیاتی می‌باشد.

در دیدگاه شیءگرا، به مؤلفه پیمانه نیز گفته می‌شود. در حیطه مهندسی نرم‌افزار شیءگرا، **واحد مؤلفه**، یک قطعه‌ی عملیاتی مبتنی بر کلاس است که بر دو نوع می‌باشد:

۱) کلاس کاربردی: مانند کلاس دانشجو که برنامه‌نویس آن را می‌نویسد و به دلیل داشتن شرایط قابل حمل به شکل ذاتی، می‌تواند به عنوان یک قطعه‌ی آماده و قابل استفاده‌ی مجدد در پروژه‌های بعدی مورد استفاده مجدد قرار گیرد.

۲) کلاس سیستمی: مانند کلاس جعبه‌ی متن که کامپایلر تعاریف آن را فراهم می‌کند و می‌تواند به عنوان یک قطعه‌ی آماده و قابل استفاده‌ی مجدد، در پروژه‌ها مورد استفاده مجدد قرار گیرد.

این مدل از نظر ماهیت، مدلی تکاملی است و ساختاری تکرارشونده را برای توسعه‌ی نرم‌افزار در پیش می‌گیرد. اما در تولید برنامه‌های کاربردی از مؤلفه‌های آماده استفاده می‌کند، در واقع این مدل، برنامه را با استفاده از ترکیب و سرهم‌کردن قطعات نرم‌افزاری از پیش ساخته شده می‌سازد.

روال استقرار کلاس در معماری نرم‌افزار (ساختار برنامه یا اسکلت برنامه) براساس مدل توسعه‌ی مبتنی بر مؤلفه شیءگرا به صورت زیر است:

۱- شناسایی کلاس مورد نیاز برنامه

۲- جستجوی کلاس در کتابخانه‌ی کلاس‌های سیستمی یا کاتالوگ کلاس‌های کاربردی

توجه: کلاس‌های کاربردی ایجاد شده در پروژه‌های قبلی در کاتالوگ کلاس‌های کاربردی، نگهداری می‌شوند.

۳- در صورت وجود کلاس‌های مورد نیاز در کتابخانه‌ی کلاس‌های سیستمی یا کاتالوگ کلاس‌های کاربردی، کلاس استخراج و دوباره استفاده می‌شود. در غیراینصورت کلاس مورد نیاز ایجاد می‌گردد.

۴- کلاس در معماری نرم‌افزار (اسکلت برنامه) استقرار می‌یابد.

۵- انجام تست برای اطمینان از صحت کار انجام می‌شود.

توجه: دقت کنید که مراحل فوق مبتنی بر تکرار و تکامل صورت می‌گیرد، یعنی پروژه به تدریج و براساس تکرار، تکامل می‌یابد.

توجه: زمان و هزینه‌ی فرآیند تولید نرم‌افزار براساس مدل توسعه‌ی مبتنی بر مؤلفه شیءگرا به دلیل استفاده از قطعات آماده (قابلیت استفاده مجدد) کاهش چشمگیری دارد. در واقع یک قطعه با قابلیت استفاده‌ی مجدد یک بار ساخته می‌شود و بارها و بارها استفاده می‌شود و این یعنی سودآوری!

تست‌های فصل دوم

۱- در شرکتی کار می‌کنید که در بازار رقابتی و در حال رشد شبکه‌های نوری فعالیت می‌کند. باید در مدت کوتاه و معینی، نرم‌افزاری برای پیکربندی و تست سخت‌افزار بنویسید (به عنوان مثال، نرم‌افزاری برای راه‌اندازی و پیکربندی مسیریاب‌های نوری در شبکه). نرم‌افزار باید به همراه سخت‌افزار به فروش برسد، در غیر این صورت سخت‌افزار استفاده‌ای نخواهد داشت. متأسفانه سخت‌افزار پیوسته تغییر می‌کند و تا چند هفته پیش از عرضه‌ی محصول به بازار، نهایی نخواهد شد. مدل فرایند مناسب برای توسعه‌ی این نرم‌افزار چیست؟ **(مهندسی IT – دولتی ۸۳)**

۱) نمونه‌سازی (Prototyping)
۲) افزایشی (Incremental)
۳) آبشاری (Waterfall)
۴) Rapid Application Development

۲- کدام عبارت صحیح است؟ **(مهندسی IT – دولتی ۸۴)**

۱) بسته به شرایط محیطی که یک نرم‌افزار برای آن توسعه می‌یابد، می‌توان از ترکیبی از متدولوژی‌های توسعه نرم‌افزار استفاده نمود. از آنجایی که متدولوژی به مفهوم ترکیبی از فرآیند و ابزار می‌باشد، می‌توان فرایند یک متدولوژی را با ابزارهای استفاده شده در متدولوژی دیگر تلفیق نموده و مورد استفاده قرار داد.

۲) بسته به شرایط محیطی که یک نرم‌افزار برای آن توسعه می‌یابد، می‌توان از ترکیبی از متدولوژی‌های توسعه نرم‌افزار استفاده نمود. از آنجایی که متدولوژی به مفهوم ترکیبی از فرآیند و ابزار می‌باشد، نمی‌توان فرایند یک متدولوژی را با ابزارهای استفاده شده در متدولوژی دیگر تلفیق نموده و مورد استفاده قرار داد.

۳) بسته به شرایط محیطی که یک نرم‌افزار برای آن توسعه می‌یابد، نمی‌توان از ترکیبی از متدولوژی‌های توسعه نرم‌افزار استفاده نمود. از آنجایی که متدولوژی به مفهوم ترکیبی از فرآیند و ابزار می‌باشد، می‌توان فرایند یک متدولوژی را با ابزارهای استفاده شده در متدولوژی دیگر تلفیق نموده و مورد استفاده قرار داد.

۴) بسته به شرایط محیطی که یک نرم‌افزار برای آن توسعه می‌یابد، نمی‌توان از ترکیبی از متدولوژی‌های توسعه نرم‌افزار استفاده نمود. از آنجایی که متدولوژی به مفهوم ترکیبی از فرآیند و ابزار می‌باشد، نمی‌توان فرآیند یک متدولوژی را با ابزارهای استفاده شده در متدولوژی دیگر تلفیق نموده و مورد استفاده قرار داد.

۳- شما مدیر بخش نرم‌افزار در یک شرکت نرم‌افزاری بزرگ هستید. از شما خواسته می‌شود که برای پروژه‌ی خودکارسازی سیستم کنترل ترافیک یک فرودگاه بزرگ پیشنهاد فنی آماده کنید. سیستم بزرگ و دستی فعلی به شکل معقولی کار می‌کند، اما مشتری اعتقاد دارد که خودکارسازی سیستم باعث

صرفه‌جویی در هزینه می‌گردد. مشتری در رابطه با مسائل قراردادی بسیار انعطاف‌پذیر است. سیستمی که ایجاد می‌شود باید بسیار ایمن و قابل اطمینان باشد و شرکت شما باید مسئولیت حقوقی تمامی صدمات ناشی از نامناسب عمل کردن سیستم را به عهده بگیرد. علاوه بر این، در رابطه با امکان‌پذیری سیستم، مسائل ناشناخته‌ای وجود دارد که در طی مراحل توسعه‌ی سیستم مشخص خواهند گردید. برای توسعه‌ی این سیستم کامپیوتری چه مدل فرآیندی را انتخاب می‌کنید؟ **(مهندسی IT – دولتی ۸۴)**

۱) مدل پیچشی (Spiral) ۲) مدل آبشاری (Waterfall)

۳) روش‌های صوری (Formal Methods) ۴) مدل Rapid Application Development

---

۴- کدام عبارت صحیح است؟ **(مهندسی IT – دولتی ۸۴)**

۱) در متدولوژی شیءگرا، همواره از روش برنامه‌نویسی شیءگرا (Object Oriented Programming) استفاده می‌شود. معمولاً در متدولوژی ساخت‌یافته (SSADM) دیدگاه تابع‌گرا در تجزیه و تحلیل سیستم و ایجاد نرم‌افزار حاکم است و می‌توان بسته به شرایط در مرحله برنامه‌نویسی از روش شیءگرا نیز استفاده نمود.

۲) در متدولوژی شیءگرا، می‌توان از روش برنامه‌نویسی شیءگرا (Object Oriented Programming) یا از روش برنامه‌نویسی تابع‌گرا استفاده نمود. معمولاً در متدولوژی ساخت‌یافته (SSADM) دیدگاه تابع‌گرا در تجزیه و تحلیل سیستم و ایجاد نرم‌افزار حاکم است و می‌توان بسته به شرایط در مرحله برنامه‌نویسی از روش شیءگرا نیز استفاده نمود.

۳) در متدولوژی شیءگرا، می‌توان از روش برنامه‌نویسی شیءگرا (Object Oriented Programming) یا از روش برنامه‌نویسی تابع‌گرا استفاده نمود. معمولاً در متدولوژی ساخت‌یافته (SSADM) دیدگاه تابع‌گرا در تجزیه و تحلیل سیستم و ایجاد نرم‌افزار حاکم است و در مرحله برنامه‌نویسی همواره از روش تابع‌گرا استفاده می‌شود.

۴) در متدولوژی شیءگرا، همواره از روش برنامه‌نویسی شیءگرا (Object Oriented Programming) استفاده می‌شود. معمولاً در متدولوژی ساخت‌یافته (SSADM) دیدگاه تابع‌گرا در تجزیه و تحلیل سیستم و ایجاد نرم‌افزار حاکم است و در مرحله برنامه‌نویسی همواره از روش تابع‌گرا استفاده می‌شود.

۵- کدام‌یک از موارد زیر صحیح است؟ **(مهندسی IT – دولتی ۸۴)**

۱) متدولوژی‌های تابع‌گرا (Function Oriented) همواره بهتر از بقیه متدولوژی‌های توسعه نرم‌افزار بوده است.

۲) متدولوژی‌های شیءگرا (Object Oriented) همواره از بقیه متدولوژی‌های توسعه نرم‌افزار بهتر می‌باشد.

۳) متدولوژی‌های شیءگرا (Object Oriented) و متدولوژی‌های تابع‌گرا (Function Oriented) هر دو زیرمجموعه‌ای از متدولوژی‌های نسل دوم یا ساخت‌یافته (Structured) می‌باشند.
۴) هیچ‌کدام

۶- مدیریت نیازها (requirements management) عبارت است از: **(مهندسی IT – دولتی ۸۳)**

۱) مجموعه‌ای از فعالیت‌ها برای مدیریت مرحله تجزیه و تحلیل نیازها
۲) مجموعه‌ای از فعالیت‌ها برای به دست آوردن نیازها در مرحله تجزیه و تحلیل نیازها
۳) مجموعه‌ای از فعالیت‌ها برای دنبال نمودن (track) نیازها و تغییرات آنها در طی انجام پروژه
۴) مجموعه‌ای از فعالیت‌ها برای به دست آوردن نیازها و دنبال نمودن (track) نیازها و تغییرات آنها در طی انجام پروژه

۷- کدام‌یک از گزینه‌های زیر مربوط به مشخصه‌ی نیازمندی‌های غیرعملیاتی نیست؟
(مهندسی IT – آزاد ۸۴)

۱) سرویس کاربر ۲) نمایش داده‌ها ۳) زمان پاسخگویی ۴) حافظه موردنیاز

۸- به چه منظور از مدل آبشاری جهت توسعه‌ی نرم‌افزار استفاده می‌شود؟ **(مهندسی IT – آزاد ۸۴)**

۱) جهت پایین آمدن هزینه‌ی نگهداری سیستم
۲) به منظورساده کردن مدیریت فرآیند نرم‌افزار
۳) جهت جلوگیری از تکرار فرآیندها
۴) چون می‌تواند مدل‌های دیگر را نیز دربرگیرد.

۹- کدام‌یک از مراحل زیر در مراحل تولید و به‌کارگیری نرم‌افزار به ترتیب دارای کمترین و بیشترین هزینه می‌باشند؟ **(مهندسی IT – آزاد ۸۴)**

۱) تجزیه و تحلیل، پیاده‌سازی ۲) امکان‌پذیری، نگهداری
۳) طراحی، تجزیه و تحلیل ۴) تعریف، نگهداری

۱۰- کدام‌یک از موارد زیر صحیح است؟ **(مهندسی IT – دولتی ۸۵)**

۱) نتایج متدولوژی SSADM با نتایج حاصل از متدولوژی شیءگرا کاملاً با یکدیگر مشابه و متناظر است.
۲) متدولوژی ساخت‌یافته (SSADM) و شیءگرا (O.O) کاملاً از یکدیگر مستقل بوده و نتایج غیرمشترکی را دارند.

۳) از روی نتایج حاصل از متدولوژی SSADM می‌توان برخی از خروجی‌های متدولوژی شیءگرا (O.O) را به دست آورد و بالعکس.
۴) متدولوژی شیءگرا (O.O) فقط برای طراحی و متدولوژی SSADM فقط برای تحلیل نرم‌افزار استفاده می‌شود.

۱۱- تفاوت متدولوژی آبشاری با متدولوژی ساخت‌یافته SSADM چیست؟ (مهندسی IT – دولتی ۸۵)

۱) انجام هر مرحله از متدولوژی SSADM در مدت زمان محدودی بوده و نتایج هر مرحله غیرقابل تغییر و ورودی مرحله بعد می‌باشد.
۲) در متدولوژی SSADM مدت زمان انجام مراحل محدود و معین بوده ولی نتایج هر مرحله قابل تغییر می‌باشد.
۳) انجام متدولوژی آبشاری نیازمند دانستن وضع موجود در یک سازمان یا سیستم می‌باشد.
۴) اساساً متدولوژی آبشاری و SSADM با هم یکسان بوده و فرقی ندارند.

۱۲- منظور از نرم‌افزارهای CASE چیست؟ (مهندسی IT – دولتی ۸۵)

۱) نرم‌افزارهایی است که انجام برخی یا تمامی مراحل موردنظر در یک متدولوژی توسعه نرم‌افزار را پشتیبانی و تسهیل نماید.
۲) نرم‌افزارهایی است که باید تمامی مراحل یک متدولوژی نرم‌افزاری یا مجموعه‌ای از متدولوژی را انجام دهد.
۳) نرم‌افزارهایی را شامل می‌شود که می‌توان به کمک آنها نمودارهای موردنظر در یک یا چند متدولوژی توسعه نرم‌افزار را ترسیم نمود.
۴) نرم‌افزارهایی است که فقط عمل تبدیل طرح یک سیستم نرم‌افزاری را به شکل کد و نرم‌افزار نهایی انجام می‌دهد.

۱۳- کدام‌یک از عبارات زیر، بیان بهتری برای مفهوم توسعه نرم‌افزار (RAD) می‌باشد؟
(مهندسی IT – دولتی ۸۵)

۱) اجرای سریع متدولوژی SSADM را RAD گویند.
۲) روشی است که در دوره‌های کوتاه زمانی، نرم‌افزار قسمت‌های مختلف یک سازمان (Enterprise) را ایجاد کرده و سپس آنها را با هم تلفیق می‌کند.
۳) مجموعه فرآیندهای سازمانی اولویت‌بندی شده و متدولوژی توسعه نرم‌افزار مشخصی در دوره‌های کوتاه برای تهیه نرم‌افزار هر فرآیند استفاده می‌شود.
۴) هر دو گزینه ۲ و ۳ با هم بیان مناسبی از RAD است.

۱۴- در کدام‌یک از مدل‌های فرآیند نرم‌افزار بر کوتاه نمودن چرخه توسعه نرم‌افزار (تولید سریع) تأکید می‌گردد؟ **(مهندسی IT – آزاد ۸۵)**

۱) مدل Incremental
۲) مدل مارپیچی
۳) مدل مارپیچی Win Win
۴) مدل RAD

۱۵- کدام‌یک از گزینه‌های زیر جزء لایه‌های مهندسی نرم‌افزار نمی‌باشد؟ **(مهندسی IT – آزاد ۸۶)**

۱) فرآیند (Process)
۲) محصول (Product)
۳) روش‌ها (Methods)
۴) ابزار (Tools)

۱۶- کدام عبارت در مورد مدل‌ها (روش‌ها)ی توسعه نرم‌افزار صحیح است؟ **(مهندسی IT – دولتی ۸۷)**

۱) مدل‌های الگوسازی و آبشاری را می‌توان در مدل حلزونی جمع کرد.
۲) مدل‌های آبشاری و حلزونی را می‌توان در مدل الگوسازی جمع کرد.
۳) مدل‌های الگوسازی و حلزونی را می‌توان در مدل آبشاری جمع کرد.
۴) هر سه مورد صحیح است.

۱۷- فرض نمایید به شما پیشنهاد ایجاد یک نرم‌افزار ارایه می‌گردد. نرم‌افزار مذکور به عناصری وابسته می‌باشد که پیوسته در حال تغییر بوده و در فاصله کوتاهی قبل از عرضه محصول، نهایی می‌گردند. مدل فرآیند مناسب برای توسعه‌ی این نرم‌افزار کدام است؟ **(مهندسی IT – آزاد ۸۷)**

۱) مدل افزایشی (Incremental)
۲) مدل ترتیبی خطی (Liner)
۳) مدل نمونه‌سازی (Prototyping)
۴) مدل Rapid Application Development

۱۸- کدام مورد به عنوان لایه‌های مهندسی نرم‌افزار صحیح‌تر می‌باشد؟ **(مهندسی IT – دولتی ۸۸)**

۱) مدل فرآیند – ابزار – متدولوژی – کیفیت
۲) مدل فرآیند – متدولوژی – ابزار – کیفیت
۳) کیفیت – مدل فرآیند – متدولوژی – ابزار
۴) کیفیت – مدل فرآیند – روش‌ها – ابزار

۱۹- مهندسی نرم‌افزار اتاق تمیز (Clean Room)، به کدام دسته از مدل‌های فرآیند نرم‌افزار تعلق دارد؟ **(مهندسی IT – آزاد ۸۸)**

۱) مدل بسط هم زمان
۲) مدل مارپیچی Win-Win
۳) مدل توسعه بر مبنای مؤلفه
۴) مدل روش‌های رسمی

۲۰- کدام‌یک از مدل‌های فرآیند توسعه‌ی نرم‌افزار زیر تأکید بر قابلیت استفاده مجدد محصولات تولید شده‌ی نرم‌افزاری ندارد؟ **(مهندسی IT – دولتی ۸۹)**

۱) افزایشی (Incremental) ۲) Rapid Application Development
۳) Component Base Development ۴) نمونه‌سازی (Prototype)

۲۱- کدام‌یک از مدل‌های فرآیند زیر بر اصول ریاضی در توسعه نرم‌افزار تأکید می‌نماید؟
(مهندسی IT – آزاد ۸۹)

۱) مدل گام به گام (Incremental Model) ۲) مدل روش‌های رسمی
۳) مدل مارپیچی ۴) مدل بسط همزمان

۲۲- نیازهای غیروظیفه‌مندی (Non functional) در کدام‌یک از مراحل زیر مورد رسیدگی قرار می‌گیرد؟
(مهندسی IT – آزاد ۸۹)

۱) تحلیل ۲) پیاده‌سازی ۳) طراحی ۴) آزمون

۲۳- تیمی از مهندسین نرم‌افزار با تجربه سیستم جدیدی را در دست تهیه دارند. اگرچه سیستم جدید نسبتاً بزرگ است، اما انتظار نمی‌رود که با سیستم‌هایی که قبلاً توسط این تیم تهیه شده است تفاوت‌های فاحشی داشته باشد. کدام‌یک از مدل‌های چرخه تولید نرم‌افزار (SDLC) زیر برای این پروژه مناسب‌تر است؟ **(مهندسی IT – دولتی ۹۰)**

۱) آبشاری ۲) حلزونی (مارپیچی)
۳) نمونه‌سازی ۴) تکاملی

۲۴- دو پروژه زیر را در نظر بگیرید:
الف) مشتری تأکید بر انجام سریع پروژه حداکثر در مدت سه ماه می‌نماید. تغییرات در درخواست مشتری اندک بوده و امکان تقسیم پروژه به قسمت‌های مستقل وجود دارد.
ب) پروژه نیازمند عواملی می‌باشد که در طول انجام پروژه امکان دارد تغییر نماید و تا چند روز به عرضه نهایی محصول قطعی نمی‌گردد.
مدل‌های فرآیند مناسب جهت پروژه‌های الف و ب به ترتیب از راست به چپ عبارتند از:
(مهندسی IT – آزاد ۹۰)

۱) مدل RAD و مدل Spiral ۲) مدل RAD و مدل Linear
۳) مدل Spiral و مدل RAD ۴) مدل Spiral و مدل Spiral

۲۵- کدام‌یک از ویژگی‌های زیر جزء خصایص مدل روش‌های رسمی نمی‌باشد؟ (مهندسی IT – آزاد ۹۰)
۱) از آنجا که به صورت کلیشه‌ای می‌باشد، هزینه‌ی زمانی اندکی دارد.
۲) با اعمال یک نظم ریاضی شدید سیستم کامپیوتری را توسعه می‌دهد.

۳) در نرم‌افزارهای بحرانی – امنیتی (مثلاً نرم‌افزارهای مرتبط با دستگاه‌های پزشکی و هوافضا) کاربردهای فراوانی دارد.
۴) استفاده از این مدل به عنوان راهکار ارتباطی با مشتریانی که دید فنی ندارند، دشوار است.

۲۶- کدام‌یک از روش‌های زیر، روشی برای استخراج خواسته‌ها نیست؟ **(مهندسی IT – دولتی ۹۱)**

۱) تحلیل ریسک ۲) مصاحبه ۳) مشاهده ۴) ساختن نمونه اولیه

۲۷- کدام‌یک از روش‌های زیر برای پروژه‌ای که از تکنولوژی mobile برای اطلاع‌رسانی در مورد خاصی استفاده می‌کند مناسب است، در صورتی که ما با این تکنولوژی آشنایی نداشته باشیم؟
(مهندسی IT – دولتی ۹۳)

۱) RAD Model ۲) Sequential Model
۳) Rapid Prototyping ۴) Incremental Model

۲۸- کدام‌یک از روش‌های زیر قادر هستند به راحتی ابهامات، ناسازگاری‌ها و نواقص یک سیستم را حین تولید مشخص نمایند؟ **(مهندسی IT – دولتی ۹۳)**

۱) Formal Method ۲) Object Oriented Analysis, RUP
۳) Component Based Development ۴) Concurrent Development Model

---

۲۹- در کدام یک از شیوه‌های زیر، مهندسی نیازمندی‌ها، مکانیسم و ابزار مناسب را فراهم می‌سازد؟
(مهندسی IT – دولتی ۹۴)

۱) ابهام در مشخصات راه‌حل ۲) اعتبارسنجی مشخصات
۳) تحلیل نیازها ۴) هر سه موارد بالا

۳۰- کدام یک از عناصر زیر متعلق به جمع‌آوری نیازها (requirements elicitation) است؟
(مهندسی IT – دولتی ۹۴)

۱) تحلیل نیازها ۲) ارزیابی خطر ۳) مشاهده و بازدید ۴) پیاده‌سازی سیستم

۳۱- کدام یک از موارد زیر اگر وجود داشته باشد از روش تولید سریع برنامه کاربردی (Rapid Application Development) نباید استفاده کرد؟ **(مهندسی IT – دولتی ۹۵)**

۱) ریسک فنی بالا
۲) واحدمند (Modular) بودن سیستم
۳) منابع انسانی کافی برای پروژه‌های بزرگ
۴) تعامل کامل کاربر و تولیدکننده سیستم

پاسخ تست‌های فصل دوم

۱- گزینه (۲) صحیح است.

مدل‌های RAD و Sequential (ترتیبی) یا Linear (خطی) یا Waterfall (آبشاری) جزو مدل‌های غیرتکاملی و مدل افزایشی (Incremental Model) جزو مدل‌های تکاملی سنتی فرآیند تولید نرم‌افزار هستند.

Prototyping (نمونه‌سازی دورانداختنی) یا Rapid Prototyping (نمونه‌سازی سریع) یک مکانیزم، صرفاً جهت شناسایی نیازمندی‌های مشتری است، در یک بیان کلی‌تر نمونه‌سازی دورانداختنی، گفتگو، مشاهده و مصاحبه روش‌هایی جهت شناسایی نیازمندی‌های مشتری و تهیه لیست نیازمندی‌ها می‌باشند. توجه کنید که نمونه‌سازی دورانداختنی یک مدل فرآیند تولید نرم‌افزار به شمار نمی‌آید، بلکه یک مکانیزم صرفاً جهت شناسایی نیازمندی‌های مشتری می‌باشد. به حاصل جمع مکانیزم نمونه‌سازی دورانداختنی و مدل آبشاری «مدل نمونه‌سازی» گفته می‌شود که این مدل نیز یک مدل غیرتکاملی به حساب می‌آید. در واقع مدل نمونه‌سازی از ترکیب مکانیزم نمونه‌سازی دورانداختنی و مدل آبشاری ایجاد شده است. به این نحو که در ابتدای کار توسط مکانیزم نمونه سازی دورانداختنی، تمامی لیست نیازمندی‌های مشتری تکمیل می‌گردد، سپس توسط مدل آبشاری به شکل خطی نرم‌افزار ایجاد می‌گردد و کار تمام می‌شود. در واقع نیازمندهای مشتری طی تکرارهای مکانیزم نمونه‌سازی دورانداختنی شناخته می‌شوند. سپس طی یک روال خطی، توسط مدل آبشاری، نرم‌افزار تولید می‌گردد و کار تمام می‌شود و دیگر تکامل نمی‌یابد. زیرا مدل نمونه سازی یک مدل غیرتکاملی است و پس از ساخت نهایی امکان ادامه‌ی پروژه و تغییر وجود ندارد.

پروژه‌ای که نرم‌افزار آن باید در کنار سخت افزار مرتبط با آن به فروش برسد و سخت افزار آن پیوسته تغییر می‌کند و دستخوش تغییر است، باعث می‌شود تا نرم‌افزار آن نیز پیوسته تغییر کند و دستخوش تغییر باشد.

در چنین شرایطی که نیازها، نرم‌افزار و سخت افزار به طور پیوسته در حال تغییر و تحول هستند، مناسب‌ترین مدل فرآیند تولید نرم‌افزار، مدل‌های تکاملی سنتی (افزایشی و پیچشی)، مدل تکاملی سنتی خاص یعنی مبتنی بر مؤلفه ساخت‌یافته (مبتنی بر تابع) و مدل تکاملی مدرن یعنی مبتنی بر مؤلفه‌ی شیءگرا (مبتنی بر کلاس) هستند. زیرا در اثر مرور زمان و با تغییرات سخت‌افزاری می‌توان نرم‌افزار را نیز در هر بار تکرار (Iteration) تغییر داد.

از آنجا که در صورت سوال، از وجود قطعات و مولفه‌های آماده و قابل استفاده مجدد صحبت نشده است، مدل‌های مبتنی بر مولفه ساخت‌یافته و مبتنی بر مولفه شیءگرا را کنار می‌گذاریم، همچنین از آنجا که شرایط امن بودن پروژه در صورت سوال مطرح نشده است، مدل پیچشی را نیز کنار می‌گذاریم، همچنین مدل‌های غیرتکاملی RAD و آبشاری (Waterfall) را نیز با توجه به

شرایط پروژه کنار می‌گذاریم.

بنابراین با توجه به گزینه‌ها پرواضح است که مدل افزایشی پاسخ درست خواهد بود.

مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. در این مدل، با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در افزایش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد و این تکامل نیز ادامه خواهد داشت.

دقت کنید که می‌توان از مکانیزم نمونه‌سازی دورانداختنی در ابتدای هر تکرار (افزایش) در مدل افزایشی به جهت کاهش ریسک مربوط به شناسایی دقیق لیست نیازمندی‌های مشتری استفاده نمود.

به طور کلی نمونه‌سازی دورانداختنی، به عنوان راه‌کاری برای شناسایی لیست نیازمندی‌های مشتری می‌باشد و در اغلب مدل‌ها می‌توان آن را گنجاند به غیر از مدل آبشاری و مدل RAD .

از آنجا که مدل RAD یک مدل غیرتکاملی است، پرواضح است که مدل مناسبی در این پروژه نخواهد بود، اما به غیر از این مطلب شرط لازم برای استفاده از مدل RAD تقسیم‌پذیر بودن پروژه است که از این مسأله هم صحبتی نشده است.

۲- گزینه (۱) صحیح است.

می‌توان مدل تحلیل و طراحی را به روش و ابزارهای ساخت‌یافته انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان برنامه‌نویسی شیءگرا انجام داد و از امکانات شیءگرایی زبان استفاده نکرد اما عکس این مطلب امکان‌پذیر نیست، یعنی نمی‌توان مدل تحلیل و طراحی را به روش شیءگرا انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان ساخت‌یافته انجام داد، زیرا زبان ساخت‌یافته امکانات شیءگرایی (همچون کلاس، وراثت و چندریختی) را پشتیبانی نمی‌کند.

۳- گزینه (۱) صحیح است.

مدل‌های RAD و Sequential (ترتیبی) یا Linear (خطی) یا Waterfall (آبشاری) جزو مدل‌های غیرتکاملی، روش‌های صوری (Formal Methods) جزو مدل‌های غیرتکاملی مطمئن و مبتنی بر روابط ریاضی و مدل پیچشی (Spiral Model) جزو مدل‌های تکاملی سنتی فرآیند تولید نرم‌افزار هستند.

پروژه‌ای که در ابتدای کار، لیست نیازمندی‌های مشخصی ندارد (مسائل ناشناخته وجود دارد)، در روند ساخت، مدام دچار تغییر و تحول می‌شود. از آنجا که پروژه مربوط به کنترل ترافیک یک فرودگاه بزرگ است، بنابراین پس از ساخت پروژه هم به دلیل تغییر قوانین و نیازهای پروژه امکان تغییر و تحول وجود دارد.

در چنین شرایطی که نیازها در ابتدای کار مشخص نیستند و پس از تحویل محصول باز هم

امکان تغییر و تحول در محصول وجود دارد، مناسب‌ترین مدل فرآیند تولید نرم‌افزار، مدل‌های تکاملی سنتی (افزایشی و پیچشی)، مدل تکاملی سنتی خاص یعنی مبتنی بر مؤلفه ساخت‌یافته (مبتنی بر تابع) و مدل تکاملی مدرن یعنی مبتنی بر مؤلفه‌ی شیءگرا (مبتنی بر کلاس) هستند.

از آنجا که در صورت سوال، از وجود قطعات و مولفه‌های آماده و قابل استفاده مجدد صحبت نشده است، مدل‌های مبتنی بر مولفه ساخت‌یافته و مبتنی بر مولفه شی گرا را کنار می‌گذاریم، همچنین مدل روش‌های صوری را نیز به دلیل وجود مسائل ناشناخته در پروژه و غیرتکاملی بودن ماهیت این مدل، کنار می‌گذاریم، همچنین مدل افزایشی را به دلیل اهمیت ایمنی مطرح شده در پروژه و عدم پشتیبانی ایمنی توسط مدل افزایشی، کنار می‌گذاریم، همچنین مدل‌های غیرتکاملی RAD و آبشاری (Waterfall) را نیز با توجه به شرایط پروژه کنار می‌گذاریم.

بنابراین با توجه به گزینه‌ها و اهمیت ایمنی و مدیریت کلیه ریسک‌های پروژه بر اساس شرایط پروژه، پرواضح است که مدل پیچشی (Spiral Model) پاسخ درست خواهد بود.

مدل پیچشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. در این مدل، با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در پیچش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد و این تکامل نیز ادامه خواهد داشت.

توجه کنید که می‌توان از مکانیزم نمونه‌سازی دورانداختنی در ابتدای هر تکرار (پیچش) در مدل پیچشی به جهت کاهش ریسک مربوط به شناسایی دقیق لیست نیازمندی‌های مشتری استفاده نمود.

به طور کلی نمونه‌سازی دورانداختنی، به عنوان راه‌کاری برای شناسایی لیست نیازمندی‌های مشتری می‌باشد و در اغلب مدل‌ها می‌توان آن را گنجاند به غیر از مدل آبشاری و مدل RAD .

مدل پیچشی از مکانیزم نمونه‌سازی دورانداختنی به عنوان راهکاری برای کاهش ریسک مربوط به شناسایی نیازمندی‌ها استفاده می‌کند و همچنین به دلیل تحلیل ریسک به واسطه‌ی مدیریت ریسک تمامی ریسک‌های مربوط به کلیت پروژه (شناسایی نیازمندی‌های دقیق مشتری، مقرون به صرفه بودن و در زمان مورد انتظار بودن) را کنترل می‌کند. در بیان ساده، مدل پیچشی به واسطه‌ی مدیریت ریسک (تحلیل ریسک) بستری امن، برای تولید نرم‌افزار ایده آل مشتری فراهم می‌سازد.

از آنجا که مدل RAD یک مدل غیرتکاملی است، پرواضح است که مدل مناسبی در این پروژه نخواهد بود، اما به غیر از این مطلب شرط لازم برای استفاده از مدل RAD تقسیم‌پذیر بودن پروژه است که از این مسأله هم صحبتی نشده است.

۴- گزینه (۱) صحیح است.

می‌توان مدل تحلیل و طراحی را به روش و ابزارهای ساخت‌یافته انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان برنامه‌نویسی شیءگرا انجام داد و از امکانات شیءگرایی زبان استفاده نکرد، اما عکس این مطلب امکان‌پذیر نیست، یعنی نمی‌توان مدل تحلیل و طراحی را به روش شیءگرا انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان ساخت‌یافته انجام داد زیرا زبان ساخت‌یافته امکانات شیءگرایی (همچون کلاس، وراثت و چندریختی) را پشتیبانی نمی‌کند.

متدولوژی SSADM متداول‌ترین نمونه از متدولوژی ساخت‌یافته براساس روش ساخت‌یافته، مدل فرآیند تولید آبشاری و ابزارهای ساخت‌یافته می‌باشد. و بر دو نوع داده‌گرا (جدولی) مانند نرم‌افزار حقوق و دستمزد که داده‌ها درون جداول ذخیره می‌شوند و تابع‌گرا (متغیری) مانند نرم‌افزار ماشین‌حساب که داده‌ها درون متغیرها ذخیره می‌شوند، می‌باشد.

بنابراین گزینه اول درست و گزینه‌های دوم، سوم و چهارم نادرست هستند.

۵- گزینه (۳) صحیح است.

اساس به وجود آمدن متدولوژی شیءگرا به وجود آمدن نیازهای جدید بوده که توسط متدولوژی ساخت‌یافته (تابع‌گرا یا داده‌گرا) قابل پوشش دادن نبوده‌اند. متدولوژی شیءگرا برخی از نیازمندی‌های جدید را پوشش داده است و این طور نبوده است که استفاده از متدولوژی ساخت‌یافته را منسوخ کند. متدولوژی شیءگرا و متدلوژی ساخت‌یافته هر دو زیرمجموعه‌ای از متدولوژی‌های نسل دوم یا ساختارمند می‌باشند. منظور از کلمه «ساخت‌یافته» در گزینه چهارم، ساختارمند بودن متدولوژی است و نه مفهوم متدولوژی ساخت‌یافته. بنابراین گزینه سوم درست است.

۶- گزینه (۴) صحیح است.

مهم‌ترین کار در فعالیت ارتباطات، مدیریت شناسایی نیازمندی‌ها و پیگیری تغییرات نیازمندی‌های مشتری است.

۷- گزینه (۱) صحیح است.

به طور کلی انواع نیازمندی‌های نرم‌افزار را می‌توانیم به دو دسته زیر تقسیم کنیم:

۱- وظیفه‌مندی (کارکردی، عملکردی)

نیازمندی‌های وظیفه‌مندی، کمّی و قابل اندازه‌گیری هستند و در قالب قابلیت‌ها، کارکردها، ویژگی‌ها و سرویس‌های سیستم در حال توسعه یا تولید محقق می‌گردند. به عبارت دیگر، نیازمندی‌های کارکردی به بیان سرویس‌هایی که سیستم باید فراهم نماید، می‌پردازد. چگونگی واکنش سیستم در برابر ورودی‌های خاص و چگونگی رفتار سیستم در شرایط خاص نیز توسط

این نیازمندی‌ها تعریف می‌شود.

مانند نیازمندی‌های مربوط به محاسبه‌ی فاکتوریل یک عدد و یا اجرای تابع فیبوناچی در یک نرم‌افزار و یا نیازمندی‌های مربوط به یک نرم‌افزار حقوق و دستمزد.

۲- غیروظیفه‌مندی (غیرکارکردی، غیر عملکردی)

نیازمندی‌های غیروظیفه‌مندی، نیازمندی‌های کیفی و نه الزاماً قابل اندازه‌گیری هستند که به بیان کیفیت مورد انتظار از نیازمندی‌های وظیفه‌مندی و همچنین محدودیت‌هایی نظیر محدودیت‌های زمانی، مالی و استانداردها می‌پردازند. برخی از انواع نیازمندی‌های غیروظیفه‌مندی عبارتند از:

قابلیت استفاده، سهولت یادگیری، قابلیت اعتماد، کارایی، زمان پاسخ، قابلیت پشتیبانی، قابلیت نگهداری، قابلیت حمل، بهره‌وری، محدودیت‌های طراحی، پیاده‌سازی و فیزیکی و همچنین نیازمندی‌های واسط کاربردی.

مانند محاسبه تابع فاکتوریل توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی $O(n)$ و مصرف زیاد حافظه به دلیل استفاده از استک در پی هر فراخوانی.

و یا مانند محاسبه تابع فیبوناچی توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی نمایی زیاد $O(۲^{\frac{n}{۲}})$ و چاق و به تبع، کند و هم مصرف زیاد حافظه به دلیل استفاده از استک، در پی هر فراخوانی.

انتخاب نوع الگوریتم براساس شرایط، حائز اهمیت می‌باشد.

محصول نرم‌افزاری که نیازمندی‌های وظیفه‌مندی را برآورده می‌کند، ولی برآورنده نیازهای غیروظیفه‌مندی و کیفی نباشد، معمولاً با نارضایتی مشتریان همراه می‌شود.

در فعالیت تست، نیازمندی‌های وظیفه‌مندی و غیروظیفه‌مندی جهت اطمینان از صحت عملکرد آنها مورد ارزیابی مجدد قرار می‌گیرند.

۸- گزینه (۲) صحیح است.

در صورتی که کلیه نیازمندی‌های مشتری در ابتدای پروژه مشخص باشد، مدل آبشاری، به عنوان مدلی کارآمد و ساده جهت مدیریت ساده فرآیند تولید نرم‌افزار مورد استفاده قرار می‌گیرد. مدل آبشاری تکرار و بازگشت به عقب ندارد و مدل‌های دیگر را در بر نمی‌گیرد، یعنی از اجتماع مدل‌های دیگر ایجاد نشده است.

۹- گزینه (۲) صحیح است.

نگهداری نرم‌افزار بیشترین هزینه و امکان‌سنجی کمترین هزینه را دارا است.

۱۰- گزینه (۳) صحیح است.

فعالیت ارتباطات و بخش مدل‌سازی لیست نیازمندی‌ها در مدل تحلیل، در متدولوژی

ساخت‌یافته و متدولوژی شیءگرا یکسان است. بنابراین در این دو مرحله از هریک از متدولوژی‌های ساخت‌یافته و شیءگرا می‌توان به دیگری تغییر مسیر داد. اما پس از عبور از این دو مرحله دیگر تحت هیچ شرایطی نمی‌توان از متدولوژی شیءگرا به متدولوژی ساخت‌یافته تغییر مسیر داد. اما در متدولوژی ساخت‌یافته می‌توان مدل تحلیل و طراحی را به روش و ابزارهای ساخت‌یافته انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان برنامه‌نویسی شیءگرا انجام داد و از امکانات شیءگرایی زبان استفاده نکرد، اما عکس این مطلب امکان‌پذیر نیست، یعنی نمی‌توان مدل تحلیل و طراحی را به روش شیءگرا انجام داد ولی فعالیت پیاده‌سازی را توسط یک زبان ساخت‌یافته انجام داد زیرا زبان ساخت‌یافته امکانات شیءگرایی (همچون کلاس، وراثت و چندریختی) را پشتیبانی نمی‌کند. بنابراین گزینه سوم درست است و گزینه‌های اول و دوم نادرست هستند.

متدولوژی ساخت‌یافته و شیءگرا هر یک به تنهایی کلیه فعالیت‌های چارچوبی مربوط به فرآیند تولید نرم‌افزار (ارتباطات، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست و استقرار) را پشتیبانی می‌کنند. بنابراین گزینه چهارم نیز نادرست است.

۱۱- گزینه (۱) صحیح است.

اشکالی که به این سوال وارد است این است که مدل فرآیند تولید نرم‌افزار آبشاری یک متدولوژی در نظر گرفته شده است. بنابراین صورت سوال باید به صورت زیر اصلاح گردد:

«تفاوت مدل آبشاری با متدولوژی ساخت‌یافته SSADM چیست؟»

متدولوژی مجموعه‌ای از فرآیندها، روش‌ها و ابزارهای مرتبط با هم و همه از یک متدولوژی خاص همچون ساخت‌یافته یا شیءگرا برای ایجاد یک محصول نرم‌افزاری، مطابق با استانداردهای مهندسی نرم‌افزار می‌باشد و بر دو طبقه‌ی ساخت‌یافته و شیءگرا است.

متدولوژی ساخت‌یافته یا مهندسی نرم‌افزار ساخت‌یافته نظامی است یکپارچه شامل مدل فرآیندهای ساخت‌یافته (سنتی)، روش ساخت‌یافته و ابزارهای ساخت‌یافته که منجر به ایجاد نرم‌افزاری در بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و دقیقاً مطابق با نیازمندی‌های واقعی مشتری می‌گردد.

متدولوژی SSADM متداول‌ترین نمونه از متدولوژی ساخت‌یافته براساس روش ساخت‌یافته، مدل فرآیند تولید آبشاری و ابزارهای ساخت‌یافته می‌باشد.

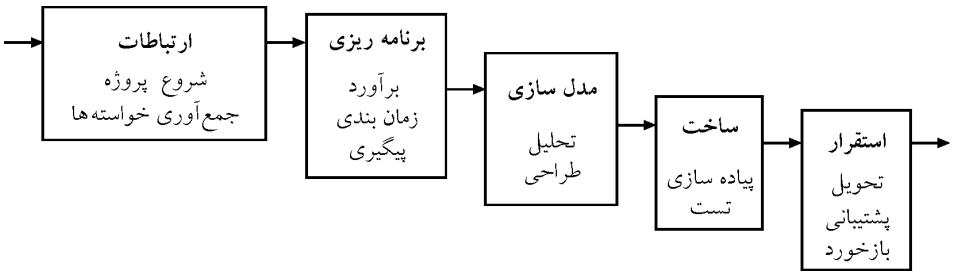
نسبت نمونه متدولوژی ساخت‌یافته SSADM به متدولوژی ساخت‌یافته، مثل نسبت سیستم عامل ویندوز سنتی به مفاهیم سنتی سیستم عامل است.

مدل آبشاری، که گاه از آن به عنوان چرخه‌ی حیات کلاسیک یاد می‌شود، روشی ترتیبی برای تولید نرم‌افزار پیشنهاد می‌کند. این مدل با فعالیت ارتباطات شروع می‌شود و با فعالیت‌های برنامه‌ریزی، مدل‌سازی و ساخت پیش می‌رود و با فعالیت استقرار پایان می‌یابد، تحویل پروژه

انجام می‌گیرد و کار تمام می‌شود. و دیگر فرصت و تکرار دیگری در کار نخواهد بود تا خواسته‌های فراموش شده یا جدید به پروژه اضافه گردد.

خصوصیت اصلی این مدل این است که هیچ‌گونه بازخوردی بین مراحل این مدل وجود ندارد. مانند آب که نمی‌تواند در آبشار به عقب برگردد، در این مدل نیز بعد از ورود به یک فعالیت به فعالیت‌های قبلی نمی‌توان بازگشت. این مدل زمانی کاربرد دارد که کلیه‌ی نیازمندی‌های مشتری در همان ابتدای پروژه مشخص، ثابت و بدون تغییر باشد.

مدل آبشاری در شرایطی که خواسته‌ها به طور کامل و جامع مشخص، ثابت و پایدار است و قرار است که کار تا پایان به شیوه‌ای خطی پیش برود، می‌تواند به عنوان مدلی مفید، مورد استفاده قرار گیرد.



مدل آبشاری

متدولوژی SSADM، یک نمونه متدولوژی ساخت‌یافته یا یک نمونه مهندسی‌نرم‌افزار ساخت‌یافته است. بنابراین از آنجا که تعریف متدولوژی ساخت‌یافته یا مهندسی‌نرم‌افزار ساخت‌یافته، براساس روش ساخت‌یافته، مدل فرآیند تولید آبشاری و ابزارهای ساخت‌یافته برای آن برقرار است، لذا SSADM منجر به ایجاد نرم‌افزاری در بازه‌ی زمانی از قبل برنامه‌ریزی شده، بودجه‌ای از قبل پیش‌بینی شده و دقیقاً مطابق با نیازمندی‌های واقعی مشتری می‌گردد.

از آنجاکه SSADM بسته‌ای کامل از فرآیند، روش و ابزار است، بنابراین می‌تواند در فضایی مناسب خصوصیات SSADM، برآورده‌سازی نیازمندی‌های مشتری، محدودیت زمانی (مدت زمان محدود) و مقرون به صرفه بودن را گارانتی کند. همچنین از آنجاکه SSADM جهت هدایت فرآیند تولید نرم‌افزار، مدل فرآیند آبشاری را مورد استفاده قرار می‌دهد، لذا مطابق خصوصیات مدل آبشاری و به تبع در SSADM نتایج هر مرحله غیر قابل تغییر و ورودی مرحله بعد می‌باشد.

همچنین از آنجاکه مدل آبشاری به تنهایی بسته‌ای کامل از فرآیند، روش و ابزار نیست و صرفا به تنهایی فقط یک مدل فرایند تولید نرم‌افزار است، بنابراین نمی‌تواند مطابق تعریف متدولوژی و مهندسی نرم‌افزار، برآورده‌سازی نیازمندی‌های مشتری، محدودیت زمانی(مدت زمان محدود) و مقرون به صرفه بودن را گارانتی کند. که همین مورد اخیر تفاوت متدولوژی SSADM

و مدل آبشاری است. بنابراین گزینه اول درست و گزینه دوم نادرست است.

همچنین از آنجاکه SSADM جهت هدایت فرآیند تولید نرم‌افزار، مدل فرآیند آبشاری را مورد استفاده قرار می‌دهد، بنابراین، هم SSADM و هم مدل آبشاری، مطابق خصوصیات مدل آبشاری نیازمند دانستن وضع موجود در یک سازمان یا سیستم می‌باشند، زیرا مدل آبشاری زمانی کاربرد دارد که کلیه‌ی نیازمندی‌های مشتری در همان ابتدای پروژه مشخص، ثابت و بدون تغییر باشد. که این شباهت است و نه تفاوت SSADM و مدل آبشاری. بنابراین گزینه سوم نیز نادرست است.

SSADM یک نمونه متدولوژی ساخت‌یافته و مدل آبشاری یک مدل فرآیند تولید نرم‌افزار است. بنابراین شاید در برخی موارد مشابه باشند، اما در کل متفاوت هستند، پس گزینه چهارم نیز نادرست است.

۱۲- گزینه (۱) صحیح است.

ابزارهای CASE فقط نمودار ترسیم نمی‌نمایند بلکه می‌توانند کلیه مراحل فرآیند تولید نرم‌افزار (ارتباطات، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست) و استقرار) را انجام دهند که حتمی و اجباری نمی‌باشد.

۱۳- گزینه (۴) صحیح است.

مدل RAD، شکل پُرسرعت مدل آبشاری می‌باشد، با این تفاوت که پروژه به بخش‌های مختلف تقسیم شده و هر بخش، توسط یک تیم، مطابق مدل آبشاری ایجاد می‌گردد و در پایان نتیجه‌ی تیم‌ها، برای خلق محصول نهایی ترکیب می‌گردد. مدل RAD، سرعت خود را مدیون بهره‌گیری از تکنیک بخش‌بندی و موازی‌سازی بخش‌های مختلف پروژه است. چنانچه نیازمندی‌ها به خوبی شناسایی شده و دامنه پروژه کوچک باشد این مدل قادر است یک سیستم کاملاً عملیاتی را در مدت زمان بسیار کوتاه (مثلاً بین ۶۰ تا ۹۰ روز) تولید نماید. در این مدل نرم‌افزار به قسمت های مختلف تقسیم شده و همواره سعی می‌شود که نرم‌افزار موردنظر سریع‌تر تولید شود. نکته قابل توجه این است که نرم‌افزار موردنظر باید خاصیت تفکیک‌پذیری داشته باشد تا بتوان این مدل را پیاده‌سازی کرد.

SSADM یک نمونه متدولوژی ساخت‌یافته است، RAD یک مدل فرآیند تولید نرم‌افزار است.

منظور از عبارت «مجموعه فرآیندهای سازمانی اولویت بندی شده» در گزینه سوم، امکان سنجی بخش‌های (فرآیندهای) مختلف محیط عملیاتی کسب و کار سنتی برای مکانیزه‌کردن بخش مورد نظر است.

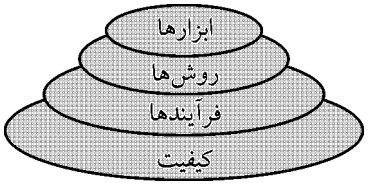
۱۴- گزینه (۴) صحیح است.

مدل RAD، شکل پُرسرعت مدل آبشاری می‌باشد، با این تفاوت که پروژه به بخش‌های

مختلف تقسیم شده و هر بخش، توسط یک تیم، مطابق مدل آبشاری ایجاد می‌گردد و در پایان نتیجه‌ی تیم‌ها، برای خلق محصول نهایی ترکیب می‌گردد. مدل RAD، سرعت خود را مدیون بهره‌گیری از تکنیک بخش‌بندی و موازی‌سازی بخش‌های مختلف پروژه است. چنانچه نیازمندی‌ها به خوبی شناسایی شده و دامنه پروژه کوچک باشد این مدل قادر است یک سیستم کاملاً عملیاتی را در مدت زمان بسیار کوتاه (مثلاً بین ۶۰ تا ۹۰ روز) تولید نماید. در این مدل نرم‌افزار به قسمت‌های مختلف تقسیم شده و همواره سعی می‌شود که نرم‌افزار موردنظر سریع‌تر تولید شود. نکته قابل توجه این است که نرم‌افزار موردنظر باید خاصیت تفکیک‌پذیری داشته باشد تا بتوان این مدل را پیاده‌سازی کرد.

۱۵- گزینه (۲) صحیح است.

مهندسی نرم‌افزار یک فرآیند لایه‌ای است. به بیان دیگر مهندسی نرم‌افزار از چهار لایه تشکیل شده است. در شکل زیر، چهار لایه نشان داده شده است:



۱۶- گزینه (۱) صحیح است.

مدل پیچشی نیز همانند مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. بنابراین مدل پیچشی مبتنی بر مدل آبشاری و مکانیزم نمونه‌سازی تکاملی است. نمونه‌سازی تکاملی به فرآیند تولید نرم‌افزار روح، حرکت و تکرار می‌دهد و مدل آبشاری فعالیت‌های چارچوبی هر تکرار (پیچش) را مشخص می‌کند.

به مکانیزم‌نمونه‌سازی تکاملی، **الگوسازی باز** نیز گفته می‌شود.

همچنین، هرگاه نیاز به شناسایی خواسته‌های مبهم مشتری وجود داشت می‌توان از مکانیزم نمونه‌سازی دورانداختنی استفاده نمود. بنابراین قبل از هر تکرار (پیچش) جهت شناسایی نیازمندی‌ها می‌توان از مکانیزم نمونه‌سازی دورانداختنی استفاده نمود و نمونه را دور انداخت. اما حرکت، تکرار و تکامل همچنان می‌تواند توسط نمونه‌سازی تکاملی ادامه یابد تا محصولی نهایی آماده گردد.

این مدل از یک سری فعالیت‌های چارچوبی تکراری تشکیل شده است که هر تکرار (پیچش) شبیه به مدل آبشاری است. با این تفاوت که روی قسمتی از نرم‌افزار انجام می‌شود. هر کدام از این قسمت‌ها یک «قطعه» قابل تحویل را ایجاد می‌کند.

به مکانیزم نمونه‌سازی دورانداختنی، **الگوسازی بسته** نیز گفته می‌شود.

۱۷- گزینه (۱) صحیح است.

مدل‌های RAD و Sequential (ترتیبی) یا Linear (خطی) یا Waterfall (آبشاری) جزو مدل‌های غیرتکاملی و مدل افزایشی (Incremental Model) جزو مدل‌های تکاملی سنتی فرآیند تولید نرم‌افزار هستند.

Prototyping (نمونه‌سازی دورانداختنی) یا Rapid Prototyping (نمونه‌سازی سریع) یک مکانیزم، صرفاً جهت شناسایی نیازمندی‌های مشتری است، در یک بیان کلی‌تر نمونه‌سازی دورانداختنی، گفتگو، مشاهده و مصاحبه روش‌هایی جهت شناسایی نیازمندی‌های مشتری و تهیه لیست نیازمندی‌ها می‌باشند. توجه کنید که نمونه‌سازی دورانداختنی یک مدل فرآیند تولید نرم‌افزار به شمار نمی‌آید، بلکه یک مکانیزم صرفاً جهت شناسایی نیازمندی‌های مشتری می‌باشد. به حاصل جمع مکانیزم، نمونه‌سازی دورانداختنی و مدل آبشاری «مدل نمونه‌سازی» گفته می‌شود که این مدل نیز یک مدل غیرتکاملی به حساب می‌آید. در واقع مدل نمونه‌سازی از ترکیب مکانیزم نمونه‌سازی دورانداختنی و مدل آبشاری ایجاد شده است. به این نحو که در ابتدای کار توسط مکانیزم نمونه‌سازی دورانداختنی، تمامی لیست نیازمندی‌های مشتری تکمیل می‌گردد، سپس توسط مدل آبشاری به شکل خطی نرم‌افزار ایجاد می‌گردد و کار تمام می‌شود. در واقع نیازمندهای مشتری طی تکرارهای مکانیزم نمونه‌سازی دورانداختنی شناخته می‌شوند. سپس طی یک روال خطی، توسط مدل آبشاری، نرم‌افزار تولید می‌گردد و کار تمام می‌شود و دیگر تکامل نمی‌یابد. زیرا مدل نمونه‌سازی یک مدل غیرتکاملی است و پس از ساخت نهایی امکان ادامه‌ی پروژه و تغییر وجود ندارد.

پروژه‌ای که نرم‌افزار آن به عناصری وابسته است که پیوسته در حال تغییر بوده و در فاصله کوتاهی قبل از عرضه محصول، باید نهایی گردد، در روند ساخت، مدام دچار تغییر و تحول می‌شود.

در چنین شرایطی که نیازها، نرم‌افزار و عناصر مرتبط با نرم‌افزار به طور پیوسته در حال تغییر و تحول هستند، مناسب‌ترین مدل فرآیند تولید نرم‌افزار، مدل‌های تکاملی سنتی (افزایشی و پیچشی)، مدل تکاملی سنتی خاص یعنی مبتنی بر مؤلفه ساخت‌یافته (مبتنی بر تابع) و مدل تکاملی مدرن یعنی مبتنی بر مؤلفه‌ی شیءگرا (مبتنی بر کلاس) هستند. زیرا در اثر مرور زمان و با تغییرات سخت‌افزاری می‌توان نرم‌افزار را نیز در هر بار تکرار (Iteration) تغییر داد.

از آنجا که در صورت سوال، از وجود قطعات و مولفه‌های آماده و قابل استفاده مجدد صحبت نشده است، مدل‌های مبتنی بر مولفه ساخت‌یافته و مبتنی بر مولفه شی گرا را کنار می‌گذاریم، همچنین از آنجا که شرایط امن بودن پروژه در صورت سوال مطرح نشده است، مدل پیچشی را نیز کنار می‌گذاریم، همچنین مدل‌های غیرتکاملی RAD و آبشاری (Waterfall) را نیز با توجه به

شرایط پروژه کنار می‌گذاریم.

بنابراین با توجه به گزینه‌ها پرواضح است که مدل افزایشی پاسخ درست خواهد بود.

مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. در این مدل، با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در افزایش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد و این تکامل نیز ادامه خواهد داشت.

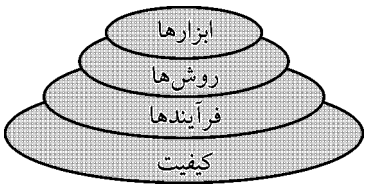
توجه کنید که می‌توان از مکانیزم نمونه‌سازی دورانداختنی در ابتدای هر تکرار (افزایش) در مدل افزایشی به جهت کاهش ریسک مربوط به شناسایی دقیق لیست نیازمندی‌های مشتری استفاده نمود.

به طور کلی نمونه‌سازی دورانداختنی، به عنوان راه‌کاری برای شناسایی لیست نیازمندی‌های مشتری می‌باشد و در اغلب مدل‌ها می‌توان آن را گنجاند به غیر از مدل آبشاری و مدل RAD .

از آنجا که مدل RAD یک مدل غیرتکاملی است، پرواضح است که مدل مناسبی در این پروژه نخواهد بود، اما به غیر از این مطلب شرط لازم برای استفاده از مدل RAD تقسیم‌پذیر بودن پروژه است که از این مسأله هم صحبتی نشده است.

۱۸- گزینه (۴) صحیح است.

مهندسی نرم‌افزار یک فرآیند لایه‌ای است. به بیان دیگر مهندسی نرم‌افزار از چهار لایه تشکیل شده است. در شکل زیر، چهار لایه نشان داده شده است:



۱۹- گزینه (۴) صحیح است.

مدل روش‌های رسمی (فرمال، قراردادی و صوری) شامل مجموعه‌ای از فعالیت‌ها است که سعی دارد پروژه‌ی نرم‌افزاری را در قالب روابطی رسمی و ریاضی، سیستم‌های کامپیوتری را تعریف، توسعه، پیاده‌سازی و ارزیابی نماید. در این مدل، با استفاده از تحلیل‌های ریاضی، بسیاری از ابهامات، نواقص و عدم سازگاری نرم‌افزار را تا حد زیادی می‌توان به سادگی کشف و تصحیح نمود. گونه‌ای از روش‌های رسمی وجود دارد که به مهندسی نرم‌افزار **اتاق تمیز** معروف است.

این مدل، امکان کشف و تصحیح خطاهای زیادی را که تا زمان اجرا غیرقابل تشخیص هستند را در طول مراحل اولیه‌ی تولید نرم‌افزار، فراهم می‌کند.

یکی از مهمترین کاربردهای مدل روش‌های رسمی، ساخت سیستم‌های حساس و امن مانند

ساخت نرم‌افزارهای کنترل هواپیما و موشک است.

۲۰- گزینه (۲) صحیح است.

مدل‌های تکاملی، محصولات تولید شده در تکرارهای قبلی را ارتقاء داده و از قابلیت‌های آن‌ها در تکرارهای بعدی استفاده می‌کنند. اما در مدل‌های غیرتکاملی محصول نهایی یکجا و در انتهای کار ایجاد می‌گردد.

مدل RAD جزو مدل‌های غیرتکاملی، مدل افزایشی جزو مدل‌های تکاملی سنتی، مدل مبتنی بر مولفه ساخت‌یافته (مبتنی بر تابع) جزو مدل تکاملی سنتی خاص و مدل مبتنی بر مولفه شیءگرا (مبتنی بر کلاس) جزو مدل تکاملی مدرن فرآیند تولید نرم‌افزار است.

مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. بنابراین مدل افزایشی مبتنی بر مدل آبشاری و مکانیزم نمونه‌سازی تکاملی است.

نمونه‌سازی تکاملی به فرآیند تولید نرم‌افزار روح، حرکت و تکرار می‌دهد و مدل آبشاری فعالیت‌های چارچوبی هر تکرار (افزایش) را مشخص می‌کند. همچنین، هرگاه نیاز به شناسایی خواسته‌های مبهم مشتری وجود داشت می‌توان از مکانیزم نمونه‌سازی دورانداختنی استفاده نمود. بنابراین قبل از هر تکرار (افزایش) جهت شناسایی نیازمندی‌ها می‌توان از مکانیزم نمونه‌سازی دورانداختنی نیز استفاده نمود و نمونه را دور انداخت. اما حرکت، تکرار و تکامل همچنان می‌تواند توسط نمونه‌سازی تکاملی ادامه یابد تا محصول نهایی آماده گردد.

این مدل از یک سری فعالیت‌های چارچوبی تکراری تشکیل شده است که هر تکرار (افزایش) شبیه به مدل آبشاری است. با این تفاوت که روی قسمتی از نرم‌افزار انجام می‌شود. هر کدام از این قسمت‌ها یک «قطعه» قابل تحویل را ایجاد می‌کند.

در این مدل، با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در افزایش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد.

هنگامی که از یک مدل افزایشی استفاده می‌شود، افزایش نخست غالباً هسته‌ی اصلی محصول است، یعنی نیازهای اولیه رفع می‌شوند، اما بسیاری از ویژگی‌های مکمل (که برخی معلوم و برخی نامعلوم هستند) تحویل داده نمی‌شوند. هسته‌ی اصلی محصول، توسط مشتری مورد استفاده و ارزیابی قرار می‌گیرد. سپس در افزایش بعدی، قابلیت‌ها و ویژگی‌های دیگر مورد انتظار مشتری به هسته‌ی اصلی محصول اضافه می‌گردد. این فرآیند به دنبال تحویل هر قطعه تکرار می‌شود تا اینکه محصول کامل تولید شود. بنابراین مدل افزایشی دیدی پیوسته به توسعه و تکامل نرم‌افزار دارد و در نسخه افزایشی خود نرم‌افزار کامل‌تری را ارائه می‌کند. در واقع ورودی هر مرحله از آن نسخه‌ای می‌باشد که در افزایش قبلی تولید شده است، پس این مدل تاکید بر قابلیت استفاده مجدد

از محصولات تولید شده قبلی را دارد.

مدل RAD، شکل پُرسرعت مدل آبشاری می‌باشد، با این تفاوت که پروژه به بخش‌های مختلف تقسیم شده و هر بخش، توسط یک تیم، مطابق مدل آبشاری ایجاد می‌گردد و در پایان نتیجه‌ی تیم‌ها، برای خلق محصول نهایی ترکیب می‌گردد. مدل RAD، سرعت خود را مدیون بهره‌گیری از تکنیک بخش‌بندی و موازی‌سازی بخش‌های مختلف پروژه است. چنانچه نیازمندی‌ها به خوبی شناسایی شده و دامنه پروژه کوچک باشد این مدل قادر است یک سیستم کاملاً عملیاتی را در مدت زمان بسیار کوتاه (مثلاً بین ۶۰ تا ۹۰ روز) تولید نماید. در این مدل نرم‌افزار به قسمت‌های مختلف تقسیم شده و همواره سعی می‌شود که نرم‌افزار موردنظر سریع‌تر تولید شود. نکته قابل توجه این است که نرم‌افزار موردنظر باید خاصیت تفکیک‌پذیری داشته باشد تا بتوان این مدل را پیاده‌سازی کرد. مدل RAD یک مدل غیر تکاملی است، پس این مدل تاکید بر قابلیت استفاده مجدد از محصولات تولید شده قبلی را ندارد. زیرا مدل‌های غیرتکاملی، محصول نهایی را یکجا و در انتهای کار ایجاد می‌کنند. تولید تکاملی یعنی نرم‌افزار قطعه قطعه، ذره ذره و کم‌کم، کم‌کم تکامل می‌یابد و نه در قالب یک قطعه و یکجا.

مؤلفه یک قطعه‌ی آماده، کاربردی و پیاده‌سازی شده است که دارای واسط لازم جهت اتصال با سایر قطعات و استقرار در بخشی از یک سیستم عملیاتی می‌باشد.

در دیدگاه ساخت‌یافته به مؤلفه پیمانه نیز گفته می‌شود. در حیطه‌ی مهندسی نرم‌افزار ساخت‌یافته، مولفه یک قطعه عملیاتی مبتنی بر تابع است که بر دو نوع می‌باشد:

۱) تابع کاربردی: مانند تابع جمع که برنامه‌نویس آن را می‌نویسد. اگر تابع کاربردی، شرایط قابل حمل بودن (مانند تعریف متغیر درون تابع به صورت‌محلی و صریح و عدم استفاده از متغیرهای سراسری) را داشته باشد می‌تواند به عنوان یک قطعه‌ی آماده‌ی قابل استفاده‌ی مجدد در پروژه‌های بعدی مورد استفاده مجدد قرار گیرد.

۲) تابع سیستمی: مانند تابع سینوس که کامپایلر تعاریف آن را فراهم می‌کند و می‌تواند به عنوان یک قطعه‌ی آماده و قابل استفاده‌ی مجدد، در پروژه‌ها مورد استفاده مجدد قرار گیرد.

مدل توسعه‌ی مبتنی بر مؤلفه‌ی ساخت‌یافته از بسیاری از خصوصات مدل‌های تکاملی سنتی استفاده می‌کند. این مدل از نظر ماهیت، مدلی تکاملی است و ساختاری تکرارشونده را برای توسعه‌ی نرم‌افزار در پیش می‌گیرد. اما در تولید برنامه‌های کاربردی از مؤلفه‌های آماده استفاده می‌کند، در واقع این مدل برنامه را با استفاده از ترکیب و سرهم کردن قطعات نرم‌افزاری از پیش ساخته شده می‌سازد.

زمان و هزینه‌ی فرآیند تولید نرم‌افزار براساس مدل توسعه‌ی مبتنی بر مؤلفه ساخت‌یافته به دلیل استفاده از قطعات آماده (قابلیت استفاده مجدد) کاهش چشمگیری دارد. در واقع یک قطعه با قابلیت استفاده‌ی مجدد یک بار ساخته می‌شود و بارها و بارها استفاده می‌شود و این یعنی سودآوری!

در دیدگاه شیءگرا به مؤلفه پیمانه نیز گفته می‌شود. در حیطه مهندسی نرم‌افزار شیءگرا، **واحد مؤلفه** یک قطعه‌ی عملیاتی مبتنی بر کلاس است که بر دو نوع می‌باشد:

۱) کلاس کاربردی: مانند کلاس دانشجو که برنامه‌نویس آن را می‌نویسد و به دلیل داشتن شرایط قابل حمل به شکل ذاتی، می‌تواند به عنوان یک قطعه‌ی آماده و قابل استفاده‌ی مجدد در پروژه‌های بعدی مورد استفاده مجدد قرار گیرد.

۲) کلاس سیستمی: مانند کلاس جعبه‌ی متن که کامپایلر تعاریف آن را فراهم می‌کند و می‌تواند به عنوان یک قطعه‌ی آماده و قابل استفاده‌ی مجدد، در پروژه‌ها مورد استفاده مجدد قرار گیرد.

این مدل از نظر ماهیت، مدلی تکاملی است و ساختاری تکرارشونده را برای توسعه‌ی نرم‌افزار در پیش می‌گیرد. اما در تولید برنامه‌های کاربردی از مؤلفه‌های آماده استفاده می‌کند، در واقع این مدل برنامه را با استفاده از ترکیب و سرهم کردن قطعات نرم‌افزاری از پیش ساخته شده می‌سازد.

زمان و هزینه‌ی فرآیند تولید نرم‌افزار براساس مدل توسعه‌ی مبتنی بر مؤلفه شیءگرا به دلیل استفاده از قطعات آماده (قابلیت استفاده مجدد) کاهش چشمگیری دارد. در واقع یک قطعه با قابلیت استفاده‌ی مجدد یک بار ساخته می‌شود و بارها و بارها استفاده می‌شود و این یعنی سودآوری!

بنابراین مستقل از ساخت‌یافته یا شیءگرا بودن مدل مبتنی بر مولفه، مولفه‌ها، قطعات و محصولات آماده شده در پروژه‌های قبلی در پروژه‌های آتی مورد استفاده قرار می‌گیرد، پس این مدل تاکید بر قابلیت استفاده مجدد از محصولات تولید شده قبلی را دارد.

نمونه‌سازی بر دو شکل نمونه‌سازی دور انداختنی و نمونه‌سازی تکاملی وجود دارد.

علاوه بر روش‌های مشاهده، مصاحبه و گفتگو، مکانیزم نمونه‌سازی دورانداختنی نیز برای شناسایی نیازمندی‌های مشتری در فعالیت ارتباطات مورد استفاده قرار می‌گیرد. در مکانیزم نمونه‌سازی دورانداختنی، یک پیاده‌سازی عملیاتی از سیستم با ابزارهای ارزان‌قیمت، فقط و فقط به منظور شناسایی نیازمندی‌های مشتری، ایجاد و سپس دور انداخته می‌شود، سپس سیستم نهایی براساس فرآیند تولید مجزایی تولید می‌گردد، توجه کنید که سیستم نهایی براساس فرآیند تولید مجزای دیگری تولید می‌گردد، مثلاً پس از شناسایی تمامی نیازمندی‌های مشتری توسط مکانیزم نمونه‌سازی دورانداختنی، لیست تمامی نیازمندی‌های مشتری آماده است، بنابراین در ادامه برای تولید نرم‌افزار از مدل آبشاری می‌توان استفاده نمود.

هدف از مکانیزم دورانداختنی، استخراج نیازمندی‌های مشتری است. شروع مکانیزم نمونه‌سازی دورانداختنی براساس نیازمندی‌هایی است که کمتر درک شده‌اند. یکی از مزایای این روش، کاهش ریسک نیازمندی‌ها است. مکانیزم نمونه‌سازی دورانداختنی صرفا یک ابزار جهت تهیه لیست نیازمندی‌های مشتری است و نمونه‌های ساخته شده دورانداخته می‌شوند، پس این

مکانیزم تاکید بر قابلیت استفاده مجدد از محصولات تولید شده قبلی را ندارد.

به مکانیزم‌نمونه‌سازی دورانداختنی، **الگوسازی بسته** نیز گفته می‌شود.

اما در نمونه‌سازی تکاملی، یک نمونه‌ی اولیه تولید می‌شود. در ادامه با اعمال اصلاحات بر روی نمونه‌ی اولیه، طی چند مرحله سیستم نهایی تولید می‌گردد. هدف از نمونه‌سازی تکاملی، تحویل یک سیستم عملیاتی به کاربران نهایی و شروع فرآیند تولید براساس نیازمندی‌هایی است که بهتر و بیشتر درک شده‌اند. کاربرد آن در مواردی است که نیازمندی‌های مشتری به طور کامل در ابتدای کار مشخص نباشد و یا نیاز به توسعه‌ی مبتنی بر تکرار و تکامل داشته باشیم. از مزایای نمونه‌سازی تکاملی می‌توان به درگیرکردن مشتری با فرآیند تولید سیستم که منجر به شناسایی بهتر نیازمندی‌ها می‌گردد، اشاره نمود. مکانیزم نمونه‌سازی دورانداختنی در تولید تکاملی نرم‌افزار مورد استفاده قرار می‌گیرد و نمونه‌های ساخته شده دورانداخته نمی‌شوند، در واقع محصولات اولیه، محصولاتی اولیه از محصولات نهایی هستند، اما قابلیت ارائه خدمات به کاربر را دارند. پس این مکانیزم تاکید بر قابلیت استفاده مجدد از محصولات تولید شده قبلی را دارد. بنابراین گزینه چهارم نیز می‌تواند به واسطه وجود مکانیزم نمونه‌سازی تکاملی، بر قابلیت استفاده مجدد محصولات تولید شده قبلی تاکید کند.

به مکانیزم نمونه‌سازی تکاملی، **الگوسازی باز** نیز گفته می‌شود.

۲۱- گزینه (۲) صحیح است.

مدل روش‌های رسمی (فرمال، قراردادی و صوری) شامل مجموعه‌ای از فعالیت‌ها است که سعی دارد پروژه‌ی نرم‌افزاری را در قالب روابطی رسمی و ریاضی، سیستم‌های کامپیوتری را تعریف، توسعه، پیاده‌سازی و ارزیابی نماید. در این مدل، با استفاده از تحلیل‌های ریاضی، بسیاری از ابهامات، نواقص و عدم سازگاری نرم‌افزار را تا حد زیادی می‌توان به سادگی کشف و تصحیح نمود. گونه‌ای از روش‌های رسمی وجود دارد که به مهندسی نرم‌افزار **اتاق تمیز** (Clean Room) معروف است.

این مدل، امکان کشف و تصحیح خطاهای زیادی را که تا زمان اجرا غیرقابل تشخیص هستند را در طول مراحل اولیه‌ی تولید نرم‌افزار، فراهم می‌کند.

یکی از مهمترین کاربردهای مدل روش‌های رسمی، ساخت سیستم‌های حساس و امن مانند ساخت نرم‌افزارهای کنترل هواپیما و موشک است.

۲۲- گزینه (۱) صحیح است.

به طور کلی انواع نیازمندی‌های نرم‌افزار را می‌توانیم به دو دسته زیر تقسیم کنیم:

۱- وظیفه‌مندی (کارکردی، عملکردی)

نیازمندی‌های وظیفه‌مندی، کمّی و قابل اندازه‌گیری هستند و در قالب قابلیت‌ها، کارکردها،

ویژگی‌ها و سرویس‌های سیستم در حال توسعه یا تولید محقق می‌گردند. به عبارت دیگر، نیازمندی‌های کارکردی به بیان سرویس‌هایی که سیستم باید فراهم نماید، می‌پردازد. چگونگی واکنش سیستم در برابر ورودی‌های خاص و چگونگی رفتار سیستم در شرایط خاص نیز توسط این نیازمندی‌ها تعریف می‌شود.

مانند نیازمندی‌های مربوط به محاسبه‌ی فاکتوریل یک عدد و یا اجرای تابع فیبوناچی در یک نرم‌افزار و یا نیازمندی‌های مربوط به یک نرم‌افزار حقوق و دستمزد.

۲- غیروظیفه‌مندی (غیرکارکردی، غیرعملکردی)

نیازمندی‌های غیروظیفه‌مندی، نیازمندی‌های کیفی و نه الزاماً قابل اندازه‌گیری هستند که به بیان کیفیت مورد انتظار از نیازمندی‌های وظیفه‌مندی و همچنین محدودیت‌هایی نظیر محدودیت‌های زمانی، مالی و استانداردها می‌پردازند. برخی از انواع نیازمندی‌های غیروظیفه‌مندی عبارتند از:

قابلیت استفاده، سهولت یادگیری، قابلیت اعتماد، کارایی، زمان پاسخ، قابلیت پشتیبانی، قابلیت نگهداری، قابلیت حمل، بهره‌وری، محدودیت‌های طراحی، پیاده‌سازی و فیزیکی و همچنین نیازمندی‌های واسط کاربردی.

مانند محاسبه تابع فاکتوریل توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی $O(n)$ و مصرف زیاد حافظه به دلیل استفاده از استک در پی هر فراخوانی.

و یا مانند محاسبه تابع فیبوناچی توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی نمایی زیاد $O(۲^{\frac{n}{۲}})$ و چاق و به تبع، کند و هم مصرف زیاد حافظه به دلیل استفاده از استک، در پی هر فراخوانی.

انتخاب نوع الگوریتم براساس شرایط، حائز اهمیت می‌باشد.

محصول نرم‌افزاری که نیازمندی‌های وظیفه‌مندی را برآورده می‌کند، ولی برآورنده نیازهای غیروظیفه‌مندی و کیفی نباشد، معمولاً با نارضایتی مشتریان همراه می‌شود.

در فعالیت تست، نیازمندی‌های وظیفه‌مندی و غیروظیفه‌مندی جهت اطمینان از صحت عملکرد آنها مورد ارزیابی مجدد قرار می‌گیرند.

۲۳- گزینه (۱) صحیح است.

مدل‌های Sequential (ترتیبی) یا Linear (خطی) یا Waterfall (آبشاری) جزو مدل‌های غیرتکاملی و مدل Spiral (پیچشی یا حلزونی) جزو مدل‌های تکاملی سنتی فرآیند تولید نرم‌افزار هستند.

Prototyping (نمونه‌سازی دورانداختنی) یا Rapid Prototyping (نمونه‌سازی سریع) یک مکانیزم، صرفاً جهت شناسایی نیازمندی‌های مشتری است، در یک بیان کلی‌تر نمونه‌سازی دورانداختنی، گفتگو، مشاهده و مصاحبه روش‌هایی جهت شناسایی نیازمندی‌های مشتری و تهیه

لیست نیازمندی‌ها می‌باشند. توجه کنید که نمونه‌سازی دورانداختنی یک مدل فرآیند تولید نرم‌افزار به شمار نمی‌آید، بلکه یک مکانیزم صرفاً جهت شناسایی نیازمندی‌های مشتری می‌باشد. به حاصل جمع مکانیزم، نمونه‌سازی دورانداختنی و مدل آبشاری «مدل نمونه‌سازی» گفته می‌شود که این مدل نیز یک مدل غیرتکاملی به حساب می‌آید. در واقع مدل نمونه‌سازی از ترکیب مکانیزم نمونه سازی دورانداختنی و مدل آبشاری ایجاد شده است. به این نحو که در ابتدای کار توسط مکانیزم نمونه‌سازی دورانداختنی، تمامی لیست نیازمندی‌های مشتری تکمیل می‌گردد، سپس توسط مدل آبشاری به شکل خطی نرم‌افزار ایجاد می‌گردد و کار تمام می‌شود. در واقع نیازمندهای مشتری طی تکرارهای مکانیزم نمونه‌سازی دورانداختنی شناخته می‌شوند. سپس طی یک روال خطی، توسط مدل آبشاری، نرم‌افزار تولید می‌گردد و کار تمام می‌شود و دیگر تکامل نمی‌یابد. زیرا مدل نمونه‌سازی یک مدل غیرتکاملی است و پس از ساخت نهایی امکان ادامه‌ی پروژه و تغییر وجود ندارد.

از آنجا که سیستم جدید، در ابعاد کوچکتر قبلاً ایجاد شده است و تفاوت‌های زیادی با سیستم سابق ندارد، لذا مسائل ناشناخته زیادی وجود ندارد. بنابراین مدل آبشاری برای توسعه سیستم جدید مناسب است. گزینه‌های دیگر برحسب کاربردشان زمانی مورد استفاده قرار می‌گیرند که نیازمندی‌های ناشناخته‌ای موجود باشد. بنابراین گزینه اول درست و گزینه‌های دوم، سوم و چهارم نادرست هستند.

۲۴- گزینه (۱) صحیح است.

مدل RAD، شکل پُرسرعت مدل آبشاری می‌باشد، با این تفاوت که پروژه به بخش‌های مختلف تقسیم شده و هر بخش، توسط یک تیم، مطابق مدل آبشاری ایجاد می‌گردد و در پایان نتیجه‌ی تیم‌ها، برای خلق محصول نهایی ترکیب می‌گردد. مدل RAD، سرعت خود را مدیون بهره‌گیری از تکنیک بخش‌بندی و موازی‌سازی بخش‌های مختلف پروژه است. چنانچه نیازمندی‌ها به خوبی شناسایی شده و دامنه پروژه کوچک باشد این مدل قادر است یک سیستم کاملاً عملیاتی را در مدت زمان بسیار کوتاه (مثلاً بین ۶۰ تا ۹۰ روز) تولید نماید. در این مدل نرم‌افزار به قسمت‌های مختلف تقسیم شده و همواره سعی می‌شود که نرم‌افزار موردنظر سریع‌تر تولید شود. نکته قابل توجه این است که نرم‌افزار موردنظر باید خاصیت تفکیک‌پذیری داشته باشد تا بتوان این مدل را پیاده‌سازی کرد.

از آنجا که در مورد «الف» مشتری تأکید بر انجام سریع پروژه حداکثر در مدت سه ماه می‌نماید و همچنین تغییرات در درخواست مشتری اندک بوده و امکان تقسیم پروژه به قسمت‌های مستقل وجود دارد. بنابراین در چنین شرایطی مدل RAD مناسب است.

از آنجا که در مورد «ب» پروژه دستخوش تغییر در نیازمندی‌ها می‌باشد بنابراین یکی از مدل‌های تکاملی فرآیند نرم‌افزار همچون مدل افزایشی (Incremental) یا حلزونی (Spiral)

می‌تواند برای این پروژه مناسب باشد. در مدل‌های تکاملی برای ایجاد سیستم‌های بسیار ایمن و قابل اطمینان، مدل حلزونی (پیچشی) می‌تواند بهترین انتخاب باشد زیرا این مدل در هر تکرار از توسعه نرم‌افزار، فعالیت کامل تحلیل ریسک را انجام می‌دهد. ضمن این که برای مورد «ب» بین مدل‌های افزایشی و حلزونی مدل افزایشی مناسب‌تر بود زیرا بحثی از امنیت مطرح نشده است.

۲۵- گزینه (۱) صحیح است.

مدل روش‌های رسمی (فرمال، قراردادی و صوری) شامل مجموعه‌ای از فعالیت‌ها است که سعی دارد پروژه‌ی نرم‌افزاری را در قالب روابطی رسمی و ریاضی، سیستم‌های کامپیوتری را تعریف، توسعه، پیاده‌سازی و ارزیابی نماید. در این مدل، با استفاده از تحلیل‌های ریاضی، بسیاری از ابهامات، نواقص و عدم سازگاری نرم‌افزار را تا حد زیادی می‌توان به سادگی کشف و تصحیح نمود. گونه‌ای از روش‌های رسمی وجود دارد که به مهندسی نرم‌افزار **اتاق تمیز** معروف است.

این مدل، امکان کشف و تصحیح خطاهای زیادی را که تا زمان اجرا غیرقابل تشخیص هستند را در طول مراحل اولیه‌ی تولید نرم‌افزار، فراهم می‌کند.

یکی از مهمترین کاربردهای مدل روش‌های رسمی، ساخت سیستم‌های حساس و امن مانند ساخت نرم‌افزارهای کنترل هواپیما و موشک است.

استفاده از مدل روش‌های رسمی بسیار وقتگیر و گران است.

آموزش گسترده‌ی سازندگان نرم‌افزار به علت اینکه تعداد اندکی از تولیدکنندگان نرم‌افزار پیش زمینه‌ی لازم برای به کاربردن روش‌های فرمال (رسمی) را دارند.

از این مدل نمی‌توان برای برقراری ارتباط با مشتریان معمولی که دید فنی ندارند، استفاده نمود.

۲۶- گزینه (۱) صحیح است.

در فعالیت ارتباط لیست نیازمندی‌های مشتری از طریق ارتباط با مشتری و ابزارهایی همچون گفتگو، مشاهده، پرسش‌نامه، نمونه‌سازی دورانداختنی و نمونه‌سازی تکاملی، جمع‌آوری و آماده می‌گردد.

تحلیل ریسک در مدل پیچشی (حلزونی) به فرآیند تولید نرم‌افزار اضافه شده‌است. گام اول مدیریت ریسک، شناسایی ریسک (تحلیل ریسک) است و گام دوم مقابله با ریسک، برای مقابله با ریسک باید هزینه کرد. برای‌مثال ریسک از دست دادن اطلاعات را می‌توان با هزینه و خریداری یک سیستم پشتیبان‌گیری از اطلاعات مرتفع نمود یا ریسک از دست دادن مدیران را می‌توان با هزینه و استخدام یک نیروی انسانی پشتیبان در کنار مدیران دارای درجه اهمیت بالا مرتفع نمود. در واقع در این فعالیت، ریسک‌های محتملی که ممکن است بر روی خروجی‌های پروژه و یا کیفیت محصول نهایی پروژه یا به عبارت دقیق‌تر بر روی خصوصیات پرژوه‌های موفق نرم‌افزاری (نیاز، زمان و هزینه) تأثیر ناگواری بگذارند شناسایی، مدیریت و در صورت امکان تقلیل می‌یابند.

دقت‌کنید که احتمال وقوع ریسک، تابعی از زمان است، بنابراین مدیریت ریسک می‌بایست در سراسر چرخه‌ی حیات نرم‌افزار، حضوری فعال و پررنگ داشته باشد. بنابراین گزینه اول درست و گزینه‌های دوم، سوم و چهارم نادرست هستند.

۲۷- گزینه (۴) صحیح است.

مدل‌های RAD و Sequential (ترتیبی) یا Linear (خطی) یا Waterfall (آبشاری) جزو مدل‌های غیرتکاملی و مدل افزایشی (Incremental Model) جزو مدل‌های تکاملی سنتی فرآیند تولید نرم‌افزار هستند.

Prototyping (نمونه‌سازی دورانداختنی) یا Rapid Prototyping (نمونه‌سازی سریع) یک مکانیزم، صرفاً جهت شناسایی نیازمندی‌های مشتری است، در یک بیان کلی‌تر نمونه‌سازی دورانداختنی، گفتگو، مشاهده و مصاحبه روش‌هایی جهت شناسایی نیازمندی‌های مشتری و تهیه لیست نیازمندی‌ها می‌باشند. توجه کنید که نمونه‌سازی دورانداختنی یک مدل فرآیند تولید نرم‌افزار به شمار نمی‌آید، بلکه یک مکانیزم صرفاً جهت شناسایی نیازمندی‌های مشتری می‌باشد. به حاصل جمع مکانیزم، نمونه‌سازی دورانداختنی و مدل آبشاری «مدل نمونه‌سازی» گفته می‌شود که این مدل نیز یک مدل غیرتکاملی به حساب می‌آید. در واقع مدل نمونه‌سازی از ترکیب مکانیزم نمونه‌سازی دورانداختنی و مدل آبشاری ایجاد شده است. به این نحو که در ابتدای کار توسط مکانیزم نمونه‌سازی دورانداختنی، تمامی لیست نیازمندی‌های مشتری تکمیل می‌گردد، سپس توسط مدل آبشاری به شکل خطی نرم‌افزار ایجاد می‌گردد و کار تمام می‌شود. در واقع نیازمندهای مشتری طی تکرارهای مکانیزم نمونه‌سازی دورانداختنی شناخته می‌شوند. سپس طی یک روال خطی، توسط مدل آبشاری، نرم‌افزار تولید می‌گردد و کار تمام می‌شود و دیگر تکامل نمی‌یابد. زیرا مدل نمونه‌سازی یک مدل غیرتکاملی است و پس از ساخت نهایی امکان ادامه‌ی پروژه و تغییر وجود ندارد.

پروژه‌ای که از تکنولوژی mobile برای اطلاع‌رسانی در مورد خاصی استفاده می‌کند، در ابتدای کار لیست نیازمندی‌های مشخصی ندارد، همچنین تکنولوژی mobile مدام در حال تغییر و تحول می‌باشد، بنابراین پس از ساخت پروژه هم امکان تغییر و تحول وجود دارد.

در چنین شرایطی که نیازها در ابتدای کار مشخص نیستند، و پس از تحویل محصول باز هم امکان تغییر و تحول در محصول وجود دارد، مناسب‌ترین مدل فرآیند تولید نرم‌افزار، مدل‌های تکاملی سنتی (افزایشی و پیچشی)، مدل تکاملی سنتی خاص یعنی مبتنی بر مؤلفه ساخت‌یافته (مبتنی بر تابع) و مدل تکاملی مدرن یعنی مبتنی بر مؤلفه‌ی شیءگرا (مبتنی بر کلاس) هستند.

از آنجا که در صورت سوال، از وجود قطعات و مولفه‌های آماده و قابل استفاده مجدد صحبت نشده است، مدل‌های مبتنی بر مولفه ساخت‌یافته و مبتنی بر مولفه شی گرا را کنار می‌گذاریم،

همچنین مدل‌های غیرتکاملی RAD و Sequential را نیز با توجه به شرایط پروژه کنار

می‌گذاریم.

بنابراین با توجه به گزینه‌ها پرواضح است که مدل افزایشی پاسخ درست خواهد بود.

مدل افزایشی، مراحل مدل آبشاری را با رویکرد تکرار و تکامل مکانیزم نمونه‌سازی تکاملی ترکیب نموده است. در این مدل، با توجه به خاصیت نمونه‌سازی تکاملی، پروژه به تدریج کامل می‌شود یعنی هر مرحله‌ای که می‌گذرد، پروژه کامل‌تر شده و در افزایش‌های بعدی این تکامل ادامه می‌یابد تا به محصول نهایی برسد و این تکامل نیز ادامه خواهد داشت.

توجه کنید که می‌توان از مکانیزم نمونه‌سازی دورانداختنی در ابتدای هر تکرار (افزایش) در مدل افزایشی به جهت کاهش ریسک مربوط به شناسایی دقیق لیست نیازمندی‌های مشتری استفاده نمود. این قضیه برای مدل‌های پیچشی و مبتنی بر مؤلفه ساخت‌یافته و شیءگرا نیز صادق است.

به طور کلی نمونه‌سازی سریع، به عنوان راه‌کاری برای شناسایی لیست نیازمندی‌های مشتری می‌باشد و در اغلب مدل‌ها می‌توان آن را گنجاند به غیر از مدل آبشاری و مدل RAD .

از آنجا که مدل RAD یک مدل غیرتکاملی است، پرواضح است که مدل مناسبی در این پروژه نخواهد بود، اما به غیر از این مطلب شرط لازم برای استفاده از مدل RAD تقسیم‌پذیر بودن پروژه است که از این مسأله هم صحبتی نشده است.

۲۸- گزینه (۱) صحیح است.

مدل روش‌های رسمی (فرمال، قراردادی و صوری) شامل مجموعه‌ای از فعالیت‌ها است که سعی دارد پروژه‌ی نرم‌افزاری را در قالب روابط رسمی و ریاضی، سیستم‌های کامپیوتری را تعریف، تولید، پیاده‌سازی و ارزیابی نماید. در این مدل، با استفاده از تحلیل‌های ریاضی، بسیاری از ابهامات، نواقص و عدم سازگاری نرم‌افزار را تا حد زیادی می‌توان به سادگی کشف و تصحیح نمود. بنابراین گزینه اول درست است.

گزینه دوم نادرست است. زیرا، RUP یک نمونه متدولوژی شیءگرا براساس مدل فرآیند مبتنی بر مؤلفه شیءگرا، روش شیءگرا (با استفاده از مفاهیم کلاس، وراثت و چندریختی) و ابزار UML است. همچنین OOA (Object Oriented Analysis) به معنی تحلیل شیءگرا، یک فعالیت تحلیل شیءگرا از فعالیت‌های فرآیند تولید نرم‌افزار (ارتباط، برنامه‌ریزی، تحلیل، طراحی، پیاده‌سازی، تست و استقرار) است.

گزینه سوم نیز نادرست است. زیرا، مدل توسعه مبتنی بر مؤلفه نیازمند وجود قطعات آماده در حد قابل قبولی می‌باشد و این پیش شرط، با شرایط مسأله سازگاری ندارد.

گزینه چهارم نادرست است. زیرا، مدل توسعه همروند، برای مدیریت وضعیت فعالیت‌های مختلف چندین پروژه‌ی مختلف در حال انجام در یک سازمان نرم‌افزاری است. اغلب سازمان‌های نرم‌افزاری، در یک بازه‌ی زمانی، احتمالاً چندین پروژه را در دست تولید دارند. به عنوان مثال ممکن است، پنج پروژه از پروژه‌های سازمان در مرحله‌ی ایجاد باشند، اما احتمالاً وضعیت آن‌ها با

یکدیگر متفاوت است (مثلاً یکی در مرحله‌ی تولید کد است در حالی که دیگری در حال تست قرار دارد). مدل توسعه‌ی همروند که با نام مهندسی هم‌روند نیز شناخته می‌شود، جهت کنترل اجرای چندین پروژه‌ی همزمان مورد استفاده قرار می‌گیرد که هر یک از فعالیت‌های چارچوبی فرآیند تولید نرم‌افزار مربوط به هر پروژه می‌تواند در وضعیت‌های مختلفی قرار داشته باشند. وضعیت‌های مختلف مربوط به هر یک از فعالیت‌های چارچوبی توسط یک گراف نشان داده می‌شود.

۲۹- گزینه (۴) صحیح است.

به طور کلی فعالیت‌های مرتبط با فرآیند تولید نرم‌افزار صرف نظر از اندازه، پیچیدگی پروژه و زمینه‌ی کاربردی آن و مستقل از متدولوژی (مهندسی نرم‌افزار) ساخت‌یافته وشی‌گرا به پنج فعالیت ارتباط، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست) و استقرار تقسیم می‌شود، به بیان دیگر فعالیت‌ها در هر دو دسته‌ی متدولوژی ساخت‌یافته و شی‌گرا همین‌ها خواهند بود، اما کارهایی که در هر فعالیت در متدولوژی ساخت‌یافته و شی‌گرا انجام می‌شود شباهت‌ها و تفاوت‌هایی خواهد داشت.

فعالیت ارتباط (مهندسی نیازمندی‌ها)

در این مرحله لیست نیازمندی‌های مشتری از طریق ارتباط با مشتری و استفاده از ابزارها و مکانیسم‌هایی همچون گفتگو، مشاهده، مصاحبه، پرسش‌نامه، بازدید، نمونه‌سازی دوراندااختنی و نمونه‌سازی تکاملی، جمع‌آوری و آماده می‌گردد. این ابزارها و مکانیسم‌ها کمک می‌کنند، تا مشتری خواسته‌ها و نیازهای خود را دقیق‌تر بیان کند و سازنده نیز دقیق‌تر نیازهای مشتری را بشناسد. به این ارتباطات میان مشتری و سازنده برای شناخت نیازها، تحلیل نیازها نیز گفته می‌شود که منجر به شناخت نیازهایی بدون ابهام در مشخصات راه‌حل، سنجش اعتبار، اعتبارسنجی و معتبر بودن مشخصات نیازها می‌گردد.

مهم‌ترین کار در فعالیت ارتباطات، مدیریت شناسایی نیازمندی‌ها و پیگیری تغییرات نیازمندی‌های مشتری است.

به طور کلی انواع نیازمندی‌های نرم‌افزار را می‌توانیم به دو دسته زیر تقسیم کنیم:

۱- وظیفه‌مندی (کارکردی، عملکردی)

نیازمندی‌های وظیفه‌مندی، کمّی و قابل اندازه‌گیری هستند و در قالب قابلیت‌ها، کارکردها، ویژگی‌ها و سرویس‌های سیستم در حال توسعه یا تولید محقق می‌گردند. به عبارت دیگر، نیازمندی‌های کارکردی به بیان سرویس‌هایی که سیستم باید فراهم نماید، می‌پردازد. چگونگی واکنش سیستم در برابر ورودی‌های خاص و چگونگی رفتار سیستم در شرایط خاص نیز توسط این نیازمندی‌ها تعریف می‌شود.

مانند نیازمندی‌های مربوط به محاسبه‌ی فاکتوریل یک عدد و یا اجرای تابع فیبوناچی در یک

نرم‌افزار و یا نیازمندی‌های مربوط به یک نرم‌افزار حقوق و دستمزد.

۲- غیروظیفه‌مندی (غیرکارکردی، غیر عملکردی)

نیازمندی‌های غیروظیفه‌مندی، نیازمندی‌های کیفی و نه الزاماً قابل اندازه‌گیری هستند که به بیان کیفیت مورد انتظار از نیازمندی‌های وظیفه‌مندی و همچنین محدودیت‌هایی نظیر محدودیت‌های زمانی، مالی و استانداردها می‌پردازند. برخی از انواع نیازمندی‌های غیروظیفه‌مندی عبارتند از:

قابلیت استفاده، سهولت یادگیری، قابلیت اعتماد، کارایی، زمان پاسخ، قابلیت پشتیبانی، قابلیت نگهداری، قابلیت حمل، بهره‌وری، محدودیت‌های طراحی، پیاده‌سازی و فیزیکی و همچنین نیازمندی‌های واسط کاربردی.

مانند محاسبه تابع فاکتوریل توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی $O(n)$ و مصرف زیاد حافظه به دلیل استفاده از استک در پی هر فراخوانی.

و یا مانند محاسبه تابع فیبوناچی توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی نمایی زیاد $O(۲^{\frac{n}{۲}})$ و چاق و به تبع، کند و هم مصرف زیاد حافظه به دلیل استفاده از استک، در پی هر فراخوانی.

انتخاب نوع الگوریتم براساس شرایط، حائز اهمیت می‌باشد.

محصول نرم‌افزاری که نیازمندی‌های وظیفه‌مندی را برآورده می‌کند، ولی برآورنده نیازهای غیروظیفه‌مندی و کیفی نباشد، معمولاً با نارضایتی مشتریان همراه می‌شود.

در فعالیت تست، نیازمندی‌های وظیفه‌مندی و غیروظیفه‌مندی جهت اطمینان از صحت عملکرد آنها مورد ارزیابی مجدد قرار می‌گیرند.

۳۰- گزینه (۳) صحیح است.

به طور کلی فعالیت‌های مرتبط با فرآیند تولید نرم‌افزار صرف نظر از اندازه، پیچیدگی پروژه و زمینه‌ی کاربردی آن و مستقل از متدولوژی (مهندسی نرم‌افزار) ساخت‌یافته وشی‌گرا به پنج فعالیت ارتباط، برنامه‌ریزی، مدل‌سازی (تحلیل و طراحی)، ساخت (پیاده‌سازی و تست) و استقرار تقسیم می‌شود، به بیان دیگر فعالیت‌ها در هر دو دسته‌ی متدولوژی ساخت‌یافته و شی‌گرا همین‌ها خواهند بود، اما کارهایی که در هر فعالیت در متدولوژی ساخت‌یافته و شی‌گرا انجام می‌شود شباهت‌ها و تفاوت‌هایی خواهد داشت.

فعالیت ارتباط (مهندسی نیازمندی‌ها)

در این مرحله لیست نیازمندی‌های مشتری از طریق ارتباط با مشتری و استفاده از ابزارها و مکانیسم‌هایی همچون گفتگو، مشاهده، مصاحبه، پرسش‌نامه، بازدید، نمونه‌سازی دوراندااختنی و نمونه‌سازی تکاملی، جمع‌آوری و آماده می‌گردد. این ابزارها و مکانیسم‌ها کمک می‌کنند، تا مشتری خواسته‌ها و نیازهای خود را دقیق‌تر بیان کند و سازنده نیز دقیق‌تر نیازهای مشتری را بشناسد. به

این ارتباطات میان مشتری و سازنده برای شناخت نیازها، تحلیل نیازها نیز گفته می‌شود که منجر به شناخت نیازهایی بدون ابهام در مشخصات راه‌حل، سنجش اعتبار، اعتبارسنجی و معتبر بودن مشخصات نیازها می‌گردد.

مهم‌ترین کار در فعالیت ارتباطات، مدیریت شناسایی نیازمندی‌ها و پیگیری تغییرات نیازمندی‌های مشتری است.

به طور کلی انواع نیازمندی‌های نرم‌افزار را می‌توانیم به دو دسته زیر تقسیم کنیم:

۱- وظیفه‌مندی (کارکردی، عملکردی)

نیازمندی‌های وظیفه‌مندی، کمّی و قابل اندازه‌گیری هستند و در قالب قابلیت‌ها، کارکردها، ویژگی‌ها و سرویس‌های سیستم در حال توسعه یا تولید محقق می‌گردند. به عبارت دیگر، نیازمندهای کارکردی به بیان سرویس‌هایی که سیستم باید فراهم نماید، می‌پردازد. چگونگی واکنش سیستم در برابر ورودی‌های خاص و چگونگی رفتار سیستم در شرایط خاص نیز توسط این نیازمندی‌ها تعریف می‌شود.

مانند نیازمندی‌های مربوط به محاسبه‌ی فاکتوریل یک عدد و یا اجرای تابع فیبوناچی در یک نرم‌افزار و یا نیازمندی‌های مربوط به یک نرم‌افزار حقوق و دستمزد.

۲- غیر وظیفه‌مندی (غیر کارکردی، غیر عملکردی)

نیازمندی‌های غیروظیفه‌مندی، نیازمندی‌های کیفی و نه الزاماً قابل اندازه‌گیری هستند که به بیان کیفیت مورد انتظار از نیازمندی‌های وظیفه‌مندی و همچنین محدودیت‌هایی نظیر محدودیت‌های زمانی، مالی و استانداردها می‌پردازند. برخی از انواع نیازمندی‌های غیروظیفه‌مندی عبارتند از:

قابلیت استفاده، سهولت یادگیری، قابلیت اعتماد، کارایی، زمان پاسخ، قابلیت پشتیبانی، قابلیت نگهداری، قابلیت حمل، بهره‌وری، محدودیت‌های طراحی، پیاده‌سازی و فیزیکی و همچنین نیازمندی‌های واسط کاربردی.

مانند محاسبه تابع فاکتوریل توسط الگوریتم حلقه با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی $O(n)$ و مصرف زیاد حافظه به دلیل استفاده از استک در پی هر فراخوانی. و یا مانند محاسبه تابع فیبوناچی توسط الگوریتم حلقه و با مرتبه اجرایی $O(n)$ و یا توسط الگوریتم بازگشتی با مرتبه اجرایی نمایی زیاد $O(۲^{\frac{n}{۲}})$ و چاق و به تبع، کند و هم مصرف زیاد حافظه به دلیل استفاده از استک، در پی هر فراخوانی.

انتخاب نوع الگوریتم بر اساس شرایط،حائز اهمیت می‌باشد.

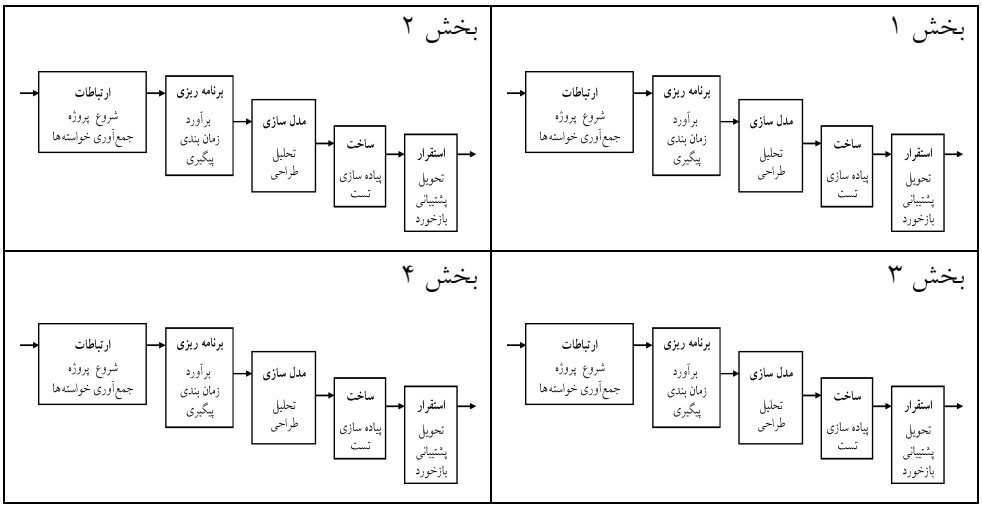
تحلیل نیازها، عدم ابهام در مشخصات راه‌حل و اعتبارسنجی مشخصات نیازمندی‌ها، نتیجه اقداماتی (ابزارهایی) همچون گفتگو، مشاهده، مصاحبه، پرسش‌نامه، بازدید، نمونه‌سازی دورانداختنی و نمونه‌سازی تکاملی است.

ارزیابی خطر (مدیریت ریسک) در فعالیت برنامه‌ریزی، از فعالیت‌هایی چارچوبی انجام

می‌گردد. فعالیت‌های چتری نیز بر فعالیت برنامه‌ریزی و به تبع مدیریت ریسک نظارت دارد.

۳۱- گزینه (۱) صحیح است.

مدل RAD یا Rapid Application Development یا تولید سریع برنامه کاربردی، شکل پُرسرعت مدل آبشاری می‌باشد، با این تفاوت که پروژه به بخش‌های مختلف تقسیم شده و هر بخش، توسط یک تیم، مطابق مدل آبشاری ایجاد می‌گردد و در پایان نتیجه‌ی تیم‌ها، برای خلق محصول نهایی ترکیب می‌گردد. مدل RAD، سرعت خود را مدیون بهره‌گیری از تکنیک بخش‌بندی و موازی‌سازی بخش‌های مختلف پروژه است. چنانچه نیازمندی‌ها به خوبی شناسایی شده و دامنه پروژه کوچک باشد این مدل قادر است یک سیستم کاملاً عملیاتی را در مدت زمان بسیار کوتاه (مثلاً بین ۶۰ تا ۹۰ روز) تولید نماید. در این مدل نرم‌افزار به قسمت‌های مختلف تقسیم شده و همواره سعی می‌شود که نرم‌افزار موردنظر سریع‌تر تولید شود. نکته قابل توجه این است که نرم‌افزار موردنظر باید خاصیت تفکیک‌پذیری داشته باشد تا بتوان این مدل را پیاده‌سازی کرد.



مدل RAD

ویژگی‌های مدل RAD

۱- شرط لازم برای انجام پروژه‌های نرم‌افزاری توسط مدل RAD، قابلیت بخش‌بندی پروژه است.

۲- از آنجا که در مدل RAD، هر بخش از مدل آبشاری استفاده می‌کند، پس بنا بر ویژگی‌های مدل آبشاری، باید تمامی نیازمندی‌های پروژه (لیست نیازمندی) در ابتدای پروژه مشخص باشد. در این صورت این مدل می‌تواند ظرف مدت بسیار کوتاهی (۶۰ تا ۹۰ روز) محصول نهایی را ایجاد نماید.

۳- در پروژه‌های بزرگ، تعداد بخش‌های مختلف پروژه زیاد می‌شود، به همین دلیل به تیم‌های نرم‌افزاری بیشتری نیاز خواهد بود. بنابراین با افزایش حجم پروژه باید نیروی انسانی کافی برای پیمانه‌ها وجود داشته باشد.

۴- این مدل در پروژه‌هایی با ریسک‌های فنی بالا، به دلیل عدم امکان شناسایی نیازمندی‌های مشتری در ابتدای پروژه کارآمد نخواهد بود. مدل RAD در محیط عملیاتی که نیازها به طور کامل واضح، مشخص و ثابت است، کارآمد است، در غیر اینصورت این مدل ناکارآمد خواهد بود.

۵- موفقیت مدل RAD، وابسته به تعامل مناسب سازنده و مشتری است و هر دو باید برای انجام سریع فعالیت‌ها با یکدیگر هماهنگ باشند تا بتوانند در موعد مقرر تولید نهایی را تحویل مشتری دهند. چون اگر یک قسمت از این پروژه انجام نشده باشد. تحویل پروژه میسر نیست، بنابراین مدیریت این مدل اهمیت فراوانی دارد.

صورت سوال به این شکل است:

کدام یک از موارد زیر اگر وجود داشته باشد از روش تولید سریع برنامه کاربردی (Rapid Application Development) نباید استفاده کرد؟

۱) ریسک فنی بالا

گزینه اول پاسخ سوال است، زیرا این مدل در پروژه‌هایی با ریسک‌های فنی بالا، به دلیل عدم امکان شناسایی نیازمندی‌های مشتری در ابتدای پروژه کارآمد نخواهد بود. مدل RAD در محیط عملیاتی که نیازها به طور کامل واضح، مشخص و ثابت است، کارآمد است، در غیر اینصورت این مدل ناکارآمد خواهد بود.

۲) واحدمند (Modular) بودن سیستم

گزینه دوم پاسخ سوال نیست، زیرا شرط لازم برای انجام پروژه‌های نرم‌افزاری توسط مدل RAD، قابلیت بخش‌بندی یا پیمانه‌ای یا واحدمند (Modular) بودن سیستم یا پروژه است.

۳) منابع انسانی کافی برای پروژه‌های بزرگ

گزینه سوم پاسخ سوال نیست، زیرا در پروژه‌های بزرگ، تعداد بخش‌های مختلف پروژه زیاد می‌شود، به همین دلیل به تیم‌های نرم‌افزاری بیشتری نیاز خواهد بود. بنابراین با افزایش حجم پروژه باید نیروی انسانی کافی برای پیمانه‌ها وجود داشته باشد.

۴) تعامل کامل کاربر و تولیدکننده سیستم

گزینه چهارم پاسخ سوال نیست، زیرا موفقیت مدل RAD، وابسته به تعامل مناسب و کامل سازنده (تولید کننده سیستم) و مشتری (کاربر) است و هر دو باید برای انجام سریع فعالیت‌ها با

یکدیگر هماهنگ باشند تا بتوانند در موعد مقرر تولید نهایی را تحویل مشتری دهند. چون اگر یک قسمت از این پروژه انجام نشده باشد. تحویل پروژه میسر نیست، بنابراین مدیریت این مدل اهمیت فراوانی دارد.