موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

مهندسی نرم‌افزار

(حل تشریحی سوالات دولتی ۱۳۹۷)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی IT

براساس کتاب مرجع

راجر اس. پرسمن هشتم ۲۰۱۴

ارسطو خلیلی‌فر

تست‌های فصل دوازدهم: طراحی مبتنی بر الگوها

۱- توصیف «یک قاعده سه-بخشی که رابطه‌ی بین یک حوزه (Context)، یک مساله (Problem) و یک راه حل (Solution) را بیان می‌کند»، مربوط به کدام مورد است؟

(مهندسی فناوری اطلاعات-دولتی ۹۷)

۱) معماری MVC

۲) داستان کاربر (User Story)

۳) الگوی طراحی (Design Pattern)

۴) اصل جایگزینی لیسکوف (Liskov Substitution Principle)

پاسخ تست‌های فصل دوازدهم: طراحی مبتنی بر الگوها

۱- گزینه (۳) صحیح است.

صورت سوال به این شکل است:

توصیف «یک قاعده سه-بخشی که رابطه‌ی بین یک حوزه (Context)، یک مساله (Problem) و یک راه حل (Solution) را بیان می‌کند»، مربوط به کدام مورد است؟

برای هر کدام از ما پیش آمده که در مواجهه با یک مساله طراحی از خود بپرسیم که آیا کسی برای این مساله راهکاری پیدا کرده است؟ پاسخ تقریبا همیشه مثبت است! حل مساله، یافتن راهکار است.

الگوهای طراحی یا Design Patterns به عنوان راهکار در حل مساله، شما را از«اختراع دوباره چرخ» مصون نگاه می‌دارد؛ الگوهای طراحی به عنوان راهکار در حل مساله؛ اگر به طور موثر استفاده شود، همان نتایج موفق را مجددا تکرار می کند.

الگوی غیرمولد(non generative) مساله را تعریف می کند اما راهکار ارائه نمی دهد؛ اما الگوی مولد(generative) مساله را تعریف می کند و راهکار حل مساله را نیز ارائه می دهد.

الگوی طراحی را می‌توان «یک قاعده‌ی سه بخشی دانست که واسط میان یک حیطه یا حوزه معین(Context)، یک مساله(Problem) و یک راهکار(Solution) را بیان می‌کند.»

برای طراحی نرم‌افزار، حیطه به طراح نرم‌افزار این امکان را می‌دهد تا محیطی را که مساله در آن جای دارد، درک کند و دریابد چه راهکاری ممکن است در این محیط مناسب باشد. 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! بنابراین محیط عملیاتی یک مساله، بر ارائه‌ی راهکار برای حل آن مساله تاثیر دارد.

به دو روش می‌شود به نتیجه رسید:

روش آزمون و خطا شده توسط دیگران که ابهام ندارد و روش آزمون و خطا توسط خودمان که ممکن است در مسیر دچار ابهام شویم.

به طور کلی محیط عملیاتی، مجموعه‌ای از خواسته‌ها، محدودیت‌ها و قید و بندها، بر روی درک خود صورت مساله و سپس ارائه راهکار برای حل مساله تاثیر دارد.

دقت کنید که اغلب مسائل چندین راهکار برای حل مساله دارند، ولی تنها یکی از آنهاست که در حیطه‌ی مساله موجود، بسیار مناسب است و می‌تواند موثر و نتیجه‌گرا واقع شود.

همه‌ی نرم‌افزارها به رفاه و کیفیت زندگی انسان کمک می‌کنند، بنابراین بهترین الگوهای طراحی، الگوهایی هستند که بر زیبایی و رفاه بیشتر برای انسان تاکید دارند.
از الگوهای طراحی می‌توان برای حل مسائل مربوط به طراحی داده، طراحی معماری، طراحی مولفه و طراحی واسط استفاده کرد.

تست‌های فصل هفتم: مدل تحلیل و مدل طراحی شیءگرا

۱- کدام نمودار UML، اساسا برای مدل‌سازی ساختار(معماری) محصول نرم‌افزاری به کار می‌رود؟
(مهندسی فناوری اطلاعات-دولتی ۹۷)

۱) نمودار فعالیت (Activity Diagram)
۲) نمودار مولفه (Component Diagram)
۳) نمودار توالی (Sequence Diagram)
۴) نمودار مورد کاربرد (Use-Case Diagram)

پاسخ تست‌های فصل دوازدهم: طراحی مبتنی بر الگوها

۱- گزینه (۲) صحیح است.

صورت سوال به این شکل است:

کدام نمودار UML، برای اساسا برای مدل‌سازی ساختار(معماری) محصول نرم‌افزاری به کار می‌رود؟

۱) نمودار فعالیت (Activity Diagram)

گزینه اول پاسخ سوال نیست، زیرا Activity Diagram یا نمودار فعالیت و Swimlane Diagram یا نمودار خط شنا، جهت مدل‌سازی سناریوی اصلی و فرعی داخل یک use case (نیاز یا مورد کاربرد یا زیرسیستم) مورد استفاده قرار می‌گیرد. به بیان دیگر نمودار فعالیت یا نمودار خط شنا، جهت مدل‌سازی روال انجام کارها داخل یک use case مورد استفاده قرار می گیرد. همچنین، در طراحی مؤلفه، باید جزئیات الگوریتمی متدهای کلاس تشریح شود. برای این منظور Activity Diagram یا نمودار فعالیت و Swimlane Diagram یا نمودار خط‌شنا مورد استفاده قرار می‌گیرد. نمودار فعالیت و نمودار خط‌شنا ساختاری مشابه فلوچارت دارد.

۲) نمودار مولفه (Component Diagram)

گزینه دوم پاسخ سوال است، زیرا Component Diagram یا نمودار مؤلفه جهت مدل‌سازی ساختار کلی پیاده‌سازی برنامه به عبارت دیگر ساختار(معماری) محصول نرم‌افزاری و ترتیب کامپایل مؤلفه‌های فرعی، برای ایجاد مؤلفه‌اصلی (مؤلفه کلی یا برنامه اصلی) مورد استفاده قرار می گیرد. به بیان دیگر نمودار مؤلفه یک دید فیزیکی از مدل سیستم به همراه مؤلفه‌های فرعی نرم‌افزار و روابط بین آنها را نشان می‌دهد.

توجه: دقت کنید که در صورت سوال به ساختار(معماری) یک «**محصول نرم افزاری**» تاکید شده است، یعنی فعالیت پیاده‌سازی.

۳) نمودار توالی (Sequence Diagram)

گزینه سوم پاسخ سوال نیست. زیرا Sequence Diagram یا نمودار توالی، Object Diagram یا نمودار شیء جهت مدل‌سازی تعاملات پویای میان اشیاء همکار داخل یک use case مورد استفاده قرار می‌گیرد.

۴) نمودار مورد کاربرد (Use-Case Diagram)

گزینه چهارم پاسخ سوال نیست. زیرا Use Case Diagram یا نمودار مورد کاربرد، Requirement Diagram یا نمودار نیاز جهت مدل‌سازی لیست نیازمندی‌های مشتری مورد استفاده قرار می‌گیرد.

تست‌های فصل یازدهم: متدولوژی‌های چابک

۱- کدام محصول، به طور معمول در روش XP تولید می‌شود؟

(مهندسی فناوری اطلاعات-دولتی ۹۷)

۱) کارت‌های CRC
۲) مدل پیکربندی
۳) نمودارهای حالت
۴) تابلوی Kanban

پاسخ تست‌های فصل یازدهم: متدولوژهای چابک

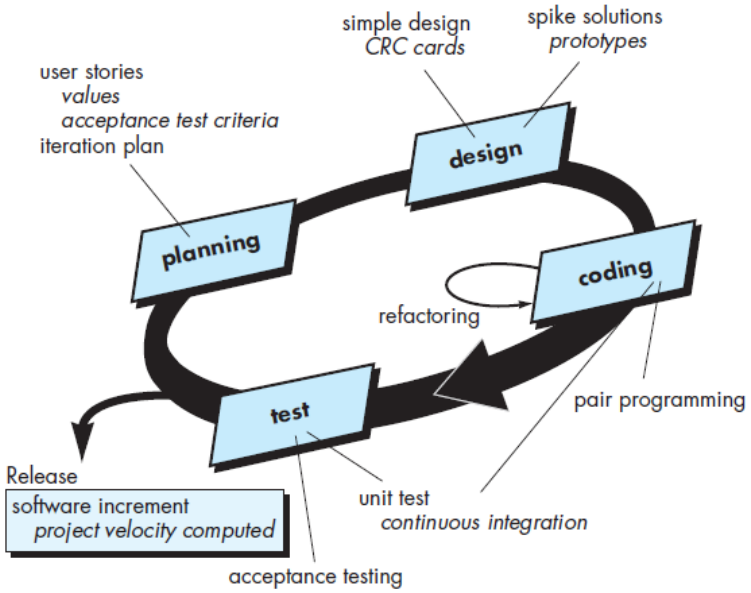
۱- گزینه (۱) صحیح است.

صورت سوال به این شکل است:

کدام محصول، به طور معمول در روش XP تولید می‌شود؟

۱) کارت‌های CRC

گزینه اول پاسخ سوال است، زیرا متدولوژی XP یا متدولوژی برنامه‌نویسی حدی (Extreme Programming) پرکابردترین رویکرد در توسعه نرم‌افزار به شیوه چابک است. این متدولوژی با نام‌های XP و یا X-programming نیز شناخته می‌شود. علت نامگذاری این متدولوژی به برنامه‌نویسی حدی این است که نسبت به متدولوژی‌های دیگر، مرحله‌ی برنامه‌نویسی را با تاکید بیشتری انجام می‌دهد. متدولوژی XP از روش شیءگرا جهت توسعه برنامه استفاده می‌کند. در این متدولوژی فعالیت‌های چارچوبی (framework activities) شامل چهار فعالیت برنامه‌ریزی، طراحی، برنامه‌نویسی و تست می‌باشد. شکل زیر گویای روند کارکرد متدولوژی XP است:



فعالیت طراحی در متدولوژی XP برپایه‌ی سادگی بنا شده است، این اصل که سادگی را حفظ‌کن

(keep it simple). در متدولوژی XP همواره یک طراحی ساده بر یک طراحی پیچیده برتری دارد. در این متدولوژی صرفا طراحی سناریوهایی انجام می‌شود که نیاز وضع موجود است و قرار به پیاده‌سازی قطعی آن است، بنابراین طراحی سناریوهایی که نیاز به آن در آینده محتمل است و قرار به پیاده‌سازی آن قطعی نیست، در این متدولوژی جایگاهی ندارد. متدولوژی XP در جهت حفظ سادگی در فعالیت طراحی خود فقط از کارت‌های CRC استفاده می‌کند. پس از شناسایی موارد کاربرد و سناریونویسی برای هر یک از موارد کاربرد، زمان تعریف کلاس‌ها، صفات، متدها و ارتباطات میان کلاس‌های همکار برای هر یک از موارد کاربرد می‌رسد. برای شناسایی کلاس‌ها، صفات، متدها و ارتباطات میان کلاس‌ها برای هر یک از موارد کاربرد، از تکنیکی موسوم به CRC که سرواژه‌ی عبارت Class – Responsibility Collaborator و به معنی «مدل همکاری مسئولیت های کلاس‌ها» می‌باشد، استفاده می‌شود. مدل‌سازی CRC روشی ساده جهت تعیین و سازماندهی کلاس‌های داخل هر مورد کاربرد است. اگر در بخشی از طراحی یک سناریو، سختی ایجاد شود، متدولوژی XP، ایجاد فوری یک نمونه اولیه عملیاتی (operational prototype) را برای آن بخش از طراحی توصیه می‌کند. استفاده از نمونه اولیه عملیاتی که موسوم به راهکار خیزشی (spike solution) نیز می‌باشد، دو پیامد دارد، اول اینکه، این راهکار منجر به کاهش ریسک فنی در پیاده‌سازی واقعی سناریوی مورد نظر می‌شود و دوم اینکه، این راهکار منجر به اعتبارسنجی برآوردهای اولیه یعنی «برآورد میزان کار»، «برآورد زمان لازم برای انجام کار»، «برآورد هزینه لازم برای انجام کار»، «مدیریت ریسک» و «زمان‌بندی» مربوط به سناریوی مورد نظر می‌شود.

۲) مدل پیکربندی

گزینه دوم پاسخ سوال نیست، زیرا مدل پیکربندی 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.

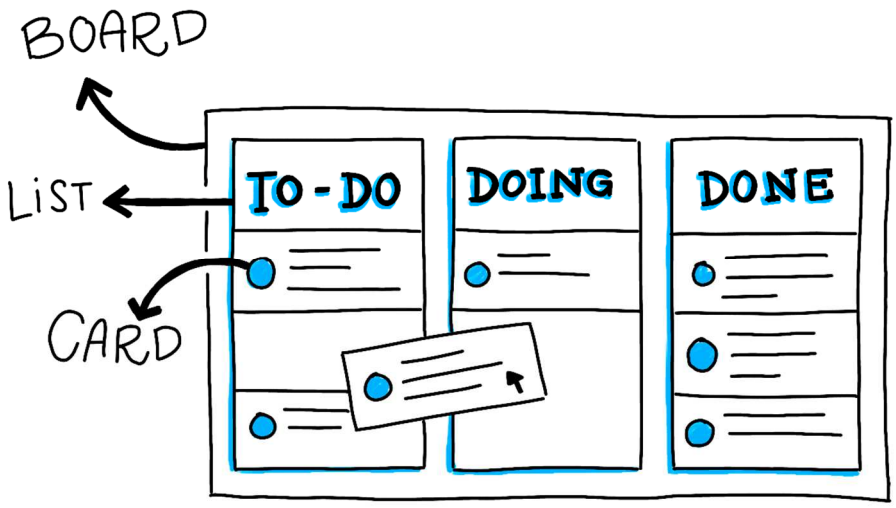
۳) نمودارهای حالت

گزینه سوم پاسخ سوال نیست. زیرا Sequence Diagram یا نمودار توالی، Object Diagram یا نمودار شیء جهت مدل‌سازی تعاملات پویای میان اشیاء همکار داخل یک use case مورد استفاده قرار می‌گیرد.

۴) تابلوی Kanban

گزینه چهارم پاسخ سوال نیست. زیرا تابلوی Kanban در متدولوژی چابک Kanban مورد استفاده قرار می‌گیرد. کانبان (Kanban) واژه‌ای ژاپنی به معنای «نشانه بصری» یا «کارت» است که کمپانی

تویوتا آن را برای نخستین بار وارد فضای مدیریت کرد. تابلوی Kanban به بهبود مهارت مـدیریت زمان کمک می‌کند. در اغلب موارد متدولوژی چابک Scrum با متـدولوژی چابـک Kanban ادغـام می‌شود. شکل زیر گویای روند کارکرد متدولوژی Kanban است:



متدولوژی چابک Kanban چهارچوبی مصور است که برای پیاده‌سازی و اجرای مدیریت پروژه چابک استفاده می‌شود و نشان می‌دهد که چه محصولی، در چه زمانی و به چه مقدار باید تولید گردد. همانطور که گفتیم روش کانبان از سیستم تولید شرکت تویوتا و تولید ناب الهام گرفته شده است. در سال ۱۹۴۰ تویوتا پروسه‌ی مهندسی خود را با مدل‌سازی آن بر اساس چگونگی عملکرد قفسه‌های سوپرمارکت‌ها بهبود بخشید. مهندس تایچی اوهنو (Taiichi Ohno) متوجه این نکته شد که سوپرمارکت‌ها تنها به اندازه‌ای محصول ذخیره می‌کنند که پاسخگوی تقاضای مشتری باشد، که موجب بهینه‌سازی جریان میان سوپرمارکت و مشتری می‌شود. موجودی نیز تنها زمانی ذخیره می شود که فضای خالی روی قفسه‌ها موجود باشد (نشانه بصری). تویوتا همین اصول ساده را به کارخانه خود راه داد. تیم‌های مختلف کارت‌هایی (یا کانبان) می‌ساختند تا اعلام کنند که ظرفیت اضافی دارند و برای دریافت مواد بیشتر آماده‌اند. روش کانبان گاهی سیستم کششی (pull system) نیز نامیده می‌شود چرا که تمام بخش‌های آن از سفارشات دریافت می‌شود.

امروزه همین ایده‌ها و اصول بر روی تیم‌های نرم‌افزاری و پروژه‌های فناوری اطلاعات نیز اعمال می‌شود. در این زمینه، توسعه کار در حال انجام (WIP) جای موجودی را می‌گیرد و کار جدید تنها زمانی اضافه می‌شود که فضای خالی بر روی تخته بصری تیم (team’s visual Kanban board)

وجود داشته باشد. کانبان میزان کار درحال انجام (WIP) را با ظرفیت تیم تطبیق می‌دهد که موجب افزایش انعطاف، شفاف‌سازی و تولید می‌شود. کانبان یک تکنیک برای مدیریت یک پروسه توسعه نرم‌افزاری توسط روشی با اثربخشی بالاست. زیربنای روش کانبان، سیستم تولید همزمان تویوتا (JIT: Just In Time) است. هرچند که توسعه نرم‌افزار یک فعالیت خلاقانه و مبتکرانه و متفاوت از تولید انبوه اتومبیل است؛ اما همچنان مکانیزم پایه برای مدیریت خط تولید می‌تواند بر روی آن پیاده‌سازی شود. تخته کانبان ابزاری برای پیاده‌سازی متدولوژی کانبان در پروژه هاست. اصولا این ابزار یک تخته فیزیکی است که دارای آهن ربا، چسب‌های پلاستیکی یا برگه‌هایی با قابلیت چسبیدن بر روی تخته کانبان است و یا گاهی به صورت دستی آیتم‌های کاری بر روی تخته نوشته می‌شوند. با این حال در سال‌های اخیر نرم‌افزارهای مدیریت پروژه بسیاری این تخته را به صورت آنلاین ارائه داده‌اند.

یک تخته کانبان، چه به صورت فیزیکی و چه آنلاین، از ستون‌ها و سطرهای متفاوتی تشکیل شده است. ساده‌ترین شکل این تخته‌ها شامل سه ستون است: ستون انجام دادن(To Do)، در حال انجام (Doing) و انجام شده (Done). ستون‌های مربوط به یک پروژه توسعه نرم‌افزاری ممکن است شامل ستون‌های بانک اطلاعاتی (Backlog)، آماده انجام(ready)، کدزنی (Coding) ، آزمایش (Testing)، تأیید (approval) و انجام شده (Done) باشد.

کارت‌های کانبان (مانند یادداشت‌های چسبانند) نشان دهنده کارها است و هر کارت بر روی تخته نصب و در ستونی قرار می‌گیرد تا بازگو کننده وضعیت کار مورد نظر باشد. برای مثال کاری که هنوز شروع نشده است و در مرحله پیش از انجام است باید در ستون انجام دادن (To Do) قرار گیرد. این کارت‌ها می‌توانند با یک نگاه وضعیت کارها را به شما بازگو کنند. می‌توان از کارت‌هایی با رنگ‌های مختلف برای نشان دادن جزئیات بیشتر استفاده کرد. برای مثال کارت سبز می‌تواند نشان دهنده یک ویژگی باشد و یا رنگ نارنجی نشان دهنده یک وظیفه یا عملیات.

طبیعت تصویری بودن روش کانبان یک مزیت منحصر به فرد برای چابک‌سازی پروژه می‌دهد. عملکرد تخته کانبان بسیار ساده و سریع بوده و گردش کار را بهبود می‌بخشد و زمان چرخه را کاهش می‌دهد. اسکرام بر پایه ترکیب تکرار دوره‌های ثابت زمانی با برنامه‌ریزی، بهبود فرآیند و انتشار شکل گرفته است. اما در کانبان هیچ چهارچوب زمانی در نظر گرفته نشده است. اسکرام

در مقابل تغییر مقاوم است، در حالی که کانبان به سادگی با هر نوع تغییری مطابقت پیدا می‌کند. در اسکرام زمانی که تیم، داستان‌های کاربری (User Story) را به اسپرینت‌ها (Sprints) متصل می کنند، اضافه کردن داستان کاربری جدید به اسپرینت امکان‌پذیر نخواهید بود. اما در کانبان می‌توان داستان‌های کاربری را در هر زمان اضافه و یا ویرایش کرد.

تست‌های فصل دوم: مدل‌های فرآیند تولید نرم‌افزار

۱- کدام مدل فرآیندی، تکاملی و ریسک-رانه (Risk-Driven) محسوب می‌شود؟

(مهندسی فناوری اطلاعات-دولتی ۹۷)

۱) مدل V

۲) مدل مارپیچی (Spiral)

۳) مدل آبشاری (Waterfall)

۴) مدل افزایشی (Incremental)

پاسخ تست‌های فصل دوم: مدل‌های فرآیند تولید نرم‌افزار

۱- گزینه (۲) صحیح است.

صورت سوال به این شکل است:

کدام مدل فرآیندی، تکاملی و ریسک-رانه (Risk-Driven) محسوب می‌شود؟

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

۱) مدل V

گزینه اول پاسخ سوال نیست، زیرا زیرا مدل V یک مدل غیرتکاملی است.

۲) مدل مارپیچی (Spiral)

گزینه دوم پاسخ سوال است، زیرا رویکرد مدل پیچشی همانند مدل افزایشی است با این تفاوت اساسی که در مدل پیچشی **مدیریت ریسک** (تحلیل ریسک) در فعالیت برنامه‌ریزی به طور جدی و در تمام جوانب پروژه انجام می‌گیرد. اما در مدل افزایشی فقط **ریسک مربوط به شناسایی درست نیازمندی‌ها** در هر تکرار توسط مکانیزم نمونه‌سازی دوراندااختنی و **ریسک تکنیکی** مدیریت می‌گردد.

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**[۱]» 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

[۱] Anchor Point Milestones

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.

۳) مدل آبشاری (Waterfall)

گزینه سوم پاسخ سوال نیست. زیرا مدل آبشاری یک مدل غیرتکاملی است.

۴) مدل افزایشی (Incremental)

گزینه چهارم پاسخ سوال نیست. زیرا 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.

در مدل افزایشی، 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.

تست‌های فصل هشتم: تست و استقرار نرم‌افزار

۱- کدام عبارت در مورد آزمون آلفا (Alpha Testing)، درست نیست؟

(مهندسی فناوری اطلاعات-دولتی ۹۷)

۱) توسط کاربران انجام می‌شود.
۲) در یک محیط کنترل شده انجام می‌شود.
۳) در غیاب ایجادکننده سیستم انجام می‌شود.
۴) برای اعتبارسنجی (Validation) انجام می‌شود.

پاسخ تست‌های فصل هشتم: تست و استقرار نرم‌افزار

۱- گزینه (۳) صحیح است.

صورت سوال به این شکل است:

کدام عبارت در مورد آزمون آلفا (Alpha Testing)، درست نیست؟

به طور کلی مراحل مرتبط با فرآیند تست نرم‌افزار صرف‌نظر از اندازه، پیچیدگی پروژه و زمینه‌ی کاربردی آن و مستقل از متدولوژی ساخت‌یافته و شیءگرا به چهار مرحله تست واحد، تست جامعیت (تست یکپارچگی یا تست تدریجی)، تست اعتبارسنجی و تست سیستم تقسیم می‌شود.

تست اعتبارسنجی(Validation Testing)، در پایان تست جامعیت یا یکپارچگی آغاز می‌شود، یعنی زمانی‌که اجتماع کل مولفه‌های برنامه در کنار یکدیگر قرار گرفتند و نرم‌افزار به طور کامل مونتاژ شد و خطاهای واسط و نقاط اتصال مابین مولفه‌ها کشف و برطرف شدند.

تست اعتبارسنجی، روند کار یا درستی کار را در سطح اجتماع کل مولفه‌های برنامه در کنار یکدیگر در شرایط **سهل و آسان** مورد تست و ارزیابی قرار می‌دهد. تا مشخص شود نرم‌افزاری که ایجاد شده است منطبق بر نیازمندی‌های مشتری است یا خیر. به عبارت دیگر مشخص شود که آیا نرم‌افزار ایجاد شده بر اساس ورودی‌های مورد نظر مشتری **در** شرایط **سهل** و **آسان**، خروجی‌های مورد انتظار مشتری را برآورده می‌سازد یا خیر.

به بیان دیگر تست اعتبارسنجی، اعتبارسنجی عملکرد یا روند اجرای کل مولفه‌های نرم‌افزار ایجاد شده در کنار یکدیگر را در شرایط **سهل و آسان** مورد تست و ارزیابی قرار می‌دهد.

مثال: تست اعتبارسنجی مانند تست قطعات (مولفه‌های) یک خودرو توسط یک خودروساز به شکل کامل در کنار یکدیگر در شرایط سهل و آسان داخل محیط کارخانه خودروساز بدون لحاظ کردن شرایط سخت و دشوار همچون شرایط بارانی و کویری است.

توجه: تست اعتبارسنجی جهت محقق کردن اصل اعتبارسنجی(validation)، روش تست جعبه سیاه و تکنیک‌های مرتبط با آنرا مورد استفاده قرار می‌دهد.

توجه: در این مرحله کلیه‌ی موارد پیاده‌سازی شده، براساس لیست نیازمندی‌های وظیفه‌مندی و غیروظیفه‌مندی مشتری (چک لیست) که در فعالیت ارتباطات تهیه و در مدل تحلیل و طراحی مدل‌سازی شده است مورد وارسی قرار می‌گیرد تا مشخص شود نرم‌افزار ایجاد شده براساس ورودی‌های مورد نظر مشتری، خروجی‌های مورد انتظار مشتری را برآورده می‌سازد یا خیر. در انتهای این تست یک لیست از نیازمندی‌های مشتری که برآورده نشده است تهیه می‌گردد تا رفع

گردند. هنگامی‌که یک **نرم‌افزار سفارشی**(Custom Software) تنها برای یک مشتری توسعه می‌یابد، «**آزمون پذیرش**[۱]» اجرا می‌شود تا مشتری را قادر به اعتبارسنجی کلیه‌ی خواسته‌ها کند. آزمون پذیرش، که به جای مهندس نرم‌افزار، توسط مشتری انجام می‌شود، می‌تواند از یک آزمون غیررسمی تا یک سری آزمون‌های برنامه‌ریزی شده‌ی سیستماتیک را شامل شود. درواقع، آزمون پذیرش را می‌توان در عرض یک هفته یا یک ماه انجام داد و لذا کشف خطاهایی که ممکن است سیستم را با گذشت زمان تنزل دهد، امکان‌پذیر می‌شود. اگر نرم‌افزار به عنوان محصولی توسعه می‌یابد که قرار است مشتریان زیادی از آن استفاده کنند، انجام «آزمون پذیرش» توسط یکایک آنها امکان‌پذیر نیست. اکثر سازندگان محصولات نرم‌افزاری از فرآیندی موسوم به تست آلفا (α) و بتا (β)، برای کشف خطاهایی استفاده می‌کنند که به نظر می‌رسد فقط کاربر نهایی قادر به یافتن آنها می‌باشد.

تست آلفا (α)

تست آلفا در مکان سازنده نرم‌افزار و توسط مشتری انجام می‌شود. بدین صورت که مشتری در یک محیط کنترل‌شده توسط سازنده، کنار سازنده می‌نشیند و نرم‌افزار را تست می‌کند. سازنده نیز همزمان خطاهای موجود را یادداشت می‌کند.

تست بتا (β)

تست بتا در مکان مشتری و توسط یک یا چند کاربر نهایی انجام می‌شود. برخلاف تست آلفا، سازنده معمولاً حضور ندارد و سیستم در محیط واقعی خود مستقر می‌باشد. بنابراین، تست بتا، یک کاربرد «زنده» از نرم‌افزار در محیطی است که سازنده قادر به کنترل آن نیست. مشتری کلیه مشکلات (واقعی یا تصوری) را که طی تست بتا یافته است، یادداشت می‌کند و آنها را در فاصله های زمانی منظم به سازنده گزارش می‌دهد. مهندسان نرم‌افزار، با توجه به مشکلات گزارش شده طی تست بتا، اصلاحات لازم را انجام می‌دهند و سپس محصول نرم‌افزاری نهایی را برای ارایه به مشتریان آماده می‌کنند.

شکل دیگری از تست بتا، که به «**آزمون پذیرش مشتری**[۲]» موسوم است، گاهی اجرا می‌شود، یعنی زمانی‌که یک **نرم‌افزار سفارشی** تحت **قرارداد** به مشتری تحویل داده می‌شود. مشتری یک سری تست‌های مشخص انجام می‌دهد تا خطاها را قبل از پذیرفتن نرم‌افزار از سازنده آن کشف کند.

۱) توسط کاربران انجام می‌شود.

گزینه اول پاسخ سوال نیست، زیرا آزمون آلفا (**Alpha Testing**)، در مکان سازنده نرم‌افزار و

---

[۱] Acceptance Test

[۲] Customer Acceptance Testing

توسط مشتری انجام می‌شود.

۲) در یک محیط کنترل شده انجام می‌شود.

گزینه دوم پاسخ سوال نیست، زیرا آزمون آلفا (**Alpha Testing**)، در مکان سازنده نرم‌افزار و توسط مشتری انجام می‌شود. بدین صورت که **مشتری در یک محیط کنترل‌شده** توسط سازنده، کنار سازنده می‌نشیند و نرم‌افزار را تست می‌کند. سازنده نیز همزمان خطاهای موجود را یادداشت می‌کند.

۳) در غیاب ایجادکننده سیستم انجام می‌شود.

گزینه سوم پاسخ سوال است. زیرا آزمون آلفا (**Alpha Testing**)، در مکان سازنده نرم‌افزار و توسط مشتری انجام می‌شود. بدین صورت که مشتری در یک محیط کنترل‌شده توسط سازنده، **کنار سازنده** می‌نشیند و نرم‌افزار را تست می‌کند. سازنده نیز همزمان خطاهای موجود را یادداشت می‌کند.

۴) برای اعتبارسنجی (Validation) انجام می‌شود.

گزینه چهارم پاسخ سوال نیست. زیرا اکثر سازندگان محصولات نرم‌افزاری از فرآیندی موسوم به تست آلفا (α) و بتا (β)، برای کشف خطاهایی استفاده می‌کنند که به نظر می‌رسد فقط کاربر نهایی قادر به یافتن آنها می‌باشد. تست آلفا (α) و بتا (β) در مرحله تست **اعتبارسنجی (Validation)** انجام می‌شود.

تست‌های فصل دوازدهم: طراحی مبتنی بر الگوها

۱- مدل مسیریابی (Navigation)، در مدل‌سازی کدام‌یک از انواع سیستم‌ها کاربرد عمده دارد؟

(مهندسی فناوری اطلاعات-دولتی ۹۷)

۱) بحرانی

۲) کنترل خطا

۳) کنترل پروژه

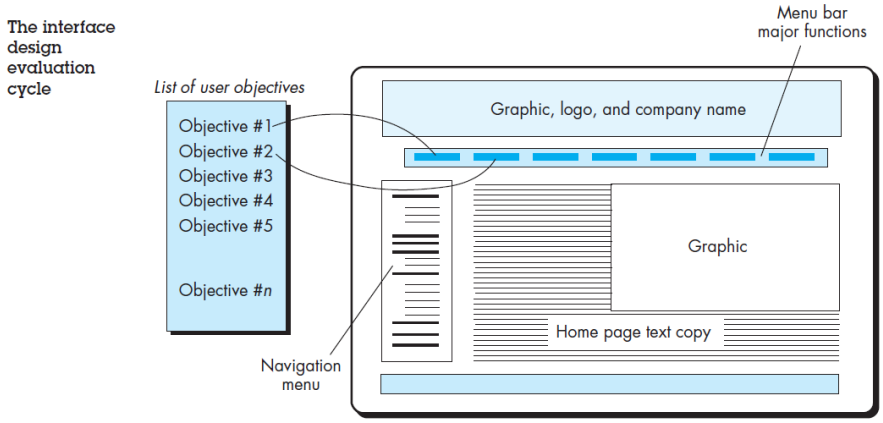
۴) مبتنی بر وب

پاسخ تست‌های فصل دوازدهم: طراحی مبتنی بر الگوها

۱- گزینه (۴) صحیح است.

صورت سوال به این شکل است:

مدل مسیریابی (Navigation)، در مدل‌سازی کدام‌یک از انواع سیستم‌ها کاربرد عمده دارد؟



اگر این احتمال وجود دارد که کاربران در مکان‌ها و سطوح گوناگونی از سلسله مراتب وب سایت و محتوا وارد برنامه‌ی تحت وب شوند، حتما هر صفحه با ویژگی‌های گشت و گذار یا مسیریابی (**Navigation**) باید طوری طراحی شود که کاربر را به سایر نقاط مورد نظر دیگر، هدایت و حمایت کند. بهترین سفر، آن است که با چند گام انجام شود، فاصله‌ی میان کاربر و هدفش باید کوتاه باشد.