سوالات تستی

جواب سوالات تستی

1. B
2. D
3. C
4. D
5. C

سوال ۱

تفاوت اساسی بین نرم‌افزار و سخت‌افزار با توجه به زمان چیست؟

جواب سوال ۱

تغییرپذیری

نرم‌افزارها به راحتی قابل تغییر و به‌روزرسانی هستند، در حالی که سخت‌افزارها برای تغییر نیاز به تعویض فیزیکی یا ارتقا دارند. با گذشت زمان، این انعطاف‌پذیری نرم‌افزار امکان پاسخ‌گویی به نیازهای جدید را فراهم می‌کند.

عمر مفید

سخت‌افزارها دارای یک عمر مفید فیزیکی هستند و با گذشت زمان فرسوده می‌شوند. در مقابل، نرم‌افزار فرسودگی فیزیکی ندارد اما ممکن است به دلیل تغییرات فناوری یا نیازهای کاربری، منسوخ شود.

هزینه‌های به‌روزرسانی و نگهداری

در طول زمان، هزینه‌های نگهداری و به‌روزرسانی نرم‌افزار می‌تواند بسیار بیشتر از هزینه‌های اولیه توسعه آن باشد. در حالی که هزینه‌های سخت‌افزار بیشتر به خرید و نصب اولیه محدود می‌شود.

وابستگی به فناوری

نرم‌افزارها معمولاً به سرعت تحت تأثیر تغییرات فناوری قرار می‌گیرند. اما سخت‌افزار ممکن است برای مدت زمان طولانی‌تری قابل استفاده باقی بماند، حتی اگر فناوری پیشرفت کند.

مقیاس‌پذیری

نرم‌افزارها معمولاً به راحتی قابل مقیاس‌پذیری هستند، به این معنی که می‌توان آن‌ها را برای پاسخگویی به نیازهای در حال تغییر تنظیم کرد. سخت‌افزار اغلب نیاز به ارتقا یا تعویض دارد تا بتواند با نیازهای مقیاس بزرگ‌تر سازگار شود.

سوال ۲

توضیح دهید Prototyping در چه مواقعی می‌تواند در پیش‌برد پروژه موثر باشد؟

جواب سوال ۲

شفاف‌سازی نیازمندی‌ها

Prototyping به شناسایی و تصحیح نیازمندی‌های کاربران و مشتریان کمک می‌کند، به ویژه زمانی که این نیازمندی‌ها به‌طور کامل شناخته نشده باشند.

بازخورد سریع

پروتوتایپ‌ها امکان دریافت بازخورد سریع از کاربران و ذینفعان را فراهم می‌کنند، که این امر به بهبود و تکامل سریع‌تر محصول کمک می‌کند.

کاهش ریسک

ایجاد نمونه‌های اولیه به تیم توسعه کمک می‌کند تا ریسک‌های مربوط به فناوری یا طراحی را در مراحل اولیه پروژه شناسایی و رفع کنند.

ارتباط بهتر با ذینفعان

پروتوتایپ‌ها به تیم توسعه این امکان را می‌دهند تا ایده‌ها و پیشنهادات خود را به شکل عینی به ذینفعان نشان دهند، که این امر می‌تواند به ارتباط و درک متقابل بهتر کمک کند.

انعطاف‌پذیری در توسعه

استفاده از پروتوتایپ‌ها به تیم توسعه اجازه می‌دهد تا انعطاف‌پذیری بیشتری در تغییر جهت یا اصلاح طرح‌ها در طول فرآیند توسعه داشته باشند.

سوال ۳

چرا ممکن است یک سیستم با عمر طولانی به اسناد طراحی بیشتری نیاز داشته باشد؟

جواب سوال ۳

پشتیبانی و نگهداری

اسناد طراحی کامل و به‌روز به تیم‌های مختلف کمک می‌کنند تا درک بهتری از سیستم داشته باشند، به ویژه در زمان انتقال مسئولیت نگهداری. به خصوص زمان انتقال یک سیستم و ملحقاتش، اسناد طراحی باعث می‌شوند این انتقال مسئولیت، با درک خوبی همراه شود و جزئیات زیادی که ممکن است در نگاه‌های اول دیده نشوند، ثبت شده باشند و توسط تیم جدید، تشخیص داده شوند.

تغییرات و به‌روزرسانی‌ها

اسناد طراحی دقیق می‌توانند به ثبت تغییرات و اطمینان از انسجام کلی سیستم در طول زمان کمک کنند. در کنار کیفیت بالای داکیومنتیشن یا همون مستندات، کیفیت بالای به‌روزرسانی‌ها و تغییرات نیز می‌تواند رخ دهد با توجه به اسناد طراحی دقیق.

کاهش خطای انسانی

اسناد طراحی کامل به کاهش خطر از دست دادن دانش و تجربه مرتبط با سیستم و حفظ دانش حیاتی کمک می‌کنند. این مورد نیز مشابه مورد اول در زمان انتقال سیستم تاثیرگذار است.

سازگاری با محیط‌های جدید

اسناد طراحی به شناسایی بخش‌هایی از سیستم که نیاز به تغییر یا به‌روزرسانی دارند، کمک می‌کنند، به خصوص در زمان انطباق با فناوری‌های جدید.

سوال ۴

.A

چرا باید برای ایجاد یک نرم‌افزار براساس یک مدل پیش برویم و در طول پروژه پایبند به آن مدل باشیم؟

.B

یک تیم مهندس نرم‌افزار برای پروژه‌ای در یک شرکت نفت بزرگ دعوت شده است. این شرکت چندین دپارتمان دارد و تیم مهندسی نرم‌افزار با دپارتمان مدیریت اطلاعات (MIS) تعامل می‌کند. سیستم MIS این شرکت قدیمی (Legacy) است و هدف، انتقال داده‌ها به یک سیستم جدید (مهاجرت داده) است. فرآیندها، قراردادهای قانونی و معیارهای پذیرش این شرکت بسیار خاص و حساس هستند. به نظر شما چه مدل ایجاد نرم‌افزاری برای راه‌اندازی این سیستم انتقال داده را تیم مهندس نرم‌افزار انتخاب خواهد کرد؟ نام مدل و علت اصلی انتخاب آن کافی است.

.C

مهم‌ترین مشکلات مدل‌های سنتی (مثل مدل آبشاری) نسبت به مدل‌های چابک، چیست؟ ()اشاره به ۳ مورد و توضیح کامل آنها کفایت می‌کند.)

جواب سوال ۴

A. اهمیت استفاده از یک مدل در ایجاد نرم‌افزار

- **ساختار و راهنمایی:** مدل‌ها چارچوب و راهنمایی لازم را برای توسعه نرم‌افزار فراهم می‌کنند، کمک به تمرکز تیم بر اهداف و مراحل مشخص. در واقع نظم تیم با استفاده از یک مدل، خیلی دقیق برنامه‌ریزی می‌شود.
- **مدیریت پیچیدگی:** استفاده از یک مدل پیچیدگی‌ها را مدیریت می‌کند و اطمینان می‌دهد که تمام جنبه‌های پروژه پوشش داده شوند. بعضا برای ایجاد نظم در تیم‌ها، از روش‌های مختلفی استفاده می‌شود ولی در پیچیدگی‌ها، مدیریت سخت می‌شود و بعضی بخش‌ها پوشش داده نمی‌شوند. برای همین بهتر از از یک مدل برای مدیریت بخش‌های پیچیده‌ی یک پروژه استفاده شود.
- **کنترل کیفیت:** پایبندی به مدل امکان بررسی و ارزیابی مرحله به مرحله پروژه را می‌دهد، که برای حفظ کیفیت نرم‌افزار ضروری است. همانند مورد قبل، کیفیت در بعضی بخش‌های پروژه ممکن است حفظ نشود در مدیریت عادی ولی با استفاده از مدل‌ها، این موضوع نیز پوشش داده می‌شود.
- **پیش‌بینی و برنامه‌ریزی:** مدل‌ها به تیم توسعه امکان پیش‌بینی و برنامه‌ریزی مناسب پیشرفت پروژه را می‌دهند.
- **همکاری و ارتباطات:** مدل‌ها زبان مشترکی برای ارتباط بین اعضای تیم و ذینفعان فراهم می‌کنند.
- **مستندسازی و توسعه مجدد:** مدل‌ها به مستندسازی فرایندها و تصمیمات کمک کرده و برای تحلیل و توسعه مجدد نرم‌افزار در آینده مهم هستند.

B. مدل ایجاد نرم‌افزار برای شرکت نفتی

مدل انتخابی: مدل V

دلیل انتخاب:

الف) **تاکید بر آزمون و اعتبارسنجی:** مدل V به خصوص برای پروژه‌هایی که نیاز به تایید دقیق و مستمر دارند، مناسب است. این مهم است زیرا در پروژه‌های مربوط به داده‌های حساس مانند مهاجرت داده‌ها در یک شرکت نفتی، اطمینان از دقت و امنیت داده‌ها در هر مرحله حیاتی است.

ب) **مدیریت ریسک در مهاجرت داده:** مهاجرت داده‌ها از یک سیستم قدیمی به سیستم جدید می‌تواند پیچیده باشد. مدل V با تاکید بر آزمون‌ها در هر مرحله از فرآیند توسعه، این امکان را فراهم می‌کند که اشکالات و مسائل احتمالی زودتر شناسایی و برطرف شوند.

ج) **پاسخ‌گویی به نیازمندی‌های خاص:** توجه به فرآیندها و قراردادهای قانونی خاص و حساسیت‌های شرکت نفتی نشان‌دهنده نیاز به یک رویکرد دقیق و ساختارمند است. مدل V با ارائه یک چارچوب مرحله‌به‌مرحله، این نیازها را برآورده می‌کند.

د) **ساختارمند و منظم:** این مدل با فرض اینکه نیازمندی‌ها به درستی تعریف شده‌اند، یک رویکرد منظم و ساختارمند ارائه می‌دهد که برای پروژه‌های با اهداف و فرآیندهای واضح مفید است.

با این حال، اگر فرآیند انتقال داده دارای ابهامات زیادی باشد و نیاز به تغییرات مکرر و ارزیابی‌های مداوم ریسک داشته باشد، مدل Spiral گزینه بهتری خواهد بود. مدل Spiral به خصوص برای پروژه‌های پیچیده و نوآورانه که نیازمند انعطاف‌پذیری و ارزیابی مداوم ریسک هستند، مناسب است. این مدل اجازه می‌دهد تا تیم مهندسی در طول پروژه به طور مداوم فرآیندها و نتایج را ارزیابی کرده و در صورت لزوم تغییرات را اعمال کند.

C. مشکلات مدل‌های سنتی (مانند مدل آبشاری) نسبت به مدل‌های چابک

- **انعطاف‌پذیری کم:** مدل‌های سنتی، معمولاً انعطاف‌پذیری کمی دارند. این بدان معناست که تغییرات در الزامات پروژه، می‌تواند منجر به مشکلات زیادی شود. مدل‌های چابک، انعطاف‌پذیری بیشتری دارند و به تیم توسعه نرم‌افزار اجازه می‌دهند تا تغییرات در الزامات پروژه را به سرعت شناسایی و اعمال کنند.
- **کمبود تعامل با مشتری:** مدل‌های سنتی، معمولاً تعامل کمی با مشتری دارند. این امر می‌تواند منجر به عدم رضایت مشتری شود. مدل‌های چابک، بر تعامل نزدیک با مشتری تأکید دارند. این امر به تیم توسعه نرم‌افزار کمک می‌کند تا نیازهای مشتری را به طور دقیق و کامل شناسایی کنند و محصولاتی را تولید کنند که رضایت مشتری را جلب کند.
- **تأخیر در بازخورد:** در مدل‌های سنتی، بازخورد کاربران و ذینفعان معمولاً در مراحل پایانی پروژه جمع‌آوری می‌شود، که می‌تواند منجر به تأخیر در شناسایی و حل مشکلات شود.
- **ریسک بالا و هزینه‌های تغییر:** به دلیل تأخیر در دریافت بازخورد و انعطاف‌پذیری کم، ریسک شکست پروژه‌ها و هزینه‌های ایجاد تغییرات افزایش می‌یابد.
- **تمرکز بر مستندات:** مدل‌های سنتی، معمولاً بر مستندات تأکید دارند. این امر می‌تواند منجر به صرف زمان و هزینه زیاد شود. مدل‌های چابک، بر تحویل ارزش به مشتری در فواصل زمانی کوتاه تأکید دارند. این امر می‌تواند منجر به صرف زمان و هزینه کمتر شود.

سوال ۵

با در نظر گرفتن رویکرد چابک به سوالات زیر پاسخ دهید:

A. در اکثر پروژه‌های نرم‌افزاری پیش‌بینی موارد زیر سخت است:

- اینکه کدام نیازمندی‌های مشتری تغییر خواهند کرد و کدام نیازمندی‌ها ثابت خواهند بود؟
- اینکه به چه میزان طراحی پیش از پیاده‌سازی احتیاج داریم؟
- و چه مقدار زمان از نظر برنامه‌ریزی برای تحلیل و طراحی، پیاده‌سازی و تست محصول نیاز خواهد بود؟

فرآیندهای چابک چگونه در جهت رفع این شرایط‌های نیاز به پیش‌بینی پاسخ می‌دهند؟

.B

اگر برای سیستم‌های بزرگ و با عمر طولانی که توسط یک شرکت نرم‌افزاری برای مشتریان خارجی توسعه داده می‌شوند، از رویکرد چابک استفاده شود، چه مشکلاتی ممکن است بوجود آید؟. ۳ مورد از مشکلات ممکن را ذکر کنید.

.C

فکر می‌کنید مدل‌های چابک خود چه مشکلاتی داشته باشند؟ (حداقل ۴ مورد)

جواب سوال ۵

.A

در فرآیندهای چابک، به جای تلاش برای پیش‌بینی دقیق تمام نیازمندی‌ها و طراحی‌ها از ابتدا، تمرکز بر توسعه تدریجی و تکراری محصول است. این رویکرد اجازه می‌دهد تا تیم‌ها به سرعت و با انعطاف‌پذیری بالا به تغییرات نیازمندی‌ها واکنش نشان دهند. زمان‌بندی برای تحلیل، طراحی، پیاده‌سازی و تست نیز به صورت انعطاف‌پذیر در چرخه‌های کوتاه مدت مدیریت می‌شود.

فرآیندهای چابک، بر تعامل نزدیک با مشتری تأکید دارند. این امر به تیم توسعه نرم‌افزار کمک می‌کند تا نیازمندی‌های مشتری را به طور دقیق و کامل شناسایی کنند و تغییرات را به سرعت شناسایی و اعمال کنند.

فرآیندهای چابک، بر آزمایش مداوم تأکید دارند. این امر به تیم توسعه نرم‌افزار کمک می‌کند تا کیفیت محصول را در طول توسعه تضمین کنند و از تغییرات غیرمنتظره در نیازمندی‌های مشتری جلوگیری کنند.

فرآیندهای چابک، انعطاف‌پذیر هستند و اجازه می‌دهند تا تغییرات در نیازمندی‌ها، طراحی و برنامه‌ریزی پروژه به سرعت اعمال شوند. این امر به تیم توسعه نرم‌افزار کمک می‌کند تا به تغییرات محیطی و نیازهای مشتری پاسخ دهند.

فرض کنید یک شرکت نرم‌افزاری برای یک مشتری جدید، یک سیستم اتوماسیون فروش توسعه می‌دهد. مشتری نیازهای خود را به صورت کلی به تیم توسعه نرم‌افزار ارائه می‌دهد. تیم توسعه نرم‌افزار با تعامل نزدیک با مشتری،

نیازهای مشتری را به صورت دقیق‌تر شناسایی می‌کند. سپس، محصول را در فواصل زمانی کوتاه تحویل می‌دهد تا مشتری بتواند آن را بررسی کند و تغییرات را درخواست کند.

در این مثال، فرآیندهای چابک به تیم توسعه نرم‌افزار کمک می‌کند تا به تغییرات در نیازمندی‌های مشتری پاسخ دهند. به عنوان مثال، اگر مشتری نیاز به اضافه کردن یک ویژگی جدید به سیستم را داشته باشد، تیم توسعه نرم‌افزار می‌تواند این ویژگی را در نسخه بعدی محصول اعمال کند.

در نتیجه فرآیندهای چابک، با توجه به چالش‌های پیش‌بینی در پروژه‌های نرم‌افزاری، از رویکردهای مختلفی استفاده می‌کنند. این رویکردها به تیم‌های توسعه نرم‌افزار کمک می‌کند تا تغییرات در نیازمندی‌های مشتری را به سرعت شناسایی و اعمال کنند و محصولی با کیفیت بالا تحویل دهند.

.B

برای سیستم‌های بزرگ و با عمر طولانی، استفاده از رویکرد چابک ممکن است مشکلاتی از قبیل:

- دشواری در مدیریت و هماهنگی بین تیم‌های بزرگ و پراکنده.
- چالش‌های مربوط به برقراری ارتباط مؤثر با مشتریان بین‌المللی.
- مسائل مربوط به مقیاس‌پذیری و ادغام مداوم تغییرات در یک سیستم بزرگ.

سیستم‌های بزرگ و با عمر طولانی، معمولاً توسط تیم‌های بزرگ و پراکنده توسعه داده می‌شوند. این تیم‌ها ممکن است در مکان‌های مختلف قرار داشته باشند و از فرهنگ‌ها و زبان‌های مختلف باشند. مدیریت و هماهنگی بین این تیم‌ها، در رویکرد چابک که بر تعامل نزدیک با مشتری تأکید دارد، می‌تواند دشوار باشد.

برای حل این مشکل، تیم‌های توسعه نرم‌افزار باید از ابزارها و تکنیک‌های ارتباطی و همکاری موثر استفاده کنند. همچنین، باید یک برنامه‌ریزی دقیق برای مدیریت و هماهنگی بین تیم‌ها داشته باشند.

اگر سیستم توسط یک شرکت نرم‌افزاری برای مشتریان خارجی توسعه داده شود، تیم توسعه نرم‌افزار باید بتواند با مشتریان بین‌المللی به طور موثر ارتباط برقرار کند. این امر می‌تواند چالش‌هایی را ایجاد کند، به خصوص اگر مشتریان از زبان‌ها و فرهنگ‌های مختلف باشند.

برای حل این مشکل، تیم توسعه نرم‌افزار باید مهارت‌های ارتباطی بین‌المللی داشته باشد. همچنین، باید از ابزارها و تکنیک‌های ارتباطی موثر استفاده کند.

سیستم‌های بزرگ و با عمر طولانی، معمولاً پیچیده هستند و نیاز به نگهداری و پشتیبانی مداوم دارند. در رویکرد چابک، تغییرات در سیستم به صورت مداوم اعمال می‌شوند. این امر می‌تواند چالش‌هایی را در زمینه مقیاس‌پذیری و ادغام تغییرات ایجاد کند.

برای حل این مشکل، تیم توسعه نرم‌افزار باید از ابزارها و تکنیک‌های مناسب برای مدیریت تغییرات استفاده کند. همچنین، باید یک برنامه‌ریزی دقیق برای مدیریت مقیاس‌پذیری سیستم داشته باشد.

البته، استفاده از رویکرد چابک برای سیستم‌های بزرگ و با عمر طولانی، مزایای زیادی نیز دارد. به عنوان مثال، چابکی می‌تواند به تیم‌های توسعه نرم‌افزار کمک کند تا به تغییرات نیازهای مشتری به سرعت پاسخ دهند.

در نهایت، انتخاب رویکرد مناسب برای توسعه یک سیستم نرم‌افزاری، به عوامل مختلفی مانند اندازه سیستم، پیچیدگی سیستم، و الزامات مشتری بستگی دارد.

.C

مدل‌های چابک، به دلیل انعطاف‌پذیری و تمرکز بر تحویل ارزش به مشتری، مزایای زیادی نسبت به مدل‌های سنتی دارند. با این حال، این مدل‌ها نیز می‌توانند مشکلاتی داشته باشند. در ادامه، چهار مورد از مشکلات احتمالی مدل‌های چابک را بررسی می‌کنیم:

مدل‌های چابک خود می‌توانند مشکلاتی داشته باشند از جمله:

- نیاز به ارتباط و همکاری مداوم و نزدیک با مشتری.

یکی از اصول کلیدی مدل‌های چابک، ارتباط و همکاری مداوم و نزدیک با مشتری است. این امر به تیم توسعه نرم‌افزار کمک می‌کند تا نیازهای مشتری را به طور دقیق و کامل شناسایی کنند و تغییرات را به سرعت شناسایی و اعمال کنند.

با این حال، این امر می‌تواند چالش‌هایی را برای تیم توسعه نرم‌افزار ایجاد کند. به عنوان مثال، ممکن است مشتری در دسترس نباشد یا زمان کافی برای همکاری با تیم توسعه نرم‌افزار را نداشته باشد. همچنین، ممکن است مشتری نیازهای خود را به طور دقیق نداند و تغییرات مکرر در نیازهای خود ایجاد کند.

- دشواری در پیش‌بینی هزینه‌ها و زمان‌بندی‌های بلندمدت.

یکی دیگر از مشکلات احتمالی مدل‌های چابک، دشواری در پیش‌بینی هزینه‌ها و زمان‌بندی‌های بلندمدت است. این امر به دلیل انعطاف‌پذیری بالای مدل‌های چابک و امکان تغییرات مکرر در الزامات پروژه است.

در مدل‌های سنتی، معمولاً یک برنامه‌ریزی دقیق در ابتدای پروژه انجام می‌شود و هزینه‌ها و زمان‌بندی‌های بلندمدت بر اساس این برنامه‌ریزی تخمین زده می‌شوند. با این حال، در مدل‌های چابک، این برنامه‌ریزی معمولاً در طول پروژه انجام می‌شود و هزینه‌ها و زمان‌بندی‌های بلندمدت به صورت مداوم به روز می‌شوند.

- خطر انحراف از اهداف اصلی در صورت عدم وجود برنامه‌ریزی دقیق.

در مدل‌های چابک، بر تحویل ارزش به مشتری در فواصل زمانی کوتاه تأکید می‌شود. این امر می‌تواند منجر به انحراف از اهداف اصلی پروژه شود، به خصوص اگر برنامه‌ریزی دقیقی برای پروژه انجام نشود.

برای جلوگیری از این مشکل، تیم توسعه نرم‌افزار باید اهداف اصلی پروژه را به خوبی شناسایی کند و یک برنامه‌ریزی دقیق برای پروژه داشته باشد. همچنین، باید در طول پروژه، اهداف اصلی پروژه را به طور مداوم بررسی کند و در صورت لزوم، تغییرات لازم را اعمال کند.

- ممکن است کیفیت کد در اثر تغییرات مکرر کاهش یابد.

یکی از ویژگی‌های مدل‌های چابک، تغییرات مکرر در محصول است. این امر می‌تواند منجر به کاهش کیفیت کد شود، به خصوص اگر تیم توسعه نرم‌افزار از روش‌های صحیح مدیریت تغییرات استفاده نکند.

برای جلوگیری از این مشکل، تیم توسعه نرم‌افزار باید از روش‌های صحیح مدیریت تغییرات استفاده کند. همچنین، باید کیفیت کد را در طول پروژه به طور مداوم بررسی کند و در صورت لزوم، اقدامات لازم را برای بهبود کیفیت کد انجام دهد.
