|  |  |
| --- | --- |
| Software process | |
| عبدالرحمان بیگناهی | 09164265024 | rahman-bigonahi@gmail.com | |
|  | |
| درس : مهندسی نرم افزار  مدرس : محمد احمدزاده  پاییز 1403 | فرایند تولید نرم‌افزار، که با عنوان «چرخه حیات تولید نرم‌افزار» نیز شناخته می‌شود، ساختاری است که روی توسعه و تولید محصولات نرم‌افزاری اعمال می‌شود.  از مهم‌ترین کارها در تولید نرم‌افزار استخراج نیازمندی‌ها یا تحلیل نیازمندی‌های آن سامانه است. مشتریان عمومی معمولاً تصور مفهومی، انتزاعی و مبهمی از نتیجه نهایی خواسته‌هایشان دارند و نمی‌دانند به درستی نرم‌افزار مورد نظرشان چه کاری باید انجام دهد. در این مرحله نیازمندی‌های ناتمام، پیچیده و مبهم، و حتی متضاد توسط مهندسان نرم‌افزار ماهر شناسایی می‌شوند. |
|  | |

1. :( User-Driven Development)UDD

UDD یا توسعه محور کاربر که بر نیازهای کاربر نهایی تمرکز دارد. در این مدل، توسعه‌دهندگان سعی می‌کنند تا با تعامل مداوم با کاربران، تجربیات و نیازهای آنها را شناسایی و درک کنند. بازخورد کاربر در فرآیند توسعه به‌طور مکرر مورد استفاده قرار می‌گیرد. این روش به افزایش رضایت کاربران و همچنین بهبود کیفیت نرم‌افزار منجر می‌شود. به طور کلی، UDD به ایجاد محصولات نرم‌افزاری که واقعاً با نیازهای کاربران همسو هستند کمک می‌کند.

**کاربردها:**

* استفاده در پروژه‌های نرم‌افزاری که نیازمند ارتباط نزدیک با کاربران هستند.
* طراحی و توسعه نرم‌افزارهای سفارشی بر اساس نیازهای خاص مشتریان.
* بهبود تجربه کاربری در تکرارهای مداوم.

**مزایا:**

* ارتقاء رضایت و وفاداری کاربران.
* نتایج بهتر ناشی از آزمون‌های مداوم و بازخورد سریع.
* کاهش احتمال ایجاد ویژگی‌های غیرضروری بر اساس نیاز واقعی کاربران.

**معایب:**

* ممکن است زمان‌بر باشد و منابع بیشتری مصرف کند، زیرا نیاز به تعامل مداوم با کاربران دارد.
* تغییر مکرر در نیازها ممکن است منجر به تاخیر در روند توسعه و افزایش هزینه‌ها شود.
* افراد درگیر در فرآیند ممکن است به نظر متفاوتی درباره نیازها داشته باشند.

1. :( Value Chain Design)VCD

VCD یا طراحی زنجیره ارزش به تجزیه و تحلیل فعالیت‌های یک سازمان پرداخته و به شناسایی و بهینه‌سازی زنجیره‌ای می‌پردازد که منجر به ایجاد ارزش برای مشتریان می‌شود. این رویکرد کمک می‌کند تا نقاط قوت و ضعف سازمان در فرآیند تولید و خدمات شناسایی شوند. با تحلیل زنجیره ارزش، می‌توان به بهبود بهره‌وری، کاهش هزینه‌ها و افزایش رضایت مشتری دست یافت. طراحی زنجیره ارزش معمولاً شامل چهارچوب‌ها و ابزارهای خاصی برای تجزیه و تحلیل هزینه‌ها و ارزش‌های افزوده است. اهداف اصلی آن، ایجاد مزیت رقابتی و بهینه‌سازی منابع است.

**کاربردها:**

* تحلیل و بهینه‌سازی فرآیندها و فعالیت‌های تجاری در سازمان‌ها.
* شناسایی مزیت‌های رقابتی و نقاط قوت در زنجیره تأمین.
* ارزیابی و بهبود کارایی مالی و عملیاتی.

**مزایا:**

* کمک به ایجاد بینش عمیق نسبت به فعالیت‌های تولید و ارائه خدمات.
* توانایی شناسایی و حذف فعالیت‌های غیرضروری یا پرهزینه.
* ارتقاء کیفیت و کاهش زمان تحویل محصولات و خدمات به مشتریان.

**معایب:**

* نیاز به وقت و منابع برای تجزیه و تحلیل دقیق و مستندات مربوطه دارد.
* ممکن است در سازمان‌های بزرگ پیچیدگی‌هایی ایجاد کند که مدیریت آن سخت باشد.
* ممکن است در برخی موارد به نتایج غیرقابل پیش‌بینی منجر شود، اگر تحلیل به درستی انجام نشود.

1. :( Behavior-Driven Development)BDD

BDD یا توسعه بر اساس رفتار، به معیارهای رفتار نرم‌افزار از منظر کاربران نهایی و ذی‌نفعان متمرکز است. در این روش، توسعه‌دهندگان با همکاری تحلیلگران کسب‌وکار و تست‌کنندگان داستان‌های کاربری نوشته و آنها را به سناریوهای قابل آزمون تبدیل می‌کنند. BDD با استفاده از زبان‌های قابل فهم (مانند Gherkin) مستندات دقیقی از رفتار سیستم ایجاد می‌کند که می‌تواند به عنوان راهنما برای توسعه و تست استفاده شود. این رویکرد به تسهیل ارتباطات در تیم‌های چندنفره کمک کرده و تضمین می‌کند که نرم‌افزار نیازهای واقعی کاربران را برآورده کند. در نهایت، BDD به بهبود کیفیت نرم‌افزار و کاهش نیاز به اصلاحات پس از تولید منجر می‌شود.

**کاربردها:**

* توسعه نرم‌افزاری که بر اساس نیازهای کاربران و سناریوهای واقعی کار می‌کند.
* تست نرم‌افزار به صورت همکاری بین تیم‌های مختلف (توسعه، تست، کسب‌وکار).
* ایجاد مستندات زنده و قابل فهم که شیوه‌های توسعه را هدایت می‌کند.

**مزایا:**

* بهبود همکاری و ارتباط در تیم‌های توسعه نرم‌افزار.
* تسهیل در آزمایش رفتار نرم‌افزار و افزایش کیفیت نهایی محصول.
* امکان ایجاد مستندات دقیق و دائمی که با تغییرات کد به‌روزرسانی می‌شود.

**معایب:**

* ممکن است نیاز به تغییرات مهم در فرهنگ سازمانی و تیمی برای پیاده‌سازی داشته باشد.
* ممکن است در مراحل اولیه زمان‌بر باشد تا تیم توانایی‌های جدید را توسعه دهد.
* در مواردی که نیازمندی‌ها کاملاً واضح نیست، ارزیابی رفتار می‌تواند مشکل‌ساز شود.

1. :( **Feature-Driven Development**) FDD

FDD یا توسعه مبتنی بر ویژگی، تمرکز اصلی آن بر روی شناسایی، طراحی و تحویل ویژگی‌های خاص نرم‌افزار است. در این رویکرد، کل پروژه به بخش‌های کوچکتر (ویژگی‌ها) تقسیم می‌شود و هر ویژگی به‌عنوان یک واحد مستقل برای توسعه و تست در نظر گرفته می‌شود. FDD به شدت بر همکاری بین اعضای تیم، بررسی‌های پیوسته و تحویل تدریجی محصولات تأکید دارد. با استفاده از این متدولوژی، تیم‌ها می‌توانند بر روی نیازهای واقعی کاربران تمرکز کرده و اطمینان حاصل کنند که هر ویژگی به‌طور موثر طراحی و پیاده‌سازی می‌شود.

**کاربردها:**

* مناسب برای پروژه‌های نرم‌افزاری بزرگ که نیاز به تحویل تدریجی ویژگی‌های مستقل دارند.
* در تیم‌های توسعه بزرگ برای مدیریت مؤثر وظایف و ویژگی‌ها به‌کار می‌رود.
* در محیط‌های Agile که به تحویل مکرر و اطلاع‌رسانی منظم نیاز دارند.

**مزایا:**

* ایجاد شفافیت در فرآیند توسعه با تمرکز بر ویژگی‌های کاربردی.
* امکان تقسیم کار بر اساس ویژگی‌ها که موجب بهبود همکاری تیمی می‌شود.
* تسهیل در اولویت‌بندی ویژگی‌ها برای دستیابی به تأثیر سریع‌تر در بازار.

**معایب:**

* نیاز به زمان و تلاش بیشتر برای تدوین ویژگی‌ها در مراحل ابتدایی.
* در صورت عدم مدیریت مناسب، می‌تواند به پراکندگی و عدم تمرکز در توسعه منجر شود.
* موفقیت آن وابسته به تجربه و تسلط تیم بر این روش است.

1. TDD (Test-Driven Development):

توسعه مبتنی بر تست، در آن تست‌ها قبل از نوشتن پیاده‌سازی واقعی برنامه نوشته می‌شوند. در TDD، چرخه کار به‌طور معمول در سه مرحله اصلی انجام می‌شود: اول، یک آزمون جدید برای عملکرد موردنظر نوشته می‌شود، که در ابتدا این آزمون باید شکست بخورد؛ سپس کد جدید نوشته می‌شود تا آزمون را پاس کند؛ و در نهایت، کد نوشته شده بهینه و خالص می‌شود. این فرآیند نه تنها به شفافیت و کارآیی کمک می‌کند بلکه طراحی نرم‌افزار را نیز تحت تأثیر قرار می‌دهد و کد را به‌شکلی ساختاریافته‌تر و کارآمدتر می‌سازد.

**کاربردها:**

* توسعه نرم‌افزارهایی که نیاز به کیفیت بالا و کد مستحکم دارند.
* استفاده در پروژه‌هایی که در آن تست‌های خودکار ضروری هستند.
* محیط‌های Agile که نیازمند توسعه مکرر و تکراری کد می‌باشند.

**مزایا:**

* کاهش احتمال بروز خطا در کد به دلیل وجود تست‌های خودکار.
* تسهیل در فرآیند نگهداری و به‌روزرسانی کد با داشتن مستندات واضح از رفتار مورد انتظار.
* بهبود طراحی کد با تمرکز بر نیازها و قابلیت‌های سطح بالا.

**معایب:**

* ممکن است در مراحل اولیه زمان‌بر باشد زیرا نوشتن تست‌ها قبل از کد نیازمند دقت و ارتباط واضح است.
* عدم وجود تست‌های جامع در مراحل اولیه شاید منجر به ایجاد زیرساخت‌های نامناسب شود.
* نیاز به تجربه و دانش در زمینه نوشتن تست‌های موثر دارد و کارکنان جدیدی که با آن آشنا نیستند ممکن است با چالش‌های زیادی مواجه شوند.

1. CDD (**Contract-Driven Development**):

توسعه مبتنی بر قرارداد، بر اساس توافقات رسمی و مستندات نوشته شده بین ذی‌نفعان، توسعه نرم‌افزار را هدایت می‌کند. در این روش، قراردادها به‌عنوان معیارهایی برای الزامات، شرایط و توافقات پروژه تعریف می‌شوند. این قراردادها می‌توانند شامل الزامات عملکرد، ویژگی‌ها، زمان‌بندی و معیارهای قبول باشند CDD می‌تواند به عنوان یک زیرساخت برای همکاری بین تیم‌های مختلف و همچنین به‌عنوان ابزاری برای ارزیابی موفقیت پروژه در برابر معیارهای مشخص به‌کار رود.

**کاربردها:**

* استفاده در پروژه‌های با ذی‌نفعان متعدد و متفاوت که نیاز به توافقات رسمی دارند.
* توسعه سیستم‌های پیچیده که شامل تعاملات متعدد بین اجزا هستند.
* محیط‌هایی که نیاز به دریافت تأییدیه‌های مشخص دارند، مانند پروژه‌های دولتی.

**مزایا:**

* وجود مستندات واضح و دقیق از الزامات و توافقات کارفرما و پیمانکار.
* کاهش احتمال سوء تفاهم در مراحل توسعه.
* قابلیت پیگیری و بررسی انطباق با قرارداد در طول پروژه.

**معایب:**

* ممکن است به کندی در فرآیند توسعه منجر شود، چرا که نیاز به توافق‌های رسمی دارد.
* در مواجهه با تغییرات نیاز، ممکن است انعطاف‌پذیری کمتری داشته باشد.
* نیاز به هزینه و زمان بیشتر برای تدوین و مدیریت قراردادها.

نتیجه گیری :

| **روش** | **توضیح** | **کاربردها** | **مزایا** | **معایب** |
| --- | --- | --- | --- | --- |
| **توسعه محور کاربر (UDD)** | تمرکز بر نیازها و بازخورد کاربر، با تعامل مداوم با کاربران | پروژه‌های نرم‌افزاری با نیاز به ارتباط نزدیک با کاربران | افزایش رضایت کاربران، بهبود تجربه کاربری، کاهش ویژگی‌های غیرضروری | زمان‌بر و پرهزینه به دلیل تعامل مداوم با کاربران، احتمال تاخیر در توسعه |
| **طراحی زنجیره ارزش (VCD)** | تحلیل و بهینه‌سازی فعالیت‌های سازمان برای افزایش ارزش به مشتریان | بهینه‌سازی فرآیندها، شناسایی مزیت‌های رقابتی و نقاط قوت در زنجیره تأمین | بهبود بهره‌وری و کاهش هزینه‌ها، شناسایی و حذف فعالیت‌های غیرضروری، افزایش کیفیت و کاهش زمان تحویل | نیاز به زمان و منابع زیاد، پیچیدگی برای سازمان‌های بزرگ، نتایج غیرقابل پیش‌بینی در صورت تحلیل نادرست |
| **توسعه بر اساس رفتار (BDD)** | تمرکز بر معیارهای رفتاری نرم‌افزار از دید کاربر، همکاری تحلیل‌گر کسب‌وکار و تست‌کننده در نوشتن سناریوهای قابل آزمون | توسعه نرم‌افزاری بر اساس نیازهای کاربران، همکاری بین تیم‌های مختلف، ایجاد مستندات دقیق و دائمی | بهبود همکاری تیم‌ها، تسهیل آزمایش رفتار نرم‌افزار، امکان مستندسازی دقیق و به‌روز | نیاز به تغییر فرهنگ تیمی، زمان‌بر در مراحل اولیه، مشکل در تحلیل رفتار در صورت نامشخص بودن نیازها |
| **توسعه مبتنی بر ویژگی (FDD)** | تقسیم پروژه به ویژگی‌های کوچک و مستقل و تحویل تدریجی هر ویژگی | پروژه‌های بزرگ با نیاز به تحویل تدریجی ویژگی‌ها، تیم‌های توسعه بزرگ | تمرکز بر ویژگی‌ها، بهبود همکاری تیمی، اولویت‌بندی ویژگی‌ها برای تاثیر سریع‌تر در بازار | نیاز به زمان و تلاش برای تدوین ویژگی‌ها، احتمال پراکندگی و عدم تمرکز در توسعه |
| **توسعه مبتنی بر تست (TDD)** | نوشتن تست‌ها پیش از کد، چرخه توسعه شامل نوشتن تست، کد و بهینه‌سازی | نرم‌افزارهایی با نیاز به کیفیت بالا، پروژه‌های نیازمند تست خودکار | کاهش خطاها، تسهیل نگهداری و به‌روزرسانی کد، بهبود طراحی کد | زمان‌بر در مراحل اولیه، نیاز به تجربه در نوشتن تست‌های موثر، احتمال ایجاد زیرساخت‌های نامناسب در مراحل ابتدایی |
| **توسعه مبتنی بر قرارداد (CDD)** | توسعه بر اساس قراردادهای رسمی بین ذی‌نفعان، با تعیین الزامات و توافقات مشخص | پروژه‌های با ذی‌نفعان متعدد، سیستم‌های پیچیده با تعاملات بین اجزا، پروژه‌های نیازمند تاییدیه‌های رسمی | مستندسازی واضح الزامات، کاهش سوء تفاهم، قابلیت پیگیری و بررسی انطباق با قرارداد | زمان‌بر و نیازمند منابع زیاد، انعطاف‌پذیری کمتر در مواجهه با تغییرات، هزینه و زمان بیشتر برای تدوین قراردادها |