**02-software-process(alireza-afrang)**

**TDD (Test-Driven Development)**

* : TDD یا توسعه مبتنی بر تست، یک روش توسعه است که ابتدا برنامه‌نویس تست‌های لازم را برای هر قسمت از کد ایجاد می‌کند و سپس کدی را می‌نویسد که این تست‌ها را پاس کند. این تست‌ها به طور مداوم اجرا می‌شوند تا اطمینان حاصل شود که تمام تغییرات، قابلیت اجرای صحیح را دارند. این روش به صورت چرخه‌ای تکرار می‌شود تا تمامی قسمت‌های نرم‌افزار، در کنار هم یکپارچه و بدون باگ باشند.
* **مزایا**:
  + کشف سریع مشکلات و کاهش خطاها در مراحل اولیه توسعه
  + تضمین کیفیت کد با تشویق به نوشتن کد بهینه و قابل تست
  + امکان توسعه راحت‌تر و تغییرات بهتر در کد به دلیل مستندسازی خودکار
* **معایب**:
  + نیاز به زمان و تلاش بیشتر برای نوشتن تست‌های اولیه، به‌ویژه در پروژه‌های کوچک
  + نیاز به تسلط بالا در نوشتن تست‌های کارآمد و مهارت کافی در ایجاد سناریوهای تست
  + ممکن است برای برنامه‌نویسان تازه‌کار پیچیده به نظر برسد
* **کاربردها**: این روش برای توسعه نرم‌افزارهای حساس و پیچیده که نیاز به اطمینان از کیفیت بالا و پوشش تستی کامل دارند، مناسب است. همچنین برای برنامه‌های کاربردی که نیاز به انعطاف در تغییرات دارند، انتخابی ایده‌آل است.

**2. FDD (Feature-Driven Development)**

* : FDD یک روش مبتنی بر تکمیل تدریجی ویژگی‌ها است. در این روش هر ویژگی به عنوان یک واحد مستقل تعریف و توسعه داده می‌شود. این رویکرد به تیم توسعه اجازه می‌دهد تا به صورت مرحله به مرحله بر روی هر ویژگی تمرکز کند، و هر بار بخشی از پروژه را به صورت کامل تحویل دهد. هر ویژگی در طول یک دوره کوتاه‌مدت تکمیل می‌شود و نتایج قابل مشاهده‌ای دارد.
* **مزایا**:
  + نمایش مستمر پیشرفت پروژه به دلیل تکمیل تدریجی ویژگی‌ها
  + تقویت کار تیمی و امکان تخصیص وظایف به اعضای مختلف بر اساس تخصص‌ها
  + مدیریت و پیگیری آسان ویژگی‌ها، به‌ویژه در پروژه‌های بزرگ
* **معایب**:
  + دشواری در هماهنگی و پیگیری چندین ویژگی به‌صورت همزمان
  + نیازمند برنامه‌ریزی و تقسیم وظایف به صورت دقیق
  + ممکن است در پروژه‌های کوچک کارایی کمتری داشته باشد
* **کاربردها**: این روش برای پروژه‌های بزرگ و پیچیده که شامل ویژگی‌های متعدد هستند و نیاز به مدیریت کارآمد این ویژگی‌ها وجود دارد، مناسب است.

**3. BDD (Behavior-Driven Development)**

* : BDD یا توسعه مبتنی بر رفتار، روشی است که رفتارهای مورد انتظار سیستم را بر اساس نیازهای کاربران تعریف می‌کند. در این روش، سناریوهایی براساس رفتار و تعامل کاربران نوشته می‌شوند تا اطمینان حاصل شود که نرم‌افزار مطابق با نیازهای واقعی کاربران توسعه می‌یابد. این روش به تسهیل همکاری بین تیم‌های فنی و غیر فنی کمک می‌کند.
* **مزایا**:
  + افزایش فهم مشترک بین تیم‌های مختلف با تمرکز بر رفتارهای سیستم
  + تولید کدهایی که به طور مستقیم نیازهای کاربران را پوشش می‌دهد
  + مستندسازی طبیعی کدها از طریق تعریف سناریوهای رفتاری
* **معایب**:
  + نوشتن سناریوهای رفتاری پیچیده و نیازمند زمان و دقت بالا است
  + نیاز به تجربه و درک بالا از نیازهای کاربر و سناریوهای تعامل
  + ممکن است منابع بیشتری را در فرآیند برنامه‌ریزی و توسعه مصرف کند
* **کاربردها**: برای پروژه‌هایی که در آن‌ها درک صحیح از نیازهای کاربران و همکاری بین تیم‌های مختلف از اهمیت بالایی برخوردار است، BDD روشی موثر است.

**4. CDD (Context-Driven Development)**

* : CDD یا توسعه مبتنی بر بستر، روشی است که توسعه نرم‌افزار را به محیط و شرایط خاص هر پروژه بستگی می‌دهد. این روش، انعطاف‌پذیری بالایی دارد و به توسعه‌دهندگان این امکان را می‌دهد که تصمیمات خود را بر اساس نیازهای واقعی و شرایط موجود در لحظه بگیرند.
* **مزایا**:
  + انعطاف‌پذیری بالا و قابلیت سازگاری با شرایط مختلف
  + تمرکز روی نیازها و اولویت‌های اصلی پروژه
  + کاهش محدودیت‌های رویکردهای ثابت و ارائه یک فرآیند پویا و سازگار
* **معایب**:
  + ممکن است در نبود معیارهای ثابت، تمرکز تیم‌ها کاهش یابد
  + نیاز به تصمیم‌گیری‌های مستمر و تغییرات زیاد
* **کاربردها**: مناسب برای پروژه‌هایی که در آن‌ها عدم قطعیت زیاد و نیاز به تغییرات مستمر وجود دارد، مانند پروژه‌های استارتاپی یا پروژه‌های پیچیده با متغیرهای محیطی زیاد.

**5. DB (Database-Driven Development)**

* : DB یا توسعه مبتنی بر دیتابیس، یک رویکرد توسعه است که در آن سیستم به گونه‌ای طراحی می‌شود که بر اساس نیازهای دیتابیس ساخته شود. در این روش، ساختار و سازمان‌دهی دیتابیس اهمیت بالایی دارد و تمام قسمت‌های نرم‌افزار براساس داده‌ها و تعاملات مربوط به آن‌ها طراحی می‌شوند.
* **مزایا**:
  + بهینه‌سازی مدیریت داده‌ها و سهولت دسترسی به اطلاعات
  + افزایش دقت و سازگاری با داده‌ها و امکان تحلیل بهتر داده‌ها
  + فراهم‌آوری بستر مناسب برای پردازش داده‌های بزرگ
* **معایب**:
  + تمرکز بیشتر روی ساختار دیتابیس و کمبود توجه به منطق برنامه
  + وابستگی زیاد به دیتابیس که باعث پیچیدگی در تغییرات آتی می‌شود
* **کاربردها**: در پروژه‌های داده‌محور مانند سیستم‌های اطلاعات تجاری و برنامه‌هایی که داده‌های بزرگ را پردازش و مدیریت می‌کنند، DB روش مناسبی است.

**6. UCD (User-Centered Design)**

* : UCD یا طراحی کاربرمحور، یک فرآیند طراحی است که بر ایجاد تجربه کاربری مناسب از طریق درک نیازهای کاربران و طراحی براساس آن‌ها تمرکز دارد. این روش به معنای بهبود تجربه و راحتی استفاده کاربران در تعامل با نرم‌افزار است.
* **مزایا**:
  + بهبود تجربه و رضایت کاربران از نرم‌افزار
  + افزایش دسترسی‌پذیری و تطابق بیشتر با نیازهای واقعی کاربران
  + افزایش پذیرش نرم‌افزار توسط کاربران نهایی
* **معایب**:
  + نیازمند زمان و منابع بیشتر برای شناخت دقیق نیازهای کاربران
  + ممکن است برای همه کاربران مناسب نباشد و به طراحی خاص نیاز داشته باشد
* **کاربردها**: در پروژه‌هایی که تجربه کاربر در اولویت است، مانند نرم‌افزارهای تجاری و برنامه‌هایی که کاربران عام درگیر آن‌ها هستند، UCD بهترین گزینه است.

**7. UDD (User-Driven Development)**

* : UDD یا توسعه کاربرمحور، یک روش توسعه است که در آن نظرات و بازخوردهای کاربران نقش تعیین‌کننده‌ای در توسعه و اولویت‌بندی ویژگی‌ها دارد. این رویکرد به‌ویژه در پروژه‌هایی که تعاملات مستمر با کاربران وجود دارد، مورد استفاده قرار می‌گیرد.
* **مزایا**:
  + افزایش هم‌راستایی نرم‌افزار با نیازهای واقعی کاربران
  + بهبود رضایت کاربران به دلیل تعامل مستقیم در توسعه
  + افزایش تطابق نرم‌افزار با نیازهای کاربر نهایی
* **معایب**:
  + نیازمند هماهنگی مستمر با کاربران برای اجتناب از پیچیدگی و سردرگمی
  + ممکن است منجر به کاهش تمرکز تیم بر روی اهداف اصلی شود
* **کاربردها**: برای پروژه‌هایی که در آن‌ها تعامل با کاربران و بازخورد مستمر کاربران مهم است، مانند پروژه‌های اجتماعی و پلتفرم‌های تحت وب که نیاز به تطابق با نیازهای کاربران دارند.

|  |  |  |  |
| --- | --- | --- | --- |
| Software process | مزایا | معایب | موارد استفاده |
| TDD  توسعه مبتنی بر تست | کاهش باگ‌ها و بهبود کیفیت کد از طریق نوشتن تست‌ها پیش از کدنویسی که به افزایش امکان اصلاح کد و مستندسازی خودکار کمک می‌کند. | ممکن است زمان‌بر باشد، به‌ویژه در پروژه‌های کوچک، و نیازمند مهارت در نوشتن تست است. | مناسب برای پروژه‌هایی که کیفیت و تست‌پذیری بالا نیاز دارند، مانند نرم‌افزارهای حساس یا در سطح سازمانی. |
| FDD  توسعه مبتنی بر ویژگی | امکان پیگیری پیوسته پیشرفت و سازماندهی بهتر تیم‌ها را فراهم می‌کند و به تدریج ویژگی‌های مشخص را تحویل می‌دهد که برای پروژه‌های بزرگ مناسب است. | مدیریت و پیگیری ویژگی‌های متعدد ممکن است پیچیده باشد و نیاز به برنامه‌ریزی دقیق و مدیریت تیم دارد. | در پروژه‌های بزرگ که به تحویل تدریجی و ویژگی به ویژگی نیاز دارند و تیم‌های متعددی در آن دخیل هستند کاربرد دارد. |
| BDD  توسعه مبتنی بر رفتار | ارتباط بین تیم‌های فنی و غیر فنی را با تمرکز بر سناریوهای کاربرمحور تقویت می‌کند و کد را با نیازهای کاربر هماهنگ می‌سازد. | نوشتن سناریوهای رفتاری می‌تواند پیچیده و نیازمند هماهنگی دقیق با انتظارات واقعی کاربر باشد که ممکن است زمان‌بر شود. | در پروژه‌هایی با تیم‌های چندرشته‌ای برای تضمین تطابق رفتار نرم‌افزار با نیازهای کاربر و اهداف کسب و کار کاربرد دارد. |
| CDD  توسعه مبتنی بر شرایط | انعطاف‌پذیری بالایی دارد و امکان سازگاری با شرایط و نیازهای خاص هر مرحله از پروژه را فراهم می‌کند. | تغییرات و تصمیم‌گیری‌های مکرر ممکن است تمرکز را کاهش دهد و نبود معیارهای ثابت می‌تواند چالش‌برانگیز باشد. | مناسب برای پروژه‌های پویا و در حال تغییر که به تنظیمات مستمر بر اساس نیازهای پروژه و تغییرات دامنه نیاز دارند. |
| DB  توسعه مبتنی بر دیتابیس | مدیریت داده بهینه‌ای را فراهم می‌کند و امکان دسترسی و تجزیه و تحلیل داده‌ها را به صورت مؤثر می‌سازد که برای برنامه‌های داده‌محور مناسب است. | ممکن است تمرکز بر منطق برنامه را کاهش داده و منجر به وابستگی بیش از حد به ساختار دیتابیس شود. | مناسب برای پروژه‌های داده‌محور، به ویژه زمانی که مدیریت و پردازش حجم بالای داده‌های ساختاریافته اهمیت دارد. |
| UCD  طراحی کاربرمحور | با تمرکز بر نیازهای کاربران، رضایت آن‌ها را افزایش داده و تجربه کاربری و دسترسی‌پذیری را بهبود می‌بخشد. | نیازمند زمان، تحقیق و منابع زیاد است و ممکن است برای همه کاربران یا برنامه‌ها مناسب نباشد. | در پروژه‌هایی که اولویت با رضایت و تجربه کاربر است، به ویژه در برنامه‌های کاربرپسند، استفاده می‌شود. |
| UDD  توسعه مبتنی بر کاربر | با هم‌راستایی بیشتر با نیازهای واقعی کاربران، رضایت کاربران را افزایش می‌دهد و به کاربران اجازه می‌دهد در تعریف ویژگی‌ها نقش بیشتری داشته باشند. | احتمال سردرگمی در اولویت‌بندی ویژگی‌ها را به همراه دارد و نیازمند هماهنگی پیوسته با کاربران است. | مناسب برای برنامه‌هایی که بازخورد کاربران اهمیت بالایی دارد، مانند نرم‌افزارهای سفارشی که برای نیازهای خاص مشتری طراحی شده‌اند. |