

methodologies

[Document subtitle]



[Date]

[Company name]

[Company address]

محمد حسن محرابی

**اسکرام : (Scrum)** یک چارچوب چابک برای مدیریت پروژه‌ها، به خصوص پروژه‌های توسعه نرم‌افزار، است که بر همکاری، بازخورد مداوم و ارائه سریع تمرکز دارد. این چارچوب به تیم‌ها کمک می‌کند تا به شکلی تکرارپذیر و فازبندی‌شده به پروژه‌ها رسیدگی کنند و به تدریج محصول نهایی را تکمیل کنند.در اسکرام، پروژه‌ها به دوره‌های زمانی مشخصی به نام **اسپرینت** (Sprint) تقسیم می‌شوند که معمولاً بین ۱ تا ۴ هفته طول می‌کشند. تیم اسکرام در هر اسپرینت تعدادی از وظایف را از لیست کارها (Product Backlog) انتخاب و بر روی آنها تمرکز می‌کند. هدف این است که در پایان هر اسپرینت، نسخه‌ای قابل ارائه از محصول تولید شود.

**مزایای متدولوژی اسکرام:**

* **پاسخگویی به تغییرات**: با چرخه‌های کوتاه و مکرر اسپرینت، تیم‌ها می‌توانند به سرعت به تغییرات نیازمندی‌ها یا اولویت‌ها پاسخ دهند.
* **شفافیت و بازخورد سریع**: جلسات منظم و بازبینی‌های دوره‌ای کمک می‌کند تا شفافیت بیشتری در پروژه باشد و بازخورد سریع دریافت شود.
* **بهبود مداوم**: جلسه بازنگری اسپرینت (Retrospective) به تیم کمک می‌کند تا پس از هر اسپرینت به دنبال بهبود روش‌های کاری خود باشد.

متدولوژی اسکرام، با وجود مزایای فراوانی که دارد، مانند هر چارچوب دیگری دارای معایب و محدودیت‌هایی است که بسته به شرایط پروژه و تیم می‌تواند چالش‌برانگیز باشد. برخی از **معایب اسکرام** عبارتند از:

**1.عدم مناسب بودن برای پروژه‌های ثابت و بسیار ساختاریافته**

* **پروژه‌های با نیازمندی‌های مشخص و ثابت**: اسکرام به دلیل طبیعت انعطاف‌پذیر و چابک خود بیشتر برای پروژه‌هایی مناسب است که نیازمندی‌ها به طور پیوسته تغییر می‌کنند. اگر پروژه‌ای نیازمندی‌های کاملاً مشخص و از پیش تعیین شده داشته باشد، رویکردهای سنتی مثل مدل آبشاری (Waterfall) ممکن است بهتر عمل کنند.

**2.نیاز به تیم‌های خودسازمانده**

* اسکرام برای تیم‌هایی طراحی شده که **خودسازمانده** هستند و می‌توانند به طور مستقل تصمیم‌گیری کنند. اگر تیمی نیاز به هدایت دقیق و نظارت مستقیم داشته باشد یا افراد تیم تجربه کافی نداشته باشند، پیاده‌سازی اسکرام ممکن است با مشکل مواجه شود.

**3. وابستگی به تعهد و همکاری تیم**

* موفقیت اسکرام به شدت به **تعهد** و **همکاری تمام اعضای تیم** وابسته است. اگر یکی از اعضا به درستی وظایف خود را انجام ندهد یا نتواند با دیگر اعضای تیم هماهنگ باشد، کل پروژه ممکن است به تأخیر بیفتد یا مشکلاتی در عملکرد تیم ایجاد شود.

**4.فشار زمانی در اسپرینت‌ها**

* اسپرینت‌های کوتاه ممکن است برای تیم‌ها فشار زمانی ایجاد کنند، مخصوصاً اگر وظایف به درستی برآورد نشده باشند. این موضوع می‌تواند منجر به **فشار زیاد** و حتی **فرسودگی شغلی** در بین اعضای تیم شود.

**5.کمبود مستندات**

* یکی از اصول اسکرام این است که تمرکز بیشتر بر **نرم‌افزارهای قابل اجرا** و **بازخوردهای سریع** باشد تا مستندات. این رویکرد در برخی پروژه‌ها که نیاز به مستندسازی دقیق دارند می‌تواند مشکل‌ساز باشد. به ویژه در پروژه‌هایی که با رگولاتوری‌ها و استانداردهای خاصی سر و کار دارند، این کمبود مستندات می‌تواند به چالش تبدیل شود.

**6.عدم قطعیت در تحویل نهایی**

* به دلیل طبیعت انعطاف‌پذیر اسکرام و تغییرات مداوم در طول اسپرینت‌ها، ممکن است **برآورد دقیقی از زمان تحویل نهایی** محصول وجود نداشته باشد. این عدم قطعیت می‌تواند برای ذی‌نفعان و مدیران پروژه که به دنبال زمان‌بندی‌های دقیق هستند چالش‌برانگیز باشد.

**7.مشکل در مدیریت وظایف بزرگ**

* اسکرام بر **وظایف کوچک و قابل تحویل در هر اسپرینت** تمرکز دارد. اما اگر پروژه دارای **وظایف بسیار بزرگ و پیچیده** باشد که به راحتی قابل تقسیم به بخش‌های کوچک‌تر نباشد، مدیریت آن در چارچوب اسکرام می‌تواند دشوار باشد.

**8.تمرکز بر نتیجه کوتاه‌مدت**

* در برخی موارد، تمرکز شدید بر ارائه‌ی محصول قابل تحویل در پایان هر اسپرینت می‌تواند باعث شود تیم به **نتایج کوتاه‌مدت** بیشتر اهمیت دهد و **اهداف بلندمدت** پروژه را نادیده بگیرد.

**9.نیاز به تجربه زیاد در ارزیابی**

* برای اینکه اسپرینت‌ها موفقیت‌آمیز باشند، تیم باید بتواند به درستی **نیازمندی‌ها و حجم کار** را ارزیابی کند. این ارزیابی نیاز به **تجربه بالا** در تخمین زمان و منابع دارد. اگر تخمین‌ها نادرست باشند، ممکن است تیم نتواند اهداف اسپرینت را به موقع به پایان برساند.

**10.چالش در سازگاری با تیم‌های بزرگ**

* اسکرام برای **تیم‌های کوچک** (معمولاً ۵ تا ۹ نفره) طراحی شده است. در پروژه‌های بزرگ‌تر که نیاز به چندین تیم موازی وجود دارد، هماهنگی بین تیم‌ها می‌تواند دشوار و پیچیده شود. چارچوب‌هایی مانند **اسکرام مقیاس‌پذیر (Scaled Scrum)** برای این موارد پیشنهاد می‌شوند، اما همچنان مدیریت و هماهنگی پروژه‌های بزرگ با اسکرام چالش‌های خاص خود را دارد.

**کاربردهای اصلی متدولوژی اسکرام:**

1. **توسعه نرم‌افزار**:
   * پرکاربردترین حوزه اسکرام است. در پروژه‌های نرم‌افزاری که نیاز به پاسخ سریع به تغییرات و تعامل مداوم با مشتری وجود دارد، اسکرام به تیم‌ها کمک می‌کند تا محصولاتی با کیفیت بالا و در زمان کوتاه تولید کنند.
2. **مدیریت پروژه‌های فناوری اطلاعات**:
   * در پروژه‌هایی مثل پیاده‌سازی سیستم‌های IT یا توسعه زیرساخت‌های جدید، اسکرام می‌تواند فرآیند توسعه را تسریع کند و تیم‌ها را برای پاسخگویی به تغییرات مداوم آماده کند.
3. **پروژه‌های تحقیق و توسعه (R&D)**:
   * تیم‌های تحقیق و توسعه اغلب نیاز دارند تا ایده‌ها و راه‌حل‌های جدید را به‌طور مستمر بررسی و ارزیابی کنند. اسکرام با چرخه‌های کوتاه اسپرینت، به تیم‌ها کمک می‌کند تا به‌سرعت نتایج را مشاهده و بازخورد دریافت کنند.
4. **توسعه محصول**:
   * در پروژه‌های توسعه محصول، به‌ویژه محصولات تکنولوژیک و دیجیتال، اسکرام برای تقسیم‌بندی و تحویل تدریجی ویژگی‌ها بسیار موثر است.
5. **مدیریت تغییرات و نوآوری در سازمان‌ها**:
   * بسیاری از شرکت‌ها از اسکرام برای مدیریت تغییرات داخلی و پروژه‌های نوآوری استفاده می‌کنند. این رویکرد به آن‌ها کمک می‌کند تا فرآیندهای سازمانی را بهبود داده و نوآوری‌ها را با سرعت بیشتری به کار بگیرند.
6. **توسعه بازی‌های ویدیویی**:
   * صنعت بازی‌های ویدیویی از اسکرام برای مدیریت توسعه بازی‌ها استفاده می‌کند. این متدولوژی به تیم‌ها کمک می‌کند تا نسخه‌های اولیه بازی را به‌سرعت آماده کنند و سپس آن را بر اساس بازخورد کاربران ارتقا دهند.

**چند نمونه از پروژه‌های بزرگ که با متدولوژی اسکرام انجام شده‌اند:**

1. : **Spotify**
   * اسپاتیفای، یکی از معروف‌ترین سرویس‌های پخش موسیقی، از اسکرام در فرآیند توسعه‌ی خود استفاده کرده است. این شرکت با تیم‌های کوچک و چابک توانسته است به‌طور مداوم ویژگی‌های جدیدی به پلتفرم خود اضافه کند و به بازخورد کاربران پاسخ دهد.
2. : **Google**
   * گوگل نیز در پروژه‌های متعددی از اسکرام استفاده کرده است. تیم‌های گوگل به‌خصوص در پروژه‌هایی مثل توسعه گوگل کروم و گوگل ادز از این متدولوژی بهره برده‌اند تا بتوانند محصولاتی نوآورانه و با کیفیت بالا ارائه دهند.
3. : **Yahoo**
   * یاهو از اسکرام در پروژه‌های بزرگ نرم‌افزاری خود استفاده کرده است. به‌عنوان یکی از شرکت‌های پیشرو در زمینه فناوری، یاهو با کمک اسکرام توانسته است محصولات دیجیتال جدیدی تولید کرده و به بهبود مداوم پلتفرم‌های خود بپردازد.
4. : **Salesforce**
   * Salesforce، یکی از بزرگ‌ترین شرکت‌های ارائه‌دهنده نرم‌افزارهای ابری (Cloud-based)، در پروژه‌های خود از اسکرام استفاده می‌کند. این متدولوژی به تیم‌های توسعه‌ی Salesforce کمک کرده تا محصولات خود را با چرخه‌های کوتاه توسعه و بهبود دهند.
5. : **BBC**
   * BBC برای توسعه پلتفرم‌های دیجیتالی خود از جمله BBC iPlayer از اسکرام استفاده کرده است. این متدولوژی به این سازمان کمک کرده تا به‌طور مستمر سرویس‌های خود را بر اساس نیازهای کاربران به‌روزرسانی کند.
6. : **Microsoft**
   * مایکروسافت در پروژه‌های متعددی از اسکرام استفاده کرده است. به عنوان مثال، تیم‌های توسعه Windows و Visual Studio از اسکرام برای تحویل منظم به‌روزرسانی‌ها و ویژگی‌های جدید استفاده کرده‌اند.

**متدولوژی (Extreme Programming) XP** یکی از روش‌های چابک (Agile) برای توسعه نرم‌افزار است که بر تعامل مستمر با مشتری، بهبود کیفیت کد، و تغییرات سریع تأکید دارد. XP به ویژه در پروژه‌هایی که نیازمند پاسخگویی سریع به تغییرات مداوم هستند و کیفیت بالای کد اهمیت زیادی دارد، به کار می‌رود.

**کاربردهای اصلی متدولوژی XP :**

1. **توسعه نرم‌افزار در محیط‌های چابک و پویا**:
   * XP برای پروژه‌هایی که نیاز به پاسخ سریع به تغییرات دارند و تعاملات روزانه با مشتری و کاربران مهم است، به کار می‌رود. این متدولوژی به تیم‌ها کمک می‌کند تا به‌طور پیوسته محصول خود را با توجه به بازخوردها بهبود دهند.
2. **پروژه‌هایی با نیاز به کیفیت بالای کد**:
   * XP تأکید بسیاری بر **کیفیت کد** و **اصول مهندسی نرم‌افزار** دارد. تکنیک‌هایی مثل **کدنویسی دوتایی** (Pair Programming) و **تست خودکار (Automated Testing)** به تیم‌ها کمک می‌کند تا کیفیت بالایی در توسعه محصول داشته باشند.
3. **توسعه سیستم‌های حیاتی و حساس**:
   * در پروژه‌هایی که سیستم‌های نرم‌افزاری حیاتی و حساس به خطا هستند، XP با استفاده از تکنیک‌های تست خودکار و بازبینی کد، تضمین می‌کند که محصول با کمترین خطا تحویل داده شود.
4. **پروژه‌های کوچک تا متوسط**:
   * XP بیشتر برای تیم‌های کوچک تا متوسط مناسب است، زیرا نیاز به تعاملات نزدیک و روزانه بین اعضای تیم و مشتری دارد.
5. **پروژه‌هایی با نیاز به تعامل مداوم با مشتری**:
   * در پروژه‌هایی که مشتریان نیاز دارند به طور مداوم نیازمندی‌های خود را تغییر دهند یا بازخورد دهند، XP کمک می‌کند تا این تغییرات به‌سرعت در فرآیند توسعه اعمال شوند.

**برخی از ویژگی‌ها و تکنیک‌های کلیدی متدولوژی: XP**

1. **تکرارهای کوتاه (Short Iterations)** :
   * XP مانند اسکرام، پروژه را به دوره‌های کوتاه توسعه تقسیم می‌کند. هر تکرار بین ۱ تا ۳ هفته طول می‌کشد و در پایان هر دوره، نسخه‌ای از نرم‌افزار که قابل استفاده است تحویل داده می‌شود.
2. **کدنویسی دوتایی (Pair Programming)** :
   * یکی از تکنیک‌های ویژه XP است که در آن دو برنامه‌نویس به‌طور همزمان روی یک سیستم کدنویسی می‌کنند. یکی از آن‌ها کد می‌نویسد و دیگری به بررسی کد و فکر کردن به راه‌حل‌های بهینه می‌پردازد. این روش به بهبود کیفیت کد و کاهش خطاها کمک می‌کند.
3. **توسعه مبتنی بر تست (Test-Driven Development)** :
   * در XP، ابتدا تست‌ها نوشته می‌شوند و سپس کدی نوشته می‌شود که آن تست‌ها را پاس کند. این روش تضمین می‌کند که کد از همان ابتدا به‌درستی تست شده و خطاهای کمتری داشته باشد.
4. **بازنگری و بهبود مستمر (Continuous Refactoring)** :
   * XP به تیم‌ها اجازه می‌دهد که به‌طور مداوم کدهای نوشته‌شده را بازبینی و بهینه کنند تا کد همیشه تمیز و قابل نگهداری باشد.
5. **ارتباطات نزدیک با مشتری**:
   * در XP، مشتری به‌طور روزانه با تیم در تعامل است و به‌طور مداوم بازخورد می‌دهد. این ارتباط نزدیک به تیم کمک می‌کند تا نیازهای مشتری را بهتر درک کرده و تغییرات سریع‌تری در نرم‌افزار اعمال کنند.
6. **تست خودکار (Automated Testing)** :
   * در XP، تست خودکار یکی از اصول کلیدی است. تیم‌ها باید سیستم‌های تست خودکاری ایجاد کنند که با هر تغییر، نرم‌افزار به‌طور خودکار تست شود تا مطمئن شوند تغییرات به درستی اعمال شده و خطاها کاهش پیدا کرده است.
7. **تحویل تدریجی (Incremental Delivery)** :
   * XP بر تحویل تدریجی نرم‌افزار تأکید دارد. به جای این که کل پروژه در انتها تحویل داده شود، تیم به‌صورت تدریجی و در هر تکرار نسخه‌ای از نرم‌افزار را تحویل می‌دهد که قابلیت استفاده داشته باشد.

**چند نمونه از پروژه‌های بزرگ که با متدولوژی XP انجام شده‌اند:**

1. : **Chrysler Comprehensive Compensation (C3) System**
   * یکی از پروژه‌های موفق که از متدولوژی XP استفاده کرد، سیستم جبران خسارت کرایسلر (Chrysler C3) بود. این پروژه که برای مدیریت پرداخت‌های کارمندان طراحی شده بود، به دلیل پیچیدگی‌های موجود و نیاز به تغییرات مداوم، با روش XP توسعه داده شد.
2. : **Ford Motor Company**
   * شرکت فورد برای توسعه برخی از سیستم‌های اطلاعاتی خود از متدولوژی XP استفاده کرده است. استفاده از XP به فورد کمک کرد تا تیم‌های توسعه به‌طور مداوم بازخورد مشتریان را دریافت کرده و به‌سرعت به نیازهای آن‌ها پاسخ دهند.
3. : **First American Real Estate Solutions**
   * این شرکت در توسعه نرم‌افزارهای مرتبط با املاک و مستغلات از XP استفاده کرده است. این متدولوژی به آن‌ها کمک کرده تا به‌طور تدریجی نرم‌افزار خود را توسعه دهند و بازخوردهای مداوم از مشتریان دریافت کنند.
4. : **Sabre Airline Solutions**
   * این شرکت که سیستم‌های نرم‌افزاری مربوط به خطوط هوایی را توسعه می‌دهد، از متدولوژی XP برای بهبود کیفیت نرم‌افزارهای خود و تسریع در تحویل پروژه‌ها استفاده کرده است.
5. : **IBM**
   * در برخی از پروژه‌های نرم‌افزاری IBM، به‌ویژه پروژه‌هایی که نیاز به تغییرات مداوم و کیفیت بالای کد داشته‌اند، از متدولوژی XP استفاده شده است. این متدولوژی به تیم‌های توسعه IBM کمک کرده است تا به‌طور مداوم کدهای خود را بازنگری و بهبود دهند.

**مزایای متدولوژی: XP**

1. **کیفیت بالای کد** به دلیل استفاده از تست خودکار و کدنویسی دوتایی.
2. **پاسخگویی سریع به تغییرات** به دلیل ارتباط نزدیک با مشتری و بازخوردهای مداوم.
3. **بهبود مستمر کد** از طریق بازنگری و بهینه‌سازی مداوم.
4. **تحویل تدریجی و قابل استفاده نرم‌افزار** در هر تکرار.

**معایب متدولوژی: XP**

1. **نیاز به تعهد بالای تیم و مشتری** به دلیل تعاملات روزانه.
2. **چالش در پیاده‌سازی برای تیم‌های بزرگ**.
3. **احتمال افزایش هزینه‌ها** به دلیل نیاز به بازنگری و بهینه‌سازی مداوم کد.
4. **مناسب نبودن برای پروژه‌های با نیازمندی‌های ثابت**.

متدولوژی XP به‌ویژه برای پروژه‌هایی که تغییرات مداوم دارند و کیفیت بالا برایشان اهمیت زیادی دارد، مناسب است.

**کانبان (Kanban)** یک روش مدیریت پروژه و تولید چابک است که بر **تصویری کردن جریان کار**، **محدود کردن کارهای در حال انجام** و **بهبود مستمر** تمرکز دارد. برخلاف متدولوژی‌هایی مانند اسکرام یا XP که چارچوب و ساختارهای مشخصی دارند، کانبان انعطاف‌پذیرتر است و بیشتر به دنبال بهبود فرآیندهای موجود در طول زمان است، نه ایجاد تغییرات بزرگ و ناگهانی.

**کاربردهای اصلی متدولوژی کانبان:**

1. **مدیریت فرآیندهای تولیدی و خدماتی**:
   * کانبان برای اولین بار توسط شرکت تویوتا در سیستم تولید خودروسازی به کار گرفته شد. این سیستم برای مدیریت موجودی و اطمینان از بهینه بودن فرآیندهای تولیدی استفاده شد. اکنون نیز در بسیاری از صنایع تولیدی برای مدیریت بهینه جریان کار به کار می‌رود.
2. **توسعه نرم‌افزار و فناوری اطلاعات**:
   * کانبان در توسعه نرم‌افزار به‌طور گسترده استفاده می‌شود. تیم‌های توسعه از آن برای مدیریت و تصویری کردن وظایف و بررسی میزان پیشرفت استفاده می‌کنند. برخلاف متدولوژی‌های دیگر، کانبان بدون نیاز به اسپرینت‌های از پیش تعریف‌شده و چرخه‌های ثابت، امکان مدیریت انعطاف‌پذیر پروژه‌ها را فراهم می‌کند.
3. **مدیریت عملیات و پشتیبانی فنی (IT Operations)** :
   * تیم‌های پشتیبانی فناوری اطلاعات و عملیات، اغلب با تعداد زیادی وظیفه و تیکت روبرو هستند که باید به سرعت پردازش شوند. کانبان به آن‌ها کمک می‌کند تا کارها را در صف نگه دارند و به ترتیب اولویت‌بندی شده به آن‌ها رسیدگی کنند.
4. **مدیریت پروژه‌های کوچک و بزرگ**:
   * کانبان می‌تواند هم برای پروژه‌های کوچک و هم برای پروژه‌های بزرگ استفاده شود. به دلیل انعطاف‌پذیری بالا، این روش به‌خوبی با نیازهای مختلف تیم‌ها و پروژه‌ها هماهنگ می‌شود.
5. **پروژه‌های با تغییرات مداوم**:
   * کانبان برای پروژه‌هایی که نیاز به انعطاف و تغییرات سریع دارند مناسب است. این متدولوژی به تیم‌ها اجازه می‌دهد تا تغییرات را به سرعت در فرآیندهای کاری خود اعمال کنند بدون اینکه نیاز به تغییرات ساختاری بزرگ باشد.
6. **بهبود مستمر در فرآیندهای کسب و کار**:
   * کانبان برای تیم‌ها و شرکت‌هایی که به دنبال بهبود مداوم فرآیندهای کسب و کار خود هستند، بسیار مناسب است. با استفاده از کانبان، تیم‌ها می‌توانند به‌طور پیوسته فرآیندهای کاری خود را بررسی کنند و آن‌ها را بهینه کنند.

**برخی از ویژگی‌ها و تکنیک‌های کلیدی متدولوژی کانبان:**

1. **تصویری کردن جریان کار (Visualizing Workflow)** :
   * یکی از اصول کلیدی کانبان، نمایش بصری وظایف در یک **بُرد کانبان** است. این بُرد شامل ستون‌هایی است که هر کدام نشان‌دهنده یک مرحله از فرآیند کار هستند. وظایف مختلف به‌صورت کارت‌هایی در بُرد کانبان نمایش داده می‌شوند که از یک ستون به ستون دیگر منتقل می‌شوند. این نمایش بصری به تیم کمک می‌کند تا به‌راحتی میزان پیشرفت و تراکم کارها را مشاهده کند.
2. **محدود کردن کارهای در حال انجام (Work In Progress Limit)** :
   * یکی از اصول مهم کانبان این است که تیم‌ها نباید هم‌زمان تعداد زیادی کار انجام دهند. به این منظور، برای هر مرحله از فرآیند محدودیتی برای تعداد کارهای در حال انجام تعریف می‌شود. این محدودیت‌ها به تیم کمک می‌کند تا بر روی تعداد محدودی از وظایف تمرکز کند و به‌سرعت آن‌ها را به پایان برساند.
3. **مدیریت جریان (Managing Flow)** :
   * کانبان بر **بهبود جریان کار** تمرکز دارد. تیم‌ها به‌طور مداوم فرآیند کاری خود را بررسی می‌کنند تا نقاطی را که ممکن است باعث تاخیر یا اتلاف وقت شوند، شناسایی و رفع کنند. به این شکل، سرعت انجام کارها بهبود می‌یابد و فرآیند کار روان‌تر می‌شود.
4. **استفاده از چرخه‌های بازبینی و بهبود مستمر (Continuous Improvement)** :
   * تیم‌هایی که از کانبان استفاده می‌کنند، به‌طور مداوم فرآیندهای کاری خود را بررسی می‌کنند و به دنبال روش‌هایی برای بهبود هستند. این بهبودها می‌تواند شامل کاهش تعداد کارهای در حال انجام، بهینه‌سازی گردش کار یا بهبود روش‌های ارتباطی تیم باشد.
5. **انعطاف‌پذیری در مدیریت زمان**:
   * برخلاف متدولوژی‌هایی مثل اسکرام که نیازمند چرخه‌های زمانی ثابت (اسپرینت) هستند، کانبان هیچ‌گونه محدودیت زمانی ندارد. تیم‌ها به‌صورت پیوسته وظایف را از طریق بُرد کانبان مدیریت می‌کنند و به محض تکمیل یک کار، کار بعدی را آغاز می‌کنند.
6. **شفافیت و ارتباطات بهتر**:
   * به دلیل نمایش بصری جریان کار، تمام اعضای تیم و حتی مشتریان می‌توانند به‌راحتی پیشرفت کارها را ببینند و در جریان وضعیت پروژه قرار بگیرند. این شفافیت باعث بهبود ارتباطات بین اعضای تیم و مشتریان می‌شود.

**چند نمونه از پروژه‌های بزرگ که با متدولوژی کانبان انجام شده‌اند:**

1. **Spotify** :
   * اسپاتیفای، مشابه اسکرام، از کانبان برای مدیریت فرآیندهای توسعه نرم‌افزار و بهبود مستمر استفاده می‌کند. تیم‌های توسعه اسپاتیفای از بُردهای کانبان برای مدیریت وظایف و نمایش پیشرفت پروژه استفاده می‌کنند.
2. **Toyota** :
   * کانبان از ابتدا در شرکت تویوتا به‌عنوان بخشی از سیستم تولید ناب (Lean Manufacturing) استفاده شده است. این روش به تویوتا کمک کرد تا فرآیندهای تولید خود را بهینه کند و به یکی از پیشروهای جهانی در صنعت خودروسازی تبدیل شود.
3. **Microsoft** :
   * مایکروسافت در تیم‌های توسعه ویژوال استودیو (Visual Studio) از کانبان استفاده کرده است. این شرکت از کانبان برای مدیریت وظایف و بهبود فرآیندهای توسعه نرم‌افزار استفاده کرده تا بتواند سریع‌تر به تغییرات پاسخ دهد.
4. **BBC Worldwide** :
   * BBC برای مدیریت پروژه‌های دیجیتالی خود، از جمله توسعه سیستم‌های رسانه‌ای و برنامه‌های کاربردی، از کانبان استفاده کرده است. این روش به BBC کمک کرده تا فرآیندهای کاری را به‌طور پیوسته بهبود دهد و پروژه‌ها را به‌موقع تحویل دهد.
5. **Riot Games** :
   * Riot Games، سازنده بازی معروف League of Legends، از کانبان برای مدیریت تیم‌های توسعه بازی استفاده کرده است. این روش به آن‌ها کمک کرده تا به‌طور پیوسته ویژگی‌های جدید بازی را توسعه دهند و به سرعت به بازخوردهای کاربران پاسخ دهند.

**مزایای متدولوژی کانبان:**

1. **انعطاف‌پذیری بالا**: کانبان بسیار انعطاف‌پذیر است و می‌تواند به راحتی با هر فرآیند کاری موجود تطبیق یابد. به‌ویژه برای تیم‌هایی که به دنبال تغییرات تدریجی هستند، مناسب است.
2. **بهبود مستمر**: کانبان بر بهبود مداوم تأکید دارد و تیم‌ها را تشویق می‌کند که همواره به دنبال راه‌های بهبود فرآیندهای خود باشند.
3. **شفافیت و همکاری بهتر**: به دلیل تصویری کردن فرآیند کار، همه اعضای تیم و ذی‌نفعان می‌توانند وضعیت پروژه را مشاهده کنند. این به بهبود همکاری و شفافیت کمک می‌کند.
4. **تمرکز بر تکمیل وظایف**: محدود کردن کارهای در حال انجام باعث می‌شود تیم‌ها به‌جای انجام چند کار هم‌زمان، تمرکز بیشتری بر تکمیل وظایف خود داشته باشند.
5. **کاهش هدررفت زمان و منابع**: با بررسی مداوم فرآیندها، نقاطی که باعث هدررفت زمان و منابع می‌شوند شناسایی و بهبود می‌یابند.
6. **مدیریت بهتر وظایف پیش‌بینی‌نشده**: برخلاف روش‌هایی مثل اسکرام که نیازمند برنامه‌ریزی دقیق اسپرینت‌ها هستند، کانبان به‌راحتی می‌تواند وظایف جدید و غیرمنتظره را مدیریت کند.

**معایب متدولوژی کانبان:**

1. **نیاز به مدیریت دقیق**: کانبان نیازمند مدیریت دقیق جریان کار است تا از انباشته شدن کارها در یک مرحله جلوگیری شود. اگر تیم نتواند به‌درستی وظایف را مدیریت کند، فرآیند بهینه‌سازی به درستی پیش نخواهد رفت.
2. **عدم ساختار زمان‌بندی مشخص**: در کانبان هیچ محدودیت زمانی مشخصی برای تحویل کار وجود ندارد، که ممکن است برای پروژه‌هایی که نیاز به تاریخ‌های تحویل دقیق دارند مشکل‌ساز باشد.
3. **مناسب نبودن برای تیم‌های غیرخودگردان**: کانبان بهترین کارایی را در تیم‌هایی دارد که می‌توانند به‌صورت مستقل و خودگردان کار کنند. تیم‌هایی که نیاز به نظارت و مدیریت مداوم دارند، ممکن است در اجرای کانبان با مشکل مواجه شوند.

**کانبان** یکی از متدولوژی‌های بسیار انعطاف‌پذیر و قدرتمند برای مدیریت پروژه‌ها است که به تیم‌ها اجازه می‌دهد با سرعت و کارایی بیشتری وظایف خود را مدیریت کرده و همواره به دنبال بهبود مستمر باشند.

**متدولوژی (Unified Process) UP** یک چارچوب توسعه نرم‌افزار است که به‌ویژه برای پروژه‌های بزرگ و پیچیده طراحی شده است. UP به‌عنوان یک متدولوژی فرایند (Process Framework) شناخته می‌شود که بر پایه چهار فاز اصلی: **تعیین نیازمندی‌ها، تحلیل، طراحی و پیاده‌سازی** بنا شده است. این روش بر اساس اصول چابک (Agile) و آبشاری (Waterfall) کار می‌کند و به تیم‌ها کمک می‌کند تا به‌طور سیستماتیک و تکرارشونده به توسعه نرم‌افزار بپردازند.

**ویژگی‌ها و مراحل اصلی متدولوژی UP :**

1. **فازهای توسعه**:
   * **فاز پیش‌نویس (Inception)** : در این فاز، نیازهای اولیه پروژه شناسایی و تعریف می‌شوند. اهداف پروژه، محدوده، ذینفعان و نیازمندی‌های کلیدی در این مرحله مشخص می‌شود. در این فاز، مطالعه امکان‌سنجی و ارزیابی ریسک‌ها نیز انجام می‌شود.
   * **فاز اقتدار (Elaboration)** : در این فاز، جزئیات بیشتر نیازمندی‌ها و معماری سیستم بررسی و توسعه می‌شود. تیم به شناسایی نیازمندی‌های دقیق‌تری می‌پردازد و طراحی سیستم به‌طور کامل‌تری مورد بررسی قرار می‌گیرد. این فاز معمولاً شامل طراحی‌های اصلی و تجزیه و تحلیل‌های فنی است.
   * **فاز ساخت (Construction)** : در این فاز، توسعه واقعی نرم‌افزار آغاز می‌شود. تیم به پیاده‌سازی و تست نرم‌افزار می‌پردازد. کد نویسی و تست‌های واحد (Unit Testing) در این فاز انجام می‌شود و ویژگی‌های جدید به‌طور تدریجی اضافه می‌شوند.
   * **فاز انتقال (Transition)** : در این فاز، نرم‌افزار به مشتری تحویل داده می‌شود. این شامل انتقال داده، آموزش کاربران و پشتیبانی اولیه است. هدف از این فاز اطمینان از این است که نرم‌افزار به‌درستی
   * مستقر شده و کاربران بتوانند به‌راحتی از آن استفاده کنند.
2. **تکرارها و سطوح**:
   * UP بر اساس تکرار (Iteration) و تکرارهای متوالی کار می‌کند. هر فاز ممکن است چندین بار تکرار شود تا به نتایج مطلوب دست یابد. این تکرارها به تیم اجازه می‌دهد تا بازخورد دریافت کنند و به‌طور مداوم پروژه را بهبود دهند.
3. **مدل‌سازی و مستندسازی**:
   * UP بر روی مستندسازی دقیق و مدل‌سازی نرم‌افزار تأکید دارد. تیم‌ها باید نیازمندی‌ها، طراحی‌ها و مستندات فنی را به‌دقت ثبت کنند تا اطمینان حاصل کنند که همه اعضای تیم و ذینفعان درک روشنی از پروژه دارند.
4. **استفاده از UML (Unified Modeling Language)** :
   * UP به‌طور معمول از زبان مدل‌سازی UML برای ترسیم و توصیف نیازمندی‌ها، طراحی‌ها و فرآیندها استفاده می‌کند. UML به تیم‌ها کمک می‌کند تا الگوهای موجود در سیستم را به‌طور بصری نمایش دهند.
5. **رویکرد چابک**:
   * UP در تلاش است تا به بهترین شیوه‌های متدولوژی‌های چابک نزدیک شود. این رویکرد به تیم‌ها کمک می‌کند تا از فرآیندهای خود به‌طور مستمر بهبود دهند و به نیازهای متغیر مشتریان پاسخ دهند.

**مزایای متدولوژی UP :**

1. **ساختار واضح و جامع**:
   * UP یک چارچوب ساختاریافته و منظم برای مدیریت فرآیند توسعه نرم‌افزار فراهم می‌کند که می‌تواند به تیم‌ها کمک کند تا پروژه‌های پیچیده را به‌طور موثری مدیریت کنند.
2. **تاکید بر نیازمندی‌ها**:
   * UP بر روی شناسایی و تحلیل دقیق نیازمندی‌ها تأکید دارد که به تیم کمک می‌کند تا درک بهتری از نیازهای مشتریان و کاربران پیدا کند.
3. **تکرار و بازخورد**:
   * با استفاده از تکرارهای متوالی، تیم‌ها می‌توانند به‌سرعت بازخورد دریافت کنند و پروژه را بر اساس نیازهای جدید به‌روز کنند.
4. **استفاده از استانداردها**:
   * UP به‌طور معمول از UML و دیگر استانداردهای صنعتی استفاده می‌کند که به تیم‌ها کمک می‌کند تا از الگوهای موفق و بهترین شیوه‌ها استفاده کنند.
5. **پشتیبانی از تست و تضمین کیفیت**:
   * UP بر روی تست و تضمین کیفیت نرم‌افزار تأکید دارد و به تیم‌ها کمک می‌کند تا محصول نهایی با کیفیت بالا و بدون نقص ارائه دهند.

**معایب متدولوژی UP :**

1. **پیچیدگی در پیاده‌سازی**:
   * UP به‌دلیل ساختار پیچیده و فرایندهای دقیق ممکن است برای تیم‌های کوچک یا پروژه‌های ساده مناسب نباشد.
2. **نیاز به مستندسازی زیاد**:
   * تمرکز بر مستندسازی می‌تواند زمان‌بر باشد و ممکن است باعث افزایش بار کاری تیم شود.
3. **نیاز به تخصص در UML** :
   * استفاده از UML نیاز به تخصص در این زبان مدل‌سازی دارد، که ممکن است برای همه اعضای تیم آشنا نباشند.
4. **پاسخگویی به تغییرات**:
   * اگرچه UP به‌عنوان یک متدولوژی تکرارشونده طراحی شده، اما در مقایسه با روش‌های چابک‌تر مانند اسکرام، ممکن است نسبت به تغییرات کمتری پاسخگو باشد.

**کاربردهای متدولوژی UP :**

1. **پروژه‌های بزرگ و پیچیده**:
   * UP به‌ویژه برای پروژه‌های بزرگ و پیچیده مناسب است که نیاز به مدیریت دقیق و ساختار مشخص دارند.
2. **توسعه نرم‌افزارهای سازمانی**:
   * برای توسعه نرم‌افزارهای سازمانی و سیستم‌های اطلاعاتی که نیاز به تحلیل دقیق و معماری‌های پیچیده دارند، UP می‌تواند بسیار مفید باشد.
3. **پروژه‌هایی با نیازمندی‌های متغیر**:
   * UP به‌خوبی می‌تواند در پروژه‌هایی که نیاز به بازخورد مداوم و تغییرات دارند، مورد استفاده قرار گیرد.
4. **توسعه محصولات فناورانه**:
   * برای توسعه محصولات جدید و فناوری‌های نوین، UP می‌تواند به تیم‌ها کمک کند تا مراحل مختلف توسعه را به‌طور منظم و کنترل‌شده پیش ببرند.

**نمونه‌هایی از پروژه‌هایی که با متدولوژی UP انجام شده‌اند:**

1. **پروژه‌های سازمانی بزرگ**:
   * بسیاری از شرکت‌های بزرگ فناوری اطلاعات و نرم‌افزاری از UP برای توسعه سیستم‌های سازمانی خود استفاده کرده‌اند، مانند IBM و Oracle.
2. **سیستم‌های اطلاعاتی در بانک‌ها**:
   * UP به‌طور گسترده در پروژه‌های توسعه سیستم‌های اطلاعاتی در صنعت بانکداری استفاده شده است. بانک‌ها به‌دلیل پیچیدگی نیازمندی‌های خود از این متدولوژی بهره می‌برند.
3. **توسعه نرم‌افزارهای دولتی**:
   * برخی از پروژه‌های نرم‌افزاری دولتی که نیاز به شفافیت و دقت بالا دارند، از UP برای توسعه و پیاده‌سازی سیستم‌ها استفاده می‌کنند.
4. **پروژه‌های تحقیق و توسعه (R&D)** :
   * UP در پروژه‌های تحقیق و توسعه برای تجزیه و تحلیل و پیاده‌سازی نتایج تحقیقات نیز به کار می‌رود.

متدولوژی **UP** با ساختار و فرآیندهای منظم خود، به‌ویژه برای پروژه‌های بزرگ و پیچیده، یک گزینه مناسب برای تیم‌های توسعه نرم‌افزار است. این روش به تیم‌ها کمک می‌کند تا با شناسایی دقیق نیازمندی‌ها و تکرار در توسعه، محصولاتی با کیفیت بالا و مطابق با خواسته‌های مشتری تولید کنند.

**AUP (Agile Unified Process)** یا **فرآیند متحد چابک** یک نسخه‌ی سبک‌تر و انعطاف‌پذیرتر از **RUP (Rational Unified Process)** است که بر اساس اصول چابکی (Agile) طراحی شده است. AUP تلاش می‌کند بهترین ویژگی‌های RUP را با اصول و ارزش‌های Agile ترکیب کند تا فرآیندی ساده‌تر، کم‌تر پیچیده، و سریع‌تر برای توسعه نرم‌افزار فراهم کند. در AUP، توجه به **ساده‌سازی** و **تکرارهای کوتاه** و بهبود پیوسته وجود دارد و به‌ویژه برای تیم‌های کوچک یا متوسط مناسب است.

**ساختار و مراحل متدولوژی AUP :**

AUP، مانند RUP، چهار فاز اصلی دارد، اما سادگی و تمرکز بیشتری بر روی ارزش‌های چابک دارد. این چهار فاز عبارتند از:

1. **فاز آغازین (Inception Phase)** :
   * این فاز بر تعریف دامنه پروژه، جمع‌آوری نیازمندی‌ها، و تعیین اهداف پروژه تمرکز دارد. مشابه RUP، اما در AUP تلاش می‌شود تا به سرعت به نیازمندی‌های اصلی دست پیدا کرده و به جای مستندسازی دقیق، به تعامل با مشتریان پرداخته شود. تصمیمات کلیدی و اساسی در این فاز گرفته می‌شود و اهداف پروژه مشخص می‌گردد.
2. **فاز تولید (Elaboration Phase)** :
   * در این فاز، تحلیل دقیق‌تر و طراحی معماری انجام می‌شود. تمرکز بیشتر روی ساختاردهی به نیازمندی‌ها و برنامه‌ریزی بلندمدت برای پروژه است. در AUP، این فاز شامل مدل‌سازی سیستم و بررسی چالش‌ها و ریسک‌های بالقوه است، اما به جای تمرکز بیش از حد بر مستندات، به ایجاد کدهای عملی و آزمایش‌های ابتدایی پرداخته می‌شود.
3. **فاز ساخت (Construction Phase)** :
   * در این فاز، تیم‌های توسعه شروع به کدنویسی و پیاده‌سازی سیستم می‌کنند. این مرحله شامل چندین تکرار است و هر تکرار یک نسخه افزایشی از نرم‌افزار ارائه می‌دهد. AUP مانند RUP تکراری است، اما تلاش می‌کند تکرارها را کوتاه‌تر و انعطاف‌پذیرتر کند و به بازخورد سریع‌تر از مشتریان پرداخته شود. این فاز شامل تست و ارزیابی‌های مستمر نرم‌افزار است.
4. **فاز انتقال (Transition Phase)** :
   * در این فاز، سیستم برای انتقال به محیط عملیاتی آماده می‌شود. این شامل استقرار نرم‌افزار، آموزش کاربران، و جمع‌آوری بازخورد نهایی از کاربران نهایی است. در AUP این فاز بر تعامل و همکاری با مشتریان برای اطمینان از پذیرش موفقیت‌آمیز سیستم تمرکز دارد.

**ویژگی‌ها و تکنیک‌های کلیدی AUP :**

1. **تکراری و افزایشی (Iterative and Incremental)** :
   * AUP از یک رویکرد تکراری و افزایشی پیروی می‌کند که پروژه به بخش‌های کوچکتری تقسیم شده و هر بخش به‌طور مستقل توسعه داده می‌شود. تیم‌ها از طریق چندین تکرار نسخه‌های به‌روز شده از نرم‌افزار را به مشتری ارائه می‌دهند.
2. **چابکی (Agility)** :
   * یکی از اصول کلیدی AUP، ساده‌سازی فرآیندها و تمرکز بر تعامل و همکاری بیشتر با مشتری و اعضای تیم است. مستندسازی و فرآیندهای رسمی در حد نیاز و بر اساس اصول چابک کاهش یافته‌اند و تیم‌ها تشویق می‌شوند به‌طور سریع‌تر و انعطاف‌پذیرتر عمل کنند.
3. **تمرکز بر مدل‌سازی ساده**:
   * AUP تأکید می‌کند که مدل‌سازی باید ساده و کافی باشد. این یعنی تیم‌ها نباید زمان زیادی را صرف مستندات و مدل‌های پیچیده کنند، بلکه باید مستقیماً به کدنویسی و پیاده‌سازی بپردازند.
4. **بهبود مستمر (Continuous Improvement)** :
   * AUP بر بهبود مستمر در هر تکرار تمرکز دارد. هر تکرار فرصتی است برای بازبینی و اصلاح فرآیندها و نرم‌افزار. این امر به تیم کمک می‌کند تا نرم‌افزاری با کیفیت بهتر تولید کند.
5. **تست و بازخورد مداوم**:
   * یکی از نکات کلیدی AUP، انجام تست‌های مداوم در هر تکرار است. به این ترتیب، خطاها و مشکلات سریع‌تر شناسایی می‌شوند و تیم می‌تواند بر اساس بازخوردها، تغییرات لازم را اعمال کند.
6. **تمرکز بر همکاری و تعامل**:
   * AUP تأکید زیادی بر همکاری و تعامل بین اعضای تیم و مشتری دارد. در طول پروژه، مشتری به‌طور فعال در فرآیندهای توسعه درگیر است و بازخوردهای مداوم ارائه می‌دهد.
7. **تمرکز بر مشتری**:
   * مشتریان در طول چرخه زندگی پروژه همواره درگیر هستند و بازخوردهای آنان از اهمیت بالایی برخوردار است. تیم‌ها باید به تغییرات نیازمندی‌ها و درخواست‌های جدید به سرعت پاسخ دهند.

**کاربردهای متدولوژی AUP :**

1. **پروژه‌های کوچک و متوسط**:
   * AUP به دلیل سادگی و کاهش مستندات، برای پروژه‌های کوچک و متوسط مناسب است. این پروژه‌ها معمولاً نیاز به فرآیندهای چابک دارند و نیازی به ساختارهای پیچیده ندارند.
2. **تیم‌های کوچک توسعه**:
   * تیم‌های کوچکی که به دنبال یک متدولوژی انعطاف‌پذیر و ساده هستند، می‌توانند از AUP بهره‌مند شوند. این متدولوژی به تیم‌ها کمک می‌کند بدون پیچیدگی‌های اضافی، به تولید نرم‌افزار بپردازند.
3. **پروژه‌های با تغییرات مداوم**:
   * برای پروژه‌هایی که نیازمند تغییرات مداوم و سریع هستند، AUP یک متدولوژی مناسب است. با استفاده از تکرارهای کوتاه و تعاملات مداوم با مشتری، این متدولوژی به تیم‌ها اجازه می‌دهد به سرعت به تغییرات پاسخ دهند.
4. **پروژه‌های توسعه نرم‌افزار تجاری**:
   * AUP به دلیل تکیه بر مشتری و تمرکز بر تولید نسخه‌های کاربردی نرم‌افزار، برای توسعه سیستم‌های نرم‌افزاری تجاری که نیازمند تعاملات مداوم با کاربران هستند، بسیار مناسب است.
5. **پروژه‌های تیم‌های دورکار**:
   * برای تیم‌هایی که به‌صورت از راه دور کار می‌کنند، AUP می‌تواند یک گزینه مناسب باشد. چون فرآیندها ساده و متمرکز هستند و ارتباطات سریع و منعطف با مشتری اهمیت دارد.

**نمونه‌های پروژه‌های اجرا شده با AUP :**

1. **پروژه‌های توسعه نرم‌افزار کوچک در شرکت‌های استارتاپی**:
   * شرکت‌های نوپا و استارتاپ‌ها که نیاز به تحویل سریع محصولات دارند و به دنبال فرآیندهای ساده و انعطاف‌پذیر هستند، از AUP برای مدیریت توسعه نرم‌افزارهای خود بهره می‌برند.
2. **توسعه سیستم‌های مبتنی بر وب**:
   * بسیاری از شرکت‌های توسعه‌دهنده وب که به دنبال ساخت محصولات کوچک و متوسط هستند، از AUP استفاده می‌کنند. به دلیل اینکه تکرارهای سریع و تعامل مداوم با مشتری برای بهبود محصولات اهمیت دارد.
3. **پروژه‌های نرم‌افزاری داخلی شرکت‌ها**:
   * بسیاری از شرکت‌ها برای توسعه سیستم‌های داخلی خود، از AUP استفاده می‌کنند تا بتوانند به سرعت نرم‌افزارهای مورد نیاز خود را تولید کنند و در عین حال به‌طور مستمر آن‌ها را بهبود دهند.

**مزایای متدولوژی AUP :**

1. **انعطاف‌پذیری بالا**: AUP انعطاف‌پذیرتر از RUP است و امکان تغییرات سریع در فرآیندهای کاری و نیازمندی‌ها را فراهم می‌کند.
2. **کاهش مستندات و تمرکز بر کدنویسی**: برخلاف RUP، AUP تأکید کمتری بر مستندات سنگین دارد و بیشتر بر کدنویسی و تحویل سریع نرم‌افزار تمرکز دارد.
3. **تکرارهای کوتاه‌تر**: تکرارهای کوتاه در AUP به تیم‌ها کمک می‌کند تا به‌طور مداوم بازخورد دریافت کنند و بهبود‌های لازم را اعمال کنند.
4. **تمرکز بر مشتری و تعاملات مستمر**: AUP مشتری را در طول چرخه توسعه دخیل می‌کند و این امر به بهبود تعاملات و تطبیق سریع‌تر با نیازهای مشتری کمک می‌کند.
5. **ساده‌سازی فرآیندها**: AUP تلاش می‌کند فرآیندهای پیچیده RUP را ساده‌تر کند و تیم‌ها را به سمت یک رویکرد چابک‌تر و کارآمدتر هدایت کند.

**معایب متدولوژی AUP :**

1. **هنوز نیاز به برخی مستندات دارد**: هرچند مستندات در AUP کاهش یافته‌اند، اما همچنان نیاز به برخی مستندات وجود دارد که ممکن است برای پروژه‌های خیلی کوچک ضروری نباشد.
2. **پیچیدگی نسبی برای تیم‌های کوچک**: برای تیم‌های خیلی کوچک یا پروژه‌های بسیار ساده، AUP ممکن است بیش از حد پیچیده به‌نظر بیاید.
3. **نیاز به تیم‌های باتجربه**: تیم‌های کم تجربه ممکن است در پیاده‌سازی AUP به مشکلاتی بربخورند، چرا که نیاز به درک خوبی از اصول چابک و رویکردهای متحد دارد.
4. **نیاز به بازخورد مستمر**: موفقیت در AUP وابسته به دریافت مداوم بازخورد از مشتریان است که ممکن است در برخی پروژه‌ها یا محیط‌ها عملی نباشد.

**AUP (Agile Unified Process)** ترکیبی از روش‌های چابک و RUP است که برای تیم‌هایی که به دنبال انعطاف‌پذیری بیشتر هستند، مناسب است. این متدولوژی به تیم‌ها اجازه می‌دهد با ساده‌سازی فرآیندها و کاهش

**DSDM (Dynamic Systems Development Method)** یک متدولوژی چابک برای توسعه سیستم‌ها و نرم‌افزارها است که تمرکز زیادی بر **تکمیل پروژه‌ها در زمان و بودجه مقرر** دارد. این متدولوژی ابتدا در دهه ۱۹۹۰ معرفی شد و با گذشت زمان توسعه یافت تا با ارزش‌ها و اصول **Agile** هماهنگ‌تر شود. DSDM تلاش می‌کند تعادلی میان ارائه نرم‌افزار با کیفیت بالا و پاسخگویی به نیازهای متغیر مشتریان برقرار کند.

**اصول و ویژگی‌های کلیدی DSDM :**

DSDM بر ۹ اصل اساسی تأکید دارد که تمامی تصمیمات و فرآیندهای آن را هدایت می‌کنند:

1. **مشارکت فعال کاربران**:
   * تعامل مستقیم و فعال با کاربران در طول چرخه توسعه، از اصول کلیدی DSDM است. کاربران باید همواره در فرآیند توسعه درگیر باشند تا بازخوردهای مهم را ارائه کنند.
2. **تحویل مکرر و زودهنگام**:
   * DSDM تأکید می‌کند که سیستم باید به صورت مکرر و زودهنگام تحویل داده شود تا کاربران بتوانند به‌سرعت بازخورد دهند و تیم توسعه اصلاحات لازم را انجام دهد.
3. **انتظارات مشخص**:
   * در DSDM، تمام ذی‌نفعان باید نیازمندی‌ها و اولویت‌های خود را به‌وضوح مشخص کنند. تیم توسعه نیز باید این انتظارات را به درستی درک کند و در طول پروژه بر روی آن تمرکز کند.
4. **کیفیت تضمین شده**:
   * DSDM بر کیفیت تمرکز ویژه‌ای دارد. کیفیت نباید فدای سرعت یا بودجه شود و باید همواره از فرآیندهای مناسب برای تضمین کیفیت استفاده کرد.
5. **تأکید بر تحویل محصول قابل استفاده**:
   * هر تکرار باید منجر به تحویل بخشی از سیستم شود که قابل استفاده و مفید برای کاربران باشد، حتی اگر تمام ویژگی‌ها در ابتدا موجود نباشند.
6. **کنترل و مدیریت فعال**:
   * در DSDM، پروژه‌ها باید به‌طور فعال مدیریت شوند. برنامه‌ریزی، کنترل و ردیابی مستمر پروژه اهمیت ویژه‌ای دارد تا پروژه‌ها در زمان و بودجه مقرر تکمیل شوند.
7. **توسعه افزایشی**:
   * توسعه سیستم‌ها باید به‌صورت افزایشی و در چندین تکرار انجام شود. هر تکرار به بهبود و توسعه نرم‌افزار قبلی کمک می‌کند و نسخه‌های جدیدتر بهتر و کامل‌تر می‌شوند.
8. **تصمیم‌گیری سریع**:
   * در DSDM، تیم‌ها باید به سرعت تصمیم‌گیری کنند و از هرگونه تأخیر در تصمیم‌گیری که می‌تواند روند پروژه را کند کند، جلوگیری شود.
9. **همکاری نزدیک بین تیم‌ها**:
   * DSDM بر همکاری نزدیک بین تمام اعضای تیم توسعه و کاربران تأکید دارد تا همه افراد باهم همکاری کرده و به اهداف مشترک دست یابند.

**مراحل DSDM :**

متدولوژی DSDM دارای چهار فاز اصلی است که به تیم‌های توسعه کمک می‌کند تا پروژه‌ها را به‌طور مؤثر و سازمان‌یافته مدیریت کنند:

1. **فاز تحقیق و امکان‌سنجی (Feasibility and Foundations)** :
   * در این فاز، امکان‌پذیری پروژه بررسی می‌شود و اهداف و نیازمندی‌های کلیدی شناسایی می‌شوند. تیم باید یک بررسی کلی انجام دهد تا مطمئن شود که پروژه از لحاظ فنی و اقتصادی قابل انجام است.
2. **فاز بنیادها (Foundations Phase)** :
   * در این فاز، جزئیات بیشتری از نیازمندی‌ها، معماری و طرح پروژه مشخص می‌شود. طرح کلی پروژه و اهداف آن باید به‌وضوح تعریف شود. تصمیمات کلیدی درباره معماری سیستم، منابع مورد نیاز، و ریسک‌های احتمالی در این مرحله گرفته می‌شود.
3. **فاز توسعه افزایشی (Evolutionary Development)** :
   * این فاز شامل تکرارهای مداوم توسعه نرم‌افزار است. در هر تکرار، بخشی از سیستم پیاده‌سازی و تحویل داده می‌شود. تیم توسعه بازخوردهای کاربران را دریافت کرده و در تکرارهای بعدی سیستم را بهبود می‌بخشد.
4. **فاز تحویل (Deployment Phase)** :
   * پس از تکمیل تکرارهای توسعه، نرم‌افزار نهایی آماده برای استقرار است. تیم توسعه نرم‌افزار را به محیط عملیاتی منتقل کرده و کاربران نهایی را آموزش می‌دهد.

**کاربردهای DSDM :**

DSDM به‌ویژه برای پروژه‌هایی مناسب است که نیاز به تحویل سریع و تعامل مداوم با مشتریان دارند. همچنین، این متدولوژی برای پروژه‌های بزرگ‌تر و پیچیده‌تر نیز مفید است که نیاز به **مدیریت دقیق و کنترل کیفیت** دارند. برخی از کاربردهای آن شامل موارد زیر می‌شود:

1. **پروژه‌های نرم‌افزار تجاری**:
   * DSDM برای پروژه‌های تجاری که نیازمند تعامل مستمر با مشتریان و تحویل سریع محصول هستند، بسیار مناسب است. این پروژه‌ها ممکن است شامل توسعه سیستم‌های مدیریت مشتری (CRM) یا پلتفرم‌های تجارت الکترونیک باشند.
2. **پروژه‌های دولتی و عمومی**:
   * پروژه‌های دولتی که نیازمند کنترل دقیق منابع و بودجه هستند، می‌توانند از DSDM بهره‌مند شوند. این متدولوژی تضمین می‌کند که پروژه‌ها در زمان و بودجه مشخص به اتمام برسند.
3. **پروژه‌های بزرگ سازمانی**:
   * DSDM در پروژه‌های بزرگ و پیچیده سازمانی که نیاز به مدیریت دقیق دارند، مفید است. این پروژه‌ها ممکن است شامل سیستم‌های ERP (برنامه‌ریزی منابع سازمانی) یا سیستم‌های بانکی باشند.

**نمونه پروژه‌های انجام‌شده با DSDM :**

1. **سیستم‌های مالی و بانکی**:
   * بسیاری از بانک‌ها و مؤسسات مالی از DSDM برای توسعه سیستم‌های بانکی خود استفاده می‌کنند تا از کنترل دقیق فرآیندهای توسعه و تضمین کیفیت سیستم‌ها اطمینان حاصل کنند.
2. **پروژه‌های دولتی در انگلستان**:
   * DSDM در پروژه‌های بزرگ دولتی، به‌ویژه در انگلستان، مورد استفاده قرار گرفته است. یکی از نمونه‌های معروف، پروژه‌های مربوط به سیستم‌های خدمات بهداشتی و درمانی انگلستان است.
3. **پروژه‌های شرکت‌های بزرگ**:
   * بسیاری از شرکت‌های بزرگ تجاری از DSDM برای توسعه سیستم‌های داخلی و خارجی خود استفاده کرده‌اند، از جمله توسعه سیستم‌های مالی، مدیریت مشتریان، و پلتفرم‌های تجارت الکترونیک.

**مزایای DSDM :**

1. **تحویل سریع و مکرر**:
   * DSDM به تیم‌ها کمک می‌کند تا به‌طور سریع نرم‌افزارهای قابل استفاده تحویل دهند و در هر تکرار بازخورد مشتری را دریافت کنند.
2. **مشارکت فعال کاربران**:
   * یکی از بزرگ‌ترین مزایای DSDM، مشارکت فعال کاربران در فرآیند توسعه است که باعث می‌شود محصول نهایی بهتر با نیازهای مشتریان تطبیق داشته باشد.
3. **مدیریت زمان و بودجه**:
   * DSDM به تیم‌ها اجازه می‌دهد پروژه‌ها را با کنترل دقیق زمان و بودجه به پایان برسانند، که این امر در پروژه‌های بزرگ و حساس اهمیت زیادی دارد.
4. **تمرکز بر کیفیت**:
   * DSDM تضمین می‌کند که کیفیت نرم‌افزار فدای سرعت یا بودجه نشود. از طریق فرآیندهای مستمر بازبینی و تست، کیفیت پروژه‌ها حفظ می‌شود.
5. **انعطاف‌پذیری در مواجهه با تغییرات**:
   * DSDM به دلیل ساختار تکراری و تعامل مداوم با مشتریان، به تیم‌ها اجازه می‌دهد به‌سرعت به تغییرات نیازمندی‌ها پاسخ دهند.

**معایب DSDM :**

1. **نیاز به تعامل مداوم کاربران**:
   * موفقیت DSDM به مشارکت فعال و مداوم کاربران بستگی دارد. اگر کاربران به‌طور فعال در فرآیند توسعه مشارکت نکنند، موفقیت پروژه ممکن است تحت تأثیر قرار بگیرد.
2. **پیچیدگی نسبی برای پروژه‌های کوچک**:
   * برای پروژه‌های کوچک یا ساده، DSDM ممکن است پیچیده به‌نظر برسد و استفاده از آن ممکن است بیش از حد نیاز باشد.
3. **نیاز به تیم‌های باتجربه**:
   * تیم‌هایی که از DSDM استفاده می‌کنند، باید تجربه کافی در مدیریت پروژه‌ها و تکنیک‌های چابک داشته باشند. تیم‌های کم تجربه ممکن است با چالش‌هایی روبرو شوند.
4. **مستندسازی کافی لازم است**:
   * برخلاف برخی از متدولوژی‌های چابک که مستندسازی را کاهش می‌دهند، DSDM نیاز به مستندسازی کافی دارد، که ممکن است زمان‌بر باشد.

**DSDM (Dynamic Systems Development Method)** یک متدولوژی قدرتمند و ساختارمند برای مدیریت پروژه‌های نرم‌افزاری است که به‌ویژه برای پروژه‌های بزرگ و پیچیده مناسب است. DSDM به تیم‌ها کمک می‌کند پروژه‌ها را با تحویل مکرر، کنترل دقیق منابع و بودجه، و تضمین کیفیت بالا مدیریت کنند و هم‌زمان به نیازهای متغیر مشتریان پاسخ دهند.

**DevOps** یک متدولوژی یا فرهنگ توسعه نرم‌افزار است که بر **یکپارچگی توسعه (Development) و عملیات (Operations)** تمرکز دارد. هدف اصلی DevOps ایجاد همکاری نزدیک بین تیم‌های توسعه‌دهندگان و تیم‌های عملیات (یا فناوری اطلاعات) است تا فرآیند توسعه و استقرار نرم‌افزار را بهبود بخشد، سریع‌تر و با کیفیت‌تر انجام دهد. DevOps اصول چابکی (Agile) و **تحویل مداوم (Continuous Delivery)** را به کار می‌گیرد تا سرعت انتشار، تست، و به‌روزرسانی نرم‌افزارها را افزایش دهد.

**فلسفه و اهداف DevOps :**

DevOps تلاش می‌کند مشکلات و چالش‌هایی که اغلب در روش‌های سنتی بین تیم‌های توسعه و عملیات وجود دارد را از بین ببرد. در روش‌های سنتی، تیم‌های توسعه نرم‌افزار معمولاً نرم‌افزار را بدون در نظر گرفتن چالش‌های استقرار و مدیریت زیرساخت‌ها توسعه می‌دهند، در حالی که تیم‌های عملیات باید مسئولیت عملکرد و استقرار نرم‌افزار را بر عهده بگیرند. DevOps این شکاف را پر می‌کند.

اهداف اصلی DevOps شامل موارد زیر می‌شود:

1. **افزایش همکاری بین تیم‌های توسعه و عملیات**.
2. **بهبود کیفیت نرم‌افزار** از طریق تست‌های مداوم و خودکار.
3. **کاهش زمان استقرار (Deployment)** و ارائه سریع‌تر ویژگی‌های جدید.
4. **کاهش خطرات و خطاها** در فرآیند استقرار.
5. **افزایش قابلیت اطمینان سیستم‌ها** با استفاده از مانیتورینگ مداوم و رفع سریع مشکلات.

**مفاهیم و اصول کلیدی DevOps :**

1. **تحویل مداوم (Continuous Delivery)** :
   * یکی از اصول اصلی DevOps است که به تیم‌ها اجازه می‌دهد نرم‌افزار را به‌طور مداوم و با سرعت بیشتری به محیط‌های تست و تولید ارسال کنند. با استفاده از این رویکرد، تیم‌ها می‌توانند به‌طور مداوم نسخه‌های جدید را با قابلیت‌های به‌روز منتشر کنند و مشکلات نرم‌افزار را سریع‌تر برطرف کنند.
2. **یکپارچه‌سازی مداوم (Continuous Integration)** :
   * توسعه‌دهندگان به‌طور مداوم کد جدید خود را به مخزن مرکزی ارسال کرده و به‌طور خودکار آزمایش‌هایی روی آن انجام می‌شود. این کار باعث می‌شود تا مشکلات و خطاها سریع‌تر شناسایی و رفع شوند.
3. **تست مداوم (Continuous Testing)** :
   * در DevOps تست‌ها به‌طور خودکار و مداوم اجرا می‌شوند تا اطمینان حاصل شود که نرم‌افزار به‌درستی کار می‌کند. با این روش، می‌توان کیفیت نرم‌افزار را بدون نیاز به تست‌های دستی بهبود داد.
4. **استقرار مداوم (Continuous Deployment)** :
   * این مفهوم به‌معنای استقرار مداوم تغییرات کد به محیط تولید است. با استفاده از ابزارهای خودکار، نسخه‌های جدید نرم‌افزار به‌طور پیوسته منتشر می‌شوند و کاربران می‌توانند به‌سرعت از آن‌ها استفاده کنند.
5. **بازخورد سریع**:
   * یکی از اصول کلیدی DevOps این است که هر تغییر به‌سرعت باید بازخورد دریافت کند، خواه از طریق تست‌های خودکار، مانیتورینگ سیستم، یا حتی بازخورد کاربران نهایی. این بازخورد به تیم‌ها کمک می‌کند تا سریع‌تر به مشکلات پاسخ دهند و آن‌ها را اصلاح کنند.
6. **زیرساخت به‌عنوان کد (Infrastructure as Code)** :
   * DevOps استفاده از تکنیک‌هایی مانند زیرساخت به‌عنوان کد را تشویق می‌کند، که به‌معنی مدیریت و پیکربندی زیرساخت‌ها با استفاده از کدها و اسکریپت‌های خودکار است. این کار باعث می‌شود زیرساخت‌ها قابل تکرار و قابل مدیریت باشند.
7. **مانیتورینگ و لاگینگ مداوم**:
   * در DevOps، ابزارهای مانیتورینگ و لاگینگ به‌طور مداوم عملکرد سیستم‌ها و نرم‌افزار را نظارت می‌کنند تا مشکلات به‌سرعت شناسایی و رفع شوند. این ابزارها به تیم‌ها کمک می‌کنند تا عملکرد سیستم‌ها را زیر نظر داشته و از صحت و عملکرد صحیح نرم‌افزار اطمینان حاصل کنند.

**ابزارهای کلیدی در DevOps :**

برای اجرای مؤثر DevOps، مجموعه‌ای از ابزارها استفاده می‌شود که به تیم‌های توسعه و عملیات کمک می‌کنند تا به اهداف خود برسند:

1. **Git** (برای کنترل نسخه)
   * یک سیستم مدیریت نسخه که به توسعه‌دهندگان کمک می‌کند تا تغییرات کد را دنبال کنند و بر روی کدها به‌صورت همکاری‌محور کار کنند.
2. **Jenkins** (برای یکپارچه‌سازی مداوم)
   * Jenkins یک ابزار اتوماسیون است که برای اجرای تست‌های خودکار و ساخت پروژه‌ها استفاده می‌شود. این ابزار به تیم‌ها کمک می‌کند تا فرآیندهای یکپارچه‌سازی و تحویل مداوم را اجرا کنند.
3. **Docker** (برای مدیریت کانتینرها)
   * Docker به تیم‌ها اجازه می‌دهد نرم‌افزارها را به‌صورت کانتینری اجرا کنند، که باعث ساده‌سازی فرآیند استقرار و کاهش ناسازگاری‌های محیطی می‌شود.
4. **Kubernetes** (برای مدیریت کانتینرها)
   * یک پلتفرم مدیریت کانتینر که به تیم‌ها کمک می‌کند تا کانتینرهای Docker را در محیط‌های پیچیده مدیریت کنند و به‌طور خودکار مقیاس‌گذاری کنند.
5. **Terraform** (برای زیرساخت به‌عنوان کد)
   * Terraform به تیم‌ها اجازه می‌دهد زیرساخت‌های خود را با استفاده از کد مدیریت و پیاده‌سازی کنند. این ابزار برای ایجاد، مدیریت و تغییر منابع ابری استفاده می‌شود.
6. **Nagios** (برای مانیتورینگ)
   * یک ابزار مانیتورینگ که به تیم‌های DevOps کمک می‌کند تا عملکرد سیستم‌ها و سرویس‌های خود را زیر نظر داشته باشند و در صورت بروز مشکلات به‌سرعت اقدامات لازم را انجام دهند.

**کاربردهای DevOps :**

DevOps به‌ویژه در سازمان‌ها و شرکت‌هایی که نیاز به انتشار سریع نرم‌افزارها دارند و یا نرم‌افزارهای پیچیده و بزرگی را مدیریت می‌کنند، کاربرد فراوانی دارد:

1. **شرکت‌های توسعه نرم‌افزار**:
   * DevOps برای شرکت‌هایی که به‌طور مداوم نیاز به ارائه و به‌روزرسانی نرم‌افزارها دارند (مانند شرکت‌های فناوری و استارتاپ‌های نرم‌افزاری)، بسیار مؤثر است. این شرکت‌ها از طریق DevOps می‌توانند تغییرات جدید را سریع‌تر به کاربران ارائه دهند.
2. **سرویس‌های ابری و SaaS**:
   * DevOps برای مدیریت و استقرار سرویس‌های ابری و مدل‌های نرم‌افزار به‌عنوان سرویس (SaaS) ایده‌آل است. این مدل‌ها نیازمند قابلیت‌های مقیاس‌پذیری سریع و استقرار مداوم هستند که DevOps آن‌ها را فراهم می‌کند.
3. **شرکت‌های تجارت الکترونیک**:
   * در شرکت‌های تجارت الکترونیک که سرعت و پایداری نرم‌افزار بسیار مهم است، DevOps به آن‌ها کمک می‌کند تا به‌طور مداوم نرم‌افزارها را به‌روزرسانی کرده و مشکلات را سریع برطرف کنند.
4. **پروژه‌های بزرگ سازمانی**:
   * سازمان‌های بزرگ که نیاز به مدیریت چندین تیم و سیستم‌های پیچیده دارند، از DevOps برای هماهنگی بهتر بین تیم‌ها و بهبود فرآیندهای توسعه و استقرار استفاده می‌کنند.

**نمونه پروژه‌های انجام‌شده با DevOps :**

1. **Netflix** :
   * Netflix از DevOps استفاده می‌کند تا نرم‌افزارهای خود را به‌طور مداوم بهبود دهد و به‌روزرسانی‌های خود را بدون وقفه در سیستم به کاربران ارائه کند. آن‌ها از یک زیرساخت قوی DevOps استفاده می‌کنند که شامل ابزارهای مدیریت کانتینر و مانیتورینگ مداوم است.
2. **Amazon** :
   * Amazon نیز از DevOps برای مدیریت سرورهای گسترده و سیستم‌های تجارت الکترونیک خود بهره می‌برد. آن‌ها از روش‌های تحویل مداوم استفاده می‌کنند تا قابلیت‌های جدید به‌سرعت به پلتفرم ارائه شوند.
3. **Facebook** :
   * Facebook یکی از شرکت‌هایی است که فرآیندهای DevOps را برای استقرار مداوم و بدون توقف نرم‌افزار به کار گرفته است. آن‌ها از سیستم‌های یکپارچه‌سازی مداوم و مانیتورینگ پیشرفته استفاده می‌کنند.

**مزایای DevOps :**

1. **سرعت بیشتر**:
   * DevOps باعث می‌شود تیم‌ها سریع‌تر به درخواست‌ها و نیازهای کاربران پاسخ دهند و ویژگی‌های جدید را سریع‌تر ارائه دهند.
2. **کیفیت بهتر نرم‌افزار**:
   * با استفاده از تست‌های خودکار و فرآیندهای یکپارچه‌سازی مداوم، نرم‌افزارها کیفیت بالاتری دارند و خطاها سریع‌تر شناسایی می‌شوند.
3. **کاهش زمان تاخیر در استقرار**:
   * DevOps به کاهش زمان لازم برای استقرار و انتشار نرم‌افزار کمک می‌کند. فرآیندهای خودکار و تحویل مداوم، زمان‌های استقرار را به حداقل می‌رسانند.
4. **همکاری بهتر بین تیم‌ها**:
   * DevOps باعث ایجاد همکاری نزدیک‌تر بین تیم‌های توسعه و عملیات می‌شود، که به بهبود کارایی و عملکرد کلی تیم کمک می‌کند.
5. **انعطاف‌پذیری بیشتر**:
   * DevOps به تیم‌ها اجازه می‌دهد به‌سرعت به تغییرات نیازمندی‌ها و بازار پاسخ دهند و تغییرات را به‌طور مداوم اعمال کنند.

**معایب DevOps :**

1. **پیچیدگی برای تیم‌های کوچک**:
   * پیاده‌سازی DevOps ممکن است برای تیم‌های کوچک پیچیده باشد، زیرا نیاز به استفاده از ابزارهای متنوع و فرآیندهای جدید دارد.
2. **نیاز به مهارت‌های چندجانبه**:
   * DevOps به تیم‌ها نیاز دارد تا دارای مهارت‌های گسترده‌ای باشند، از جمله توسعه، عملیات، تست، و مدیریت زیرساخت‌ها. این نیاز به مهارت‌های چندجانبه می‌تواند یک چالش باشد.
3. **نیاز به فرهنگ سازمانی مناسب**:
   * پیاده‌سازی موفق DevOps نیاز به تغییرات فرهنگی در سازمان دارد که ممکن است در برخی سازمان‌ها سخت باشد.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **متدولوژی** | **ویژگی‌های کلیدی** | **مزایا** | **معایب** | **کاربردها** |
| **Scrum** | چرخه‌های کوتاه (Sprint)، جلسات روزانه (Daily Scrum)، تحویل تدریجی، تمرکز بر تیم و همکاری | انعطاف‌پذیری بالا، بازخورد سریع، تحویل مکرر ویژگی‌ها | نیاز به تعهد بالا از تیم، مناسب نبودن برای پروژه‌های با نیازهای ثابت | پروژه‌های پیچیده و تغییرپذیر، توسعه نرم‌افزار و استارتاپ‌ها |
| **XP (Extreme Programming)** | برنامه‌نویسی جفتی، بازخورد مداوم، تست مداوم، ساده‌سازی کد، انتشار مکرر | کیفیت بالای کد، کاهش باگ‌ها، تمرکز بر مشتری | ممکن است در تیم‌های بزرگ کارایی کمتری داشته باشد، نیاز به ارتباط قوی بین اعضا | توسعه نرم‌افزارهای کوچک و متوسط، جایی که کیفیت کد و بازخورد سریع مهم است |
| **Kanban** | مدیریت جریان کار، کار در حال انجام محدود، تمرکز بر بهبود مداوم | انعطاف‌پذیری بالا، بهبود مستمر فرآیند، کاربرد ساده | ممکن است در پروژه‌های پیچیده نیاز به ساختار بیشتری باشد، عدم داشتن برنامه‌ریزی دقیق برای چرخه‌ها | محیط‌های پشتیبانی، عملیات‌های روزمره، پروژه‌های بدون نیاز به چرخه‌های زمان‌بندی دقیق |
| **UP (Unified Process)** | فازهای مشخص (شروع، تحلیل، طراحی، اجرا، تست)، مبتنی بر مدل‌های RUP، تمرکز بر مستندات | رویکرد ساختارمند، کاهش خطرات در مراحل اولیه | پیچیدگی بالا، زمان‌بر بودن مراحل، نیاز به مستندات زیاد | پروژه‌های بزرگ سازمانی، جایی که نیاز به برنامه‌ریزی دقیق و مستندات کامل است |
| **AUP (Agile Unified Process)** | ترکیبی از اصول چابک و UP، کاهش مستندات غیرضروری، چرخه‌های کوتاه | انعطاف‌پذیری چابک، بهبود مستمر، کاهش مستندات غیرضروری | ممکن است برای پروژه‌های بسیار بزرگ کافی نباشد، نیاز به هماهنگی دقیق بین تیم‌ها | پروژه‌های نرم‌افزاری چابک با نیاز به مستندات متوسط، سازمان‌های بزرگ و متوسط |
| **DSDM (Dynamic Systems Development Method)** | تحویل تدریجی، زمان و بودجه ثابت، تمرکز بر همکاری با مشتری و تحویل کارکردهای با اولویت بالا | زمان و هزینه ثابت، تمرکز بر نیازهای واقعی مشتری، سازگاری با تغییرات | نیاز به تعهد بالا از طرف مشتری، مناسب نبودن برای پروژه‌های با نیازمندی‌های ثابت | پروژه‌های پیچیده با تغییرات مکرر، سازمان‌های دولتی و مالی |
| **DevOps** | یکپارچگی توسعه و عملیات، تحویل و استقرار مداوم، تست مداوم، زیرساخت به‌عنوان کد | کاهش زمان استقرار، بهبود کیفیت نرم‌افزار، افزایش همکاری بین تیم‌ها | نیاز به تغییر فرهنگ سازمانی، پیچیدگی در پیاده‌سازی برای تیم‌های کوچک | شرکت‌های فناوری و استارتاپ‌ها، سرویس‌های ابری، تجارت الکترونیک، سازمان‌های بزرگ با سیستم‌های پیچیده |