د کابل پوهنتون دطبیعي علومو علمي ـــ څېړنیزه مجله
مجـــله علـــمی- تحقیقی علـــوم طبیعی پوهنتون کابل
Journal of Natural Sciences – Kabul University
e-ISSN:3078-6665 P-ISSN:3078-5561

مقایسه روش‌های توسعه نرم‌افزار به اساس پلان و توسعه تدریجی

پوهندوی عبدالوحید صمدزی[۱]، پوهنیار محمد سلیم همدرد[۲]

[۱،۲]دیپاتمنت انجنیری نرم‌افزار، پوهنځی کمپیوترساینس، پوهنتون کابل، کابل، افغانستان
ایمیل: samadzai@mail.com

چکیده

روش‌های مختلفی برای توسعه نرم‌افزار وجود دارد. از آن جمله روش توسعه به اساس پلان و روش توسعه تدریجی می‌باشند. روش توسعه به اساس پلان دارای مراحل مختلفی هستند که در آن تمام فعالیت‌های پروسه از قبل پلان‌گذاری شده است و روش توسعه تدریجی یک روش مدرن برای مدیریت پروژه است که شامل تقسیم یک پروژه‌ی پیچیده به مادیول‌های کوچک‌تر و مستقل می‌باشد. در این تحقیق مقایسوی-مروری سیستماتیک روش توسعه به اساس پلان و روش توسعه تدریجی مقایسه گردیده است. یافته‌های این تحقیق نشان می‌دهد که روش توسعه‌ی نرم‌افزار به اساس پلان به دانش مشتری بستگی دارد و روش تدریجی به افراد وابسته به تیم بستگی دارد و نیز آوردن تغییرات در روش تدریجی ارزان است؛ اما در روش توسعه به اساس پلان قیمت است. توسعه به اساس پلان به تیم بزرگ‌تر ضرورت دارد؛ اما روش توسعه تدریجی به تیم کوچک‌تر ضرورت دارد.

واژه‌های کلیدی: افزایشی؛ پروسس؛ پلان؛ تدریجی؛ تیم؛ چابک؛ فکتورسازی مجدد؛ مادیول

Comparison of Plan-Driven Software Development Model and Incremental Model

Abdul Wahid Samadzai[1], Mohammad Salim Hamdard[2]

[1,2]Department of Software Engineering, Computer Science Faculty, Kabul University, Kabul, Afghanistan
Email: samadzai@gmail.com

Abstract

There are different methods for software development. One of them is the plan-based development method. The plan-based development method has different stages in which all the process activities are planned in advance. Another important method is called incremental development, incremental development is a modern method for project management that involves dividing a complex project into smaller and independent modules. In this article, the plan-based development method and the incremental development method are compared. In this research, a comparative and descriptive method are used and the expected results are as follows: In the plan-based development method, it depends on the knowledge of the customer and in the incremental method, it depends on the people in the team, making changes in the incremental method is cheap but, in the plan, -based development method it is expensive. Plan-based development requires a larger team, while the incremental method requires a smaller team.

Keywords: Agile; Incremental; Module; Plan; Process; Refactoring; Team

ارجاع: صمدزی، ع. و.؛ همدرد، م. س. (۱٤۰۳). مقایسه روش‌های توسعه نرم‌افزار به اساس پلان و توسعه تدریجی، *مجله علمی- تحقیقی علوم طبیعی پوهنتون کابل، ۷(٤)، ۵۷-۷۰*. https://doi.org/10.62810/jns.v7i4

مقدمه

پروگرام‌نویسی عبارت از نوشتن دستورات و سطرهای کود است، همان طور که در ساخت خانه‌ها بیشتر از کوبیدن میخ‌ها استفاده می‌شود. توسعه‌ی نرم‌افزار بیشتر شامل سازمان و پلان کردن، و قراردادهای مختلف برای ترسیم دیاگرام برای پلان‌های آن است. دانشمندان کمپیوتر از پروسه پلان، پروگرام کردن و سازماندهی یک پروگرام به عنوان توسعه‌ی نرم‌افزار یاد می‌کند. چندین رویکرد برای توسعه‌ی نرم‌افزار وجود دارد. یکی از نمونه‌ها به عنوان مودل آب‌شار شناخته می‌شود. روش‌های متفاوت برای توسعه نرم‌افزار وجود دارد که توسعه‌دهنده‌گان می‌تواند بخاطر دیزاین، ساختن پلان، ساخت و امتحان نرم‌افزار از آن استفاده نماید(Tanveer, 2016). انواع مختلفی از محصولات/خدمات نرم‌افزار هر روز در حال توسعه هستند.

توسعه محصول جدید نرم‌افزار منبع خوب رقابت و اساس رشد دوامدار برای شرکت‌ها است. شرکت های صنعتی از ضروریات استفاده نموده تا محصول جدید را به بازار عرضه نماید که مراحل زیادی را طی می‌کند. توسعه محصول نرم‌افزار شامل طیف وسیعی از زمینه‌های فنی و مراحل زیادی است. بنابراین، باید مطابق به روش‌های مشخص‌شده‌ی آن عمل کرد. توسعه‌دهنده‌گان در پروسه‌ی توسعه محصول، بین روش‌های معمول یا توسعه نرم‌افزار به اساس پلان و روش‌های تدریجی فرق قایل می‌شود. روش‌ها بر اساس پلان علاوه بر روش‌های دیگر، مودل آبشار و مودل V را نیز شامل می‌شوند(Tanveer, 2016). مشخصات آن‌ها بر اساس مراحل مختلف توسعه متفاوت می‌باشد. مودل توسعه بر اساس پلان یکی از متداول‌ترین روش‌های توسعه است. این روش یک ساختار خطی دارد؛ به این معنی که در نهایت مرحله قبلی به پایان می‌رسد و مرحله بعدی شروع می‌شود. خصوصیت آن در سادگی و سهولت مدیریت است. استفاده‌کننده‌گان تنها زمانی می‌توانند نتایج توسعه را ببینند که تا پایان پروسه منتظر بمانند که خطر توسعه را از طریق غلطی‌های کشف نشده‌یی‌که در اوایل پروسه ایجاد شده و اثرات دیرهنگام قابل توجهی دارند، افزایش می‌دهد(Albers, 2019).

یکی از مشکلات عمده و اساسی این است که در بخش انجنیری نرم‌افزار کتاب‌ها به لسان انگلیسی موجود می‌باشد. بنابراین، درک مفاهیم اساسی و فرق بین روش‌های توسعه نرم‌افزار به اکثر محصلان و توسعه‌دهنده‌گان نرم‌افزار که تسلط کامل به لسان انگلیسی ندارد مشکل است. بنابراین، نوشتن مقالات به لسان‌های ملی (پښتو و دری) ضرورت اساسی می‌باشد.

در دنیای تجارت امروزی که وابسته به قوه محرکه است، توسعه نرم‌افزار در نوآوری و رشد را تقویت حیثیت ضربان قلب را دارد. شرکت‌هایی که خدمات توسعه نرم‌افزاری اختصاصی ارائه می‌کنند، معمولاً

زمان زیادی را صرف پلان، ساخت و اجرای کارهای ضروری برای دست‌یابی به نتایج دل‌خواه می‌کند(Lom et al., 2016). این پروسه بسیار پیچیده‌تر از آن چیزی است که بسیاری از شما می‌توانید تصور کنید. در این تحقیق، پنج مرحله اساسی که ماهیت پروسه توسعه نرم‌افزار را تشکیل می‌دهند مورد مطالعه قرار داده و روش‌های توسعه نرم‌افزار بر اساس پلان و توسعه تدریجی باهم مقایسه می‌گردد(Neumann, 2012).

انجنیری نرم‌افزار با پذیرش فرصت‌های جدید و غلبه بر مشکلات به طور مستمر در حوزه‌ی کاربرد و تحقیقات خود با پیشرفت تکنالوژی در حال ارتقاء است. قابلیت و خصوصیات نرم‌افزار با مدیریت خوب دیتا و افزودن قابلیت‌های هوشمند در حال افزایش است. وسایل هوشمند، خودکارسازی صنعتی، تجهیزات هوش مصنوعی و مفاهیم تجارت (کسب‌وکار) دیجیتالی فرصت‌ها را برای صنایع نرم‌افزاری و محققان افزایش می‌دهند. بنابراین، محققان سازمان‌های استندرد برای بهبود کیفیت محصولات نرم‌افزاری و کسر نرخ شکست پروژه، تلاش می‌کند. تحول دیجیتال مشکلات جدیدتری را در صنایع نرم‌افزاری ایجاد می‌کند. ارتقاء سریع تکنالوژی، افزایش قابلیت‌ها و تغییر زیرساخت، پروژه نرم‌افزار را حیاتی می‌نماید(Hanschke et al., 2015).

این تحقیق اهداف زیر را دنبال می‌کند:

- درک تفاوت‌های اساسی بین روش توسعه نرم‌افزار مبتنی بر برنامه‌ریزی و روش توسعه تدریجی.
- انتخاب روش مناسب توسعه نرم‌افزار با توجه به ماهیت و نیازهای پروژه.

همچنین، پرسش‌های کلیدی این تحقیق شامل موارد زیر است:

- در چه شرایطی روش توسعه نرم‌افزار مبتنی بر برنامه‌ریزی برای طراحی و توسعه‌ی نرم‌افزار مناسب نیست؟
- چه زمانی استفاده از روش توسعه تدریجی گزینه‌ی بهتری خواهد بود؟

روش تحقیق

در این تحقیق، محقق از روش مقایسوی و مروری استفاده می‌نماید. روش مقایسوی عبارت است از نگاه کردن به یک موضوع مورد مطالعه در ارتباط با موضوع دیگر. موضوع مطالعه معمولاً در مکان و/یا زمان مقایسه می‌شود. روش‌های مقایسوی می‌تواند کیفی و کمی باشد. تجزیه و تحلیل مقایسه‌یی یک مقایسه جانبی است که به طور سیستماتیک دو یا چند چیز را با هم مقایسه می‌کند تا شباهت‌ها و تفاوت‌های آن‌ها را مشخص کند. تمرکز تحقیق ممکن است مفهومی باشد - یک مشکل، تصور یا

نظریه خاص یا شاید چیزی ملموس‌تر ؛ مانند دو ست متفاوت دیتا (Saleh et al., 2019). به طور مثال؛ می‌توانیم از تجزیه و تحلیل مقایسه‌یی برای بررسی این‌که چگونه خصوصیات محصول بدست آمده با رقبا مقایسه می‌شود، استفاده کنید. پس از یک تحلیل مقایسوی موفق، باید نقاط قوت و ضعف را شناسایی و به وضوح درک کنید که کدام محصول مؤثرتر است. همچنین می‌توانیم از تحلیل مقایسوی برای بررسی روش‌های مختلف تولید آن محصول استفاده کنید و تعیین نماید کدام راه کارآمدتر و مفیدتر است(Jahr, 2014).

یافته‌ها

مقایسه‌ی روش توسعه بر اساس پلان و روش تدریجی

این دو روش به اساس فکتورهای ذیل مقایسه گردیده است:

1. پیش‌بینی‌پذیری: ساختن پلان دقیق در روش توسعه نرم‌افزار بر اساس پلان امکان پیش‌بینی بهتر جدول زمانی و قیمت‌های پروژه را فراهم می‌سازد.

۲. کنترل: کنترل دقیق بر پروسه‌ها و محصولات تحویلی، پابندی به پروگرام‌های از پیش تعریف شده را تضمین می‌کند.

۳. مستندسازی: تأکید بر مستندسازی به حفظ تاریخچه پروژه و انتقال دانش کمک می‌کند.

بنابراین، یک پروژه توسعه مبتنی بر اساس پلان بر تنظیم یک پروگرام برای هر پروژه متمرکز است که شامل منابع موجود، زمان‌بندی تکمیل کار، خرابی‌های کار و غیره است.

برای اطمینان از زمان بهینه تجهیزات و انطباق با مقررات و در عین حال کاهش خطرات، روش توسعه نرم‌افزار بر اساس پلان و روش تدریجی پروگرام‌های اتکای خدمات سیستم را توسعه داده است. مثلاً؛ طیف کاملی از خدمات برای سیستم‌ها و تجهیزات شما.

طرح اتکا/اعتماد، اتکای اساسی شامل مجموعه‌یی از خدمات ایجاد و نگهداری پیشگیرانه و عیب‌یابی است؛ در حالی که طرح اتکای پیشرفته، پوشش بالاتری را با زمان پاسخ‌گویی سریع‌تر و دست‌رسی اولویت به پشتیبانی از راه دور و در محل ارائه می‌دهد.

فکتورسازی مجدد عملی برای بهبود مستمر دیزاین کودهای موجود، بدون تغییر رفتار اساسی است. در روش تدریجی به طور خاص با استفاده از میتود چابک ، تیم ها کود خود را به صورتّ افزایشی از

[1] Refactoring
[2] Agile

حد اکثر اجزا تا حد اکثر اجزا حفظ و ارتقا می‌دهند. اگر کود در یک پروژه‌ی تدریجی به طور خاص در روش چابک بازسازی نشود، منجر به کیفیت پایین کود می‌شود؛ مانند وابستگی‌های ناسالم بین کلاس‌ها یا بسته‌ها، تخصیص نامناسب مسئولیت کلاس، مسئولیت‌های بسیار زیاد در هر میتود یا کلاس، کود تکراری و انواع دیگر سردرگمی و درهم‌ریختگی، فکتورسازی مجدد به حذف این آشفتگی کمک می‌کند و کود نامشخص و پیچیده را ساده می‌کند.

جدول ۱: پارامترهای مقایسه شده بین توسعه براساس پلان و توسعه تدریجی را نشان می‌دهد

شماره	پارامتر	توسعه بر اساس پلان	توسعه تدریجی
۱	اعتماد (اتکا)	در توسعه بر اساس پلان، اتکا به سطح توانایی مشتری بستگی دارد.	در روش تدریجی، اتکا به دانش ضمنی وابسطه به رابطه میان اشخاص بستگی دارد.
۲	معماری شده است	برای ضرورتهای قابل پیش بینی و فعلی هر دو	فقط برای ضرورتهای فعلی
۳	تیم ها	تیم های بزرگتر	تیم های کوچکتر
۴	قابلیت تعویض تیم	اعضای تیم قابل تعویض نیست	اعضای تیم قابل تعویض است
۵	فکتورسازی مجدد	گران (پر خرچ)	ارزان

با تکیه بر کنترول‌ها و استندردهای سخت‌گیرانه، منابع انسانی ممکن است نامزدی بعید برای یک روش تدریجی به نظر برسد؛ اما چابکی ممکن است تنها کلید باز کردن قفل و رویکرد بهتر به مدیریت منابع انسانی باشد. در توسعه نرم‌افزار به اساس پلان اتکا به سطح توانایی مشتری وابسته می‌باشد؛ چون تعامل با مشتری بسیار کم‌تر است. چرا که محصولات تولید شده بعد از پروسه کلی توسعه، به مشتری تحویل داده می‌شوند. اما در روش توسعه تدریجی اتکا به فهم و دانش وابسطه به رابطه میان افراد ارتباط و بستگی دارد و تعامل با مشتریان در روش تدریجی بسیار زیاد است. به این دلیل که پس از هر بار تکرار، یک نسخه‌ی افزایشی برای مشتریان مستقر می‌شود. دریافت تفاوت‌ها میان روش توسعه براساس پلان و روش تدریجی به شما امکان می‌دهد تا روش بهتر توسعه نرم‌افزار را انتخاب کنید که به ضرورت‌های شما مطابقت دارد .

روش تدریجی به خوبی با محصولات و تیم‌های کوچک مطابقت دارد. تکیه بر دانش ضمنی مقیاس‌پذیری را محدود می‌کند. روش‌های توسعه بر اساس پلان، روش تکامل‌یافته برای رسیدگی به محصولات و تیم‌های بزرگ به سختی می‌توان برای پروژه‌های کوچک استفاده کرد. البته این مورد هر وقت درست نمی‌باشد. در بعضی سافت‌ویرهای کوچک که در وسایل (وسایل طبی، موتر، طیاره و

غیره) استفاده می‌شود و در آینده توسط پروگرامر تغییر داده نمی‌شود، نیز از روش توسعه به اساس پلان استفاده می‌شود؛ چون روش توسعه بر اساس پلان برای امنیت سافت‌ویر مهم است.

اتکای بیش از حد به الزامات از پیش تعیین‌شده در مناطقی که نیازمندی‌ها می‌توانند از طریق آشنایی و تجربه‌ی استفاده‌کننده پدیدار شوند، می‌تواند بر معماری و دیزاین‌های پروژه به طور غیرقابل پیش بینی تأثیر بگذارد. روش توسعه نرم‌افزار به اساس پلان برای نیازمندی‌هایی‌که قابل پیش‌بینی شده و فعلی هر دو مناسب می‌باشد، همچنان روش توسعه‌ی تدریجی تنها برای نیازمندی‌های فعلی مناسب می‌باشد.

در روش تدریجی اعضای تیم قابل تعویض و تغیر هستند. از همین‌رو آن‌ها بسیار سریع کار می‌کنند. علاوه براین، این روش توسعه نرم‌افزار، لزوم داشتن مدیر پروژه را از بین می‌برد؛ چون تمام اعضای تیم، مسئولیت مدیریت پروژه‌ی نرم‌افزاری را بر عهده دارند. در روشی به اساس پلان، مدیر پروژه مسئول گفتن حرف آخر در تمام مراحل توسعه نرم‌افزار است. علاوه براین، تعویض اعضای تیم ممکن نیست. می‌توانیم بگوییم که این یک نگرش به انجنیری نرم‌افزار است. هرجا که پروگرام‌سازی شده باشد؛ در خصوص نحوه انجام مراحل توسعه، پروگرام‌سازی شده است. این مبتنی بر روش‌های مدیریت انجنیری پروژه و روش قدیمی برای مدیریت سرمایه‌گذاری‌های بزرگ توسعه نرم‌افزار است. این مورد باعث می‌شود تا استفاده از پروگرام‌سازی، مدیریت در اندازه‌گیری پیشرفت و تصمیم‌گیری پروژه آشکار شود.

هنگامی‌که برخی از نکات ذکر شده در بالا را دنبال می‌نمایید، می‌توانید پاسخ این سؤال را پیدا کنید که چگونه وابستگی‌ها را در روش تدریجی (چابک) مدیریت کنید. با گنجاندن این استراتیژی‌ها در روش‌های تدریجی خود، می‌توانید مدیریت وابستگی را ساده کنید و چابکی کلی تیم خود را افزایش دهید. چه همه یا برخی از این استراتیژی‌ها را اجرا کنید، متوجه خواهید شد که کاهش وابستگی‌ها آسان‌تر می‌شود و به تیم‌های توسعه اجازه می‌دهد تا بهتر و کارآمدتر عمل کنند.

فکتورسازی مجدد در روش تدریجی به معنی آسان‌تر و قابل فهم کردن کود موجود در یک پروگرام، بدون تغییر عملکرد آن است. این اجازه می‌دهد تا پروگرام مقیاس‌پذیر شود و قیمت‌های نگهداری بیشتر را کاهش دهد. فکتورسازی مجدد به طور مداوم از پیچیده شدن کود جلوگیری می‌کند و به بهتر نگهداشتن کود و نگهداری آسان کمک می‌کند. در توسعه تدریجی، می‌توان سرعت‌های کوتاه و جداگانه‌یی برای تطبیق مجدد فاکتور وجود داشت.

فکتورسازی مجدد، پروسه بازسازی کودهای موجود کمپیوتر برای بهبود کارایی و حفظ عملکرد است. مزایای بازسازی مجدد شامل بهبود خوانایی، کاهش پیچیدگی، همکاری تیمی بهتر و محصول با کیفیت بالاتر است. فکتورسازی مجدد پروسه اصلاح‌سازی و بازسازی کودهای موجود کمپیوتر برای بهبود کارایی آن بدون تغییر در عملکرد آن است.

در نتیجه، تغییر آوردن در کود یک عمل حیاتی برای حفظ و افزایش کیفیت، عملکرد و قابلیت نگهداری کود نرم‌افزار است. توسعه‌دهندگان با درک اهمیت، مزایا، چالش‌ها و تکنیک‌های ضروری فکتورسازی مجدد می‌توانند کود خود را بهینه کنند و پروگرام‌های کاربردی کارآمدتر، قابل نگهداری و باکیفیت‌تر ایجاد کنند. پیاده‌سازی بهترین شیوه‌ها و بهره برداری از پشتیبانی بازسازی خودکار می‌تواند روند بازسازی را بیشتر بهبود بخشد و از یک پروژه‌ی بازسازی موفق که در نهایت منجر به یک محصول برتر می‌شود، اطمینان حاصل کند.

یکی از راه‌های بهتر برای تغییر شکل کود، پیروی از روش‌های توسعه تدریجی است که اجازه می‌دهد کود به شیوه‌ی بهتر، سازمان یافته و کارآمد از طریق امتحان جامع نگهداری شود. با پیشرفت‌ها و پیشرفت‌های نرم‌افزاری، مقیاس‌گذاری کود می‌تواند دشوار شود. فکتورسازی مجدد در روش توسعه تدریجی نسبت به روش توسعه نرم‌افزار بر اساس پلان ارزان‌تر است. در روش توسعه بر اساس پلان فکتورسازی مجدد به قیمت بلند صورت می‌گیرد.

بحث و مناقشه

در این بخش یافته های این تحقیق با روش کار و یافته های تحقیقات که قبلاً صورت گرفته باهم بحث و مناقشه صورت گرفته.

روش توسعه نرم‌افزار به اساس پلان

روش توسعه نرم‌افزار به اساس پلان ممکن است برای سیستم‌های که ضرورت به تجزیه و تحلیل زیاد قبل از پیاده‌سازی دارند، (به طور مثال سیستم بلادرنگ با الزامات زمان‌بندی پیچیده) مورد ضرورت باشد(Al-Azawi, 2014).

یافته‌های این تحقیق نشان می‌دهد که روش توسعه به اساس پلان معمولاً به نوعی قرارداد بین توسعه‌دهنده‌گان و مشتریان به طور اساسی برای روابط با مشتری بستگی دارد. آن‌ها سعی می‌کنند با کار کردن از قبل روی آن‌ها و رسمی کردن راه‌های حل در یک توافق‌نامه‌ی مستند، با مشکلات قابل پیش‌بینی کنار بیایند. این مورد، به خصوص در شرایط پایدار دارای فواید زیاد است. توسعه‌دهنده‌گان

می‌دانند که چه کاری باید انجام دهند؛ مشتریان می‌دانند که چه چیزی به دست می‌آورند و بهتر می‌توانند به مسئولیت‌های عملیاتی خود ادامه دهند(Badwe and Erkan, 2018).

یافته‌های این تحقیق نشان‌دهنده‌ی آن است که در روش توسعه به اساس پلان، مردم زمانی احساس راحتی و قدرت می‌نماید که پالیسی‌ها و طرز العمل‌های روشنی وجود داشته باشد که نقش آن‌ها را در سازمان مشخص کند. این بیشتر یک محیط خط تولید است که در آن وظایف هر شخص به خوبی تعریف شده است. انتظار این است که آن‌ها وظایف خود را تا حد مشخص انجام دهند تا محصولات قابل استفاده(کاری) آن‌ها به راحتی در محصولات قابل استفاده‌ی دیگران با دانش محدود از آن‌چه که دیگران واقعاً انجام می‌دهند، ادغام شوند (Hanschke et al., 2015). ساحه (فضای) پروگرام‌سازی شده از رویکردهای رسمی برای درک و مودل‌سازی مسئله استفاده می‌کند. در مرحله اول، این مشکلی است که استفاده‌کننده‌ی یک سیستم با آن مواجه است. در موارد کمتر، ممکن است یک مشکل فنی نیز باشد. در ساحه‌ی روش‌های پروگرام‌سازی شده، کار به درک، تحلیل و مودل‌سازی مسئله‌یی می‌رود که راه حل آن توسعه می‌یابد (Ghani et al., 2014). روش توسعه به اساس پلان شامل رویکردهای توسعه، مانند مودل آبشار، پروسه یک‌پارچه منطقی و مودلV است. همه روش‌ها به اساس پلان، دارای خصوصیات ذیل هستند: عملکردها/خصوصیات مورد نظر نرم‌افزار باید از قبل مشخص شوند. یک پلان تفصیلی از ابتدا تا ختم پروژه ساخته می‌شود. ضروریات با جزئیات و تفصیل عالی مشخص شده و پس از آن پروسه، درخواست تغییر جدی اجرا می‌شود. مشخصات معماری و دیزاین باید قبل از شروع اجرا کامل شود، کار پروگرامر فقط در مرحله پروگرام‌نویسی متمرکز است. امتحان (چک/تست) در پایان پروژه انجام می‌شود و اطمینان کیفیت به صورت رسمی انجام می‌شود(Fulbright, 2013).

توسعه بر اساس پلان، یک روش توسعه نرم‌افزار است که توسعه‌دهنده با استفاده از این روش تلاش می‌کند تمام خصوصیاتی‌که ممکن است یک استفاده‌کننده در محصول نهایی بخواهد پلان‌گذاری و توسعه دهد و روش توسعه‌ی همه‌ی آن خصوصیات را تعیین کند. در واقع این پروگرام عملیاتی مبتنی بر اجرای مجموعه‌ی منظمی از سطوح مخصوص کار است (Ardito, 2017).

توسعه بر اساس پلان یک، روش انجنیری نرم‌افزار مبتنی بر دستور کار است که بر مراحل مختلف توسعه متمایز؛ مانند ضروریات، دیزاین، ساخت، تست و توسعه متکی است. هر مرحله توسط گروپ‌های مختلف مراقبت می‌شود: گروه تجاری در دریافت ضروریات کمک می‌کند، تیم راه‌حل دیزاین را ارائه می‌دهد و تیم‌های اجرایی موارد اجرایی را انجام می‌دهند. خروجی هر مرحله در شروع

پروژه پلان‌گذاری شده است. در اینجا، تمام خصوصیات که مشتری ممکن است بخواهد، پیش‌بینی و پلان‌گذاری شده است(Goevert, 2019). همچنین در پروگرام ساخت آن خصوصیات از قبل انجام می‌شود. سپس بر اساس محدوده‌ی کار، زمان و قیمت نیز از قبل پروگرام‌سازی می‌شود. به طور خلاصه؛ محدوده، قیمت و زمان بر اساس پروگرام ثابت و اجرا می‌شوند(Matthies, 2018).

پروسه توسعه به اساس پلان قرار ذیل است: اولاً، دیزاین ساختمان را کاملاً مطابق با ضروریات مشتری قبل از شروع هر کار ساخت‌وساز دیزاین می‌کند. سپس ساخت و ساز، مانند دیزاین انجام می‌شود. در نهایت، پیشرفت با توجه به چگونگی پیشروی هر مرحله از ساختمان اندازه‌گیری می‌شود (Kumar et al., 2014). توسعه بر اساس پلان، روشی از انجنیری نرم‌افزار است که در آن پروسه‌ی توسعه با جزئیات پروگرام‌سازی می‌شود. توسعه بر اساس پلان مبتنی بر روش‌های مدیریت انجنیری و به روش معمولی مدیریت پروژه‌های بزرگ توسعه نرم‌افزار است. در روش توسعه به اساس پلان، تکرار در فعالیت‌ها با اسناد رسمی که برای برقراری ارتباط بین مراحل پروسه استفاده می‌شود، انجام می‌شود. به طور مثال؛ نیازمندی‌ها تکامل می‌یابند و در نهایت، مشخصات نیازمندی‌ها مشخص می‌شود. سپس این یک ورودی به پروسس دیزاین و پیاده‌سازی است. در یک روش چابک، تکرار در بین فعالیت‌ها اتفاق می‌افتد. بنابراین، الزامات و دیزاین به صورت جداگانه، با هم توسعه می‌یابند(Schmidt, 2019).

مزایای روش توسعه به اساس پلان

- در ابتدا ضرورت به دانش شخصی دارد؛
- مناسب برای پروژه‌های بزرگ توسعه‌ی نرم‌افزار نمی‌باشد؛
- قابل پیش‌بینی: جدول زمانی و قیمت پروژه را می‌توان با دقت بیشتری تخمین کرد؛
- مستندسازی: اسناد با قیمت زیاد در هر مرحله ایجاد می‌شود که به درک و نگهداری آینده می‌افزاید.

نواقص روش توسعه به اساس پلان

هنگامی که یک پروژه دستخوش تغییرات خاصی می‌شود، اعمال تغییرات در الزامات معمولاً دشوار است. بازخورد دیرهنگام و فرآیند جامع اصلاح آن اغلب موجب تأخیر در اجرای پروژه می‌شود (Aguilar & Zapata, 2017).

میتودولوژی‌های مبتنی بر توسعه نرم‌افزار به اساس پلان، مدیریت پروژه‌های نرم‌افزاری بزرگ به اساس یک دلیل ساده بود: این کاری است که انجینران می‌دانستند چگونه انجام دهند (Mukhtar,

(2018. تقریباً همه اشکال دیگر انجنیری از روشی پیروی می‌کنند که به دلایلی به اساس پلان در نظر گرفته می شود، از جمله:

- روی تکرارپذیری و پیش‌بینی‌پذیری؛
- پروسه‌های تعریف شده، استندردشده و به تدریج در حال بهبود؛
- مستندات کامل؛
- پروگرام‌های دقیق، گردش کار، نقش‌ها، مسئولیت‌ها و توضیحات محصول کاری(Jahr, 2014)؛
- مدیریت خطر مستمر.

روش توسعه تدریجی

با روش توسعه تدریجی، سیستم در مراحلی که به صورت واضح تعریف شده است، تجزیه و تحلیل، دیزاین، توسعه و امتحان می‌شود. به عبارت دیگر پروژه به مجموعه‌یی از ساخت‌ها تقسیم می‌شود. عملکرد اصلی در مرحله اول بررسی شده و با افزایش نهایی پروژه کامل منتشر می‌شود. این روش بر رویکرد بلاک ساختمانی تأکید می‌کند و به این معنی است که یک سیستم عملیاتی سریع‌تر ایجاد می‌شود(Lom, M., et al., 2016).

الزامات کامل ممکن است از قبل جمع‌آوری شود یا پروژه ممکن است با اهداف کلی شروع شود که با هر تکرار اصلاح می‌شوند. الزامات، اولویت‌بندی می‌شوند؛ هر چه اولویت الزامات بالاتر باشد، زودتر در یک افزایش گنجانده می‌شود. بنابراین، عملکرد جدید اضافه شده نباید مهم‌تر از آنچه در ساخت فعلی وجود دارد، باشد(Cooper and Sommer, 2018).

روش تدریجی، پروسه توسعه نرم‌افزار است که در آن نیازمندی‌ها به چندین مادیول مستقل از چرخه‌ی توسعه نرم‌افزار تقسیم می‌شوند. در این مودل، هر مادیول مراحل مورد ضرورت، دیزاین، پیاده‌سازی و تست را طی می‌کند. هر انتشار بعدی مادیول عملکردی را به نسخه قبلی اضافه می‌کند. این روند تا رسیدن به سیستم کامل ادامه می‌یابد(Doshi and Patil, 2016).

چی زمانی از روش تدریجی استفاده می‌کنیم؟

- زمانی‌که ضروریات واضح و بهتر شناسایی شده باشد.
- یک پروژه دارای پروگرام توسعه طولانی است.
- زمانی که تیم نرم‌افزار خیلی ماهر یا آموزش دیده نیستند.
- زمانی که مشتری تقاضای عرضه سریع محصول را دارد.

- ابتدا می‌توانید الزامات، اولویت‌بندی شده را توسعه دهید(Bauschmann et al., 2017).

خصوصیات روش تدریجی

- این سیستم به بسیاری از پروژه‌های کوچک توسعه تقسیم می‌شود.
- سیستم‌های جزئی برای ایجاد سیستم نهایی ساخته شده اند.
- ابتدا باید به بالاترین اولویت مورد ضرورت مقابله کرد.
- ضرورت یک بخش پس از توسعه بخش افزایش‌یافته منجمد می‌شود(Doshi and Patil, 2016).

مزیای روش تدریجی

- تشخیص غلطی‌ها آسان است.
- امتحان و رفع مشکلات آسان‌تر است(Heimicke, 2019).
- انعطاف‌پذیرتر است.
- مدیریت خطر آسان است؛ زیرا در طول تکرار آن مدیریت می‌شود.
- مشتری عملکردهای مهم را زود دریافت می‌کند(Kumar et al., 2014).

معایب روش تدریجی

- ضرورت به پروگرام کردن خوب دارد.
- قیمت کل بالاست.
- ارتباط مادیول به خوبی تعریف شده مورد ضرورت است(Saleh, S.M., et al., 2019).

نتیجه‌گیری

روش‌های توسعه نرم‌افزار بر اساس پلان و روش تدریجی دو میتود متفاوت هستند. از همین‌رو هر کدام آن‌ها برای بعضی موارد مناسب و برای بعضی دیگر غیرعملی هستند.

- پروژه‌های توسعه نرم‌افزاری که نیازمندی‌های آن در حال تکامل و یا غیرقطعی می‌باشد، روش تدریجی بهتر است.
- برعکس، روش توسعه به اساس پلان برای پروژه‌های کوچک‌تر توسعه نرم‌افزار با نیازمندی‌های مشخص و قطعی، بهترین روش است.
- در روش تدریجی تیم‌های کوچک‌تر و برخلاف در روش بر اساس پلان تیم‌های بزرگ‌تر ضرورت است.

- در روش تدریجی اعتماد به رابطه میانه بستگی دارد ؛ زیرا در هر تغییر، توسعه‌دهنده با مشتری در تماس می‌باشد. بر عکس در روش توسعه به اساس پلان ارتباط توسعه‌دهنده با مشتری در ختم پروژه می‌باشد.
- فکتورسازی مجدد در روش تدریجی بر خلاف روش بر اساس پلان ارزان‌تر می‌باشد؛ زیرا در روش تدریجی پروژه به بخش‌های کوچک تقیسم گردیده و تغییر و اصلاح کود آسان‌تر می‌باشد.
- در روش تدریجی تغیرات در اعضای تیم ممکن است؛ ولی در روش بر اساس پلان تغییرات در اعضای تیم ممکن نیست.
- روش تدریجی به خاطر ضرویات فعلی استفاده می‌شود و روش به اساس پلان هم به ضروریات فعلی و هم برای ضرویات پیش‌بینی‌شده استفاده می‌گردد.

منابع

Aguilar, M. and Zapata, C. (2017). Integrating UCD and an Agile Methodology in the Development of a Mobile Catalog of Plants, Advances in Intelligent Systems and Computing 486. *Advances in Ergonomics Modeling, Usability & Special Populations (pp.75-87)*. DOI:10.1007/978-3-319-41685-4_8.

Al-Azawi, R. et al. (2014). Multi Agent Software Engineering (MaSE) and Agile Methodology for Game Development. In *In 14th Middle Eastern Simulation and Modelling Multiconference, MESM 2014 - 4th GAMEON-ARABIA Conference, GAMEON-ARABIA 2014.*

Albers, A. et al. (2019). Agility and Its Features in Mechatronic System Development: A Systematic Literature Review, *Proceedings of 30th ISPIM Innovation Conference, Forence, I, June 16-19, 2019.*

Ardito, C. et al. (2017). Integrating a SCRUM-Based Process with Human Centred Design: An Experience from an Action Research Study. *In Proceedings - 2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry, CESI 2017*.

Jonas Heimicke and Albert Albers (2020). Agile meets plan-driven – hybrid approaches in product development: a systematic literature Review Proceedings of the Design Society DESIGN Conference 1:577-586. DOI:10.1017/dsd.2020.259.

Bauschmann, M. and Ahnert, C. (2017). *Vergleich von Web of Science Und Scopus Im Hinblick Auf Den Informationsbedarf an Der TU Chemnitz*. Dieses Gesamtwerk ist, sofern nicht an Einzelinhalten anders angegeben, lizenziertunter einer Creative Commons Namensnennung 4.0 International Lizenz, www.tu-chemnitz.de.

Cooper, R.G. and Sommer, A.F. (2018). Agile–Stage-Gate for Manufacturers: Changing the Way New Products Are Developed Integrating Agile Project Management Methods into a Stage-Gate System Offers Both Opportunities and Challenges. *16(Research Technology Management): 17–26*. https://doi.org/10.1080/08956308.2018.1421380.

Doshi, V.P. and Patil, V. (2016). Competitor Driven Development: Hybrid of Extreme Programming and Feature Driven Reuse Development. In *In 1st International Conference on Emerging Trends in Engineering, Technology and Science, ICETETS 2016 - Proceedings.*

Fulbright, R. (2013). Incorporating Innovation into Iterative Software Development Using the Inventive Problem Solving Methodology. *In International Journal of Innovation Science*, 5(4): 203–12. https://doi.org/10.1260/1757-2223.5.4.203.

Ghani, I., Azham, Z. and Jeong, S.R. (2014). Integrating Software Security into Agile-Scrum Method. *In KSII Transactions on Internet and Information Systems* 8(2): 646-663. https://doi.org/10.3837/tiis.2014.02.019.

Goevert, K. et al. (2019). Interview Study on the Agile Development of Mechatronic Systems. In *Proceedings of the 22nd International Conference on Engineering Design (ICED19), Delft, The Netherlands, The Design Society*.

Hanschke, S., Ernsting, J. and Kuchen, H. (2015). Integrating Agile Software Development and Enterprise Architecture Management. In *In Proceedings of the Annual Hawaii International Conference on System Sciences, March 2015.*,

https://doi.org/10.1109/HICSS.2015.492.

Heimicke, J. et al. (2019). Comparison of Existing Agile Approaches in the Context of Mechatronic System Development: Potentials and Limits in Implementation. In *Proceedings of the 22nd International Conference on Engineering Design (ICED19), Delft, The Netherlands, Design Society (Chair)*.

Jahr, M. (2014). A Hybrid Approach to Quantitative Software Project Scheduling within Agile Frameworks. *In Project Management Journal*, 45(3): 35–45.

Kumar, M., Shukla, M. and Agarwal, S.(2014). A Hybrid Approach of Requirement Engineering in Agile Software Development. In *In Proceedings - 2013 International Conference on Machine Intelligence Research and Advancement, ICMIRA 2013*.

Lom, M., Pribyl, O. and Zelinka, T. (2016). System Engineering for Smart Cities-Hybrid-Agile Approach in Smart Cities Procurement, In WMSCI 2016 - *20th World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings 2.*

Matthies, C.(2018). Scrum2kanban: Integrating Kanban and Scrum in a University Software Engineering Capstone Course. In *In Proceedings - International Conference on Software Engineering.*, https://dl.acm.org/doi/10.1145/3194779.3194784.

Mukhtar, M. et al. (2018). Scrum2kanban: Integrating Kanban and Scrum in a University Software Engineering Capstone Course, *In Proceedings - International Conference on Software Engineering*." https://doi.org/10.1145/3194779.3194784.

Neumann, F. (2012). Mechatronic Product Development: Potentials, Challenges, Terminology. In *Portela, L.T. and Borrego, G, "Scrumconix: Agile and Documented Method to AGSD", In Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016.*, https://doi.org/0.1109/icgse.2016.39.

Saleh, S.M., Huq, S.M. and Rahman, M.A. (2019). Comparative Study within Scrum, Kanban, XP Focused on Their Practices. In *In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, IEEE,* , 1–6.

Schmidt, T.S. et al. (2019). *Agile Development of Physical Products: An Empirical Study about Potentials, Transition and Applicability, Report, University of the German Federal Armed.*

Tanveer, M. (2016). Agile for Large Scale Projects - A Hybrid Approach. In *In 2015 National Software Engineering Conference, NSEC 2015*.