تجزیه و تحلیل تناسب توسعه نرم افزارهای مختلف مدل های چرخه دوام

چکیده

در دوره فعلی توسعه نرم افزاری، تعداد زیادی از مدل های دوام برای توسعه نظام مند طرح ها و نرم افزارهای رایانه ای در دسترس هستند. مدل های SDLC دستورالعمل های نظری مربوط به توسعه نرم افزاری را ارائه می دهند. مدل های SDLC برای توسعه نرم افزار به شیوه ای نظام مند اهمیت زیادی دارند، به گونه ای که در مدت زمانی خاص در دسترس خواهد بود و باید دارای کیفیت مناسبی نیز باشد. این مدل ها درای خصوصیات منحصربفرد خود هستند و متناسب با شرایط خاصی از توسعه نرم افزاری و انواع نرم افزار هستند. ممکن است ثابت شود که یک مدل چرخه دوام نرم افزاری، بسته به محیط توسعه، کارآمدتر از دیگری باشد. در این مقاله، تلاش شده است تا مدل های چرخه دوام نرم افزاری گوناگون از این جنبه مورد تجزیه و تحلیل قرار گیرند. انتخاب SDLC مناسب به مدیران طرح امکان می دهد تا کل راهکارهای توسعه نرم افزار را سر و سامان دهند. هر SDLC دارای مزایا و معایب خاص خود است که بر اساس آن تصمیم می گیریم چه مدلی باید در چه شرایطی اجرا شود. در این مقاله، بررسی جامعی از مدل های چرخه دوام مختلف، مانند مدل آبشاری[1]، توسعه سریع برنامه (RAD[2])، مدل نمونه، مدل مارپیچی، مدل فزاینده و برنامه نویسی نهایی[3] (XP) ارائه می دهیم.

کلمات کلیدی: چرخه دوام توسعه نرم افزاری (SDLC)، تجزیه و تحلیل تناسب، مراحل چرخه دوام، نقاط عطف

[1] Waterfall model
[2] Rapid application development
[3] Extreme programming

1. مقدمه

چرخه دوام توسعه نرم افزاری (SDLC) یکی از روش های توسعه نرم افزار به شیوه ای هموار است. SDLC احتمال تکمیل طرح نرم فزاری در مدت زمان معین و حفظ کیفیت محصول نرم افزاری به طبق نیاز را نیز افزایش می دهد. چارچوب چرخه دوام توسعه سیستم زنجیره ای از فعالیت ها برای طراحان سیستم را فراهم می آورد تا بتوانند توسعه نرم افزاری را ادامه دهند. این چارچوب اغلب به عنوان بخشی از چرخه دوام توسعه سیستم در نظر گرفته می شود. فرایند توسعه نرم افزار به مراحلی تقسیم می شود که به شرکتی نرم افزاری امکان می دهد تا فعالیت های خود را سازماندهی نماید. تمامی طرح های نرم افزاری مراحل جمع آوری ملزومات، تجزیه و تحلیل تجاری، طراحی سیستم، اجرا، و ازمون اطمینان از کیفیت پشت سر می گذارند. به کار گیری هر مدل SDLC اغلب مسئله انتخاب فردی است که به توسعه دهنده بستگی دارد. هر SCLD دارای نقاط قوت و ضعف خود می باشد که می تواند در شرایط مختلف کارکردهای بهتری را ارائه دهد. جووانوویچ دی[۴] و همکارانش اصول پایه و مقایسه مدل های توسعه نرم افزاری را ارائه دادند. رودریگوئیز- مارتینز[۵] و همکارانش به مدل های چارچوب چرخه دوام توجه کرده و به شرح دقیق فرایند چرخه های دوام توسعه نرم افزاری پرداختند و نتایج بررسی جامع چرخه های دوام توسعه نرم افزاری را گزارش دادند. ممکن است یک مدل چرخه دوام به لحاظ نظری مناسب با شرایط خاصی بوده و ممکن است همزمان مدل دیگری نیز ماناسب با مقتضیات ما به نظر برسد، اما باید در زمان تصمیم گیری در انتخاب مدل، موازنه و تعادل را در نظر داشته باشیم. دیویس ای ام[۶] و همکارانش قالبی را ارائه دادند که می تواند به عنوان مبنای تحلیل شباهت ها و تفاوت های میان مدل های چرخه دوام جایگزین به صورت ابزاری برای محققان مهندسی نرم افزار عمل نماید که به آنها کمک کند تا اثرات احتمالی مدل چرخه دوام را توصیف کنند. این قالب می تواند به کاربران نرم افزار نیز کمک کند تا در مورد مدل چرخه دوام مناسب برای استفاده در پروژه ای خاص یا حوزه کاربردی دیگری تصمیم گیری نمایند. مدل چرخه دوام توسعه نرم افزار به کارکردهای مجزایی تقسیم بندی می شود و مشخص می کند که چگونه این کارکردها در تلاش برای توسعه کل نرم افزار سازماندهی می شوند. در این پژوهش، تحقیق کاملی در مورد مدل های

[4] Jovanovich D.
[5] Rodriguez-Martinez
[6] Davis A.M.

مختلف چرخه دوام ارائه شده است که بر اساس آن تجزیه و تحلیل تناسب انواع مختلف مدل ها برای پروژه های گوناگون صورت گرفته است.

2. مراحل شامل شده در مدل SDLC

مراحل موجود در مدل چرخه دوام توسعه نرم افزار شامل موارد زیر هستند:

1. جمع آوری نیازمندی ها
2. تجزیه و تحلیل نیازمندی ها
3. طراحی سطح بالا
4. طراحی سطح پایین
5. اجرا
6. آزمودن
7. آماده سازی و نگهداری

ممکن است هر یک از خود فعالیت های اصلی به قدری گسترده باشد که نتواند در یک مرحله انجام شده و باید به مراحل کوچک تر دسته بندی و تقسیم شود. برای مثال، طراحی یک سیستم نرم افزاری گسترده همیشه به چندین مرحله طراحی مجزا تقسیم می شود که از یک طراحی سطح بالا شروع می شود که از مشخص نمودن تنها مؤلفه ها در سیستم آغاز می شود و به سمت طراحی دقیق که در آن منطق مؤلفه ها مشخص می شود، به پیش می رود.

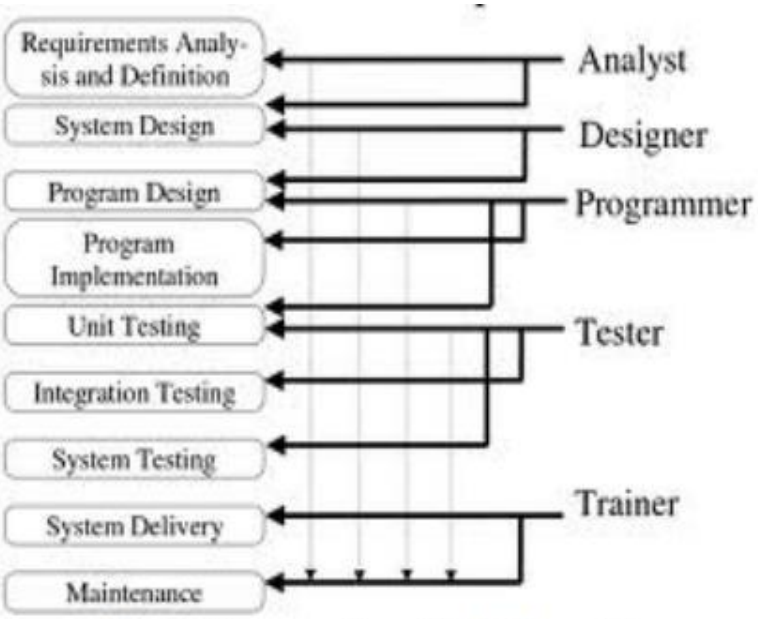

Fig.1. Phases of an SDLC model.

3. مدل های چرخه زندگی توسعه نرم افزار

مدل آبشار

در سال 1970، مدل آبشار توسط رویس[۷] معرفی شد که یک مدل چرخه دوام توسعه نرم افزاری متوالی خطی بود. مراحل گوناگون پی گرفته شده شامل تجزیه و تحلیل نیازمندی ها، طراحی، کدگذاری، آزمودن و اجرا به شیوه ای هستند که مرحله ای که یا بار انجام شود، دیگر تکرار نمی شود و توسعه به مرحله بعد منتقل نمی شود، مگر این که مرحله قبلی کاملا تمام شده باشد. بنابراین، زمانی که مقتضیات پروژه ذاتاً پویا هستند، این مدل چندان مفید نخواهد بود.

نقاط قوت مدل آبشار

1. فهم و استفاده آسان
2. فراهم آوردن ساختاری برای کاربران بی تجربه

[7] Royce

3. درک خوب نقاط عطف

4. تنظیم ثبات مقتضیات

5. مناسب برای کنترل مدیریت (طرح، کارکنان، لبه)

6. کارکرد مناسب در زمانی که کیفیت مهم تر از هزینه ها یا برنامه زمان بندی باشد

نقاط ضعف آبشار

1. تمامی الزامات باید از پیش مشخص باشد.

2. تحویلی های ایجاد شده برای هر مرحله ثابت در نظر گرفته می شوند- مانع انعطاف پذیری می شوند.

3. ممکن است برداشت اشتباهی از فرایند القا نمایند.

4. تلفیق در پایان و به شیوه انفجار بزرگ صورت می گیرد.

5. فرصت کمی برای بازبینی سیستم توسط کاربر وجود دارد (تا حدی که ممکن است دیر شود).

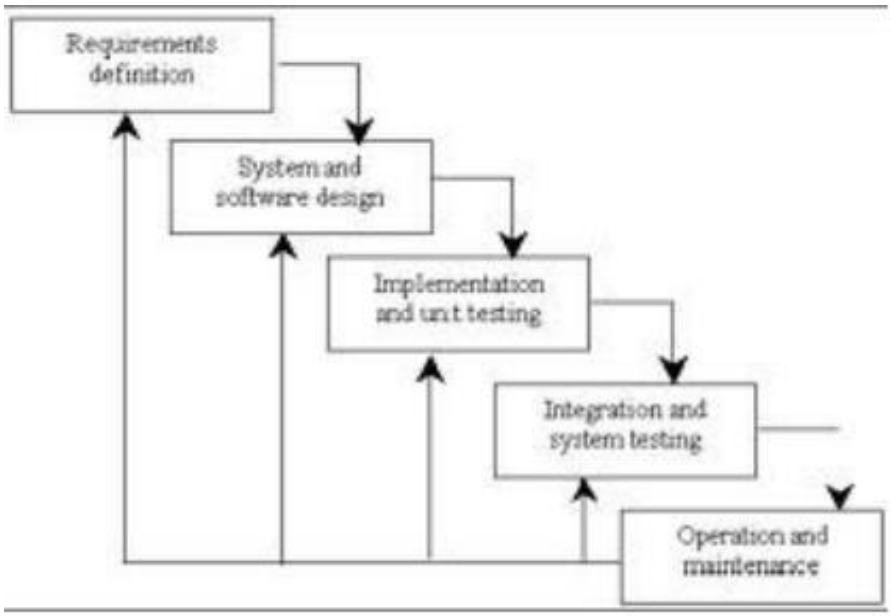

Fig.2. Waterfall Model

نقاط قوت RAD

1. زمان چرخه ای کوتاه تر و بازدهی بهتر که به کارگیری افراد کمتر در آن هزینه ها نیز کاهش می یابد

2. رویکرد جعبه- زمان هزینه ها و خطر زمان بندی برنامه ها را کاهش می دهد.

3. شامل شدن کاربر در طول چرخه کامل احتمال عدم رضایت کاربر و رفع نشدن نیازی تجاری را به حداقل می رساند.

4. این مدل از مفاهیم مدل سازی برای جمع آوری اطلاعات در مورد تجارت، داده ها، و فرایند ها استفاده می کند.

نقاط ضعف RAD

1. احتمال دست نیافتن به جمع بندی

2. دشواری استفاده با سیستم های میراثی

3. لزوم وجود سیستمی که بتواند از بخش های مختلف تشکیل شود

4. توسعه دهندگان و کاربران باید متعهد به فعالیت های پیاپی در چارچوب زمانی کوتاهی باشند.

RAD می تواند در پروژه هایی اجرا شود که برای آنها جدول زمانی محدود بوده، خطر زیاد بالا نیست و دامنه پروژه کوچک یا متوسط است. مدل RAD برای پروژه های با محدودیت زمانی 60 تا 90 روز می تواند بسیار مؤثر باشد.

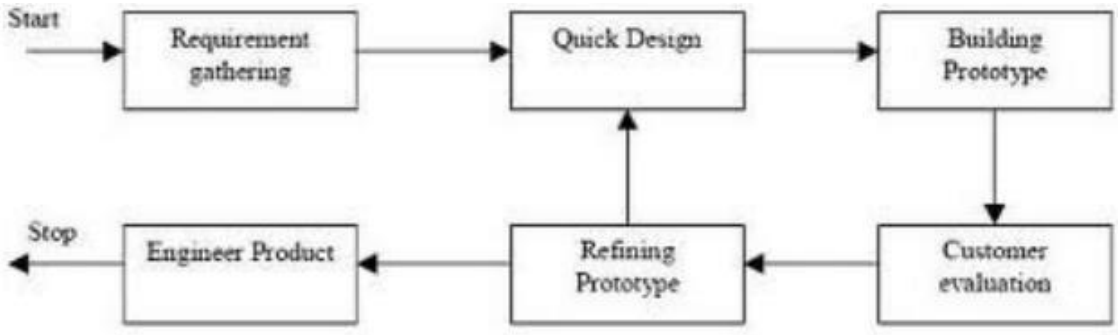

Fig.4. RAD Model

مدل فزاینده

در مدل فزاینده، به اجرای نسبی سیستمی کلی می پردازیم، و سپس به تدریج کارکردهای دیگری را نیز به آن اضافه می کنیم. مدل فزاینده به ملزومات سیستم اولویت می دهد و سچس در گروه هایی آنها را اجرا می سازد. در هر نسخه جدید این سیستم نسبت به نسخه پیشین امکاناتی اضافه می شود، تا زمانی که تمام کارکردها اجرا شود.

نقاط قوت مدل فزاینده

1. کارکردهای دارای خطر بالا یا عمده را در ابتدا توسعه می دهد.
2. هر نسخه محصولی عملیاتی را تحویل می دهد.
3. کاربر می تواند به هر شکلی واکنش نشان دهد.
4. از تفکیک تقسیم و تسخیر وظایف استفاده می کند.
5. تحویل محصول اولیه سریع تر است.

نقاط ضعف مدل فزاینده

1. نیازمند برنامه ریزی و طراحی خوب است.
2. نیازمند تعریف اولیه سیستمی کامل و کاملا کاربردی است تا تعریف افزایش ها را به حساب آورد.
3. نیازمند سطوح مشترک واحد مشخص است (برخی از آنها پیش از سایرین توسعه می یابند).

مدل نمونه

نمونه سازی نرم افزار فرایند ایجاد مدل ناقصی از نرم افزارهای کامل است. این فرایندها شامل معرفی ملزومات اصلی، توسعه نمونه اولیه، بازبینی توسط کاربران و بررسی و تقویت می باشد.

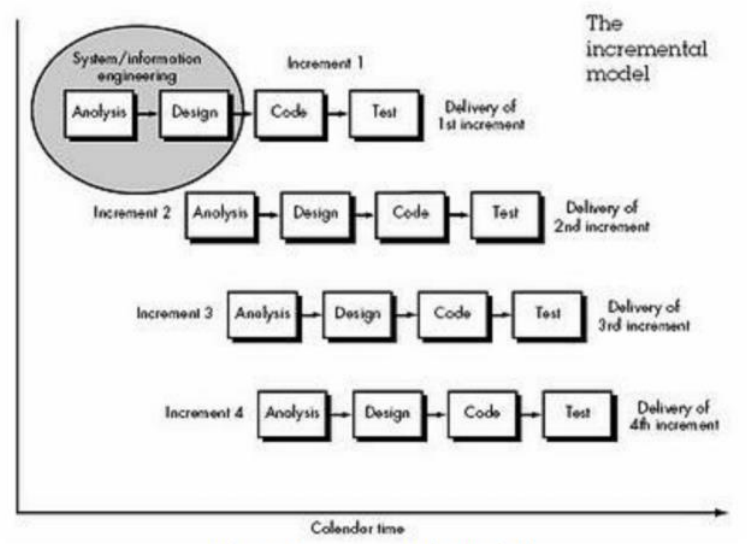

Fig.5. Incremental Model

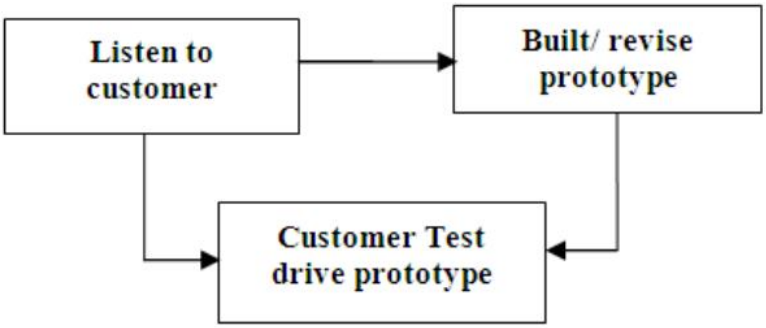

Fig.6. Prototype Model

مزایای مدل عبارت هستند از برداشت های غلط بین کاربران نرم افزار و توسعه دهندگان عیان است، خدمات ازدست رفته ممکن است شناسایی شود و خدمات آشفته کننده ممکن است مشخص شوند، یک سیستم کاری در اوایل فرایند در دسترس است، نمونه می تواند به عنوان مبنایی برای به دست آوردن مشخصات سیستم استفاده شود، و سیستم می تواند از آموزش کاربر و ازمودن سیستم پشتیبانی نماید.

مدل XP

در مدل XP، برنامه ریزی به صورت جفت صورت می گیرد. این مدل شامل توسعه مبتنی بر آزمون، برنامه ریزی مستمر، تغییر و تحویل می باشد. هیچ کار اضافی لازم نیست.

نقاط قوت XP

1. شیوه های سبک مناسب پروژه های کوچک و متوسط هستند.

2. انسجام گروهی خوبی ایجاد می کنند.

3. بر محصول نهایی تاکید می ورزند.

نقاط ضعف XP

1. ارتقای آن برای پروژه های بزرگ که در آنها ارائه سند ضروری است، مشکل است.

2. نیازمند تجربه و مهارت است.

3. برنامه ریزی به صورت جفتی هزینه بر است.

از آن جا که مدل های متنوعی از چرخه های دوام توسعه نرم افزار وجود دارد، هر کدام از این مدل ها دارای مزایا و معایب خاص خود هستند که بر اساس آنها باید تصمیم بگیریم چه مدلی را باید انتخاب نماییم. برای مثال، اگر ملزومات ما از قبل مشخص و به خوبی درک شده باشند وبخواهیم همیشه نظارت کاملی را بر پروژه داشته باشیم، آنگاه می توانیم از مدل آبشار استفاده کنیم. مدل های SDLC مختلف دارای ویژگی ها و ملزومات منحصربه فرد خود می باشند. بر اساس این جنبه ها، این مقاله به تحلیل تناسب مدل های SDLC گوناگون با کاربرد آن برای توسعه پروژه های نرم افزاری می پردازد. مدل فزاینده در بطن فرایند چرخه ای توسعه نرم افزار قرار دارد. این مدل با برنامه ریزی اولیه آغاز می شود و با آماده سازی تعاملات چرخه ای در میان پایان می یابد. آزمودن و رفع عیب کردناز این مدل در طول تکرار کوچک تر آسان تر است. مدیریت این مدل نیز آسان تر است، چون قسمت های خطرناک شناسایی شده و در طول تکرار آن رسیدگی به آن صورت می گیرد. مدل مارپیچی برای پروژه های حساس و بزرگ مثل پرتاب ماهواره مناسب است که در آنها تحلیل خطر بالایی ضروری است. مدل RAD انعطاف پذیر و قابل انطباق با تغییزات است، چون چرخه های کوچک توسعه را وارد می کند؛ به عبارت دیگر، کاربران محصول RAD را به

سرعت می بینند. این مدل شامل مشارکت کاربر و در نتیجه افزایش احتمال پذیرش زودهنگام کاربر در جامعه و کاشه کلی در خطر پروژه می شود.

جدول 1. تحلیل تناسب مدل های مختلف SDLC

SDLC	موقعیتی که در آن مناسب است
آبشار	- ملزومات ثابت - کار به صورت خطی به پیش می رود. - بدون محدودیت زمانی. - نمی تواند پروژه های حساس را مدیریت نماید.
XP	- پروژه های کوچک. - نماینده تمام وقت کاربر - اگر گروه نتواند قادر بخ پیش بینی هزینه و زمان بندی باشد، نمی تواند کار کند.
RAD	- محدودیت زمانی وجود دارد. - خطر زیادی وجود ندارد. - برای پروژه های کوچک تا متوسط مناسب است.
نمونه	- توسعه دهندگان قادر هستند چیزی را به سرعت ایجاد نمایند. - نمی تواند خطرات بالا را مدیریت نماید.
فزاینده	- خطرات فنی بالا. - محدودیت زمانی وجود دارد.
مارپیچی	- سیستم و نرم افزار با مقیاس گسترده. - سطح اعتبار بسیار بالا است. - لازم است تا در هر سطح تکاملی در مقابل خطر واکنش صورت گیرد.

جمع بندی

مدل های SDLC زیادی نظیر آبشار، RAD، مارپیچی، فزاینده، XP، نمونه و غیره وجود دارد که بسته به شرایط رایج، در سازمان های مختلف مورد استفاده قرار می گیرند. تمام این مدل های مختلف توسعه نرم افزاری دارای مزایا و معایب خاص خود هستند. در صنعت نرم افزار، مجموعه ای از این شیوه ها، با اندکی اصلاحات، مورد استفاده قرار می گیرد. در این مقاله، تلاش شده است تا مدل های مختلف SDLC با توجه به تناسب آنها با توسعه انواع گوناگون پروژه های نرم افزاری، بسته به محیط توسعه، تجزیه و تحلیل شوند. انتخاب مدل چرخه دوام صحیح در صنعت نرم افزار از

اهمیت زیادی برخوردار است و بر اساس نوع پروژه، تحلیل تناسب مدل های SDLC در انتخاب مدل مناسب SDLC برای پروژه ای خاص کمک خواهد کرد.

REFERENCES

[1] Jovanovich, D., Dogsa, T.,"Comparison of software development models," Proceedings of the 7th International Conference on, 11- 13 June 2003, ConTEL 2003, pp. 587-592.

[2] Laura C. Rodriguez Martinez, Manuel Mora, Francisco,J. Alvarez, "A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles", Proceedings of the 2009 Mexican International Conference on Computer Science IEEE Computer Society Washington,DC, USA, 2009.

[3] Roger Pressman, titled "Software Engineering - a practitioner's approach".

[4] A. M. Davis, H. Bersoff, E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", Journal IEEE Transactions on Software Engineering, Vol. 14, Issue 10, 1988

[5] Sanjana Taya, Shaveta Gupta, "Comparative Analysis of Software Development Life Cycle Models".

[6] Vishwas Massey, K.J Satao, " Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology".

[7] Klopper, R., Gruner, S., & Kourie, D. "Assessment of a framework to compare software development methodologies".

[8] Fowler, M. (2000), "Put Your Process on a Diet", Software Development".

[9] Sandeep Kumar et al, "Ontology Development and Analysis for Software Development Life Cycle Models".

[10] Manish Sharma, "A Survey of project scenario impact in SDLC models selection process".