به نام خداوند مهربان

به نام خداوند بخشنده و بخشایشگر

نام دانشجو:ابوالفضل امیری(مقطع کارشناسی)

مهندسی نرم افزار ۱

نام استاد:دکتر احمد فراهی

دانشگاه پیام نور

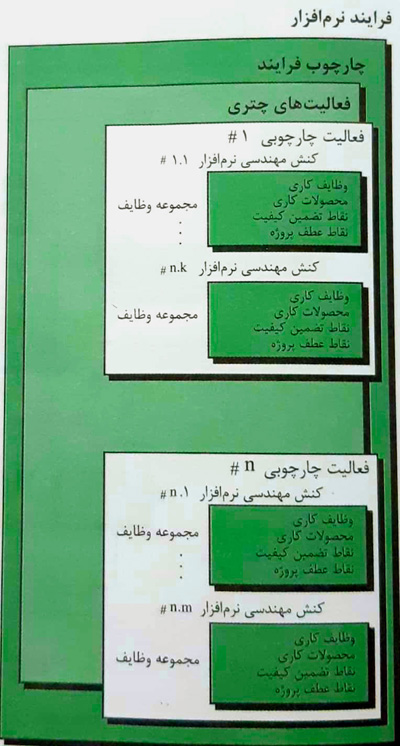
دانشکده فنی مهندسی

گروه مهندسی کامپیوتر و فناوری اطلاعات

مدل های**فرایند نرم افزار**.

**یک مدل فرایند کلی**

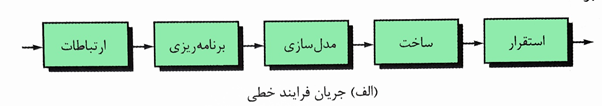
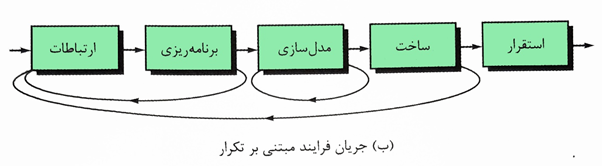
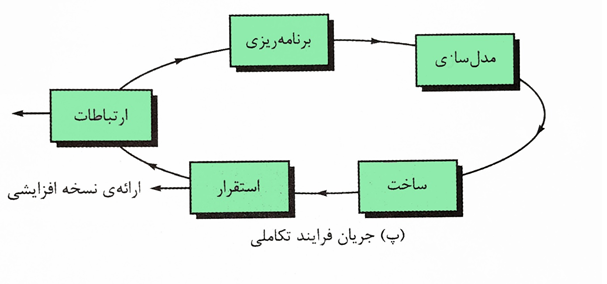
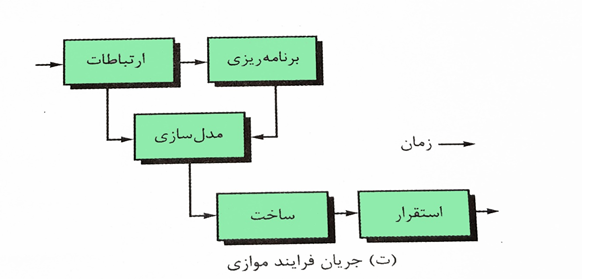
فرایند به عنوان مجموعه ای از فعالیت های کاری، کنش ها و وظایف تعریف می شود، که هنگام ایجاد یک محصول باید اجرا شوند. هر کدام از این فعالیت ها، کنش ها و وظایف در یک چارچوب یا مدل قرار دارند که رابطه آنها را با فرایند و با یکدیگر تعریف می کند. شکل زیر طرحی از فرایند نرم افزار را نشان می دهد.

شکل 1

با توجه به شکل 1 فرایند نرم افزار شامل یک چارچوب فرایند کلی است که شامل پنج فعالیت چارچوبیِ ارتباطات، برنامه ریزی، مدل سازی، ساخت و استقرار می شود. به علاوه فرایندهای نرم افزاری شامل مجموعه ای از فعالیت های چتری می شوند که عبارتند از: کنترل و پیگیری پروژه، مدیریت ریسک، تضمین کیفیت، مدیریت پیکربندی و بازبینی های فنی، که در سرتاسر پروژه به کار گرفته می شوند.

یک جنبه مهم از فرایند نرم افزار جریان فرایند است که شرح می دهد که فعالیت های چتری و کنش ها و وظایفی که در داخل هر فعالیت چارچوبی رخ می دهند از نظر ترتیب زمانی چگونه سازماندهی می شوند.

در یک جریان فرایند خطی، هر کدام از فعالیت های چارچوبی به ترتیب اجرا می شود، به طوری که با ارتباطات آغاز و به استقرار ختم می شود(شکل 2). در یک جریان فرایند مبتنی بر تکرار، پیش از رفتن به دور تکرار بعدی، یک یا چند فعالیت تکرار می شوند(شکل 3). در جریان فعالیت تکاملی، فعالیت ها به شیوه ای حلقوی اجرا می شوند. هر مدار از پنج فعالیت عبور می کند که به نسخه کامل تری از نرم افزار می انجامد(شکل 4). در جریان فرایند موازی(شکل 5) یک یا چند فعالیت به موازات سایر فعالیت ها انجام می شوند.

شکل 2شکل3شکل 4شکل 5

**تعریف یک فعالیت چارچوبی**

یک پرسش کلیدی در اینجا مطرح است که: اگر ماهیت مساله، خصوصیات افرادی که کار را انجام می دهند و طرف های ذی نفعی که پروژه را پشتیبانی می کنند، معلوم باشد، کدام کنش ها برای یک فعالیت چارچوبی مناسب خواهد بود؟

برای یک پروژه نرم افزاری کوچک که یک نفر با خواسته های صریح و ساده درخواست می کند، فعالیت برقراری ارتباط ممکن است شامل چیزی در حد یک تماس تلفنی با ذی نفع مورد نظر باشد. بنابراین تنها کنش لازم، مکالمه تلفنی است و وظایف کاری (مجموعه وظایف) که این کنش شامل آنها می شود، عبارتند از:

* بحث درباره خواسته ها و یادداشت برداشتن
* سازماندهی یادداشت ها در یک بیان مختصر از خواسته ها
* ارسال ایمیل برای بازبینی و تصویب

اگر پروژه به واسطه وجود تعداد زیادی از طرف های ذی نفع پیچیدگی بیشتری می یافت، که هر کدام مجموعه ای متفاوت از خواسته ها را دارند، فعالیت ارتباطات ممکن است شش کنش متمایز داشته باشند: شروع (inception)، استخراج (elicitation)، شناخت (elaboration)، مذاکره (negotiation)، تعیین مشخصات (specification) و اعتبارسنجی (validation). هر کدام از این کنش های مهندسی نرم افزار دارای چندین وظیفه کاری و تعدادی محصولات کاری متمایزند.

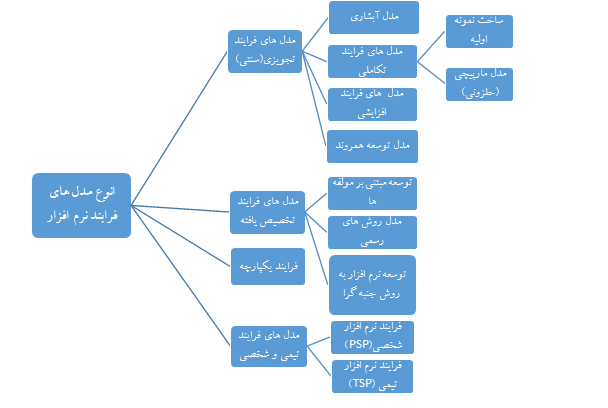
**تعیین مجموعه وظایف**

با رجوع به شکل یک، هر کنش مهندسی نرم افزار (مثلا استخراج که کنشی مرتبط با فعالیت ارتباطات است) را می توان با تعدادی از مجموعه وظایف متفاوت نشان داد که هر کدام مجموعه ای از وظایف کاری در مهندسی نرم افزار مرتبط با محصولات کاری، نقاط تضمین کیفیت و نقاط عطف پروژه اند. باید مجموعه وظایفی را برگزینید که نیازهای پروژه را به بهترین نحو برآورده سازد و خصوصیات تیم شما را دربرگیرد.

**انواع مدل های فرایند نرم افزار**

انواع مدل های فرایند نرم افزار شامل موارد زیر می شود:

* مدل های فرایند تجویزی(سنتی)
* مدل های فرایند تخصیص یافته
* فرایند یکپارچه
* مدل های فرایند تیمی و شخصی

انواع مدل های فرایند نرم افزار

**مدل های فرایند تجویزی(سنتی)**

در این بخش به بررسی روش های فرایند تجویزی خواهیم پرداخت، که در آن نظم و سازگاری پروژه، مسائل عمده به شمار می رود. ما آنها را تجویزی می نامیم زیرا مجموعه ای از عناصر فرایند- فعالیت های چارچوبی، عملیات مهندسی نرم افزار، وظایف، محصولات کاری، تضمین کیفیت و سازوکارهای کنترل تغییرات را برای هر پروژه تضمین می کنند. هر مدل فرایند همچنین یک جریان فرایند(یا جریان کاری) را تعریف می کند. که شیوه ارتباط میان عناصر فرایند را تعریف می کند. این مدل ها شامل موارد زیر هستند:

* مدل آبشاری
* مدل های فرایند افزایشی
* مدل های فرایند تکاملی
* مدل توسعه همروند

**مدل آبشاری**

گاه پیش می آیدکه خواسته های مربوط به یک مسئله به خوبی شناخته شده اند- هنگامی که کار به طریق خطی، از برقراری ارتباط تا استقرار جریان پیدا می کند(شکل 6). این وضعیت هنگامی مشاهده می شود که تطبیق خوش تعریف یا بهسازی یک سیستم موجود ضرورت پیدا می کند.

شکل 6

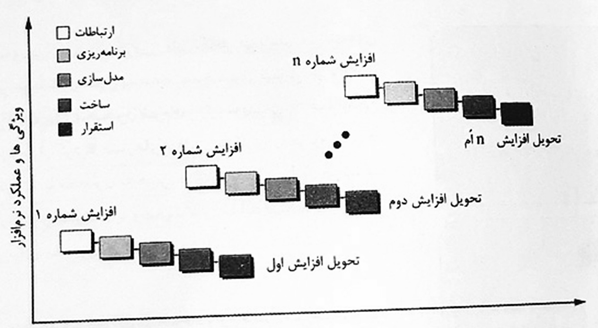
در مدل آبشاری که گاه از آن به عنوان چرخه حیات کلاسیک یاد می شود، روشی سیستماتیک و ترتیبی برای توسعه نرم افزار پیشنهاد می شود که با مشخص کردن خواسته ها توسط مشتری آغاز می شود و از طریق برنامه ریزی، مدل سازی، ساخت و استقرار پیش می رود و در پشتیبانی مستمر نرم افزار کامل شده به اوج می رسد.

از جمله مشکلاتی که به هنگام اجرای مدل ترتیبی خطی پیش می آید، می توان به موارد زیر اشاره کرد:

* پروژه های واقعی به ندرت جریان ترتیبیِ پیشنهاد شده توسط این مدل را دنبال می کنند. گر چه مدل خطی می تواند پذیرای تکرار باشد، این عمل را به طور غیرمستقیم انجام می دهد. در نتیجه، با پیش رفتن پروژه، ممکن است تغییرات باعث سردرگمی شوند.
* غالبا برای مشتری دشوار است که همه نیازهای خود را به وضوح بیان کند. مدل ترتیبی خطی، به بیان واضح نیاز دارد و به خوبی از پس موارد غیر قطعی که در آغاز اکثر پروژه ها وجود دارند بر نمی آید.
* مشتری باید حوصله داشته باشد. یک نسخه کاری از برنامه ها تا آخرین روزهای پروژه در دسترس او قرار نخواهد گرفت. یک اشتباه عمده که تا زمان بازبینی برنامه کاری از دید پنهان بماند، می تواند بسیار دردسر آفرین باشد.

**مدل های فرایند افزایشی**

با رجوع به شکل 7، مشاهده می شود که مدل افزایشی، مراحل ترتیبی خطی را به شیوه ای باورنکردنی با پیشرفت تقویم اجرا می کند، هر ترتیب خطی یک ((افزایش)) قابل تحویل از نرم افزار را ارائه می دهد.

شکل 7

در واقع در مدل فرایند افزایشی در هر گام یک نسخه از نرم افزار در اختیار مشتری قرار می گیرد که این نسخه نسبت به نسخه های قبلی خود کامل تر می باشد و خدمات بیشتری به مشتری ارائه می دهد. در تولید هر کدم از این نسخه ها از عناصر مدل ترتیبی خطی استفاده می شود.

این مدل به تحویل قطعه ای در هر گام تاکید می ورزد، قطعات اولیه نسخه ((دست و پا شکسته ای)) از محصول نهایی هستند ولی قابلیت ارائه خدمات به کاربر را دارند.

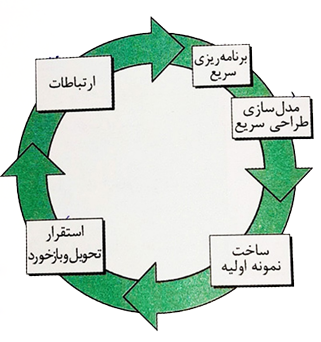
**مدل های فرایند تکاملی**

مدل های فرایند تکاملی شامل موارد زیر هستند:

* ساخت نمونه اولیه
* مدل مارپیچی(حلزونی)

**مدل ساخت نمونه اولیه**

الگوی ساخت نمونه اولیه با جمع آوری خواسته ها آغاز می شود. مشتری اهداف کلی نرم افزار را مشخص می کند، سپس یک طراحی سریع صورت می پذیرد. این طراحی سریع منجر به ساخت نمونه اولیه می شود. نمونه اولیه مورد ارزیابی مشتری قرار می گیرد و از آن برای پالایش خواسته های نرم افزار مورد نظر استفاده می شود. در حالت ایده آل نمونه اولیه به عنوان راهکاری برای تشخیص خواسته های نرم افزار عمل می کند.

شکل 8

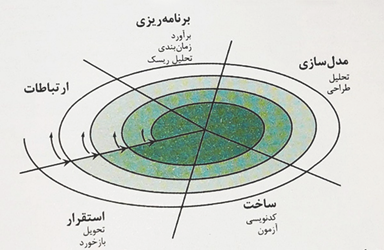
ساخت نمونه اولیه به دلایل زیر مشکل آفرین است:

* مشتری فکر می کند که با کمی تغییر در نمونه اولیه می توان از آن به عنوان محصول نهایی استفاده کرد، در حالی که او نمی داند این نمونه اولیه بدون در نظر گرفتن کیفیت کلی نرم افزار و قابلیت نگهداری دراز مدت و با شتاب طراحی و ساخته شده است.
* مهندس نرم افزار غالبا برای به کارگیری هر چه سریع تر نمونه اولیه، در پیاده سازی دقیق آن کوتاه می آید. ممکن است از الگوریتم ها و زبان های برنامه نویسی ناکارآمد استفاده کند. پس از مدتی ممکن است برنامه نویس با این انتخاب ها مانوس شود و کلا فراموش کند که چرا نامناسب بوده اند، انتخاب کمتر از ایده آل اکنون به بخشی از سیستم تبدیل شده است.

ممکن است این مشکلات به وجود آید، ولی ساخت نمونه اولیه می تواند الگوی موثری برای مهندسی نرم افزار باشد. کلید کار، تعیین قواعد بازی در همان آغاز است. یعنی افراد ذی نفع و سازنده هر دو باید بپذیرند که نمونه اولیه بدین منظور ساخته می شود که به عنوان سازوکاری برای تعیین خواسته ها عمل کند. سپس دور انداخته می شود و نرم افزار واقعی با مد نظر قرار دادن کیفیت، مهندسی می شود.

**مدل مارپیچی(حلزونی)**

مدل مارپیچی یک مدل فرایند تکاملی است که ماهیت تکراری مدل ساخته نمونه اولیه را با جنبه های کنترلی و سیستماتیک مدل ترتیبی خطی (آبشاری) تلفیق می کند. با استفاده از این مدل، نرم افزار به صورت یک سری از نگارش های تکاملی توسعه می یابد. طی نخستین دورهای تکرار، نگارش تکاملی، ممکن است یک مدل کاغذی یا یک نمونه اولیه باشد. طی تکرارهای بعدی، هر بار نسخه کامل تری از سیستم، مهندسی شده و تولید می شود.

شکل 9

نخستین مدار مارپیچ ممکن است منجر به ایجاد مشخصه ای از محصول گردد، ولی عبورهای بعدی در اطراف مارپیچ برای ایجاد یک نمونه اولیه و سپس نسخه های پیچیده تری از نرم افزار به کار می رود. هر بار گذر از ناحیه برنامه ریزی، منجر به تنظیم دوباره طرح پروژه می شود.

هزینه و زمانبندی بر اساس بازخورد حاصل از ارزیابی مشتری تنظیم می شود. به علاوه مدیر پروژه تعداد تکرارهای مورد نیاز برای کامل شدن نرم افزار را تنظیم می کند.

مدل مارپیچی یک روش واقع گرا برای توسعه نرم افزارها و سیستم هایی در مقیاس انبوه است. از آنجا که نرم افزار به موازات پیشرفت فرایند، تکامل می یابد، سازنده و مشتری در هر سطح تکامل ریسک را بهتر درک کرده به آن واکنش نشان می دهند. مدل مارپیچی از ساخت نمونه اولیه به عنوان راهکاری برای کاهش ریسک استفاده می کند، ولی مهم تر آنکه سازنده را قادر می سازد تا روش ساخت نمونه اولیه را در هر مرحله از تکامل محصول به کار بندد.

**مدل های فرایند تخصیص یافته**

این مدل ها شامل موارد زیر می شود:

* توسعه مبتنی بر مولفه ها
* مدل روش های رسمی
* توسعه نرم افزار به روش جنبه گرا

**توسعه مبتنی بر مولفه ها (Component-Based Development)**

مولفه های نرم افزاری آماده(COTS)، توسط عده ای از فروشندگان این مولفه ها توسعه داده می شوند. عملکرد مورد نظر را با واسط های مناسب فراهم می آورند، به طوری که مولفه را می توان به خوبی در سیستم در حال ساخت الحاق کرد. توسعه مبتنی بر مولفه ها بسیاری از خصوصیات مدل مارپیچی را در بر می گیرد. ماهیتی تکاملی دارد و برای ایجاد نرم افزار به رویکردی تکراری نیاز دارد. به هر حال، در مدل توسعه مبتنی بر مولفه ها، برنامه های کاربردی از به هم پیوستن مولفه های نرم افزار آماده ساخته می شوند.

مدل توسعه مبتنی بر مولفه ها شامل مراحل زیر می شود:

1. محصولات مبتنی بر مولفه موجود، از نظر دامنه کاربرد مورد نظر بررسی و ارزیابی می شوند.

2. مسائل مربوط به انسجام مولفه ها در نظر گرفته می شوند.

3. برای چیدمان مولفه ها یک معماری نرم افزار طراحی می شود.

4. مولفه ها در این معماری قرار داده می شوند.

5. آزمونی جامع برای حصول اطمینان از عملکرد درست، به عمل می آید.

مدل توسعه مبتنی بر مولفه ها، استفاده مجدد از نرم افزار را میسر می سازد. اگر استفاده مجدد از مولفه ها فرهنگ سازی شود، تیم مهندسی نرم افزار شما می تواند به کاهش زمان چرخه توسعه و نیز کاهش هزینه های پروژه دست پیدا کند.

**مدل روش های رسمی (Formal Methods Model)**

روش های رسمی، مهندس نرم افزار را قادر می سازند تا با اعمال یک نظم ریاضی شدید، سیستم کامپیوتری را مشخص کند، بسط دهد و وارسی نماید.

با استفاده از این مدل، ابهام، ناقص بودن و ناسازگاری را می توان راحت تر کشف و تصحیح کرد، نه از طریق بازبینی خاص، بلکه از طریق به کارگیری تحلیل ریاضی. هنگامی که روش های رسمی در اثنای طراحی به کار برده می شوند، به عنوان مبنایی برای وارسی برنامه عمل کرده از این رو مهندس نرم افزار را قادر به کشف و تصحیح خطاهایی می سازند که ممکن بود در غیر این صورت مخفی بمانند.

ملاحظات مربوط به قابلیت اجرای این مدل در محیط های تجاری شامل موارد زیر می شود:

* توسعه مدل های رسمی در حال حاضر بسیار وقت گیر و پرهزینه است.
* از آنجا که تعداد معدودی از نرم افزار سازان دارای زمینه لازم برای اجرای روش های رسمی هستند، آموزش گسترده ای مورد نیاز است.
* استفاده از مدل ها به عنوان راهکار ارتباطی با مشتریانی که دید فنی ندارند، دشوار است.

نظر به این ملاحظات، روش های رسمی احتمالا در میان نرم افزار نویسانی هوادار پیدا می کند که باید نرم افزارهای ایمنی-حیاتی (safety-critical) (مثلا نرم افزارهای دستگاههای پزشکی و هوافضا) بسازند یا در میان آنهایی که در صورت بروز خطا در نرم افزار دستخوش زیان های اقتصادی کلان می شوند.

**توسعه نرم افزار به روش جنبه گرا (Aspect Oriented)**

هر فرایند نرم افزار که انتخاب شود، سازندگان نرم افزارهای پیچیده، مجموعه ای از ویژگی ها، عملکردها و محتویات اطلاعاتی متمرکز را پیاده سازی می کنند. این خصوصیات نرم افزاری متمرکز به صورت مولفه هایی (مثلا کلاس های شئ گرا) مدل سازی و سپس در حیطه ی یک معماری سیستم بنا می شوند. با پیچیده تر شدن سیستم های کامپیوتری مدرن، دغدغه های خاصی کل معماری را در بر می گیرد. برخی از این دغدغه ها خواص سطح بالای سیستم (نظیر امنیت و تحمل پذیری خطا) هستند. عده ای دیگر بر وظایف سیستم تاثیر می گذارند (مثل اعمال قواعد تجاری) در حالی که عده ای هم سیستماتیک هستند (مانند همزمان سازی وظایف یا مدیریت حافظه).

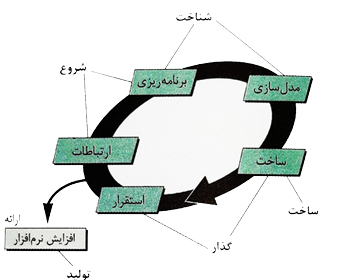
هنگامی که این دغدغه ها در وظایف، ویژگی و اطلاعات سیستمی با یکدیگر تلاقی می کنند، غالبا از آنها به عنوان دغدغه های متلاقی (Crossing Concerns) یاد می شود. خواسته های جنبه گرا، آن دسته از دغدغه های متلاقی را تعریف می کنند که بر معماری نرم افزار تاثیر می گذارند. توسعه نرم افزار به روش جنبه گرا (AOSD)، که از آن غالبا به عنوان برنامه نویسی جنبه گرا نیز یاد می شود، یک الگوی مهندسی نسبتا جدید است که رویکردی فرایندی و روش شناختی برای تعریف، مشخص سازی، طراحی و ساخت جنبه ها ارائه می دهد.((سازوکارهایی به غیر از زیر روال ها و وراثت برای متمرکز ساختن بیان یک دغدغه ی متلاقی)).

**فرایند یکپارچه (Unified Process)**

اوایل دهه 1990، جیمز رومباف، گرادی بوچ و ایوار جیکابسون کار روی یک ((روش یکپارچه)) را آغاز کردند که بهترین ویژگی های هر کدام از روش های طراحی و تحلیل شئ گرا را تلفیق می کرد و ویژگی هایی از سایر کارشناسان در مدل سازی شئ گرا را بر آن منطبق می ساخت. نتیجه، زبان مدل سازی یکپارچه (uml) است که حاوی یک نمادگذاری قدرتمند برای مدل سازی و توسعه ی سیستم های شئ گرا است.

زبان مدل سازی یکپارچه (uml) فن آوری لازم برای پشتیبانی از مهندسی نرم افزار شئ گرا را فراهم ساخت، ولی چارچوب فرایندی لازم را برای راهنمایی تیم های پروژه در به کارگیری این فن آوری ارائه نداد. طی چند سال بعد، جیکابسون، ریمباف و بوچ، فرایند یکپارچه را توسعه دادند که چارچوبی برای مهندسی نرم افزار شئ گرا با استفاده از uml است.

پیش تر پنج فعالیت چارچوبی کلی را مورد بحث قرار دادیم و استدلال کردیم که از آنها می توان برای توصیف هر مدل فرایند نرم افزار استفاده کرد. فرایند یکپارچه نیز از این قاعده مستثنی نیست. در شکل 10، مراحل up و ارتباط آنها با فعالیت های چارچوبی کلی، تصویر شده است.

شکل 10

مرحله آغازین در up شامل هر دو فعالیت برقراری ارتباط با مشتری و برنامه ریزی می شود. از طریق همکاری با طرف های ذی نفع، خواسته های تجاری برای نرم افزار شناسایی می شود، یک معماری تقریبی برای سیستم پیشنهاد می شود و برنامه ریزی بنیادی برای ماهیت تکراری و افزایشی پروژه آتی توسعه داده می شود.

مرحله شناخت شامل فعالیت های برقراری ارتباط و مدل سازی در مدل فرایند کلی می شود. در این مرحله، ((use case های)) مقدماتی که به عنوان بخشی از مرحله آغازین ایجاد شدند، پالایش یافته و بسط داده می شوند، به علاوه، در این مرحله نمایش معماری نیز بسط داده می شود تا پنج نمای متفاوت نرم افزار را در برگیرد- این پنج نما عبارتند از مدل use case، مدل خواسته ها، مدل طراحی، مدل پیاده سازی و مدل استقرار.

مرحله ساخت در up، هم ارز فعالیت ساخت است که برای فرایند نرم افزار کلی تعریف شد. مرحله ساخت با استفاده از مدل معماری به عنوان ورودی، مولفه های نرم افزاری را که هر کدام از use caseها را عملیاتی می کنند، ایجاد می کند یا آنها را توسعه می دهد.

مرحله گذار در up شامل آخرین مراحل در فعالیت ساخت در مدل کلی و اولین بخش از استقرار در مدل کلی می شود. نرم افزار برای آزمون بتا به کاربران نهایی داده می شود و بازخورد به دست آمده از کاربران هم نقایص و هم تغییرات لازم را گزارش می کند. در پایان مرحله گذار، هر ((افزایش)) نرم افزار به یک نسخه قابل استفاده تبدیل می شود.

مرحله تولید در up منطبق بر فعالیت استقرار در فرایند کلی است. طی این مرحله، استفاده از نرم افزار، پایش می شود، محیط عملیاتی (زیرساخت) پشتیبانی می شود و گزارش نقایص و درخواست برای تغییرات، تسلیم و ارزیابی می شود.

**مدل های فرایند تیمی و شخصی**

این مدل ها شامل موارد زیر هستند:

* فرایند نرم افزار شخصی (PSP)
* فرایند نرم افزار تیمی (TSP)

**فرایند نرم افزار شخصی (PSP)**

فرایند نرم افزار شخصی بر اندازه گیری شخصی محصول کاری تولید شده و کیفیت حاصل از محصول کاری تاکید دارد. به علاوه، در PSP مجری کار مسئول برنامه ریزی پروژه است و کنترل کیفیت همه محصولات کاری نرم افزاری ساخته شده بر عهده خود اوست. در مدل PSP پنج فعالیت چارچوبی تعریف می شود:

* برنامه ریزی: در این فعالیت، خواسته ها شناسایی می شود، و برآورد منابع و تعیین اندازه پروژه انجام می شود. به علاوه برآوردی از نقایص (تعداد نقایص پیش بینی شده برای کار) به عمل می آید. همه معیارها روی کاربرگ ها یا قالب ها ثبت می شوند. سرانجام، وظایف لازم برای توسعه نرم افزار تعیین و زمانبندی پروژه انجام می شود.
* طراحی سطح بالا: مشخصات خارجی برای هر کدام از مولفه هایی که قرار است تعیین شود و طراحی مولفه ها انجام می شود. نمونه های اولیه ساخته می شود، در حالی که هنوز عدم قطعیت هایی موجود است. همه مسائل و مشکلات، ثبت و پیگیری می شوند.
* مرور طراحی سطح بالا: روش های وارسی رسمی برای یافتن خطاهای طراحی، اعمال می شوند. معیارهای مربوط به وظایف مهم و نتایج کار، نگهداری می شود.
* توسعه: طراحی سطح بالا پالایش و بازبینی می شود. کدها تهیه، بازبینی، کامپایل و آزموده می شوند. معیارها برای کلیه وظایف مهم و نتایج کار، حفظ می شوند.
* پایان کار: با استفاده از معیارها و موازین جمع آوری شده (که مقادیر چشم گیری از داده ها را شامل می شود و باید مورد تحلیل آماری قرار گیرد)، اثربخشی فرایند تعیین می شود. موازین و معیارها باید راهنمایی برای اصلاح فرایند و بهبود بخشیدن به اثربخشی آن فراهم آورند.

در PSP تاکید بر شناسایی زودهنگام خطاهاست و شناخت انواع خطاهایی که احتمال ارتکاب آنها وجود دارد نیز به همان اندازه اهمیت دارد. این هدف از طریق یک فعالیت ارزیابی طاقت فرسا قابل دستیابی است که باید روی کلیه محصولات کاریِ تولید شده اجرا شود.

**فرایند نرم افزار تیمی(TSP)**

هدف TSP تشکیل یک تیم پروژه ((خود هدایت گر)) است که سازماندهی برای تولید نرم افزارهای پر کیفیت را خود عهده دار می شود. فعالیت های زیر برای TSPتعریف می شود:

* تشکیل تیم های خود هدایت گری که کار خود را برنامه ریزی و پیگیری می کنند. اهداف را تعیین می کنند و خود به تعیین فرایندها و طرح ها اقدام می نمایند. این تیم ها می توانند تیم های نرم افزاری محض یا تیم های محصولات انسجام یافته (IPT) شامل سه تا حدود بیست مهندس باشند.
* نشان دادن شیوه راهبردی و ایجاد انگیزه در تیم ها به مدیران و چگونگی کمک به آنها در حفظ حداکثر ۱
* شتاب بخشیدن به بهبود فرایند نرم افزار با نهادینه ساختن CMM سطح5. (مدل بلوغ شایستگی ها (CMM) که میزانی از اثر بخشی یک فرایند نرم افزار است).
* فراهم ساختن دستورالعمل بهسازی برای سازمان های بالغ
* تسهیل آموزش دانشگاهی مهارت های تیمی در سطح صنعتی

یک تیم ((خود هدایت گر)) درک سازگاری از اهداف و مقاصد خود دارد. نقش ها و مسئولیت ها را برای هر عضو تیم تعریف می کند، داده های کمی پروژه (مربوط به بهره وری و کیفیت) را زیر نظر دارد، تیم پروژه ای را تعیین می کند که برای تیم مناسب باشد. و برای پیاده سازی فرایند، یک راهبرد تعیین می کند، استانداردهای محلی قابل استفاده را برای کار مهندسی نرم افزار تعریف می کند، خطرات را پیوسته ارزیابی می کند و به آن واکنش نشان می دهد، و سرانجام اینکه وضعیت پروژه را پیگیری، مدیریت و گزارش می کند.

در TSP، فعالیت های چارچوبی زیر تعریف می شود: آغاز پروژه، طراحی سطح بالا، پیاده سازی، انسجام دهی و آزمون، و پایان کار. این فعالیت ها مانند همتاهای خود در PSP (توجه دارید که اصطلاحات قدری تفاوت دارند)، تیم را قادر به برنامه ریزی، طراحی و ساخت نرم افزار به شیوه ای منضبط می کند، در حالی که در عین حال، اندازه گیری کمی فرایند و محصول انجام می شود. مرحله پایان کار صحنه را برای بهسازی فرایند آماده می کند.

**نتیجه گیری**

اگر فرایند انتخابی ما برای یک پروژه ضعیف باشد، محصول نهایی بدون شک ضعیف خواهد بود. از این رو آگاهی و دانش در مورد انواع مدل های فرایند نرم افزاری امری لازم و ضروری می نماید، تا با انتخاب درست و بجای یک فرایند نرم افزاری بتوان نرم افزارهای قدرتمندی را توسعه داد. امید است که این مقاله بتواند به شما در توسعه پروژه های نرم افزاری یاری رساند. در مقاله بعد سعی می کنیم مطالب مرتبط با توسعه چابک را مطرح نموده و بیشتر با آن آشنا شویم.

**منبع:**ویراست هفتم کتاب مهندسی نرم افزار نوشته راجر اس پرسمن