درس مهندسی نرم افزار

آشنایی با

Rational Unified Process (RUP)

تهیه و تنظیم: غفور علیپور

مهر ۸۶

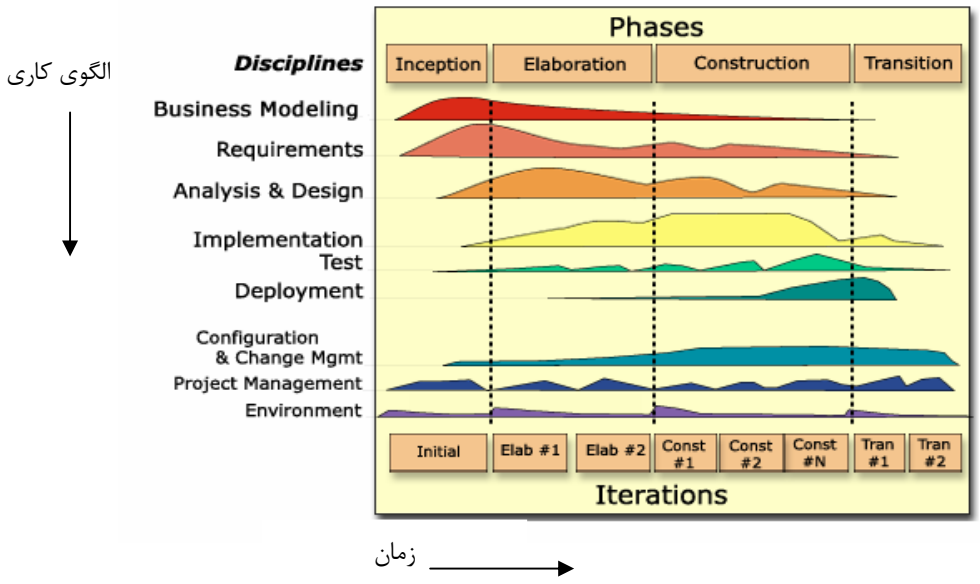
مقدمه

RUP فرآیند تولید نرم افزار می باشدکه فعالیت ها و مسئولیت ها را به ترتیب جهت توسعه نرم‌افزار مشخص می‌کند و هدف آن حصول اطمینان از تولید نرم افزار با کیفیت بالا براساس نیازهای کاربران و با توجه به زمان و بودجه در دسترس می باشد.

تاریخچه RUP

با پیشرفت تکنولوژی کامپیوتر، نیاز هرچه بیشتر به گسترش علوم نرم افزاری نیز احساس می‌شد،لذا متدولوژی‌های گوناگون تعریف شده و بمرور تکامل یافت .مهم‌ترین آنها متدولوژی‌های ساخت‌یافته بالاخص SSADM با نگرش آبشاری بود. در ابتدا، این روشها مناسب بوده و جوابگوی نیازهای آن زمان بودند، ولی با افزایش داده‌ها و پیدایش مفاهیمی همچون شبکه، Web و ... دیگر کارآیی لازم را جهت پیاده‌سازی و هدایت پروژه‌های نرم افزاری نداشتند. پس مفاهیم برنامه نویسی شیء گرا پا به عرصه وجود گذاشت و در سال ۱۹۹۱ بطور جدی مورد مطالعه و بحث قرار گرفت. استفاده از این روشها و متدهای برنامه نویسی قدرت و انعطاف بسیاری را به برنامه‌ها داد و شرکتهای نرم افزاری توانستند با کاهش هزینه‌ها و بهینه سازی کدهای خود، نرم افزارهای قویتری را به بازار عرضه کنند، ولی این روش جدید نیز نیاز به مدیریت و یکپارچگی داشت. پس روشها و متدولوژیهای جدیدی مطرح شد که شامل Booch، OMT، OSE و ... بودند. در سال ۲۰۰۰ شرکت Rational روشی را تحت عنوان RUP مطرح ساخت که بعد از روش MSF شرکت مایکروسافت، به دنیای نرم افزار عرضه شد و امروزه از طرفداران بسیاری برخوردار است.

RUP چیست؟



RUP فرآیند تولید نرم افزار می باشدکه فعالیت ها و مسئولیت ها را به ترتیب جهت توسعه نرم‌افزار مشخص می‌کند و هدف آن حصول اطمینان از تولید نرم افزار با کیفیت بالا براساس نیازهای کاربران و با توجه به زمان و بودجه در دسترس می باشد.

RUP داری دو بعد است(شکل بالا):

۱- محور افقی یا بعد زمان که فازهای چرخه عمر فرآیند را به ترتیب در طول زمان نشان می‌دهد.

۲- محور عمودی که دیسیپلین‌ها یا الگوهای کاری را نشان می‌دهد.

بعد اول جنبه دینامیکی فرآیند را نمایش می‌دهد، که در قالب فازها(Pase)، تکرار و توالی(iteration) و نقاط کلیدی(milestone) تشریح می‌شود.

بعد دوم جنبه استاتیکی فرآیند را نمایش می‌دهد که در قالب اجزای فرایند، الگوهای کاری، فعالیتها، جریانهای کاری، محصولات و نقشها(role) تشریح می‌شود.

از مزایای RUP می‌توان به این موضوع اشاره نمود که چون می‌تواند بر پایه Web باشد، پس قابل Customize است و می‌توان آنرا جهت استفاده همگان بر روی سایت قرارداد. از طرفی مثل هر نرم افزار شیءگرای دیگری، با UML کارکرده و رشد می‌کند.

یکی از خصوصیات مهم RUP توالی و تکرار می باشد. در روش‌های سنتی تحلیل و طراحی یک سیستم نرم‌افزاری، مراحل کار (Workflow) به‌صورت ترتیبی (Seqeuential) بوده و هر یک از آنها یک و فقط یک‌بار انجام می شود. این روش تحلیل و طراحی "روش آبشاری" (Waterfall) نامیده می شود.

در این روش کلیه مسایلی که هنگام تجزیه تحلیل سیستم از قلم افتاده است در هنگام پیاده‌سازی سیستم رو شده و باعث یک توقف در روند پیشرفت پروژه می‌گردند و در ادامه کار اختلال به‌وجود می‌آورند. به طوری که پروژه ناگزیر در چرخه رفع ایرادات (bug-fix cycle) بسیار سختی قرار می‌گیرد.

بهترین و کاراترین روش برای جلوگیری از وقوع چنین پیش‌آمدهایی این است که در طی مراحل تجزیه، طراحی و ساخت پروژه نرم‌افزاری بازگشت‌هایی (feedback) به مراحل قبلی انجام گردد. در این صورت نیازمندیهای سیستم بهتر درک شده و سیستم از ساختار بسیار قوی‌تری برخوردار می‌شود. به این ترتیب مراحل ساخت سیستم، تدریجی و به‌صورت بسیار کاملتری انجام می‌گیرد. فازهای مختلف پروژه در طی این تکرارها (iteration) رفته رفته تکمیل(Incremental)و نهایی می‌گردند.

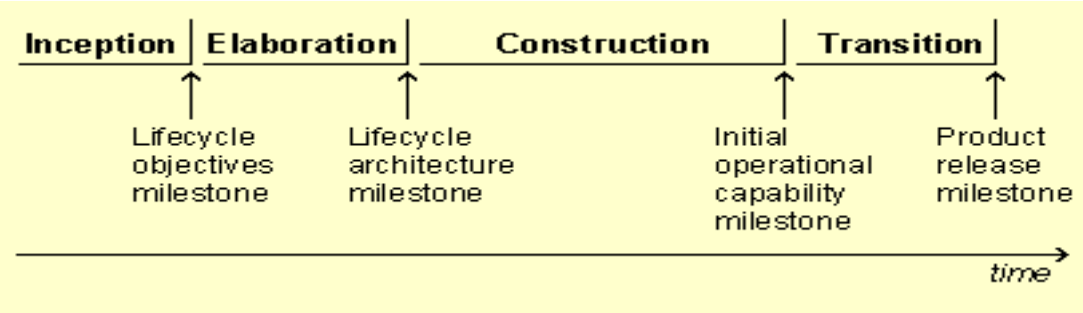
در هر تکرار (iteration) نسخه‌ای (release) اجرایی و پایدار از سیستم نهایی حاصل می‌شود. با وجودی که ممکن است این نسخه محصول کامل و جامعی نباشد ولی در دید کاربر و مهندسی سیستم بسیار مؤثر است.

شکل قبلی نشان می‌دهد که میزان اهمیت الگوهای کاری در طول زمان متغیر است. برای مثال در توالی تکرارهای اولیه، زمان بیشتری روی درخواست‌ها صرف می‌شود، ولی در تکرارهای نهایی، پیاده‌سازی زمان بشتری نیاز دارد.

بسته به حجم پروژه های نرم‌افزاری و نوع کار، تعداد تکرارها در فازهای مختلف متفاوت می‌باشد.

نگرش زمان درRUP

از دیدگاه مدیریت سیستم، چرخه زندگی یک نرم‌افزار با استفاده از متدولوژی RUP در طول زمان به ۴ فاز متوالی اصلی تقسیم می‌شود که هر فاز با یک نقطه کلیدی(milestone) به پایان می‌رسد.(شکل زیر)

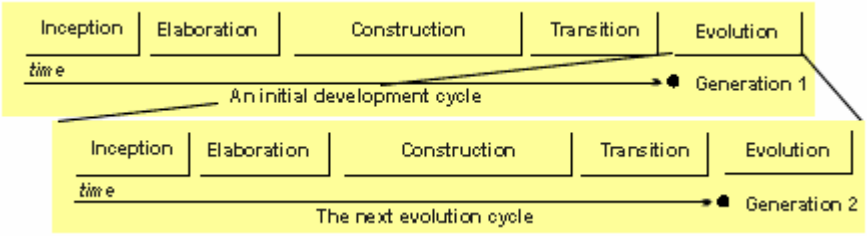


در پایان هر فاز یک ارزیابی انجام می گیرد، تا مشخص شود که آیا اهداف فاز بدست آمده‌اند یا نه. اگر نتیجه ارزیابی رضایت‌بخش بود، پروژه به فاز بعدی حرکت می کند.

برنامه‌ریزی (schedule) و فعالیت یا کار انجام شده (effort) در فازهای مختلف RUP یکسان نیستند. هر چند این تفاوتها عمدتا به پروژه بستگی دارند، اما یک چرخه توسعه اولیه برای پروژه‌ای با اندازه متوسط، معمولا از توزیع زیر بین زمان‌بندی(برنامه‌ریزی) و کار مطابقت می کند.

	Inception	Elaboration	Construction	Transition
Effort	~5 %	20 %	65 %	10%
Schedule	10 %	30 %	50 %	10%

RUP دارای ابزارهایی (tool) است که می‌توان از آنها استفاده کرد تا برخی از فعالیت‌های فاز ساخت را به صورت اتوماتیک انجام داد (مانند Rational rose). این ابزار فاز ساخت (construction) را ساده‌تر و کوتاه‌تر می کنند. در نتیجه این فاز کوتاه‌تر از فاز شناخت و معماری می‌شود. اولین باری که این ۴ فاز به‌ترتیب برای پروژه‌ای انجام شود، یک چرخه توسعه(development cycle) بوجود می‌آید و در هر بار اجرا محصول یا نسلی(generation) از نرم‌افزار بوجود می‌آید. به غیر از مواردی پروژه متوقف می‌شود (می‌میرد)، پروژه با تکرار همان توالی فازها رشد می کند، اما این بار میزان تاکید روی فازها، متفاوت است. چرخه‌های بعدی چرخه‌های تکاملی(evolution cycles) نامیده می‌شوند.



چرخه‌های تکاملی ممکن است با بهبودهای پیشنهاد شده توسط کاربران، تغییر در زمینه کاربر، تغییر در تکنولوژی مورد نظر، واکنش به رقبا و غیره، ایجاد و اجرا شوند. چرخه‌های تکاملی معمولا فازهای شناخت و معماری کوتاهتری دارند. چون معماری و تعریف محصول اولیه در چرخه‌های توسعه قبلی مشخص شده است. مگر در مواردی که محصول یا معماری دوباره تعریف شوند.

فازهای RUP

در ادامه نگرش زمان در RUP یعنی هر یک از فازهای آن به ترتیب تشریح می‌شود و همه فعالیتها (activity)، خروجی‌ها (artifact) و نقش‌های (role) مربوط به هر یک از فازها شرح داده می‌شود.

Inception (شناخت)

در این فاز مشکلات کلان سیستم تعیین شده و محدوده، چشم انداز عملیاتی و معیارهای پذیرش سیستم مشخص می گردد. در این فاز همچنین برآورد میزان ریسک‌پذیری، تشخیص منابع مورد نیاز و زمان‌بندی گام‌های اساسی توسعه سیستم تعیین می‌شود. هدف اصلی این فاز انجام یک امکان‌سنجی و شناخت کامل از وضعیت سیستم موجود، در کنار کسب رضایت کلیه ذینفعان در مسیر انجام پروژه می‌باشد. فاز شناخت برای توسعه سیستم‌های جدید مهم است. برای پروژه‌هایی که روی توسعه سیستم موجود متمرکز هستند، این فاز کوتاهتر و دارای اهمیت کمتری است. اما هنوز این اطمینان باید حاصل شود که آیا پروژه ارزش سرمایه‌گذاری و امکان انجام دارد یا نه.

اهداف اصلی فاز شناخت عبارتند از:

۱- ایجاد محدوده و شرایط مرزی نرم‌افزار: شامل چشم‌انداز عملیاتی، معیارهای پذیرش و تعیین اینکه چه‌چیزی در محصول باشد و چه‌چیزی نباشد.

۲- تخمین هزینه کلی و زمانبندی کلی پروژه (تخمین‌های جزئی‌تر بلافاصله در فاز معماری دنبال می‌شود.)

۳- برآورد ریسک‌های موجود و بالقوه

۴- تدارک محیط پشتیبانی برای پروژه

۵- مشخص کردن موارد بحرانی استفاده از سیستم

فعالیت‌های اساسی فاز شناخت عبارتند از:

- فرموله کردن دامنه پروژه، که شامل تعیین زمینه کسب وکار و مهمترین نیازمندیها و محدودیت‌ها تا حدی که بتوان معیار پذیرش محصول نهایی را استخراج کرد.
- برنامه‌ریزی و آماده کردن مورد کسب و کار، شامل ارزیابی جایگزین‌های مدیریت ریسک، کارکنان، برنامه‌ریزی پروژه و موارد قابلیت سودآوری/زمان‌بندی/هزینه
- ترکیب یک معماری انتخاب شده، شامل ارزیابی مواردی در طراحی، ساخت/خرید/استفاده دوباره و تخمین هزینه، زمان‌بندی و منابع. هدف از انجام این فعالیت اثبات موجه بودن پروژه با استفاده از برخی دلایل مفهومی است. شاید در اینجا لازم باشد شکلی از مدل را برای تعیین اینکه چه‌چیزی مورد نیاز است، شبیه‌سازی کنیم، و یا شکل اولیه‌ای(prototype) از پروژه را برای تشریح مناطقی با ریسک بالا، ایجاد کنیم. مدل‌سازی اولیه در فاز شناخت باید محدود به کسب اطمینان از اینکه آیا راه‌حل پیشنهادی برای توسعه سیستم امکان‌پذیر است یا نه، باشد. سیستم(راه‌حل) طی فازهای معماری و ساخت درک شده و تحقق می‌یابد.
- فراهم کردن محیط پروژه، شامل ارزیابی پروژه و سازمان، انتخاب ابزار، تصمیم‌گیری درباره اینکه کدام قسمت‌های فرایند باید بهبود یابد.

نقطه کلیدی(Milestone): اهداف چرخه عمر

اولین نقطه کلیدی اصلی پروژه در پایان فاز شناخت، نقطه کلیدی اهداف چرخه عمر(Lifecycle Objectives Milestone) است که بقای اولیه پروژه را ارزیابی می‌کند. در این نقطه، اهداف چرخه عمر ارزیابی و تست می‌شود، و تصمیم گرفته می‌شود که آیا پروژه انجام شود یا اینکه باید آنرا لغو کنیم. معیارهای ارزیابی عبارتند از:

- توافق ذینفعان روی تعریف دامنه و برآوردهای هزینه/زمان
- توافق روی اینکه مجموعه نیازمندیهای (درخواست‌ها) شناخته شده، واقعی و صحیح است و درک مشترکی از این نیازمندیها وجود دارد.
- توافق روی اینکه برآوردهای هزینه و زمان، اولویت‌ها، ریسک‌ها و فرایند توسعه مناسب هستند.
- همه ریسک‌ها شناسایی شده‌اند و استراتژی کاهش آنها وجود دارد.

اگر پروژه در این نقطه کلیدی شکست بخورد، ممکن است متوقف شود و یا بطور قابل ملاحظه‌ای بازنگری شود.

خروجی‌های (artifacts) اصلی این فاز به ترتیب عبارتند از:

- چشم‌انداز(Vision)
- مورد کسب و کار(Business Case)
- لیست ریسک‌ها (Risk List)

- برنامه توسعه نرم‌افزار(Software Development Plan)
- برنامه تکرار(Iteration Plan)
- Development Case
- ابزار(Tools)
- لغت‌نامه(Glossary)
- مدل قالب‌های کاربردی (مورد- کاربرد، Use-Case Model)(عامل‌ها و مورد کاربردها)
- انبار یا مخزن پروژه (Project Repository) و درخواست تغییر

و خروجی‌های موردی و اختیاری آن عبارتند از:

- مدل اشیاء کسب و کار (Business Object Model)
- راهنماهای(Guidelines) مدل مورد- کاررد
- اسناد ویژه پروژه
- Prototypes

هر کدام از این خروجی‌ها، با انجام یکی از الگوهای کاری یا دیسیپلین‌های RUP(بعد دوم RUP) بوجود می‌آید. مثلا مدل اشیاء کسب و کار (Business Object Model) با انجام مدلسازی کسب و کار و یا سند چشم‌انداز و مدل مورد کاربرد با انجام الگوی کاری نیازمندیها بوجود می‌آیند. در ادامه الگوهای کاری و خروجی‌های فاز شناخت مربوط به هر کدام از آنها تشریح می‌شود.

۱. مقدمه‌ای بر مدلسازی کسب و کار(Business Modeling)

هدف:

اهداف مدلسازی فرایند عبارتند از:

- فهم ساختار و دینامیک(پویایی) سازمانی که سیستم باید در آن مستقر شود (سازمان هدف).
- فهم مسائل موجود در سازمان هدف و شناسایی نقاط قابل بهبود
- ایجاد فهم مشترک در مشتری‌ها، کاربران نهایی و توسعه‌گرها از سازمان هدف
- استخراج نیازمندیهای سیستمی که برای پشتیبانی سازمان هدف مورد نیاز هستند.

برای دستیابی به این اهداف، مدلسازی کسب و کار تشریح می‌کند که چگونه یک چشم‌اندازی را برای سازمان هدف ایجاد کنیم و سپس بر اساس این چشم‌انداز، فرایندها، نقشها و مسئولیت‌های سازمان را در قالب مدل کسب و کار مورد کاربرد (use-case model) و مدل شیئ کسب و کار(business object model) تعریف کنیم. برای تکمیل این مدلها، خروجی‌های زیر ایجاد می‌شوند:

- مشخصات تکمیلی کسب و کار (Supplementary Business Specification)
- لغت نامه

ارتباط با سایر الگوهای کاری

الگوی کاری مدلسازی کسب و کار به صورت زیر با سایر الگوهای کاری مرتبط است:

نیازمندیها (Requirements) مدلهای کسب و کار را بعنوان ورودی مهمی برای درک نیازمندیهای سیستم استفاده می‌کنند.

آنالیز و طراحی (Analysis & Design) موجودیت‌های(entities) کسب و کار را بعنوان ورودی برای شناسایی کلاسهای موجودی در مدل طراحی استفاده می‌کند.

محیط (Environment) که خروجی‌های پشتیبانی مانند راهنمایی‌های مدلسازی کسب و کار(Business-Modeling Guidelines) را ایجاد و نگه می‌دارد.

۲. مقدمه‌ای بر نیازمندیها (Requirements):

هدف:

اهداف الگوی کاری نیازمندیها عبارتند از:

- ایجاد و حفظ توافق با مشتریها و سایر ذی‌نفعان روی اینکه سیستم باید چه کاری انجام دهد.
- ایجاد درک بهتری از نیازمندیهای سیستم برای توسعه‌گران سیستم
- تعریف مرزها و حدود سیستم
- تهیه پایه‌ای برای برنامه‌ریزی مفاهیم تکنیکی تکرارها
- تهیه پایه‌ای برای تخمین هزینه و زمان توسعه سیستم
- تعریف واسطه کاربری (user-interface) سیستم، تمرکز روی نیازها و اهداف کاربران

برای دستیابی به این اهداف، اول از همه، درک تعریف و دامنه مسئله‌ای که می‌خواهیم آنرا با سیستم مورد نظرمان حل کنیم، مهم است. قواعد کسب و کار(Business Rules)، مدل مورد کاربرد کسب و کار(Business Use-Case Model) و مدل شیء کسب و کار (Business Object Model) که در طی مدلسازی کسب و کار(Business Modeling) ایجاد می‌شوند، ورودی باارزشی را برای این فعالیت فراهم می‌کنند. ذینفعان شناسایی می‌شوند و درخواست‌های ذینفعان(Stakeholder Requests) استخراج، جمع‌آوری و تحلیل می‌شود. سند چشم‌انداز، مدل مورد کاربرد(Use-Case Model)، مورد کاربردها(Use Case) و مشخصات تکمیلی(Supplementary Specifications) برای تشریح کامل سیستم، به عبارت دیگر اینکه سیستم چه کاری انجام خواهد داد، ایجاد می‌شوند. در این فعالیت باید دیدگاه همه ذینفعان، شامل مشتریان و کاربران بالقوه را بعنوان منابع مهم اطلاعات مورد توجه قرار دهیم.

درخواست‌های ذینفعان به صورت فعال از منابع موجود استخراج و جمع‌آوری می‌شوند تا لیست خواسته‌های (wish list) ذینفعان مختلف پروژه (مانند مشتریان، کاربران و پشتیبان‌های محصول) بدست آید.

سند چشم‌انداز، چشم‌انداز کاملی را برای سیستم نرم‌افزار تحت توسعه تهیه می‌کند. سند چشم‌انداز از منظر مشتریان، روی قابلیت‌های اساسی سیستم تمرکز می‌کند و سطوح قابل قبول کیفیت را ارائه می‌کند. چشم‌انداز باید شرحی از اینکه چه قابلیت‌هایی باید در سیستم در نظر گرفته شود و چه قابلیت‌هایی نه را داشته باشد. چشم‌انداز همچنین باید ظرفیت‌های عملیاتی(حجم‌ها، زمان‌های پاسخگویی، دقت‌ها)، پروفایل‌های کاربر(چه‌کسی از سیستم استفاده خواهد کرد) و تعاملات عملیات داخلی با موجودیت‌های خارج از مرز سیستم و اینکه سیستم کجا قابل اجراست را مشخص کند. سند چشم‌انداز پایه‌ای قراردادی برای امکان دیدن نیازمندیها توسط ذینفعان را فراهم می‌کند.

مدل مورد کاربرد (Use-Case Model) بعنوان واسطه ارتباطی و همچنین بعنوان قرارداد بین مشتری، کابران و توسعه‌گران سیستم، برای قابل استفاده بودن سیستم بکار می‌رود، بطوریکه به مشتریان و کاربران این اطمینان را می‌دهد که سیستم آنچیزی است که آنها انتظار دارند و توسعه‌گران سیستم آنچیزی را که انتظار می‌رود، می‌سازند.

مدل مورد کاربرد، شامل مورد کاربردها(Use Case) و عامل‌ها(actors) است. هر مورد کاربردی در مدل به صورت دقیق و با جزئیات تشریح می‌شود و مرحله به مرحله نشان می‌دهد که چگونه سیستم با عامل‌ها تعامل دارد و سیستم چه‌چیزی را در مورد کاربردها انجام می‌دهد. مورد کاربردها بعنوان یک بخش یکپارچه در سراسر چرخه عمر نرم‌افزار استفاده می‌شود. همین مدل مورد کاربرد، در آنالیز، طراحی، اجرا و تست سیستم استفاده می‌شود.

در ادامه جریان کاری (work flow) و مراحل انجام کار الگوی کاری نیازمندیها و خروجی‌های مربوط به هر مرحله تشریح می‌شود.

۱,۲. آنالیز مسئله:

اولین گام در الگوی کاری نیازمندیها، آنالیز مسئله است.

هدف از آنالیز مسئله:

- بدست آوردن توافق روی مسئله‌ای که باید حل شود.
- شناسایی ذینفعان سیستم
- تعریف مرزهای سیستم و
- شناسایی محدودیت‌هایی که سیستم را تحت تاثیر قرار دهند.

اولین مرحله در آنالیز هر مسئله‌ای این است که مطمئن شویم همه قسمت‌های درگیر روی مسئله‌ای که ما می‌خواهیم آنرا با سیستممان حل کنیم، موافق باشند. برای اجتناب از درک اشتباه و نادرست، مهم است که روی اصطلاحات و واژگان مشترکی که در کل پروژه استفاده خواهد شد، توافق داشته باشیم. بنابراین اولین گام، تعریف اصطلاحات پروژه در لغت‌نامه‌ای(Glassory) است که باید آنرا در کل چرخه عمر پروژه نگه داشته و استفاده کنیم. نحوه تهیه و تدوین لغت‌نامه پروژه در پیوست الف آورده شده است.

اولین خروجی که در آنالیز مسئله آنرا مستند می‌کنیم سند چشم‌انداز (vision) است، که دیدگاه ذینفعان سطح بالای (high-level) سیستمی که باید ساخته شود را شناسایی می‌کند. در سند چشم‌انداز نیازمندی‌های اصلی سطح بالا، قابلیت‌های کلیدی (key features) سیستم را تعیین می‌کنند. این قابلیت‌ها، عموما مجموعه‌ای از قابلیت‌های سطح بالایی هستند که برای حل بحرانی‌ترین مسائل در سیستم پردازش می‌شوند.

ذینفعان کلیدی باید در جمع‌آوری مجموعه قابلیت‌هایی که باید مورد توجه قرار گیرند، درگیر شوند. ممکن است این قابلیت‌ها در کارگاه نیازمندیها (stakeholders workshop) جمع‌آوری شود.

برای تعیین محدوده و دامنه اولیه سیستم، باید روی مرزهای سیستم توافق حاصل شود. تحلیل‌گر سیستم، کاربران و سیستم‌هایی را که با سیستمی را که باید ساخته شود در تعامل هستند را شناسایی می‌کند (بوسیله عامل‌ها). اگر ما مدل دامنه (domain model)، مدل مورد کابرد کسب و کار(Business Use-Case Model) یا مدل شیء کسب و کار(Business Object Model) را همراه با قواعد کسب و کار(busines rules) ایجاد کنیم، اینها ورودی اصلی برای کمک به انجام آنالیز مسئله خواهند بود. برای اطلاعات بیشتر می‌توان به بحث Going from Business Models to Systems رجوع کرد.

آنالیز مسئله باید توسط چندین آیتم در طی فاز شناخت و ابتدای فاز معماری بازنگری شود.

اعضای پروژه که در آنالیز مسئله درگیر هستند، باید تسهیل کننده‌های موثری باشند و در تکنیک‌های پیدا کردن ریشه مسائل (مسئله پشت مسئله) دارای تجربه باشند. البته آشنایی با تکنولوژی‌های هدفمند مطلوب است، اما کافی نیست. تعامل فعال با ذینفعان مختلف پروژه نیز لازم است. برای درک و فهم کامل مسئله‌ای که ما باید آنرا هدایت کنیم، شناخت ذینفعان خیلی مهم است. ذینفع(Stakeholder) بعنوان هرکسی که واقعا با خروجی پروژه تحت تاثیر قرار می‌گیرد، تعریف می‌شود. حل موثر هر مسئله پیچیده‌ای عبارت است از ارضای نیازهای گروه‌های مختلف ذینفعان. معمولا ذینفعان دیدگاه‌ها و نیازهای مختلفی نسبت به مسئله دارند که باید توسط راه‌حل ارائه شده هدایت شوند. اکثر ذینفعان کاربران سیستم هستند. سایر ذینفعان، یا کاربران غیر مستقیم سیستم هستند، یا از نتایج کسب و کاری که سیستم بر آن اثر می‌گذارد، تاثیر می‌پذیرند که اکثرا خریداران یا پشتیبان‌های اقتصادی سیستم هستند. شناخت ذینفعان و نیازهای آنها عناصر کلیدی توسعه سیستم است. مثالهایی از ذینفعان عبارتند از:

- مشتری یا نماینده (representative)مشتری
- کاربر یا نماینده کاربر
- سرمایه‌گذار
- مدیر تولید
- خریدار
- طراح
- تست‌کننده
- نویسنده سند
- و غیره

در ادامه فعالیت‌ها و خروجی‌های آنالیز مسئله (اولین گام الگوی کاری نیازمندیها) به ترتیب تشریح می‌شود.

۱,۱,۲. چشم‌انداز(Vision)

چشم‌انداز، دیدگاه ذینفعان را درباره محصولی که باید ایجاد شود، تعریف می‌کند و قابلیت‌ها و نیازهای کلیدی ذینفعان را از محصول تعیین می‌کند. سند چشم‌انداز بر نیازمندیهای(requirement) محوری رویایی (envisioned) تاکید می‌کند و یک پایه قراردادی برای نیازمندیهای تکنیکی جزئیاتی‌تر فراهم می‌کند. در این سند، مشخصات نیازمندیهای رسمی نیز می‌تواند وجود باشد. چشم‌انداز، نیازمندیهای سطح بالا و محدودیت‌های طراحی را دریافت می‌کند تا درکی از سیستمی که باید توسعه یابد را در خواننده بوجود آورد. سند چشم‌انداز، ورودی فرآیند تصویب پروژه را فراهم می‌کند، بنابراین عمدتا با مورد کسب و کار(Business Case) مرتبط است و بین چراها و چه‌های مربوط به پروژه ارتباط برقرار می‌کند و *بعنوان معیاری برای تایید اعتبار همه تصمیمات آینده است*. سند چشم‌انداز ابتدا در فاز شناخت ایجاد می‌شود و بعنوان ورودی و پایه‌ای برای مورد کسب و کار(Business Case) استفاده می شود و اولین سند لیست ریسک (Risk List) است و بعنوان یک سند جداگانه بروز و نگهداری می‌شود. سند چشم‌انداز توسط مدیران، سرمایه‌گذاران و توسعه‌گرها خوانده می‌شود.

تحلیل‌گر سیستم مسئول جامعیت و کامل بودن سند چشم‌انداز است و تضمین می‌کند که آن بروز و توزیع شده است و انتظارات همه ذینفعان در آن دیده شده است. نام دیگری که برای سند چشم‌انداز استفاده می‌شود، سند درخواست محصول (Requirement Document Product) است.

در این سند مهمترین درخواستها و علایق ذینفعان آورده می‌شود و از درخواست‌های جزئی اجتناب می‌شود. از نرم‌افزار Rational RequisitePro می‌توان برای مستند کردن چشم انداز استفاده کرد.

فعالیت: نحوه ایجاد چشم‌انداز

اهداف ایجاد چشم‌انداز عبارتند از:

- توافق روی اینکه چه مسائلی نیاز به حل دارند.
- شناسایی ذینفعان سیستم
- تعریف مرزهای سیستم
- تشریح قابلیت‌های(Features) اولیه سیستم

مراحل تدوین چشم‌انداز:

۱- بدست آوردن توافق روی مسئله‌ای که باید حل شود.
۲- شناسایی ذینفعان سیستم
۳- تعریف مرزهای سیستم
۴- شناسایی محدودیت‌هایی که سیستم را تحت تاثیر قرار دهند.
۵- تنظیم و فرموله کردن بیانیه مسئله
۶- تعریف قابلیت‌های سیستم

۷- ارزیابی نتایج

۱- بدست آوردن توافق روی مسئله‌ای که باید حل شود.

یکی از ساده‌ترین راهها برای بدست اوردن توافق روی تعریف مسئله، نوشتن آن و مشاهده میزان موافقت افراد است.

از گروه سوال کنید: مسئله چیست؟

افراد قبل از آنکه درک اولیه‌ای از مسئله پیدا کنند، یورش بی‌پروایی را به تعریف مسئله می‌کنند.

مسئله را بنویسید و ببنید که آیا شما می‌توانید توافق همه افراد روی آن جلب کنید.

سپس دوباره از گروه سوال کنید؟ واقعا مسئله چیست؟

دلایل ریشه‌ای مسئله را پیدا کنید. مسئله واقعی اغلب پشت چیزی که بعنوان مسئله مشاهده می‌شود، پنهان است. اولین بار بیانیه مسئله نپذیرید. با سوال چرا ادامه دهید تا پیدا که مسئله واقعا چه‌چیز است؟

گاهی اوقات گروه روی یک روش و راه‌حل غیر واقعی تمرکز می‌کند که در این حالت بدست اوردن توافق ایشان در مورد اینکه چه‌چیزی واقعا مشکل و مسئله است خیلی سخت است. در چنین مواردی کشف مزایای راه‌حل و اینکه چه مسائلی با توجه آن مزایا حل می‌شوند، مفید خواهد بود.

تکنیکهای عمومی که برای پیدا کردن ریشه مسائل استفاده می‌شوند، عبارتند از: طوفان فکری (Brainstorming)، دیاگرام استخوان ماهی(Fishbone Diagrams) و نمودار پارتو (Pareto Diagrams).

۲- شناسایی ذینفعان

با توجه به میزان مهارت تیم توسعه، شناسایی ذینفعان شاید آسان و یا پیچیده باشد. معمولا ذینفعان شامل تصمیم‌گیرندگان، کاربران و قسمتهای علاقه‌مند و ذینفع است. سوالات زیر به شناسایی ذینفعان کمک می‌کنند:

۱- کاربران سیستم چه‌کسانی هستند؟
۲- خریدار اقتصادی سیستم چه کسی است؟
۳- چه کسانی از خروجی‌هایی که سیستم تولید می‌کند تحت تاثیر قرار می‌گیرند؟
۴- چه کسانی هنگامی که سیستم تحویل و استقرار یافت، آن را ارزیابی و از آن تمجید می‌کنند؟
۵- آیا کاربران داخلی و خارجی دیگری برای سیستم وجود دارد که باید نیازهای آنها توسط سیستم پاسخ داده شود؟
۶- چه کسی سیستم جدید را نگهداری خواهد کرد؟
۷- آیا کس دیگری وجود دارد؟
۸- بسیار خوب، آیا کس دیگری وجود دارد؟

ابتدا پروفایل‌هایی از کاربران واقعی و بالقوه سیستم را مشخص کنید. این پروفایل‌ها برای عامل‌های انسانی سیستمی که باید توسعه داده شود، ایجاد می‌شوند. اطلاعات اولیه کاربران کلیدی و محیط آنها در سند چشم انداز مستند می‌شود. اگر مدل‌سازی کسب و کار بعنوان قسمتی از پروژه یا بعنوان پیش‌نیاز آن انجام شود، مدل مورد کاربرد(Business Use-Case Model) و مدل اشیاء کسب و کار(Business Object Model) اطلاعات با ارزشی را در این حوزه فراهم می‌کنند.

۳- تعریف مرزهای سیستم

مرز سیستم بعنوان مرز بین سیستم و دنیای واقعی که سیستم را احاطه کرده است، تعریف می‌شود. به عبارت دیگر مرز سیستم بعنوان پاکتی است که سیستم درون آن است. اطلاعات به صورت ورودی و خروجی از سسیتم به طرف کاربرانی که در خارج از سیستم زندگی می‌کنند، خارج شده و یا برگشت داده می‌شوند. همه تعاملات با سیستم، از طریق روابط بین سیستم و محیط خارجی صورت می‌گیرد. در بسیاری از مواقع مرزهای سیستم مشخص هستند، برای مثال مرزهای یک کاربر. معمولا استفاده از عامل‌ها برای تعریف و تشریح مرزهای سیستم خیلی موثر خواهد بود. بعلاوه مدل مورد کاربرد(Business Use-Case Model) و مدل اشیاء کسب و کار(Business Object Model) اطلاعات با ارزشی را در این حوزه فراهم می‌کنند، اگر مدل‌سازی کسب و کار انجام شده باشد.

۴- شناسایی محدودیت‌هایی که سیستم را محدود و آنرا تحت فشار قرار می‌دهند.

چندین منبع محدودیت وجود دارد که باید مورد توجه قرار گیرند. بسیاری از اطلاعات مربوط به محدودیت‌ها ممکن است در قواعد کسب کار(Business Rules) مستند شده باشند. در ادامه لیستی از منابع بالقوه و سوالاتی برای شناسایی محدودیت‌ها ارائه می‌شود:

۱- سیاسی: آیا موضوعات سیاسی داخلی، خارجی و یا بین سازمانی وجود دارد که سیستم را تحت تاثیر قرار می‌دهند؟
۲- اقتصادی: کدام محدودیت‌های مالی یا بودجه‌ای وجود دارند؟ آیا هزینه‌های فروش کالا یا ملاحظات قیمت‌گذاری محصول وجود دارد؟ آیا موارد مربوط به اخذ مجوز وجو دارد؟
۳- قانونی: آیا محدودیت‌های قانونی و محیطی وجود دارد؟ آیا استانداردهای دیگری ما را محدود می‌کنند؟
۴- تکنیکی: آیا ما در انتخاب تکنولوژی محدودیت داریم؟ آیا ما باید با تکنولوژیها یا بسترهای موجود کار کنیم؟ آیا ما با استفاده از تکنولوژیهای جدید منع شده‌ایم؟

۵- توجیه‌پذیری یا موجه بودن: آیا یک برنامه زمانی(زمانبندی) تعریف شده وجود دارد؟ آیا ما به استفاده از منابع موجود محدود شده‌ایم؟ آیا ما می‌توانیم از منابع بیرونی استفاده کنیم (برون سپاری)؟ آیا ما می‌توانیم منابع موجود را توسعه دهیم؟ بطور موقت یا دائمی؟

۶- سیستمی: آیا راه‌حلی برای توسعه سیستم‌های موجود وجود دارد؟ آیا می‌توانیم سازگاری با سیستم‌های موجود را حفظ کنیم؟ کدام سیستمها و محیطهای عملیاتی باید پشتیبانی شوند؟

اطلاعاتی که در این قسمت جمع‌آوری می‌شوند بعنوان ورودی اولیه به تعریف محدودیت‌های طراحی در مشخصات تکمیلی(Supplementary Specifications) استفاده می‌شوند.

۵- فرموله کردن بیانیه مسئله

با همه گروه، الگوی زیر را برای هر مسئله‌ای شناسایی کرده‌اید، پر کنید:

مسئله <شرح مسئله>

ذی‌نفعان < ذی‌نفعانی که از مسئله تاثیر می‌پذیرند>

ضربه یا فشار آن < فشار یا اثر منفی مسئله چه‌چیز است>

راه‌حل یا سیستم موفق< لیستی از مزایای کلیدی راه‌حل موفق>

هدف این الگو کمک به تمایز راه‌حل‌ها/جوابها از مسائل/سوالات است.

۶- قابلیت‌های سیستم را تعریف کنید.

با توجه به مزایایی که در بیانیه مسئله لیست کرده‌اید، لیستی از خصوصیاتی که از سیستم انتظار دارید را ایجاد کنید. آنها را بطور خلاصه شرح دهید و از ویژگیهای آنها برای تعریف اولویت و وضعیت عمومی آنها در پروژه کمک بگیرید. برای اطلاعات بیشتر در این مورد می‌توان از فعالیت مدیریت وابستگی‌ها(Manage Dependencies) استفاده کرد.

۷- ارزیابی نتایج

در این مرحله باید چشم‌انداز را تست کرده و آن را به صورت کلی (نه به صورت جزئی) بازبینی کرد. چک لیستی برای ارزیابی سند چشم‌انداز در زیر ارائه شده است:

- آیا شما کاملا ریشه مسئله (مسئله پشت مسئله) را کشف کرده‌اید؟
- آیا بیانیه مسئله بطور صحیح فرموله شده است؟
- آیا لیست ذینفعان کامل و اصلاح شده است؟
- آیا همه روی تعریف مرز سیستم توافق دارند؟
- در صورتی که مرز سیستم با استفاده از عامل‌ها مشخص شده، آیا همه عامل‌ها تعریف و بطور صحیح تشریح شده‌اند؟

- آیا همه خصوصیات سیستم شناسایی و تعریف شده‌اند؟
- آیا خصوصیات سیستم مسائل شناسایی شده را حل می‌کنند؟
- آیا خصوصیات با محدودیت‌های شناسایی شده سازگار شده‌اند؟

خروجی فعالیت تدوین چشم‌انداز در قالب سند استاندارد(Template) ارائه شده در **پیوست ب** تکمیل می‌شود.

۲,۱,۲. درخواست‌های ذینفعان(Stakeholder Requests)

هدف این خروجی جمع‌آوری همه درخواست‌ها و نحوه هدایت آنها با پروژه است. گرچه تحلیل‌گر سیستم مسئول این خروجی است، اما افراد زیادی مانند کاربران نهایی و مشتریان و هر کسی که بعنوان ذینفع نتیجه پروژه در نظر گرفته می‌شود، در آن درگیر هستند.

درخواست‌های ذینفعان عمدتا طی فازهای شناخت و معماری جمع‌آوری می‌شوند، با وجود این باید این درخواست‌ها را در کل پروژه برای برنامه‌ریزی و بروز کردن محصول جمع‌آوری کنید. درخواست‌های ذینفعان در یک پایگاه داده‌ای مانند Rational ClearQuest and/or Rational RequisitePro به بهترین نحو مدیریت می‌شوند. به این ترتیب که درخواست‌ها پیگیری و اولویت‌بندی می‌شوند، گزارشات تولید می‌شود و امکان ردگیری ایجاد می‌شود. برای اطلاعات بیشتر می‌توان به بحث Tool Mentor: Managing Stakeholder Requests Using Rational ClearQuest and Rational RequisitePro مراجعه کرد.

فعالیت: استخراج درخواست‌های ذینفعان

هدف:

- فهمیدن اینکه چه‌کسانی ذینفعان سیستم هستند
- جمع‌آوری درخواست‌ها برای فهمیدن اینکه چه نیازهایی را سیستم باید برآورد کند
- اولویت بندی درخواست‌های ذینفعان

مراحل:

۱. تعیین منابعی برای نیازمندیها
۲. جمع‌آوری اطلاعات
۳. هدایت کارگاه‌های نیازمندیها
۴. ارزیابی نتایج

۱.تعیین منابعی برای نیازمندیها

هدف:

- شناسایی افرادی که در تیم پروژه بعنوان ذینفع عمل می‌کنند
- تعیین و اولویت‌بندی منابعی برای نیازمندی‌ها

در سیستم‌های موجود، اولین مجموعه ورودی برای این فعالیت، مجموعه درخواست‌های رشد و ارتقاء (enhancement requests) قبلی و بتعویق افتاده است که در کل چرخه عمر پروژه بعنوان قسمتی از فرایند مدیریت درخواست تغییر (Change Request Management) جمع‌آوری می‌شوند، که این نقطه شروع ارزشمندی برای جمع‌اوری داده و اصلاح مجموعه درخواست‌های ذینفعان فراهم می‌کند.
بعد از اینکه اطلاعات اولیه جمع‌آوری شد، شرکا، کاربران، مشتریان، کارشناسان و تحلیل‌گران صنعت که می‌توانند ذینفعان شما را نشان دهند را پیدا کنید. افرادی را که برای جمع‌اوری اطلاعات با آنها کار می‌کنید را تعیین کنید. برای این کار دانش افراد، مهارت‌های ارتباطی، دسترس‌پذیری و اهمیت را در نظر بگیرید. این افراد بعنوان ذینفعان پروژه در تیم پروژه عمل خواهند کرد.
معمولا بهتر است که یک گروه کوچک(۲-۵) از افرادی را که تا پایان پروژه با شما همراه خواهند بود انتخاب کنید. چون اگر افراد بیشتری در تیم وجود داشته باشند، مدیریت آنها زمان بیشتری نیاز دارد و نمی‌توان مطمئن بود از زمان بطور موثر استفاده می‌شود. این افراد به صورت تمام وقت در پروژه کار نخواهند کرد، بلکه معمولا در یک یا چند کارگاه جمع‌اوری نیازمندیها در فازهای شناخت و معماری و بعدا در جلسه بازنگری شرکت خواهند کرد.
سعی کنید راهی را برای یادگیری از افرادی که قبلا کاری همانند کاری که شما سعی دارید تا انجام دهید را انجام داده‌اند، پیدا کنید. اگر شما سعی دارید که یک محصول نرم‌افزاری تجاری تولید کنید، باید اطلاعاتی را از رقبا جمع‌آوری کنید. اگر می‌خواهید ویرایش جدیدی از سیستم اطلاعاتی داخلی ایجاد کنید، باید ببینید که چه افرادی از سیستم فعلی استفاده می‌کنند و چگونه می‌توان آنرا را بهبود داد. یک منبع مهم، هرگونه تشریحی از وضعیت موجود سازمان است که این می‌تواند یا مدل‌های کسب و کاری که در مدلسازی کسب و کار تولید شده‌اند باشد یا هر شکل دیگری که کسب و کار تعریف شده است.
مثالهایی از منابع درخواست‌های ذینفعان عبارتند از:

- نتایج مصاحبه با ذینفعان
- نتایج جلسات و کارگاه‌های استخراج نیازمندیها
- درخواست تغییر
- شرح کار
- درخواست برای پیشنهاد(RFP)
- بیانیه ماموریت
- بیانیه مسئله
- قواعد کسب و کار
- قوانین و آیین‌نامه‌ها
- سیستم‌های قبلی

- مدل‌های کسب و کار

۲. جمع‌اوری اطلاعات

هدف:

- فرموله کردن سوالاتی که نیاز به جواب دارند
- جمع‌آوری و مستند سازی اطلاعات

مصاحبه‌ها:

یکی از کاربردی‌ترین روش‌های جمع‌آوری اطلاعات انجام مصاحبه با گروه انتخاب شده از ذینفعان کلیدی است. برخی نمونه سوالات و تکنیک‌هایی که ممکن است استفاده شود، در Work Guidelines: Interviews ارائه شده است.

پرسشنامه‌ها:

این روش بطور وسیعی استفاده می‌شود و این امکان را به شما می‌دهد که آمارهای رسمی بهتری را از جواب‌های داده شده به سوالات بدست آورید. در این روش توانایی در فرموله کردن سوالاتی که آمارهای واقعی از نیازهای واقعی ذینفعان را ارائه کند، خیلی مهم است.

ممکن است ذینفعان بتوانند از طریق اینترنت به سوالات جواب داده و به شما بفرستند. در این صورت شما به دامنه وسیعتری از افراد نسبت به حالت مصاحبه مستقیم دسترسی دارید، اما کنترل کمتری نسبت به نتایج دارید. چون در این حالت ارتباط مستقیمی با فردی که به سوالات جواب می‌دهد ندارید که بتوانید موضوعات برای او شرح داده و برداشت‌های اشتباه او را از سوالات رفع کنید. پرسشنامه می‌تواند ابزار خیلی قوی باشد، اما نمی‌تواند جایگزین یک مصاحبه مستقیم شود.

۳. برگزاری و اداره کارگاه‌های نیازمندیها

هدف:

- ملاقات تیم پروژه با ذینفعان پروژه
- جمع‌آوری لیست خواسته (wish list) جامعی از ذینفعان پروژه
- اولویت‌بندی نیازمندی‌های جمع‌آوری شده بر اساس انتظارات ذینفعان

۴. ارزیابی نتایج

هدف:

- ارزیابی نتایج کارگاه‌های مختلف نیازمندیها
- ایجاد اطمینان از اینکه شما اطلاعات درستی را جمع‌آوری کرده‌اید

اگر شما بیش از یک کارگاه نیازمندیها را برگزار کنید، آن عادت خوبی برای تیم پروژه در جهت دستیابی به نتایج است و اطمینانی را نسبت به اولویت‌بندی درخواست‌ها و اطلاعات مربوط به اینکه چه‌چیزی و چه‌کسی منبع درخواست است، ایجاد می‌کند و ناسازگاری و ناهماهنگی‌های موجود بین درخواست‌ها را آشکار می‌کند.

نتایج کارگاه نیازمندیها باید به مجموعه منتخبی از مشتریان یا کاربران در یک جلسه بازنگری ارائه شود. در این جلسه شما باید موضوعاتی را که نیاز به شفاف شدن دارند را شناسایی کنید و آنها را بدرستی معنی کنید. همچنین شما باید مواردی که باید نیاز به تکمیل شده دارند را شناسایی و افرادی را برای چنین مواردی تخصیص دهید.

راهنمایی‌هایی برای انجام مصاحبه(Work Guidelines: Interviews):

برای یک تکنیک مصاحبه مستقیم رو در رو و موثر، شما باید لیستی از سوالات مناسب را طراحی کنید تا بتوانید درکی از مسائل واقعی و راه‌حل‌های ممکن بدست آورید.

برای بدست آوردن جواب‌های بی‌غرض، شما باید مطمئن باشید که سوالاتی که شما می‌پرسید مستقل از متن و عمومی (context-free) هستند. سوال مستقل از متن یک سوال سطح بالا و خلاصه است که می‌تواند در ابتدای پروژه مطرح شود تا اطلاعاتی را درباره اولویت‌های مسائل کاربر و راه‌حل‌های ممکن بدست آورد.

یک سوال مستقل از متن و عمومی، سوالی است که:

- همیشه مناسب است
- فرموله شده است بگونه‌ای که به شما در درک دیدگاه‌های ذینفع کمک می‌کند.
- غرض و پیش‌فرضهایی را با توجه به دانش و عقیده شما درباره اینکه مسئله چه‌چیزی باید باشد، ندارد.

*سند مصاحبه مستقل از متن (آزاد) (**Context-Free Interview Scrip**):*

فرصت‌های بزرگی در صنعت برای بهبود وجود دارند. درک نیازهای ذینفعان و کاربران قبل از انجام بهبود خیلی مهم است. تکنیک‌های زیادی برای استخراج نیازهای ذینفعان وکاربران وجود دارند. یکی از تکنیک‌های ساده و کم هزینه که برای هر وضعیتی مناسب است، مصاحبه عمومی (Generic Interview) است. مصاحبه عمومی به توسعه‌گر و تحلیل‌گر کمک می‌کند که مسائل و اهداف ذینفع و کاربر را درک کند. با این بینش، توسعه‌گران می‌توانند سیستم‌هایی را ایجاد کنند که مطابق با نیازهای واقعی ذینفع و کاربر باشد و رضایت آنها را افزایش دهد.

همچنین با استفاده از مصاحبه عمومی، توسعه‌گر یا تحلیل‌گر دانش حل مسئله را بدست می‌آورد و همزمان دید ذینفع یا کاربر را نسبت به پارامترهای راه‌حل موفق درک می‌کند.

راهنمایی‌هایی برای استفاده در عمل:

با حداقل آمادگی و مصاحبه ساخت‌یافته خوب، توسعه‌گر یا تحلیل‌گر می‌توانند مصاحبه موثری را انجام دهند. به نکات زیر توجه کنید:

- اطلاعاتی را درباره پس‌زمینه ذینفع یا کاربر و آینده شرکت بدست آورید..
- سوالات را قبل از مصاحبه بازنگری کنید.

- فرمتی را برای مصاحبه داشته باشید تا مطمئن شوید که سوالات درست و مناسبی پرسیده می‌شود.
- دو یا سه مسئله کلی را در پایان هر مصاحبه خلاصه کنید. سعی کنید که قوه درک‌تان را افزایش دهید.

اجازه ندهید سند و متن(script) خیلی شما را محدود کند. اگر رابطه خوبی با ذینفع یا کاربر ایجاد کنید، ممکن است در طی مصاحبه وی درباره مشکلاتی که آنها را تجربه کرده صحبت کند. در این صورت او را متوقف نکنید و در صورت امکان جواب‌های او را سریع یادداشت کنید. دریافت اطلاعات را با سوالات ادامه دهید.

مثالهایی از سوالات مستقل از متنی که برای پیدا کردن عاملها استفاده می‌شود:

- مشتری کیست؟
- کاربر کیست؟
- آیا نیازهای آنها متفاوت است؟
- پس‌زمینه، توانایی و محیط آنها چگونه است؟

مثالهایی از سوالات مستقل از متنی که برای فهمیدن فرایندهای کسب و کار به شما کمک می‌کند:

- مسئله چیست؟
- دلیل حل این مسئله چیست؟
- آیا دلایل دیگری برای حل این مسئله وجود دارد؟
- ارزش راه‌حل موفق چیست؟
- در حال حاضر مسئله را شما چگونه حل می‌کنید؟
- بین زمان و ارزش چه رابطه‌ای وجود دارد؟
- کجا می‌توان راه‌حل دیگری برای این پیدا کرد؟

مثالهایی از سوالات مستقل از متنی که برای فهمیدن نیازمندیهای سیستم و محصولی که باید ساخته شود، به شما کمک می‌کنند:

- این سیستم چه مسئله‌ای را حل می‌کند؟
- مسائل کسب و کاری که می‌توانند این سیستم را بوجود بیاورند، چه‌چیزهایی هستند؟
- خطراتی که برای کاربر وجود دارد چه‌چیزهایی هستند؟
- سیستم با چه محیطی مواجه می‌شود؟
- انتظارات شما برای قابلیت استفاده سیستم چه‌چیزهایی هستند؟
- انتظارات شما برای قابلیت اطمینان سیستم چه‌چیزهایی هستند؟
- چه میزان کارایی یا دقت مورد نیاز است؟

مثالهایی از سوالات مستقل از متنی که فرد مصاحبه کننده در پایان مصاحبه باید از خود بپرسد:

- آیا من سوالات زیادی را پرسیدم؟
- آیا سوالات من مناسب و مربوط بودند؟
- آیا شخص مناسبی به این سوالات پاسخ داد؟
- آیا نیازمندیها در پاسخ‌های که توسط من جمع‌آوری شده‌اند هستند؟
- آیا من می‌توانم بعدا نیز سوالاتی را بپرسم؟
- ایا شما مایل هستید در بازنگری سوالات شرکت کنید؟
- آیا چیز دیگری غیر از اینکه من از شما پرسیدم، وجود دارد؟

مثالهایی از سوالاتی که مستقل از متن نیستند:

- سوالات هدایت کننده، مانند "شما صفحه بزرگتری نیاز دارید، این‌طور نیست؟"
- سوالاتی که خود به آن پاسخ می‌دهیم، مانند "آیا حدودا پنجاه قلم درست است؟"
- جملات کنترلی مانند "آیا می‌توانیم به سوالاتم برگردیم؟"
- سوالات خیلی بلند و خیلی پیچیده مانند "من یک سوال سه قسمتی دارم، ...".

هنگامی که شما مجموعه‌ای از سوالات را فرموله کردید، موارد زیر را نیز باید در نظر بگیرید:

- از افراد نخواهید چیزهایی را که معمولا بیان نمی‌کنند، شرح دهند.
- در تدوین سوالات تصور نکنید که کاربران می‌توانند فعالیتهای پیچیده را شرح دهند، مانند بستن کفش
- معمولا افراد نمی‌توانند خیلی از چیزهایی را که می‌توانند انجام دهند، بیان کنند.
- مدارک تجربی- همبستگی ناچیز(Empirical evidence - poor correlation)
- سوالات انتها باز(open-ended) بپرسید.
- از سوالاتی که با چرا شروع می‌شوند اجتناب کنید؛ چون چنین سوالاتی حالت دفاعی فرد را تحریک می‌کنند.

هنگامی که شما جلسه مصاحبه را برگزار می‌کنید، در نظر داشته باشید:

- انتظار جواب‌های ساده را نداشته باشید.
- نگذارید مصاحبه شونده با عجله به سوالات پاسخ دهد.
- گوش کنید، گوش کنید، گوش کنید.

در خواست‌های ذینفعان در قالب سند درخواست‌های ذینفعان مستند می‌شود که نمونه‌ای از این سند در ***پیوست ج*** ارائه شده است.

خروجی: راهنمایی‌های (سرفصل) برای مدلسازی مورد کاربرد (Develop Use-Case Modeling Guidelines)

هدف: هدف راهنمایی‌های مدلسازی مورد کاربرد تشریح نحوه مدلسازی مورد کاربردهاست.
راهنمایی‌های مدلسازی مورد کاربرد در ابتدای فاز شناخت ایجاد و توسعه می‌یابند و مسئولیت آن با تحلیل گر سیستم است. این راهنمایی‌ها هنگامی مورد نیاز هستند که مورد کاربردها نقش مهمی را در نشان دادن رفتار سیستم بازی می کنند.

فعالیت: ایجاد راهنمایی‌هایی برای مدلسازی مورد کاربرد (Develop Use-Case Modeling Guidelines)

هدف:

- ایجاد راهنمایی‌هایی برای مدلسازی مورد کاربرد

مراحل:

۱. تعیین راهنمایی‌های مناسب برای مدلسازی مورد- کاربرد (Tailor the Use-Case Modeling Guidelines)
۲. تصمیم گیری

۱. تعیین راهنمایی‌های مناسب برای مدلسازی مورد- کاربرد

راهنمایی‌های مدلسازی مورد کاربرد را بگونه‌ای مشخص کنید که مناطقی را که در پروژه شما مورد توجه هستند را پوشش دهند. مورد توسعه(development case) بعنوان یک ورودی مهم برای تشریح نحوه کار پروژه با مورد کاربردها استفاده می‌شود.

۲. تصمیم گیری

قبل از شروع به تشریح مورد- کاربردها، باید چندین تصمیم را درباره مدلسازی مورد کاربردها اتخاذ کنیم، برای مثال آیا نمونه اولیه‌ای از واسط کاربر(user interface) ایجاد کنیم یا نه و کدام نوع راهنمایی را در تشریح مورد کاربردها استفاده خواهیم کرد. همه تصمیماتی که درباره راهنمایی‌ها و استراتژی‌های مدلسازی مورد کاربرد گرفته می‌شود، باید در راهنمایی‌های مدلسازی مورد-کاربرد مستند شود. در ادامه مثال‌هایی از برخی تصمیماتی که باید اتخاذ شوند، ارائه شده است:

الف. تصمیم گیری درباره اینکه چگونه مورد-کاربردها را بنویسیم

برای اجتناب از ناهماهنگی‌ها، سبک مناسبی را برای تشریح مورد-کاربردها در ابتدای پروژه انتخاب کنید. روش عمومی و مشترکی(common) را برای تشریح مورد-کاربردها انتخاب کنید. مزایا و عدم مزیت‌های سبک عمومی را ذکر کنید. مهمترین چیزی که درباره مورد-کاربردها در چنین روشی نوشته می‌شود را بخاطر بیاورید بگونه‌ای که نماینده‌ها/ناظرهای مشتری/کاربر بفهمند که چه نوع

سیستمی را شما به آنها پیشنهاد می‌کنید. برای اطلاعات بیشتر درباره نحوه نوشتن یک مورد-کاربرد به فعالیت جزئیات یک مورد- کاربرد(Activity: Detail a Use Case) مراجع کنید.

ب. تصمیم‌گیری در باره اینکه کی استفاده از روابط را شروع کنیم.
تصمیم بگیرید که کی شما باید شروع به استفاده از سه نوع رابطه مدل مورد-کاربرد کنید. این روابط عبارتند از:
actor-generalization, include-relationship, extend-relationship
قاعدتا شما نباید این روابط را در اولین نسخه مدل مورد-کاربردتان استفاده کنید، بخاطر اینکه این روابط مدل را پیچیده‌تر و فهم آنرا مشکل می‌کنند.

فعالیت: پیدا کردن عامل‌ها و مورد کاربردها (Find Actors and Use Cases)

هدف:

- ایجاد طرح کلی از قابل استفاده بودن(functionality) سیستم
- تعریف اینکه چه‌چیزهایی توسط سیستم انجام می‌شوند و چه‌چیزهایی خارج از سیستم انجام می‌شوند.
- تعریف اینکه چه‌کسانی و چه‌چیزهایی با سیستم در تعامل خواهند بود
- تقسیم مدل به بسته‌هایی(packages) از عامل‌ها و مورد کاربردها
- ایجاد نمودارهای مدل مورد کاربرد
- ممیزی مدل مورد کاربرد

مراحل:

۱. پیدا کردن عامل‌ها
۲. پیدا کردن مورد کاربردها
۳. تشریح اینکه چگونه عامل‌ها و مورد کاربردها باهم تعامل دارند
۴. بسته‌بندی مورد کاربردها و عامل‌ها
۵. نمایش مدل مورد کاربرد در قالب نمودارهای مورد کاربرد
۶. ایجاد ممیزی از مدل مورد کاربرد
۷. ارزیابی نتایج

۱. پیدا کردن عامل‌ها:

پیدا کردن عاملها یکی از اولین مراحل تعریف کاربردی سیستم است. *هر نوع پدیده خارجی که سیستم باید با آن در تعامل باشد، بوسیله یک عامل نشان داده می‌شود.* برای پیدا کردن عامل‌ها، سوالات زیر را بپرسید:

- کدام گروه‌های کاربر برای انجام وظایف‌شان نیاز به کمک سیستم دارند؟

- کدام گروه‌های کاربر برای اجرایی کردن آشکارترین وظایف(functions) اصلی سیستم مورد نیاز هستند؟
- کدام گروه‌های کاربر برای انجام وظایف(functions) ثانویه سیستم، مانند نگهداری و مدیریت سیستم مورد نیاز هستند؟
- آیا سیستم با هر گونه سیستم سخت‌افزاری و نرم‌افزاری خارجی دیگری در تعامل خواهد بود؟

هر فرد، گروه یا پدیده‌ای که مطابق با یک یا چندتا از این مقوله‌ها باشد، کاندیدی برای یک عامل است.

برای تشخیص اینکه آیا شما عامل‌های (انسانی) درستی دارید یا نه، شما می‌توانید دو یا سه فرد را که می‌توان بعنوان عامل‌ها نشان داد، نام‌گذاری کنید و سپس ببینید که آیا مجموعه عامل‌های شما برای نیازهای آنها کافی است. برای اطلاعات بیشتر برای تشکیل یک عامل Guidelines: Actor مراجعه کنید.

در ابتدا ممکن است پیدا کردن مناسب‌ترین عامل‌ها مشکل باشد و شما احتمالا نتوانید فورا همه آنها را پیدا کنید، برای اینکه شما هنوز مورد کاربردها را پیدا نکرده‌اید. کار با مورد کاربردها تنها چیزی است که فهم عمیقی از محیط سیستم و نحوه تعامل آن با سیستم به شما می‌دهد. هنگامی که شما جلوتر رفتید، ممکن است که شما بخواهید در مدل اولیه و اصلی‌تان تجدیدنظر کنید، برای اینکه در اوایل کار شما دوست دارید عامل‌های خیلی زیادی را مدل کنید. دقت کنید که هنگامی که شما عامل‌هایتان را تغییر می‌دهید، تغییرات شما ممکن است به همان صورت بر مورد کاربردهای شما اثر بگذارد. به یاد داشته باشید که هر گونه اصلاحی در عامل‌ها، تغییر و دگرگونی اساسی را در واسط‌ها(interfaces) و رفتار سیستم بوجود می‌آورد.

اگر شما مدل مورد کاربرد کسب و کار و مدل شیئ کسب و کار را ایجاد کرده‌اید می‌توانید برای راهنمایی‌های بیشتر به آنها و همچنین به بحث Going from Business Models to Systems مراجعه کنید.

عامل‌هایی را که پیدا کرده‌اید، نامگذاری کنید و بطور خلاصه شرح دهید.

نام عامل‌ها باید بطور واضح نقش عامل‌ها را بیان کند. مطمئن باشید که در هر مرحله‌ای که جلوتر می‌روید، ریسک ابهامات نام هر عامل با دیگری کمتر می‌شود. هر عامل را با نوشتن شرح خلاصه‌ای بر آن تعریف کنید. این شرح خلاصه باید منطقه مسئولیت هر عامل و نیازها و انتظارات عامل از سیستم در بر بگیرد. برای اینکه عامل‌ها چیزهای خارج از سیستم را نشان می‌دهند و نیازی برای تشریح جزئی آنها نیست. برای اطلاعات بیشتر به بخش شرح خلاصه در بحثGuidelines: Actor (بحثی که در ادامه آمده است) مراجعه کنید.

Guidelines: Actor

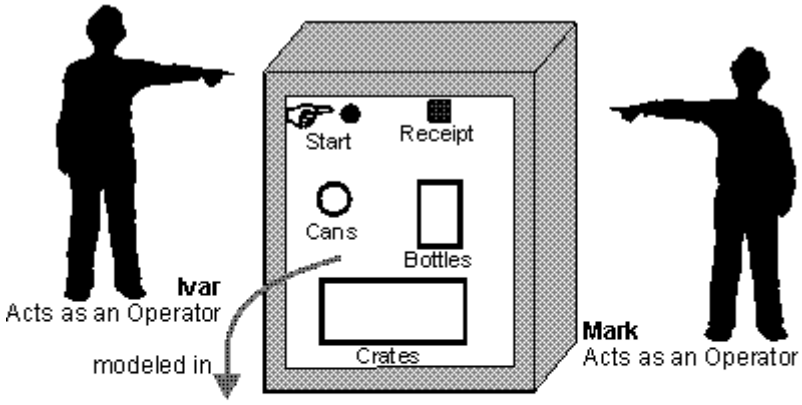
Actor: یک نمونه عامل(actor instance) هرکسی یا هرچیزی که خارج از سیستم است و با سیستم تعامل دارد.

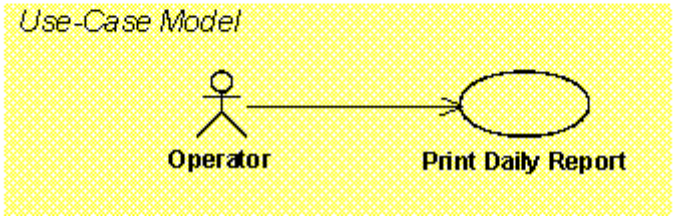
کلاس عامل(actor class) مجموعه‌ای از نمونه‌های عامل است که هر نمونه عامل آن نقش یکسانی را در ارتباط با سیستم بازی می‌کند.

شرح:

برای درک کامل هدف سیستم، شما باید بدانید سیستم برای چه‌کسی است و چه کسی از سیستم استفاده خواهد کرد. انواع مختلف کاربران بصورت عامل‌ها نشان داده می‌شوند.

هر چیزی که با سیستم داده تبادل و رد و بدل می‌کند، یک عامل است. یک عامل می‌تواند یک کاربر، سخت‌افزار خارجی یا سیستم دیگری باشد. تفاوت بین یک عامل با یک کاربر عادی و فردی سیستم این است که یک عامل در مقایسه با یک کاربر واقعی، کلاس خاصی از کاربر را نشان می‌دهد. چندین کاربر می‌توانند نقش یکسانی را در سیستم بازی کنند، بدین معنی که آنها می‌توانند یک عامل باشند. در چنین مواردی هر کاربر یک نمونه عامل را بوجود می‌آورد.

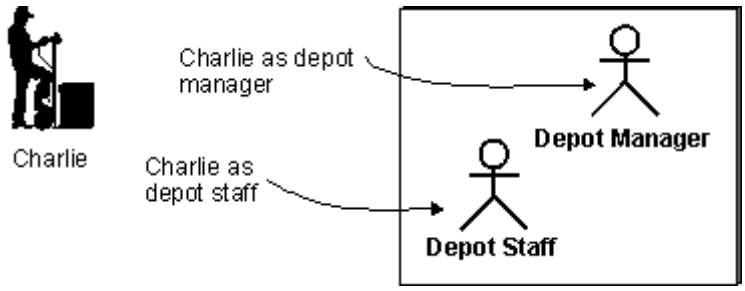




Ivar و Mark بعنوان اپراتورهای ماشین بازیافت(recycling machine) هستند. هنگامی که آنها ماشین را استفاده می‌کنند هرکدام از آنها بوسیله یک نمونه عامل اپراتور نشان داده می‌شوند.

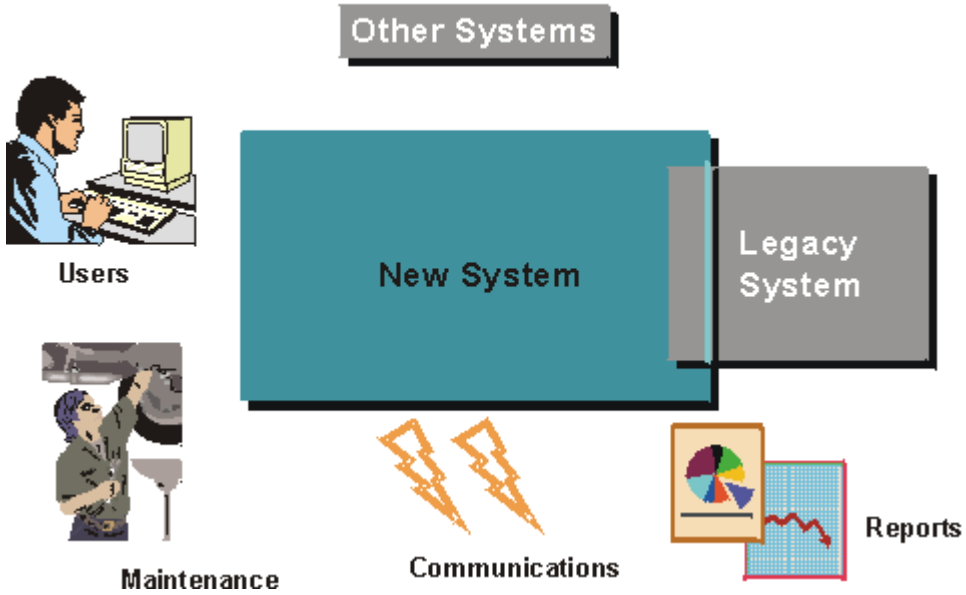
در برخی مواقع فقط یک شخص نقشی را که بعنوان یک عامل مدل شده است، بازی می‌کند. برای مثال ممکن است فقط یک نفر نقش مدیر سیستم را برای یک سیستم نسبتا کوچک بازی کند. همچنین

یک کاربر می‌تواند بعنوان چندین عامل عمل کند(مانند اینکه یک شخص می‌تواند چندین نقش مختلف را بگیرد).



در این سیستم انبار، چارلی اصولا بعنوان مدیر انبار است اما در برخی مواقع بعنوان کارمند عادی انبار نیز استفاده می‌شود.

چگونه عامل‌ها را پیدا کنیم.



چه‌چیزهایی در پیرامون و محیط سیستم، عامل‌های آن خواهند شد؟

با تفکر درباره افرادی که سیستم را استفاده خواهند کرد، شروع کنید. چگونه شما آنها را تشخیص می‌دهید؟ اغلب یک عادت خوب برای اینکار این است که تعدادی از افراد(دو یا سه نفر) را در ذهن داشته باشید و مطمئن باشید که عامل‌هایی که شما شناسایی کرده‌اید نیازهای آنها را پوشش می‌دهد.

مجموعه سوالات زیر برای شناسایی عامل‌ها مفید است:

- چه‌کسی اطلاعات را تامین، استفاده یا حذف خواهد کرد؟
- چه‌کسی این سیستم را استفاده خواهد کرد؟
- چه‌کسی نیازمندی مشخصی از سیستم دارد؟

- سیستم در کجای سازمان استفاده می‌شود؟
- چه‌کسی سیستم را پشتیبانی و نگهداری خواهد کرد؟
- منابع خارجی سیستم چی‌چیزهایی هستند؟
- چه سیستم‌های دیگری با این سیستم تعامل خواهند داشت؟

برای نشان دادن عامل‌ها از چندین جنبه مختلف در محیط سیستم می‌توان به آن نگاه کرد که این جنبه‌ها عبارتند از:

- کاربرانی که وظایف عمده سیستم را انجام می‌دهند.

مثال: برای یک سیستم انبارداری، چندین مقوله کاربر وجود دارد:

کارمند انبار، مسئول ثبت سفارش و مدیر انبار. همه این مقوله‌ها نقش‌های خاصی را در سیستم دارند. بنابراین شما باید هر یک از آنها را بعنوان یک عامل جداگانه نشان دهید.

- کاربرانی که وظایف ثانویه سیستم را اجرایی می‌کنند، مانند مدیر سیستم.

مثال: ...

- سخت‌افزار خارجی که سیستم استفاده می‌کند

مثال: در یک سیستم تهویه که بطور مداوم دما را در یک ساختمان کنترل می‌کند، داده‌های دمای اندازه‌گیری شده را از سنسورهای ساختمان می‌گیرد. بنابراین سنسور یک عامل است.

- سیستم‌های دیگری که با سیستم تعامل دارند

مثال: یک دستگاه عابر بانک خودکار باید با سیستم مرکزی که حساب‌های بانکی را نگه می‌دارد ارتباط داشته باشد.سیستم مرکزی احتمالا یک سیستم بیرونی است بنابراین می‌تواند یک عامل باشد.

اگر شما سیستمی را بر پایه اینترنت ایجاد کرده‌اید، عامل‌های اولیه شما احتمالا بی‌نام هستند. شما واقعا نمی‌دانید آنها چه کسانی هستند و شما نمی‌توانید هیچ‌گونه فرضی را درباره مهارت‌ها و پس زمینه آنها ایجاد کنید. ...

عامل‌ها به تعریف مرزهای سیستم کمک می‌کنند.

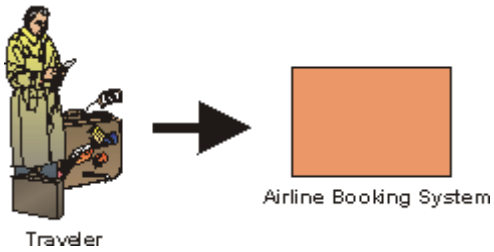
پیدا کردن عامل‌ها همچنین بمعنی ایجاد مرزهای سیستم است، چیزی که به درک هدف و وسعت سیستم کمک می‌کند. تنها آنهایی که مستقیما با سیستم ارتباط دارند باید بعنوان عامل در نظر گرفته شوند. اگر شما سعی کنید نقش‌های بیشتری را در محیط سیستم در نظر بگیرید، در این صورت شما کسب و کار را در سیستم‌تان مدل کنید نه سیستم خودتان را.

مثال:

در سیستم بلیط فروشی هواپیمایی(airline booking system)، عامل چه‌چیزی است؟ این بستگی به این دارد که آیا شما سیستم بلیط‌فروشی هواپیما را برای استفاده یک نمایندگی سفر ایجاد کرده‌اید یا برای مسافری که بتواند مستقیما از طریق اینترنت با آن ارتباط برقرار کند.

اگر شما سیستم بلیط فروشی هواپیما را ایجاد کرده‌اید که در نمایندگی سفر استفاده شود، عامل نمایندگی سفر خواهد بود. مسافر مستقیما با سیستم تعامل ندارد و بنابراین عامل نیست.(شکل زیر)

اگر شما سیستم بلیط فروشی را بگونه‌ای ایجاد کرده‌اید که کاربران بتوانند از طریق اینترنت به آن متصل شوند، در این صورت مسافر مستقیما با سیستم تعامل دارد و بنابراین یک عامل است.



شرح خلاصه:

شرح خلاصه‌ای از عامل باید اطلاعاتی را درباره موارد زیر داشته باشد:

- عامل چه‌چیزی یا چه‌کسی را نشان می‌دهد.
- چرا عامل مورد نیاز است.
- عامل چه انتظاراتی را از سیستم دارد.

شرح خلاصه باید شامل چند جمله مناسب باشد.

مثال: در مدل مورد کسب و کار ماشین بازیافت(Recycling Machine) سه عامل بطور خلاصه در زیر تشریح شده‌اند:

مشتری: مشتری بطری‌ها، قوطی‌ها و حلبی‌ها را در خانه جمع‌آوری می‌کند و سپس آنها را برای بازیافت به کارگاه می‌آورد و پول می‌گیرد.

اپراتور: اپراتور مسئول نگهداری ماشین بازیافت است.

مدیر: مدیر، مسئول جوابگویی به سوالات مشتریان درباره پول و خدمات ارائه شده است.

خصوصیات عامل(Actor Characteristics):

ویژگیها و خصوصیات یک عامل ممکن است نحوه ایجاد و توسعه سیستم را تحت تاثیر قرار دهد. ؟

مشخصات و ویژگیهای یک عامل عبارتند از:

- محدوده مسئولیت عامل

- محیط فیزیکی که عامل، سیستم را در آن استفاده خواهد کرد. انحراف از وضعیت ایده‌آل(وضعیتی که کاربر در یک محیط آرام، با هیچگونه حواس‌پرتی کار می‌کند) ممکن است تحت تاثیر چیزهایی مانند صدا، انتخاب فونت و استفاده مناسب از ترکیب وسایل ورودی(مانند صفحه کلید، ماوس، صفحه لمسی و کلیدهای حساس) قرار گیرد.
- تعداد کاربرانی که با این عامل نشان داده شده‌اند. تعداد کاربران هنگام تعیین اهمیت عامل و اهمیت قسمت‌های واسط کاربری که عامل استفاده می‌کند، یک فاکتور نسبی است.
- تناوب و تعداد دفعاتی که عامل از سیستم استفاده خواهد کرد. این تناوب استفاده ...

در اکثر موارد تخمین کلی از تعداد کاربران و تناوب استفاده کافی است. اختلاف بین ۳۰ و ۴۰ بر چگونگی شکل‌گیری واسط کاربری تاثیر نمی‌گذارد اما اختلاف بین ۳ و ۳۰ شاید تاثیر بگذارد.

سایر ویژگیهای عامل عبارتند از:

- سطح قلمرو دانش عامل. ...
- سطح مهارت عمومی کار با کامپیوتر. این سطح به تعیین اینکه چه تکنیک‌های تعاملی ساده یا پیچیده‌ای در ارتباط کاربر(user interface) مناسب هستند کمک می‌کند.
- سایر کاربردهایی که عامل استفاده می‌کند. بهره‌گیری مفاهیم واسط کاربر(user-interface) از این کاربردها زمان یادگیری عامل را کوتاه و بار حافظه او را کاهش می‌دهد. برای اینکه عاما از قبل با این مفاهیم آشناست.
- ویژگی‌های عمومی عامل، مانند سطح خبرگی(آموزش)، مفاهیم اجتماعی(زبان) و سن. این ویژگی‌ها می‌توانند جزئیات واسط کاربر(user interface) مانند فونت و زبان را تحت تاثیر قرار دهند.

این ویژگیها در ابتدا هنگام شناسایی مرز کلاسها و نمونه اولیه(prototype) استفاده می‌شود. ...

مثال:

در اینجا مثالی از ویژگی‌های عامل کاربر پست ارائه شده است. این عامل با مورد کاربرد مدیریت پیغام‌های پستی ورودی تعامل می‌کند:

- کاربر پستی یک کاربر ماهر کامپیوتر است
- محیط کاری یک کاربر پستی معمولا یک محیط آرام است.
- تعداد کاربران پستی مورد نظر ۵۰۰,۰۰۰ نفر است.

مورد کاربردها را پیدا کنید.

هنگامی که شما در ابتدا طرح کلی از عامل‌ها را تکمیل کردید، مرحله بعدی جستجو و پیداکردن مورد کاربردهای سیستم است. اولین مورد کاربردها خیلی مقدماتی هستند و بدون شک شما مجبور خواهید بود آنها را چندین بار آنها را تغییر دهید تا ثابت(stable) شوند. اگر چشم‌انداز یا نیازمندیهای سیستم

ناکارامد باشند یا آنالیز سیستم مبهم باشد، قابل استفاده بودن(functionality) سیستم نیز مبهم خواهد بود. بنابراین شما باید بطور مداوم از خود بپرسید که آیا شما مورد کاربردهای درستی را پیدا کرده‌اید. بعلاوه شما باید برای اضافه، حذف، ترکیب و یا تقسیم مورد کاربردها آماده شوید، قبل از اینکه به نسخه نهایی برسید. هنگامی که شما مورد کاربردها را بطور جزئی تشریح کردید، درک بهتری از آنها را بدست خواهید آورد.

بهترین روش برای پیدا کردن مورد کاربردها این است که خواسته‌ها و نیازهای هر عامل را از سیستم مورد توجه قرار دهیم. در نظر داشته باشید که سیستم تنها برای کاربرانش وجود دارد، بنابراین باید بر اساس نیازهای کاربرانش باشد. شما تعداد زیادی از نیازهای عامل‌ها را بواسطه نیازمندیهای وظیفه‌ای(functional requirements) سیستم تشخیص خواهید داد. برای هر عامل، انسانی یا غیره، سوالات زیر را از خودتان بپرسید:

- وظایف عمده‌ای که هر عامل انتظار دارد سیستم آنها را انجام دهد چه‌چیزهایی هستند؟
- آیا عامل داده‌هایی را در سسیتم ایجاد، ذخیره، تغییر، حذف و یا خواهد خواند؟
- آیا عامل باید سیستم را از تغییرات بیرونی ناگهانی(sudden external changes) آگاه سازد؟
- آیا عامل نیاز دارد که درباره اتفاقات و رویدادهای مشخصی از سیستم آگاه شود؟
- آیا عامل سیستم را روشن یا خاموش خواهد کرد؟

جواب به این سوالات، جریان وقایعی را نشان می‌دهد که مورد کاربردهای داوطلب و کاندید (candidate use cases) را شناسایی می‌کند. البته همه آنها مورد کاربردهای مجزا(separate) نیستند، بلکه برخی ممکن است بصورت تفاوت‌های(variants) یک مورد کاربرد یکسان مدل شده‌اند. همیشه گفتن اینکه چه‌چیزی تفاوت یک مورد کاربرد و چه‌چیزی یک مورد کاربرد مجزا و متمایز است، آسان نیست. با این وجود، این مطلب هنگامی که شما جریان وقایع را بطور جزئی و دقیق شرح می‌دهید آشکار خواهد شد.

به غیر از نیازمندیها، یک مدل شرکتی(enterprise model) از سازمان شما (که مدل کسب و کار نیز نامیده می‌شود) منبع باارزشی برای تعیین مورد کاربردهاست. این مدل تشریح می‌کند که چگونه سیستم اطلاعات می‌تواند عملیات موجود را یکپارچه کند و ایده خوبی را درباره محیط سیستم به شما بدهد. شما همچنینی مفاهیمی را پیدا خواهید کرد که باید در مدل شرکت تعریف شود، برای اینکه این مدل، اشیاء کسب و کار(business objects)شرکت را در بر می‌گیرد.

اگر شما مدلسازی کسب و کار را انجام داده باشید، مدل مورد کاربرد کسب و کار و مدل اشیاء کسب و کار را بعنوان ورودی خواهید داشت. برای اطلاعات بیشتر به بحث Going from Business Models to Systems مراجعه کنید.

یک سیستم ممکن است چندین مدل مورد کاربرد داشته باشد. بهترین راه برای پیدا کردن مدل بهینه، ایجاد دو یا سه مدل و انتخاب مدلی است که شما آنرا ترجیح می‌دهید و سپس مدل انتخاب شده را توسعه دهید. ایجاد چندین مدل جایگزین به شما کمک می‌کند سیستم را بهتر درک کنید.

هنگامی که اولین مدل مورد کاربردتان را طرح کردید، باید بررسی کنید که آیا این مدل همه نیازمندی‌های وظیفه‌ای شما را هدایت می‌کند یا نه. بررسی دقیق نیازمندیها این اطمینان را می‌دهد که مورد کاربردها، همه نیازمندیها را مورد توجه قرار داده‌اند.

برای اطلاعات بیشتر در مورد اینکه مورد کاربرد چیست و چگونه می‌توان آنها را پیدا کرد به بحث‌های مدل مورد کاربرد(Use-Case Model) و مورد کاربرد مراجعه کنید.

مورد کاربردهایی را که پیدا کرده‌اید نامگذاری کنید و بطور خلاصه شرح دهید.

هر مورد کاربرد باید نامی داشته باشد که نشان دهد در تعامل با عامل‌ها چه‌کاری انجام می‌دهد. نام ممکن است دارای چندین کلمه باشد تا درک شود. هیچ دو مورد کاربرد متفاوتی نمی‌توانند نام یکسانی داشته باشند. برای اطلاعات بیشتر می‌توانید به بحث Use Case مراجعه کنید.

Guidelines: Use Case

Use Case: یک نمونه مورد کاربرد(**use-case instance**) یک توالی از عملیاتی است که سیستم انجام می‌دهد تا یک نتیجه ارزشمند قابل مشاهده‌ای را برای یک عامل خاص بدست آورد.

یک مورد کاربرد(**use case**) مجموعه‌ای از نمونه‌های مورد کاربرد است.

تشریح مورد کاربرد:

چندین کلمه کلیدی در این تعریف وجود دارد که باید تشریح شوند:

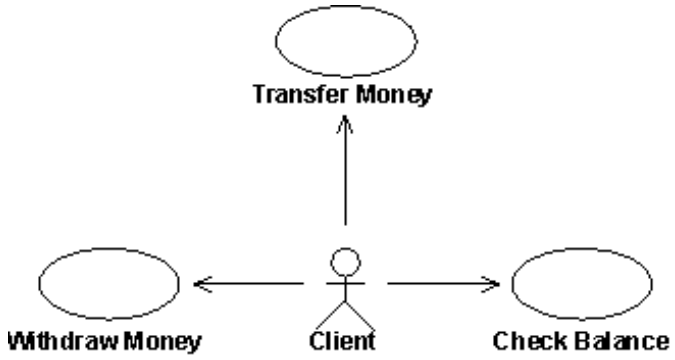
- نمونه مورد کاربرد(**Use-case instance**): منظور از توالی مطرح شده در تعریف، در واقع جریان خاصی از رویدادها و وقایع(flow of events) از میان سیستم یا نمونه است. جریان‌های وقایع و رویدادهای زیادی ممکن هست وجود داشته باشند و تعداد زیادی از آنها شاید یکسان باشند. برای ایجاد مدل مورد کاربرد قابل فهم، جریانهای وقایع مشابه را در یک مورد کاربرد گروه‌بندی کنید. شناسایی و تشریح یک مورد کاربرد در واقع بمعنی شناسایی و تشریح گروهی از جریان‌های وقایع مرتبط است.
- نمایش‌های سیستم(**System performs**): یعنی اینکه سیستم مورد کاربرد را تهیه می‌کند. یک عامل با یک نمونه مورد کاربرد از سیستم تعامل دارد.
- نتیجه ارزشی قابل مشاهده: شما می‌توانید ارزشی را به یک مورد کاربردی که بطور موفقیت آمیز اجرا شده، تخصیص دهید. یک مورد کاربرد باید تضمین کند که یک عامل می‌تواند کاری را که دارای ارزش قابل شناسایی است، انجام دهد. این مورد در تعیین سطح یا

دانه‌بندی(بزرگی یا کوچکی) صحیح یک مورد کاربرد خیلی مهم است. تعیین سطح صحیح و مناسب برای دستیابی به مورد کاربردهایی است که خیلی کوچک نیستند. در شرایط مشخصی، شما می‌توانید مورد کاربردی را بعنوان واحد برنامه‌ریزی در یک سازمان استفاده کنید.

- عملیات(**Actions**): یک عمل، رویه‌ای محاسباتی یا الگوریتمی است که یا هنگامی که عامل نشانه و سیگنالی را به سیستم می‌دهد شروع می‌شود و یا هنگامی که زمان وقوع رخدادی در سیستم فرا می‌رسد. یک عمل ممکن است انتقال سیگنال به یک یا چند عامل را نشان دهد.
- عامل خاص: عاملی کلیدی برای پیدا کردن use case درست است و به شما کمک می‌کند که از use case هایی که خیلی وسیع و بزرگ هستند اجتناب کنید. **هر عاملی تقاضاهای خاص خود را از سیستم دارد و بنابراین مجموعه** use case های مربوط به خودش را نیاز دارد.

قابلیت استفاده(functionality) **یک سیستم با** use cases های مختلفی تعریف می‌شود **که هر کدام جریان خاصی از وقایع را نشان می‌دهد. شرح هر** use case چیزی را که هنگام اجرای use case در سیستم اتفاق می‌افتد را بیان می‌کند.

برای مثال در یک ماشین پرداخت خودکار، کاربر می‌تواند برداشت پول از حساب، انتقال پول به حساب دیگر یا کنترل وضعیت حساب را انجام دهد. هر کدام از عملیات مربوط به جریانهایی است که شما می‌توانید **با** use case ها نشان دهید.



هر use case **برای اینکه اجرا شود کار (task) مربوط به خود را نیاز دارد. مجموعه** use case **ها، همه راه‌های ممکن استفاده از سیستم را تشکیل می‌دهند.**

چگونه use case **ها را پیدا کنیم**

در زیر مجموعه‌ای از سوالاتی که هنگام شناسایی use case ها مفید هستند ارائه شده است:

- وظایف هر عامل شناسایی شده در سیستم چیست؟
- **آیا عامل باید درباره وقایع مشخصی در سیستم آگاه باشد؟**
- **آیا عامل باید سیستم را از تغییرات ناگهانی بیرونی آگاه کند؟**
- **آیا همه قابلیت‌های سیستم با** use caseهایی که شما شناسایی کرده‌اید، انجام می‌شود؟
- **چه همه** use case ها سیستم را پشتیبانی یا نگهداری می‌کنند؟
- **چه اطلاعاتی را باید در سیستم اصلاح یا ایجاد کنیم؟**

جریان وقایع را طرح‌ریزی کنید.

در این نقطه شما همچنین خلاصه‌ای از جریان وقایع مورد کاربرد را بنویسید اما وارد جزئیات نشوید. هر شخصی که بعدا مورد کاربرد را تعیین می‌کند، این تشریح مرحله به مرحله را نیاز دارد.

با ارائه طرح کلی از جریان وقایع اولیه شروع کنید و هر بار جریان‌های جایگزینی را به آن اضافه کرده و روی آن توافق کنید.

مثال:....

نیازمندیهای اضافه را جمع‌آوری کنید.

برخی نیازمندیهای سیستم را نمی‌توان به مورد کاربرد خاصی تخصیص داد. این نیازمندیها را در خصوصیات تکمیلی (Supplementary Specifications) جمع‌اوری کنید.

شرح دهید که چگونه عاملها و مورد کاربردها باهم تعامل دارند.

برای اینکه نشان دادن اینکه چگونه عامل‌ها با مورد کاربردها ارتباط دارند مهم است، بنابراین شما باید یک مورد کاربرد را پیدا کنید و ارتباط آنرا با عامل‌هایی که با آن در تعامل هستند، برقرار کنید. برای انجام این کار شما باید یک ارتباط-پیوستگی(communicates-association) را تعریف کنید که در یک جهت مانند انتقال علامت بین عامل و مورد کاربرد شناور است.

انتقال علامت معمولا در هر دو جهت است. در این موارد، شما باید اجازه دهید که ارتباطات-وابستگی‌ها(communicates-associations) در هر دو جهت شناور باشد. غالبا یک ارتباط وابستگی را برای هر جفت عامل و مورد کاربرد تعریف کنید. همچنین باید بطورخلاصه هر ارتباط وابستگی را که تعریف کرده‌اید تشریح کنید. برای اطلاعات بیشتر در مورد ارتباطات – وابستگی‌ها بحث communicates-associations ببینید.

مورد کاربردها و عامل‌ها را بسته‌بندی کنید.

اگر تعدادی عامل یا مورد کاربرد خیلی بزرگ شوند، آنها را در بسته‌های مورد کاربرد تقسیم کنید تا نگهداری مدل مورد کاربرد ساده‌تر باشد. این کار همچنین فهم مدل مورد کاربرد را آسان‌تر می‌کند و تخصیص مسئولیت‌ها در مدل مورد کاربرد ساده‌تر می‌کند به این صورت که توسعه‌گران مسئول بسته‌های مورد کاربردها یا عامل‌ها باشند. برخی روشهای بسته‌بندی مورد کاربرد با همدیگر به صورت زیر است:

اگر آنها با یک عامل تعامل دارند

اگر آنها روابط شاملی یا توسعه‌ای با هم دارند.

...

مدل مورد کاربرد را در قالب نمودارها نشان دهید

شما می‌توانید روابط بین مورد کاربردها و عاملها را همانند روابط بین مورد کاربردهای مرتبط در قالب نمودارهای مدل مورد کاربرد تشریح کنید. این نمودارها ممکن است هر یک از موارد زیر را در بر بگیرد:

- متلقات و وابستگی‌های عامل‌ها به یک بسته موردکاربرد یکسان
- یک عامل و همه مورد کاربردهایی که با آن در تعامل دارند
- مورد کاربردهایی که اطلاعات یکسانی را محاسبه(handle) می‌کنند
- مورد کاربردهایی که با گروه یکسانی از عامل‌ها استفاده می‌شود
- مورد کاربردهایی که اغلب در یک توالی اجرا می‌شوند.
- مورد کاربردهایی که متعلق به یک بسته مورد کاربرد یکسانی هستند.
- مهمترین مورد کاربردها.

نمودار این نوع می تواند بعنوان خلاصه‌ای از مدل بکار رود و احتمالا در دیدگاه مورد کاربرد در نظر گرفته می‌شود.

مورد کاربردهایی که باهم ایجاد شده‌اند(در قالب یک توسعه)

برای اطلاعات بیشتر می‌توانید به بحث Guidelines: Use-Case Diagram مراجعه کنید.

یک بررسی از مدل مورد کاربرد را ایجاد کنید

در تشریح بررسی تشریح مدل مورد کاربرد موارد زیر را در نظر بگیرید:

توالی‌های نوعی که در مورد کاربردهایی که توسط کاربران استفاده می‌شوند.

وظیفه‌ای که توسط مدل مورد کاربرد انجام نمی‌شود.

برای اطلاعات بیشتر بخش تشریح بررسی را در بحثGuidelines: Use-Case Model ببینید.

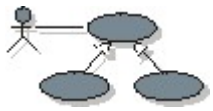
ارزیابی نتایج:

در این مرحله شما باید مدل‌مورد کاربردتان را تست کنید اما مدل را بطور دقیق بازنگری نکنید. شما همچنین باید چک‌لیستی را برای مدل مورد کاربرد داشته باشید هنگامی که شما روی آن کار می‌کنید. چک لیست‌های هر عامل، مورد کاربرد و مدل مورد کاربرد در بحث Activity: Review Requirements ارائه شده است.

در این مرحله تصویب مدل مورد کاربرد توسط افرادی که خارج از تیم توسعه هستند(مانند کاربران و مشتریان) مهم است. بنابراین شما باید کاربران و مشتریان را در بازنگری مدل مورد کاربردتان قبل از اتمام این فعالیت در نظر داشته باشید. قسمت‌های ذینفع باید موارد زیر را تعیین کنند:

- آیا همه مورد کاربردهای لازم شناسایی شده‌اند.
- آیا مورد کاربردهای غیر ضروری نیز شناسایی شده‌اند.
- آیا رفتار هر مورد کاربردی با ترتیب درست انجام می‌شود.
- آیا جریان وقایع هر مورد کاربردی در این مرحله کامل شده است.

- آیا تشریح بررسی مدل مورد کاربرد آن را قابل فهم کرده است.



مدل Use-Case

مدلی از فرایندهای سیستمی مورد انتظار و محیط مربوطه می باشد، تعاملات کاربر با سیستم را نمایش می دهد. این مدل به عنوان ورودی اصلی برای تجزیه و تحلیل و طراحی و تست می باشد.
این مدل بعنوان ورودی اصلی تحلیل و طراحی معماری سیستم است. مدل در ابتدا در فاز شناخت تهیه شده و در طی فاز معماری استفاده می‌شود. مسئولیت تولید آن با تحلیل گر سیستم است.

Elaboration (معماری)

هدف این فاز ایجاد معماری سیستم است تا پایه مستحکمی را برای فعالیت‌های طراحی و اجرا در فاز ساخت فراهم کنیم. معماری مهم‌ترین نیازمندیها یعنی آنهاییکه بیشترین تاثیر را بر معماری سیستم دارند و همچنین ارزیابی ریسک را در بر می‌گیرد. پایداری معماری با یک یا چند شکل اولیه معماری(prototype) ارزیابی می‌شود.

هدف اصلی این فاز بررسی موشکافانه کلیه جوانب و مسایل دربرگیرنده پروژه، ایجاد طرح معماری سیستم، طرح‌ریزی پروژه و برطرف نمودن کلیه عناصر مخاطره‌انگیز پروژه می‌باشد. طرح معماری سیستم برمبنای درک صحیحی از تمامی سیستم انجام می شود؛ این امر مستلزم آن است که اکثر قالب‌های عملیاتی سیستم مشخص شده باشد. ماحصل کار استخراج برخی محدودیت‌های واقع شده در خصوصیات تکمیلی سیستم می‌باشد. در پایان این فاز جزئیات، اهداف و دامنه سیستم مشخص شده و یک معماری خاص انتخاب می گردد و مخاطرات اصلی سیستم مرتفع می‌شود.

روشن است که فاز جزئیات (elaboration phase) بحرانی‌ترین فاز در بین ۴ فاز ذکر شده می‌باشد؛ چرا که در خاتمه این فاز، مهندسی بسیار دشواری که با تفکر و تعمق خاص انجام پذیرفته بود، تکمیل شده و تصمیم گرفته می‌شود که پروژه مورد نظر قابلیت ورود به فاز ساخت را داراست یا خیر.

از آنجایی که یک پروژه همواره دستخوش تغییر و تحول می باشد، در این فاز باید اطمینان حاصل نمود که معماری، نیازمندیها و طرح پروژه به اندازه کافی پایدار هستند و از عوامل مخاطره‌انگیز به میزان مطلوبی کاسته شده است تا جایی که پیش‌بینی هزینه‌ها و زمان‌بندی پروژه امکان‌پذیر باشد. در این فاز طی یک یا چند تکرار (iterate) بر مبنای دامنه، اندازه، ریسک و شکل اولیه پروژه (prototype) معماری قابل اجرای پروژه ساخته می‌شود. این معماری بر مبنای قالب‌های عملیاتی (use case) استخراج شده در فاز آغازین (inception phase) پروژه می‌باشد، که عموماً در معرض بزرگترین مخاطرات تکنیکی پروژه قرار می‌گیرد. اهداف اولیه این فاز عبارتند از:

- تعریف و تأیید اعتبار و معماری سیستم تا حدی که عملی باشد.
- بنیانگذاری چشم انداز سیستم
- بنیانگذاری یک طرح صحیح و قابل اجرا برای فاز ساخت (Construction phase)
- اثبات اینکه طرح معماری سیستم کلیه نیازها و انتظارات را با هزینه مناسب و در زمان قابل قبول برآورده خواهد کرد.

برای دستیابی به اهداف اولیه همزمان با این فاز پشتیبانی محیطی انجام می‌گیرد.

خروجی‌های اصلی این فاز عبارتند از:

۱- شکل اولیه(prototype)

۲- لیست ریسکها

۳- مورد توسعه
۴- سند معماری نرم‌افزار
۵- مدل طراحی
۶- مدل داده
۷- چشم‌انداز
۸- برنامه توسعه نرم‌افزار
۹- مدل use-case
۱۰-خصوصیات تکمیلی

و خروجی‌های اختیاری آن عبارتند از:

۱- مورد کسب و کار
۲- مدل تحلیل
۳- اسناد خاص پروژه

Construction (ساخت)

در حین انجام این فاز مکرراً محصولات کاملی تولید می‌شود که آمادگی انتقال برای استفاده کاربران را دارا می‌باشند. در این فاز طراحی سیستم استخراج شده و فعالیتهای اجرا و تست نرم‌افزار تکمیل می‌گردد. در پایان این فاز بررسی می‌شود که آیا سازمان و کاربران سیستم آمادگی لازم برای عملیاتی شدن سیستم را دارند یا خیر.

در خلال روند این فاز تمامی اجزا (component) و قابلیت‌های (feature) باقیمانده سیستم ساخته شده و در محصول مورد نظر قرار می گیرند و تمامی قابلیت‌ها تست می‌شوند. این فاز به‌عبارت دیگر یک فرآیند تولیدی است، که تکیه بر مدیریت منابع و کنترل عملیات برای بهینه‌سازی هزینه‌ها، زمان‌بندی و کیفیت پروژه دارد.

در واقع مدیریت از ساخت یک سری ویژگی‌های ذهنی در مرحله inception و elaboration به ساخت یک محصول قابل گسترش در مراحل construction و transaction می‌رسد.

اولین باری که Componentها ایجاد می‌شوند و ارتباطات آنها رسم می‌گردد، تولید کردن کد بر پایه طراحی شروع می شود. این کد عموماً شامل تعریف کلاس، صفات، محدوده، نوع توابع و دستورات و دانش می‌باشد.

بعد از کدنویسی، برنامه‌نویسان می‌توانند بر روی جنبه‌های ویژه تجاری متمرکز شوند و فعالیت های زیر را انجام دهند:

- بازبینی کد براساس استانداردهای طراحی و توافقات حاصله در مراحل قبلی
- بازبینی آبجکتها به منظور تضمین کیفیت

این فاز زمانی به پایان می‌رسد که نرم‌افزار تکمیل شده و تست گردیده است. این امر مهم است که اطمینان همزمانی مدل و نرم افزار در انتهای این فاز حاصل شود.

Transition (انتقال)

در این فاز نرم افزار در دست کاربران آن قرار می‌گیرد. ممکن است نیازهای جدیدی برای متعادل کردن سیستم بوجود آید و یا مسایل حل نشده سیستم مرتفع گردد و نیز قابلیت‌های سیستم که تولید آنها به تعویق انداخته شده است، نهایی گردد.این فاز عمدتا با تولید نسخه بتا آغاز می‌گردد. در پایان این فاز بررسی می‌شود که آیا همه جوانب سیستم دیده شده است یا باید چرخه ساخت جدیدی را آغاز نمود.

فاز انتقال زمانی آغاز می‌شود که زیربنای پروژه آنقدر محکم شده باشد که بتوان آنرا برای استفاده کاربر گسترش داد. در واقع برای کسب نتیجه ای معقول و مناسب باید برخی بخش‌های تکمیل شده سیستم که دارای کیفیت مطلوبی می‌باشند همراه با اسناد و مدارک کاربری در اختیار کاربران نهایی قرار داد.این مقوله شامل :

- تست نسخه بتا از لحاظ مطابقت با انتظارات کاربران
- تست نسخه بتا همراه با عملکرد موازی سیستم جدید با سیستم قدیمی
- تبدیل پایگاه داده قابل استفاده
- آموزش کاربران و نگهدارندگان سیستم
- بازاریابی، توزیع و فروش

این فاز هنگامی تکمیل می‌کردد که از دیدگاه کاربران نهایی، فعالیت استقرار (Deployment) سیستم کامل شده است. برای برخی نرم‌افزارها نقطه پایان این چرخه همزمان با نقطه شروع چرخه دیگری است که منجر به تولید نسخه بعدی محصول مورد نظر می‌شود.

در هر یک از فازهای اشاره شده الگوهای کاری وجود دارد که به ۹ دسته تقسیم می‌شود . یک الگوهای کاری نمایانگر تمام اعمالی است که در حین تولید محصولات نرم‌افزاری انجام می‌شودعموما الگوهای کاری از لحاظ ماهیت کار به دو دسته تقسیم می‌شوند.

- الگوهای کاری مربوط به تولید نرم‌افزار
- الگوهای کاری مربوط به مدیریت تولید نرم‌افزار

هر یک از این الگوها شامل مجموعه فعالیت‌هایی است که خروجی هریک از این فعالیت‌ها مجموعه‌ای از اسناد و مدل‌ها می‌باشد. این اسناد در قالب اسناد استاندارد RUP (Template) شناخته شده است.

پیوست الف: لغت‌نامه(glossary)

هدف:

تعریف لغات مشترکی که در همه شرح‌های متنی سیستم، می‌توان استفاده کرد، بویژه در شرح‌های مورد-کاربرد.

مراحل:

- پیدا کردن اصطلاحات مشترک
- ارزیابی نتایج

پیدا اصطلاحات مشترک:

برای پیدا کردن اصطلاحات مشترک در محدود مسئله، اصطلاحاتی را که در نیازمندیها و دانش عمومی تیم توسعه سیستم استفاده می‌شود را باید مورد توجه قرار دهیم. روی اصطلاحاتی که مفاهیم زیر را تشریح می‌کنند تمرکز کنید:

- مفاهیم نمایش اشیاء کسب و کار که در کار روزانه سازمان یا در محیط عملیاتی سیستم استفاده می‌شود. در بسیاری از موارد لیستی از این نوع مفاهیم از قبل وجود دارد.
- اشیاء دنیای واقعی که سیستم نیاز دارد که نسبت به آنها آگاه باشد. این اشیاء بطور طبیعی رخ می‌دهند و شامل مواردی از قبیل: ماشین، سگ، بطری، هواپیما، مسافر، بلیط، فاکتور و غیره.

مثال:

در یک سیستم بررسی انبار، بیشتر بحث در باره اقلامی که باید انبار شوند و محل‌های مناسب برای ذخیره آنهاست.

- وقایعی و رویدادهایی(event) که سیستم نیاز دارد تا نسبت به آنها آگاهی داشته باشد. در اینجا منظور از رویداد، نقطه‌ای از زمان یا ترتیب وقوع وقایع است، مانند ملاقات، یا وقوع یک خطا.

مثال:

یک رویداد طبیعی در سیستم انبار، تحویل کالا به انبار است. برای هر تحویل، سیستم باید زمان تحویل، فردی که کالا را دریافت می‌کند، اقلامی که تحویل می‌شوند و تعدادی که از هر نوع وجود دارد را به خاطر داشته باشد.

هر اصطلاح معمولا به صورت یک اسم همراه با تعریف آن نمایش داده می‌شود. اصطلاحات باید منفرد باشند، نه به صورت جمع. همه قسمت‌های درگیر باید روی تعاریف اصطلاحات توافق داشته باشند.

ارزیابی نتایج:

در این مرحله باید لغت نامه را برای اصلاح و هدایت کار در مسیر درست چک کنیم. البته نیازی به بازنگری در جزئیات نیست. برای این‌کار می‌توان از چک لیست زیر استفاده کرد.
الگوی (**Template**) تهیه لغت نامه در ارائه شده است.

منابع و مراجع:

1- Rational Unified Process 2003 Environment(additional in Rational CD).

2- The Enterprise Unified Process: Extending the Rational Unified Process, By Scott W. Ambler, John Nalbone, Michael J. Vizdos; Prentice Hall PTR; February 11, 2005