فهرست

مقدمه ای بر UM۱..... ٤

Polymorphism (چند درختی)..... ۷

مدلسازی بصری (visual modeling) چیست؟..... ۷

نمودارهای use case..... ۹

نمودارهای Callaboration..... ۱۲

نمودارهای class (کلاس)..... ۱۳

نمودارهای حالت (state transition digrmm)..... ۱٤

نمودارهای اجزاء (component Diagrams)..... ۱٦

نمودارهای Deployment..... ۱۸

مدلسازی بصری و پردازش تولید و توسعه نرم افزار..... ۱۹

Elagboration (مهارت)..... ۲۰

Construction(ساختار)..... ۲۱

Transtion (انتقال)..... ۲۲

کار با Use case ها..... ۲٤

مشاهده شرکت کنندگان در یک Use case..... ۲٥

ساختن Use case های Abstract (مجرد)..... ۲٦

مشاهده رابطه های متعلق به یک Use case..... ۲٦

افزودن عامل ها..... ۲۷

رابطه های extend..... ۲۸
نمودارهای interaction..... ۲۹
کار با کلاس..... ۳۷
نسبت دادن یک Stereotype به کلاس..... ۳۸
کار با بسته‌ها..... ۳۸
تنظیم visibility صفت..... ٤۱
تنظیم محدودیتهای صفت..... ٤۱
یافتن عملیاتها..... ٤۲
تعریف protocol عملیات..... ٤۳
تعریف qualifications عملیات..... ٤۳
تعریف exeptions استثنائات عملیات..... ٤٤
Rational Rose چیست؟..... ۵۲
نصب Rose ۹۸..... ۵۵
بخشهای صفحه نمایش..... ۵٦
Browser (مرورگر)..... ۵۷
Documentation window (پنجره مستند سازی)..... ۵۷
Toolbar (نوار ابزار)..... ۵۸
Diagram window (پنجره نمودار)..... ۵۸
Log هنگامی که روی مدل Rose کار می‌کنید،..... ۵۸

چهار نمای وجود در یک مدل Rose:..... ۵۹
use case..... ۵۹
نمای منطقی (logica view)..... ۵۹
نمای component..... ۶۰
نمای Deployment..... ۶۱
کاربرد برنامه Retional Rose..... ۶۱
وارد کردن و ارسال مدلها..... ۶۲
کار با واحدهای کنترل شده..... ۶۳

مقدمه ای بر UM١

- یادگیری متد object- oriented برنامه نویسی شی گرا و visual modeling (مدلسازی بصری)
- بررسی انواع نمادهای گرافیکی
- نگاهی به انواع نمودارهای UML (UML Diagrams)
- توسعه نرم افزار با استفاده رز مدلسازی بصری (visual modeling)

مقدمه ای بر متد object- oriented (شی گرایی)

در متد شی گرایی (٠,٠) برنامه به قطعات بسیار کوچک یا آبجکت هایی تقسیم می‌شود که تا اندازه ای مستقل از یکدیگرند مانند ساختمانی از بلوک ها.

در اولین گام تعدادی آبجکت های اساسی (انوع مختلف بلوک ها) را بسازید یا به دست آزمایشی آورید. اولین باری که شما این بلوک های ساختمانی را دارید, می‌توانید آنها را کنار هم گذاشته تا قصرتان را بسازید. به محض اینکه تعدادی آبجکت های اساسی در دنیای کامپیوتر ساختید یا به دست آورید می‌توانید به سادگی آنها را کنار هم بگذارید تا برنامه های جدید را ایجاد نمایید. یکی از امتیازات اساسی متد شی گرایی این است که می‌توانید یک بار component (اجزا) را ساخته و بارها و بارها از آنها استفاده کنید. درست مانند زمانی که می‌توانید یک بلاک ساختمانی را در یک قصر, یک خانه یا یک سفید فضایی دوباره استفاده کنید, می‌توانید از یک قطعه طرح یا کد شی گرایی در یک سیستم حسابداری, یک سیستم بازرگانی یا یک سیستم پردازش سفارش استفاده مجدد نمایید.

تفاوت شی گرایی با روش سنتی: در روش سنتی, روش توسعه به همراه اطلاعاتی که سیستم نگهداری خواهد کرد به خودتان وابسته است. در این روش پایگاه داده بر اساس نیازهای اطلاعاتی کار بران طراحی می‌کنیم و صفحاتی تهیه می‌کنیم تا اطلاعات را بگیرد, و گزارشاتی را چاپ می‌کنیم تا اطلاعات را برای کاربر نمایش دهد. یعنی بر روی اطلاعات متمرکز می‌شویم و کم توجه می‌کنیم که چه کاری با این اطلاعات انجام شده است یا رفتار سیستم چگونه است. این روش data- centric (مبتنی بر داده) نامیده شده است. مدلسازی data- centric مخصوص طراحی پایگاه داده و گرفتن اطلاعات خیلی سهم می‌باشد, اما انتخاب این روش در زمان طراحی برنامه های تجاری با مشکلاتی همراه است. یک چالش بزرگ این است که در خواهشهای سیستم چندین بار تغییر خواهند کرد.

سیستمی که روش data- centric استفاده می‌نماید, می‌تواند به آسانی تغییر در پایگاه داده را مدیریت نماید. اما اجرای تغییرات در قوانین تجاری یا رفتار (behavior) سیستم آن قدر آسان نمی باشد.

با استفاده از متد شی گرایی هم بر اطلاعات و هم بر رفتار متمرکز شویم.

مزیت این انعطاف پذیری با طراحی یک سیستم شی گرایی به خوبی شناخته شده است.

اصول شی گرایی عبارتند از: نهان سازی (Encapsulation), وراثت (Inheritance) و چند ریختی (Polymorphism)

Enlopsulation (نهان سازی)

در سیستم های شی گرایی, اطلاعات و رفتارها را در یک آبجکت بسته بندی می‌کنیم. این مطلب در قالب اطلاعات Encapsulation (پنهان سازی) ارجاع داده شده است و یا

می‌توانیم برنامه را به بخشهای کوچکی از توابع وابسته, تقسیم کنیم. مثلا یک حساب بانکی شامل: شماره حساب, تراز جاری, نام مشتری, آدرس., نوع حساب, نرخ بهره و تاریخ باز کردن حساب می‌باشد. رفتارهایی هم برای یک حساب بانک داریم مانند: باز کردن حساب, بستن حساب, به حساب گذاشتن, برداشت از حساب, تغییر نوع حساب, تغییر مشتری و تغییر آدرس ما این اطلاعات و رفتارها را باهم در یک آبجکت account پنهان می‌کنیم. در نتیجه, همه تغییرات سیستم بانکی تاثیرات اعمال شده به سیستم را محدود می‌کند. یک مفهوم مشابه نهان سازی,Information Hiding است, پنهان سازی اطلاعات توانایی است که جزئیات مبهم یک آبجکت را در نیای خارج پنهان می‌نماید. دنیای خارج به معنی هر چیزی از خارج از همان آبجکت دست حتی اگر چه دنیای خارج شامل بقیه سیستم باشد

Inheritance (وراثت)

در سیستم های شی گرا وراثت به شما اجازه می‌دهد تا آبجکت های جدید را بر پای ابجکت های قدیمی ایجاد کنید. آبجکت CHILD ویژگی هایی یک آبجکت PARENT را به ارث می‌برد.

یکی از مزایای اصل وراثت، سهولت در نگهداری است. وقتی چیزی تغییر می‌کند و بر همه تاثیر می گذارد، فقط آبجکت والد نیاز به تغییر دارد و آبجکت های فرزند به طور خورکار تغییرات را به ارث می برند. مثلا در طبعیت، اگر پستانداران به طور ناگهانی خونسرد شوند، فقط آبجکت پستانداران (mamaal) باید تغییر نماید. در یک سیستم بانکداری ممکن است از وراثت برای انواع مختلفی از حسابهایی که داریم استفاده کنیم.

این نوع مختلف حسابها شباهتهایی نیز دارند. هر کدام دارای یک شماره حساب، نرخ بهره و نام مالک می‌باشند بنابراین می‌توانیم یک آبجکت والد بنام account (حساب) را ایجاد نماییم تا ویژگی های مشترک همه این حسابها را نگهداری می‌کنیم آبجکت های فرزند (child) می‌توانند علاوه بر ویژگی هایی که به ارث برده اند، ویژگی ها منحصر به فرد خودشان راداشته باشند، مثلا حساب اعتباری یک حد موجودی و حداقل میزان پرداخت را خواهد داشت. سپرده گذاری نیز دارای یک موعد پرداخت می‌باشد.

تغییرات آبجکت والد بر روی همه فرزندان اثر خواهد گذاشت اما بچه ها آزاد هستند که بدون بر هم زدن آرامش فرزند دیگر یا والدشان تغییر نمایند.

Polymorphism (چند درختی)

سومین اصلی شی گرایی، ploymor phism است که به این معنی است که شکل ها یا پیامدهای زیادی از یک تابع ویژه را داشته باشیم. همانند وراثت، چند ریختی نیز در دنیای طبیعی دید می‌شود. چند ریختی در اصطلاحات یک سیستم شی گرایی به این معنی است که ما می‌توانیم بسیاری از رخداد ها یا پیامدهای یک عمل ویژه را داشته باشیم.

مثلا ممکن است یک سیستم رسم اشکال گرافیکی را بسازیم.

مدلسازی بصری (visual modeling) چیست؟

یک طرح کلی به شما کمک می‌کند تا قبل از اینکه سیستم را بسازید آن را طراحی نمایید و در این صورت سیستم می‌تواند حتی در مقابل کوهی از تغییرات درخواست، مقاومت نماید. پس از جمع آوری درخواستهای خود، آن ها را تبدیل به کد می‌نمایید با تبدیل رسمی درخواستها به کد، می‌توانید مطمئن شوید که واقعا درخواستها به وسیله که مطرح شده اند

و آن کد می‌تواند به آسانی راه برگشت به درخواستها را طی کند این پردازش modeling (مدلسازی) نامیده شده است.

نتیجه پردازش مدلسازی این توانایی است که نیازهای تجاری را به درخواستهایی تبدیل کند تا در کد به صورت مدل در آید و آن را دوباره برگردند بدون اینکه درطول راه چندی گم شود.

مدلسازی بصری (visual modeling) پردازش گرفتن اطلاعات از مدل است و آن را با استفاده از مجموعه ای از عناصر گرافیکی استاندارد به صورت گرافیکی نشان می‌دهد. هدف اصلی مدلسازی بصری، ارتباط میان کاربران، برنامه نویسان، تحلیلگران، آزمایش کننده ها، مدیران و هر شخص دیگری که با پروژه در گیر شده است می‌باشد بعد از ایجاد این مدلها، می‌توانیم آنها را به همه بخشهای وابسته نشان دهیم و آن بخشها می‌توانند اطلاعات را از مدل به دست آورند. در مدلسازی بصری از نمادهای گرافیکی (مثل object modeling technolohy oM T, Booch تکنولوژی مدلسازی شی و unified Modeling Language زبان مدلسازی یکپارچه) برای نشان دادن چره های مختلف یک سیستم استفاده می‌شود.

- نمودارهای UML
- نمودار use case
- نمودار sequence (توالی)
- نمودار collaboration (همکاری)

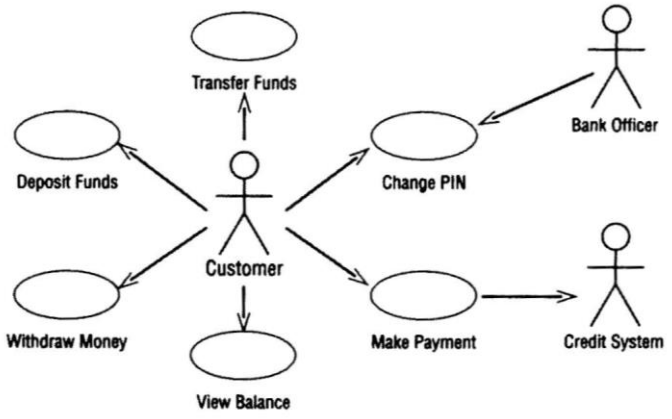
- نمودار class (کلاس)
- نمودار state transition (در حالت)
- نمودار component
- نمودار Deployment

این نمودار ها جنبه های مختلفی از سیستم را نشان می‌دهند.

نمودارهای use case

نمودار های use case محاورات میان use case ها را نشان میدهد که عملیات سیستمی و عاملها (Actor) که نشان دهنده افراد یا سیستم هایی است که اطلاعات را برای سیستم فراهم کرده است و یا از آن دریافت می‌کنند را نمایش می‌دهند. use case ها درخواستهای سیستم را از دید کاربر نشان می‌دهند. بنابراین vse case ها عملیاتی هستند که سیستم فراهم می‌کند. عامل ها در واقع نگهدارنده پول (بانکدار) یک سیستم هستند. این نمودارها نشان می‌دهند که چه عاملهایی به use case ها مقدار اولیه می‌دهند. همچنین آنها نشان می‌دهند که چه موقع یک عامل، اطلاعات را از یک use case دریافت می‌کند.

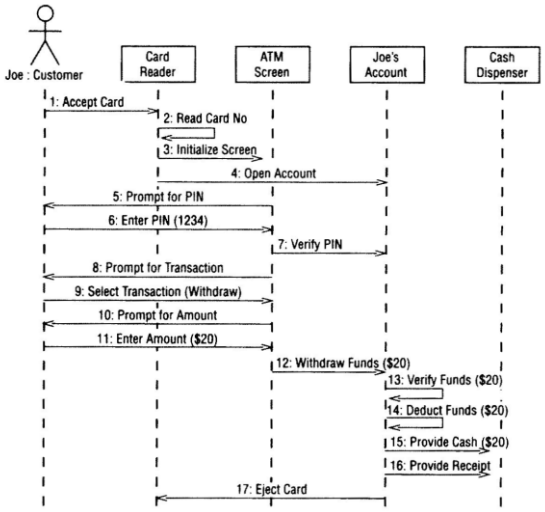
نمودار Use Case یک ATM (Automated Teller Machine)



تعدادی از ارتباطات این ارزش را دارند که بیشتر بـه آنهـا اشـاره مـی‌شـود. کارمنـد بانـک همچنین، به use case تغییر PIN مقدار اولیه می‌دهد. use case پرداخـت، فلشـی را نشـان می‌دهد که به سیستم اعتباری می‌رود سیستم های خارجی ممکن اسـت عاملهـایی باشـند و در این مورد، سیستم اعتباری به عنوان یک عامل نشان می‌دهد کـه use case اطلاعـاتی را تولید می‌کند که یک عامل از آن استفاده می‌کند. در این مورد use case پرداخـت، اطلاعـات پرداختی کارت اعتباری را برای سیسـتم اعتبـاری آمـاده مـی‌کنـد. اکثـر اطلاعـات دزدیـدن نمودارهای use case قابل فهم می‌باشند زیر ایـن نمـودار همـه عملیـات سیسـتم را نشـان می‌دهد. کاربران، مدیران پروژه، تحلیلگران، برنامه نویسان، مهندسان تضمین کیفیـت و هـر شخص دیگری که به سیستم وابسته است، می‌توانـد ماننـد همـه، ایـن نمودارهـا را ببینـد و بفهمد که چه سیستمی قرار است به انجام برسد.

نمودارهای sequence (توالی)

این نمودارها، برای نشان دادن جریان عملیات در یک use case استفاده شده اند، مثلا use case برداشت پول چند توالی sequences دارد مانند برداشت پول، تـلاش بـرای برداشـت پول از حساب بدون موجودی، تـلاش بـرای برداشـت پـول بـا PIN اشـتباه و غیـره طـرح معمولی برداشت ۲۰ دلار پول و بدون هیچ مشکلی مانند وارد کردن PIN اشتباه یـا وجـوه ناکافی در حاسب در شکل زیر نشان داده شده است.

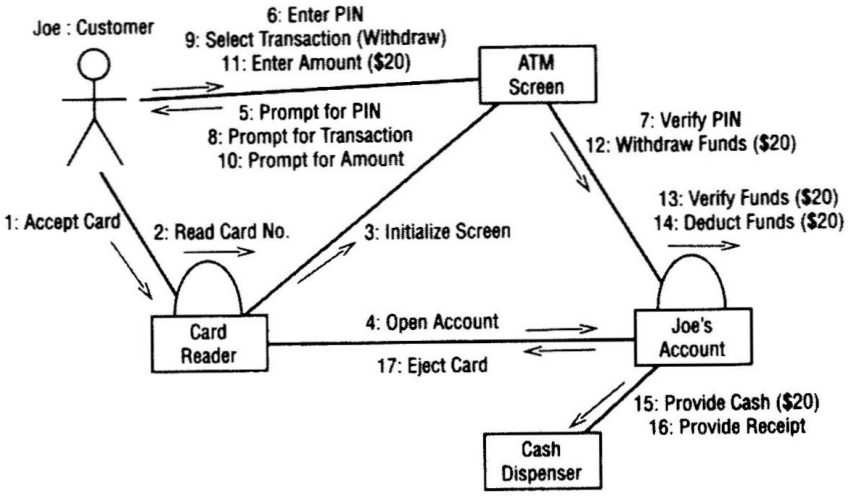


نمودار sequence جریان پردازش را در use case برداشت پول نشـان مـی‌دهـد عاملهـای وابسته در بالای نمودار نشان داده شده اند، عامل مشتری هم در آن نشان داده شده است. هر فلش یک پیغام ارسالی بین عامل و آبجکت، یا آبجکت را نمایش می‌دهد تا عملیات مـورد نیاز را به انجام برساند. نمودارهای sequence آبجکت ها را نمایش می‌دهند و نـه کلاسـها use case بدین ترتیب شروع می‌شود که مشتری کارتش را وارد کارت خوان می‌کند. کارت خوان شماره کارت را می‌خواند. آبجکت حساب joe را باز می‌کند و صـفحه نمـایش ATM را مقدار دهی می‌نماید. .صفحه نمایش از joe می‌خواهد که PIN را وارد نمایـد. او ۱۲۳٤ را وارد می‌کند. صفحه PIN را با آبجکت حساب تایید می‌کنـد و آنهـا را بـه هـم جفـت وجـور می‌کند. صفحه انتخابهایش را برای joe آماده می‌کند و او ۲۰ دلار را انتخاب می‌کند. سـپس صفحه وجـوه را از حسـاب بـر مـی‌دارد. ایـن سـری از پردازشـهایی کـه آبجکـت حسـاب (account) به انجام می‌رساند را مقدار دهی می‌کنـد. ابتـدا، حسـاب joe تاییـد مـی‌کنـد کـه حساب، حداقل شامل ۲۰ دلار است سپس وجوه را از حساب کسر می‌کند بعدا بـه صـندوق اطلاع می‌دهد که ۲۰ دلار را آماده کند. همچنین حساب joe به صندوق اطلاع می‌دهد با یـک

رسید آماده کند. سرانجام به کارت خوان اطلاع می‌دهد تا کارت را باز پس دهد. بنابراین، این نمودار sequence تمام جریان پردازشی use case برداشت پول را با نشان دادن یک مثال مشخصی از اینکه joe دلار از حسابش بر می‌دارد را توضیح می‌دهد. کاربران می‌توانند به این نمودارها نگاه کنند مشخصات پردازش تجاریشان را ببینند. تحلیلگران جریان پردازش را در نمودار sequence می‌بینند. برنامه نویسان آبجکت هایی که به کد نویسی نیاز دارند را به همراه عملکردهای آن آبجکت ها می‌بینند. مهندسین تضمین کیفیت می‌توانند جزئیات پردازش و تولید test cas مبتنی بر پردازش را ببینند. نمودارهای sequence برای همه کسانی که در پروژه مسئول نگهداری پول هستند، مفید است.

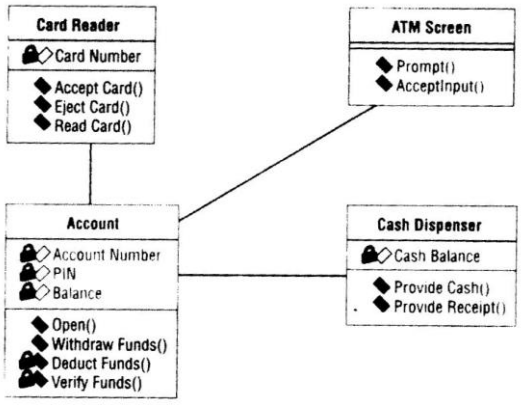
نمودارهای Callaboration

نمودارهای collaboration دقیقا همان اطلاعات نمودارهای sequence را نشان می‌دهند. اگر چه نمودارهای collaboration اطلاعات را به روشی متفاوت و با یک هدف متفاوت نشان می‌دهند. در نمودارهای sequence آبجکت ها و ارتباطات عامل ها به ترتیب زمان توضیح داده شده اند، در حالی که در نمودار collaboration آبجکت ها و فعل و انفعالات عامل ها را بدون توجه به زمان نشان میدهد. در نمودارهای Collaboratim افراد به دلایل مختلف به این نمودارها مراجعه می‌کنند.



نمودارهای class (کلاس)

نمودارهای (class) کلاس ارتباطات بین کلاسها را در سیستم نشان می‌دهند. کلاسها می‌توانند بدون طرحی کلی برای آبجکت ها دیده شوند مثلا حساب joe یک کلاس است کلاسها شامل اطلاعات و رفتاری هستند که بر روی اطلاعات عمل می‌نمایند. کلاس حساب (account) شامل PIN مشتری و رفتاری که PIN را کنترل می‌کند .در نمودار کلاس برای هر نوع آبجکتی در نمودار collaboration , sequence یک کلاس ایجاد شده است.



این نمودار مربوط به برداشت پول از حساب متعلق به سیستم ATM است و با چهار کلاس شامل card reader (کارت خوان)، Account (حساب)، Atmscreen صفحه نمایش cash Dispenser , ATM (صندوق) است. بخشهای مختلف یک کلاس در این مثال شامل:

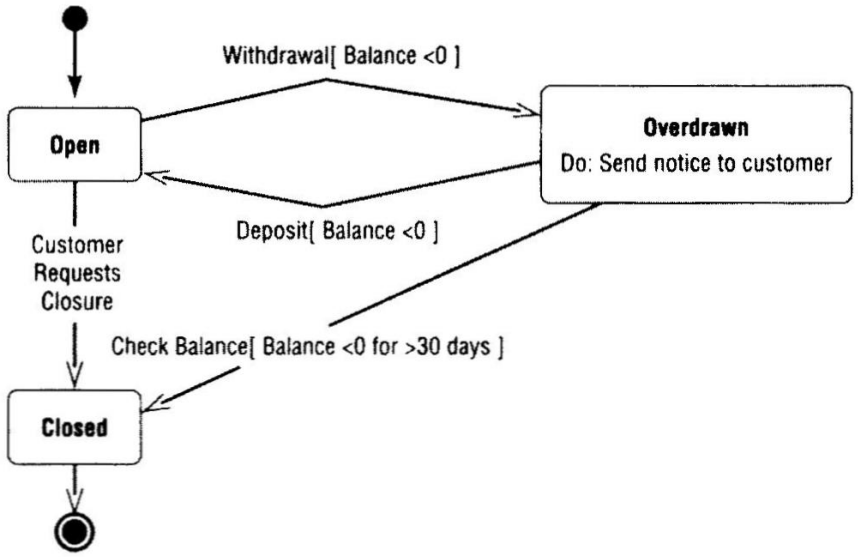
نام کلاس، صفات کلاس و عملگرهای کلاس است. همچنین در صفت ها عملگرها، قفلهای کوچکی در سمت چپشان دارند که فقط می‌توانند از طریق کلاسی که شامل آنهاست قابل دستیابی می‌باشند.

برنامه نویسان از نمودارهای class استفاده می‌کنند تا که کلاسها را به طور واقعی تولید نمایند. ابزارهایی مانند Rose چهار چوب کلاسها را تولید می‌کنند، سپس برنامه نویسان جزئیات را در زبان انتخابی خود نشان می‌دهند. تحلیلگران از نمودارهای کلاس استفاده می‌کند تا جزئیات سیستم را نشان می‌دهند. همچنین طراحان به نمودارهای class نگاه می‌کنند تا طرح سیستم را ببیند. اگر یک کلاس شامل چند تابع باشد، یک معمار می‌تواند این را در نمودار class دیده و توابع را به چند کلاس بشکند. نباید هیچ وابستگی بین کلاسهایی که با یکدیگر ارتباط دارند وجود داشته باشد. یک طراح یا برنامه نویس نیز می‌تواند این را ببیند. نمودارهای کلاس برای این ایجاد شده اند تا کلاسهایی را نشان دهنده که باهم در هر use case کار می‌کنند و نمودارهای جامع (camper hensive) شامل کل سیستم با زیر سیستم را می‌توان به همین ترتیب ایجاد نمود.

نمودارهای حالت (state transition digrmm)

نمودارهای حالت راهی را آماده می‌کنند تا حالتهای مختلف یک آبجکت را مدل کنند. در حالی که نمودارهای کلاس یک تصویر ثابت از کلاسها و وابستگی آنها را نشان می‌دهند، نمودارهای حالت استفاده می‌شوند تا بیشتر رفتارهای پویای یک و نیم را نمایش دهند. یک نمودار حالت رفتار یک آبجکت را نشان می‌دهد. مثلا یک حساب بانکی می‌تواند به چنین حالت متفاوت وجود داشته باشد. می‌تواند باز باشد، بسته شود یا به طور اضافی (بیشتر از

موجودی) از حساب برداشته شود یک حساب ممکن است در هر یک از این حالتها، به طور متفاوتی رفتار کند از نمودارهای حالت برای نشان دادن این اطلاعات استفاده می‌شود.



مثلا وقتی که حساب باز است و مشتری درخواست بستن حساب را می‌کند، حساب به حالت بسته منتقل می‌شود. در خواست مشتری Event (رخداد) نامیده می‌شود. اگر حساب باز است و مشتری برداشت از حساب را انتخاب می‌کند، حساب ممکن است به حالت برداشت بروز این فقط زمانی رخ می‌دهد که تراز (موجودی) حساب کمتر در صفر باشد که با علامت >تراز نشان داده شده است یک شرط که در براکت محصور شده است شرط حفاظتی Guard condition نامیده می‌شود و وقوع یک انتقال اینکه بتواند یا نتواند اتفاق بیفتد را کنترل می‌کند.

در حالت ویژه start state (حالت شروع) و stop state (حالت پایان) وجود دارد حالت شروع با دایره توپر سیاه نشان داده شده و نشان می‌دهد چه حالتی از آبجکت در ابتدا ایجاد شده است حالت پایانی بوسیله یک خال هدف نمایش داده شده است و نشان می‌دهد

که آبجکت درست قبل از اینکه از بین برود، در چه حالتی می‌باشد. بر روی یک نمودار حالت، فقط یک حالت شروع وجود دارد در حالی که شما می‌توانید حالت پایانی نداشته باشید یا اینکه هر چند حالت پایانی که نیاز دارید ؟ را داشته باشید.

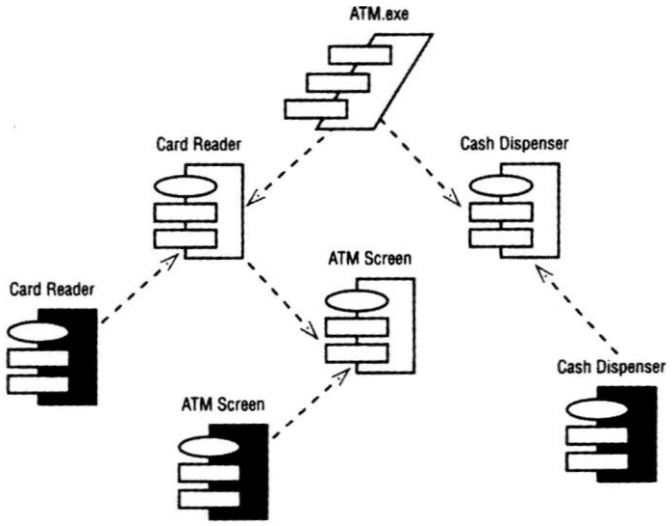
نمودارهای حالت فقط برای مستند سازی ایجاد شده اند. همچنین نمودارهای حالت برای هر کلاسی ایجاد نمی شوند و فقط برای کلاسهای پیچیده استفاده می‌شوند.

نمودارهای اجزاء (component Diagrams)

نمودارهای compnent یک دید فیزیکی از مدلتان را به شما نشان می‌دهند. یک نمودار component اجزای نرم افزاری سیستم شما و روابط بین آنها را به شما نشان می‌دهد دو نوع companent در نمودار وجود دارد.

Component های قابل اجرا و کتابخانه های کد.

در Rose، هر یک از کلاسهای موجود در مدل به یک component کد منبع نگاشت شده اند. اولین باری که Companent ها ایجاد می‌شوند آنها به نمودار component اضافه می‌شوند. سپس وابستگی های میان component ها کشیده می‌شود. وابستگی های co,ponent، وابستگی های زمان اجرا و زمان ت؟ میان component ها را نشان ی می‌دهد.



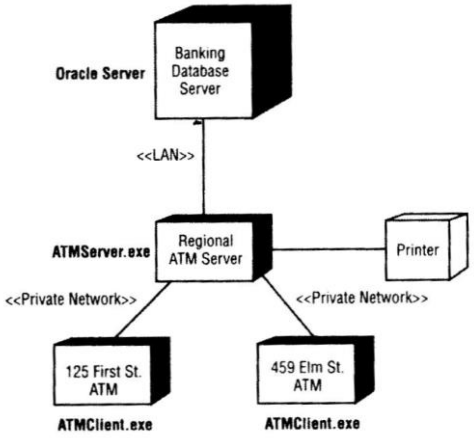
این نمودار component برای سرویس گیرنده ATM/ client است.

Component هایی که به هم وصل شده اند، با خطر چین، وابستگی روابط بین آنها را نشان می‌دهد. مثلا کلاس card Reader به کلاسATM screen وابسته است به این معنی که کلاس ATM screen باید موجود باشد تا کلاس card Reader ترجمه شود. فایل اجرایی ATM client . exe اولین باری که همه کلاسها ترجمه شده اند می‌تواند ایجاد شده باشد مثال ATM دو نخ پردازش دارد، بنابراین به دو صورت قابل اجرا است. یک مجموعه اجرایی ATM client شامل ATM screen , card Reader, cash Dispenser می‌باشد. دومین مجموعه اجرایی ATM screen شامل component حساب است. یک سیستم شبیه به تعداد زیر سیستم ها با قابلیت اجرایی می‌تواند چندین نمودار component داشته باشد. به طور عمومی بسته ها مجموعه‌هایی از آبجکت ها هستند. در این مورد، بسته ها مجموعه ای از component ها می‌باشند ATM شامل دو بسته است ATM screen , ATM client.

نمودارهای component به وسیله هر شخصی که مسئول تنظیم و تدوین سیستم است، استفاده می‌شود. نمودارها این ویژگی را بیان می‌نمایند که به چه منظوری نیاز به کامپایل component وجود دارد. همچنین نمودار نشان خواهد داد که چه component هایی در زمان اجرا به عنوان نتیجه کامپایل ایجاد خواهند شد. نمودارهای component، نگاشته شدن کلاسها به اجزای اجرا شده را نشان می‌دهد. این نمودارها در جایی که تولید تمام شده است رسم می‌شوند.

نمودارهای Deployment

این نمودار ها، لایه فیزیکی شبکه و جایی که Deployment های مختلف مقیم می‌شوند را نشان می‌دهد.



در مثال ATM, ATM از بسیاری زیر سیستم های در حال اجرا بر روی وسایل فیزیکی مجزا یا گره ها تشکیل شده است نمودار Layout, Deployment سیستم را به ما بیشتر نشان می‌دهد. سرویس گیرنده قابل اجرای ATM، بر روی چندین ATM که بر روی محلهای مختلف ایجاد شده اند، اجرا خواهد شد. سرویس گیرنده ATM بر روی یک شبکه

خصوصی، با سرویس دهنده ATM اصلی ارتباط برقرار خواهد کرد. سرویس دهنده ATM قابل اجرا بر روی سرویس دهنده ATM اصلی، اجرا خواهد شد. سرویس دهنده ATM اصلی، بر روی شبکه محلی با سرویس دهنده پایگاه داده بانکداری که proje را اجرا می‌کند ارتباط برقرار خواهد کرد. سرانجام، یک چاپگر به سرویس دهنده ATM اصلی وصل شده است این نمودار به ما نصب فیزیکی سیستم را نشان می‌دهد سیستم ATM ما یک سبک معماری سه طبقه دارند به همراه یک طبقه پایگاه داده، سرویس دهنده اصلی و سرویس گیرنده.

مدلسازی بصری و پردازش تولید و توسعه نرم افزار

تولید نرم افزار می‌تواند به چندین روش انجام شود. چندین نوع متفاوت از پردازشهای تولید شامل هر چیزی از پردازشهای آبشاری گرفته تا شی گرایی وجود دارد که پروژه ها آنها را دنبال می‌کنند و هر یک مزایا و امتیازات خودش را دارد.

در پردازش شی گرا، ما در سراسر مراحل تجزیه و تحلیل، طراحی، تولید، تست و تولید درمقاطع کوچک، بارها حرکت خواهیم کرد. این غیر ممکن است که همه درخواستها را در طول بخش نخست پروژه بفهمیم چیزهای جدیدی ظاهر می‌شوند، بنابراین با طراحی پروژه به صورت تکراری آنها را برنامه ریزی می‌کنیم این مفهوم، یک پروژه می‌تواند به عنوان یک سری از آبشارهای کوچک دیده شود. در پروژه ٤ فاز را پشت سر می‌گذاریم: Inception (انتقال)، Elaboration(مهارت)، constraction(ساختار)، Transitian (انتقال).

Inception شروع پروژه است که ما اطلاعات را جمع آوری کرده، و مفهوم و برداشت کلی را اثبات می‌نماییم پایان Inception تصمیم درباره انجام / عدم انجام پروژه است. در

Elaboration، به طور مفصل sue case توضیح داده شده و تصمیمات معماری گرفته می‌شوند. Elaboration شامل مقداری تجزیه و تحلیل، طراحی، کد نویسی، تست می‌باشد. construction (ساختار) جایی است که سخت عمده کد نویسی انجام شده است. Transition آمادگی و تولید نهایی سیستم برای کاربران است. Inception شناخت)

فاز inception شروع پروژه است و ما کشف می‌کنیم که اشکالات سطح بالای سیستم چه هستند و آنها را مستند سازی می‌کنیم و عامل های سیستم چه کسانی هستند و use case ها را تعیین می‌نماییم ولی وارد جزئیات use case ها نمی شویم بلکه فقط یک یا دو جمله را آماده می‌کنیم. همچنین تخمینی را فراهم می‌کنیم تا مدیریت را پیش ببریم.

Inception زمانی پایان می‌یابد که تحقیقات انجام شده اند و مدیریت، منابع را اختصاص می‌دهد تا بر روی پروژه کار کند. فاز Inception پروژه به طور اساسی دنباله ای غیر تکراری است. حالتهای دیگر چندین بار در طول پروژه تکرار می‌شوند.

بعضی از کارهای Iception شامل مشخص کردن use case ها و عامل ها است. Rose می‌تواند برای مستند سازی این u se caseها و عامل استفاده شده و نمودارهایی را برای نشان دادن ارتباطات آنها ایجاد کند.

Elagboration (مهارت)

فاز مهارت Elaboration پروژه شامل مقداری طراحی، تجزیه و تحلیل و طرح معماری است همراه با طرح تکرار، فاز مهارت برای هر u se caseدر تکرار جاری انجام می‌شود فاز مهارت شامل چندین جنبه از یک پروژه است مانند کد کردن، اثبات مفاهیم (proofs- of – concept)و تولید نمونه های آزمایشی و ایجاد تصمیمات طراحی، از کارهای اصلی فاز

Elaboration تکمیل use case است درخواستها سطح پایین یک use case شامل جریان پردازش در طول use case می‌باشد، چه عاملهایی با use case درخواست شده اند و نمودارهای Interation جریان پردازش را به صورت گرافیکی نشان می‌دهد و کلا هر حالتی که تغییر می‌کند ممکن است در زمان use case اتفاق بیفتد.

درخواستها به شکل use case های کامل و با جزئیات، در یک سند جمع شده اند که یک software Requirement spencification (SRS) (مشخصات درخواست نرم افزار) نامیده شده است. SRS شامل هم جزئیات درخواستهای سیستم می‌باشد.

کارهای دیگری در فاز مهارت (Elaboration) انجام می‌شود مانند اصلاح تخمین های اولیه، بررسی کیفیت SRS و مدل use case و بررسی کردن خطرها فاز مهارت (Elaboration) زمانی تمام شده است که use case کاملا وارد جزئیات شده اند و به وسیله کار بران پذیرفته شده اند، اثبات مفاهیم (proofs- of –concept) کامل شده اند تا شدت خطرها را کاهش دهند نمودارهای کلاس کامل می‌باشند به عبارت دیگر این فاز زماین کامل است که سیستم طراحی شده بازبینی شده و آماده است تا برنامه نویسان آن را تولید نمایند.

Construction(ساختار)

Construction (ساختار) به روند تولید و توسعه نرم افزار بر می‌گردد. مانند Elaboration این فاز برای هر مجموعه از use caseف در یک بار تکرار کامل شده است و کارهای فاز construction شامل مشخص کردن درخواستهای ثابت، تولید نرم افزار و تست نرم افزاری می‌باشد. از آنجایی که نرم افزار در طول فاز Elaboration به طور کامل

طراحی شده است، construction نباید درگیر تصمیم های طراحی زیادی باشد این به گروه پروژه کمک می‌کند تا تولید موازی را به انجام برسانند یعنی چند برنامه نویس بتوانند بر روی آبجکت های مختلف نرم افزاری کار کنند و بدانند که کل سیستم باهم جمع خواهند شد.

مزیت دیگر مدل کردن سیستم آن است که Rational Rose می‌تواند کد اولیه را برای سیستم تولید کند به منظور استفاده از این شکل، شما نیاز دارید تا componet ها و یک نمودار component را به عنوان یک بخش اولیه constraetion ایجاد کنید. construction جایی است که اکثر کد نویسی پروژه انجام شده است. از Rose برای ایجاد component بر حسب طول آبجکت، استفاده شده است. نمودارهای component ایجاد شده اند تا وابستگی های زمان کامپایل را میان component ها نشان دهند بعد از اینکه زبانها برای هر component انتخاب شدند، تولید کد اولیه می‌تواند انجام شود.

Transtion (انتقال)

فازTransition زمانی است که محصول نرم افزاری کامل شده، به سمت اجتماع کاربر بر می‌گردد کارها در این فاز شامل کامل کردن محصول نرم افزاری نهایی، تکمیل تست تایید نهایی، کامل کردن مستند سازی کاربرد فراهم کردن آموزش برای کاربرد می‌باشد. باید مشخصات درخواستهای نرم افزار (software requirement specification)، نمودارهای use case , class , component , Deployment بروز رسانی شده باشند تا تغییرات نهایی را منعکس نمایند. مهم است که این مدلها با محصول نرم افزاری همزمان شده باشند

زیرا مدلهایی که یکبار در محصول نرم افزاری استفاده خواهند شد به مد پشتیبانی می‌روند. چند ماه بعد از اتمام پروژه، این مدلها در کمک به ارتقا نرم افزار، گرانبها تر خواهند بود.

Use case شامل تمام آن چیزهایی است که درون سیستم قرار دارد. عامل شامل تمام آن چیزهایی است که خارج از سیستم قرار دارد.

نمودار use case برخی از use case های موجود در سیستم مورد نظر شما برخی از عامل های موجود در سیستم شما و رابطه های بین تمامی اینها را مشخص می کند. Use case عملیات سطح بالایی است که سیستم مهیا می کند عامل هر چیز و یا هر کسی است که بر سیستمی که در حال ساخت است اثر می گذارد.

یکی از مزیت های بزرگ نمودارهای Use case تبادل اطلاعات است. مراجعه کنندگان شما می توانند به این نمودارها نگاه کرده و اطلاعات وسیعی را بدست آورند. با نگاه به نمودار Use case خواهند فهمید که چه عملیاتی در سیستم انجام می شود. با نگاه به عامل ها خواهند فهمید که چه کسی بر سیستم کنش دارند. با نگاه به مجموعه Use case و عامل می فهمند که چه محدوده ای از پروژه انجام خواهد شد. بنابر این کمکی به آنها خواهد بود تا از هر عملیات از قلم افتاده ای یک ذهنیت اولیه داشته باشند.

یک نمودار سطح بالا که در main, rational rose نامیده می شود. فقط بسته های نرم افزاری یا گروه بندی Use case ها را نشان می دهد.

نمودارهای Use case کار مشخصی را برای مستند سازی عامل ها (هر چیز خارج از محدوده سیستم) Use case (هرچیز درون محدوده سیستم و ارتباط آنها انجام می دهد.

نکاتی را که باید به عنوان کسی که یک نمودار Use case را ایجاد می نماید به خاطر داشته باشید بدین ترتیب می باشند.

- ارتباطات عامل با عامل را مدل سازی نکنید.
- هیچ گاه مستقیما با فلش، Use case را به هم وصل نکنید (بجز در ارتباطات extends or uses
- هر Use case باید توسط یک عامل آغاز به کار کند.
- بانک اطلاعاتی را به عنوان لایه زیرین تکمیل نمودار Use case در نظر بگیرید.

کار با Use case ها

Use caseبخش سطح بالایی از عملیاتی است که سیستم مهیا می کند به عبارت دیگر Use case، اینکه شخص چگونه سیستم استفاده می کند را شرح می دهد. یک ماشین ATM یک سری عملیات اصلی را برای مشتری انجام می دهد. به مشتری اجازه می دهد تا پول به حساب بریزد نقدا از حساب برداشت کند پول را از یک حساب به حساب دیگر منتقل نماید مقدار موجودی را مشاهده کند، pin را تعویض نماید و یا توسط کارت اعتباری پول پرداخت نماید. هر کدام از این transaction ها روش متفاوت استفاده مشتری از سیستم می باشد. به هر حال هر کدام از آنها یک Use case متفاوت هستند در uml یک Use case با استفاده از عملیات زیر نمایش داده می شود.ژ:

یک مزیت نگاه به سیستم با استفاده از Use case این است که می توان پیاده سازی سیستم را از دلیل ایجاد سیستم در ابتدا جدا نمود.

Use case ها به صورت دیگری به متدهای سنتی نزدیک می شوند. شکستن پروژه به Use case ها یک روش نگاه کردن به پروژه به صورت پردازش گر ا است و نه به صورت عملگرا. البته با تجزیه عملیاتی که گاهی اوقات انجام می شود تفاوت دارد. تجزیه عملیاتی بر اینکه چگونه باید مشکلات سیستم را برای حل شدن به قطعات کوچک و کوچکتر تبدیل کرد تمرکز دارد در حالی که Use case تمرکز کار را بر روی آنچه مشتری از سیستم توقع دارد قرار می دهد.

Use case ها مستقل از پیاده سازی هستند و یک دید سطح بالا از آنچه کاربر از سیستم انتظار دارد می باشند بیایید هر بخش از این تعریف را جداگانه در نظر بگیریم.

اولا Use case به طور مستقل عمل می کنند.

دوما Use case ها یک دید سطح بالا از سیستم هستند.

نباید انقدر زیاد Use case باشید که مشتری به زحمت بتواند سند را بررسی کند فقط در این حد باشد که بفهمد سیستم چه کاری انجم می دهد.

نهایتا تمرکز Use case باید بر آنچه که کاربر از سیستم به دست می اورد باشد.

مشاهده شرکت کنندگان در یک Use case

ممکن است بخواهید لیستی از تمام کلاس ها و عملیات ک هدر Use case شرکت می کنند را داشته باشید. در حالی که پروژه در حال پیشروی است و شما نیازهایی را تغییر و یا اضافه می کنید اینکه بدانید چه کلاس هایی ممکن است تحت تاثیر این تغییرات قرار گیرند کمک زیادی به شما خواهد نمود.

ساختن Use case های Abstract (مجرد)

یک Abstract Use case یک Use case است که مستقیما توسط یک عامل شروع به کار نمی کند. درعوض برخی عملیات اضافی که می تواند توسط دیگر Use case ها استفاده شود را مهیا می کند. Use case های abstract، Use case هایی هستند که در ارتباطات گسترده و مورد استفاده شرکت می کنند.

مشاهده رابطه های متعلق به یک Use case

برگه relation در پنجره Use case specification تمام رابطه هایی که Use case در آنها مشارکت دارد و یا ارتباط با دیگر Use case و یا عامل ها را لیست می کند.

هر کس یا هر چیزی که با سیستم موجود برهم کنش دارد عامل actor نامیده می شود. Use case ها هر چیز موجود در محدوده سیستم را توصیف می کنند در حالی که عامل ها در خارج از محدوده سیستم قرار دارند در UML عامل ها با آدمک هایی نشان داده می شوند.

سه نوع اصلی از عامل ها وجود دارند کاربران سیستم ، سیستم های دیگری که با سیستم موجود در ارتباط هستند و زمان.

اولین نوع عامل یک انسان فیزیکی و یا به عبارت دیگر کاربر است اینها بیشترین عامل مورد استفاده هستند و تقریبا در تمام سیستم ها وجود دارند.

دومین نوع عامل سیستم دیگر است به طور مثال ممکن است بگویید که بانک ما دارای یک سیستم اعتباری است که برای پشتیبانی از اطلاعات اعتبار حساب هر مشتری استفاده می شود.

سومین نوع عامل که زیاد استفاده می شود زمان است هنگامی زمان تبدیل به یک عامل می شود که زمان در حال گذر باعث ایجاد رخدادی در سیستم گردد.

افزودن عامل ها

دو راه برای افزودن یک عامل وجود دارد: یا آن را به یک نمودار Use case باز شده بیفزایید و یا این کار را مستقیما در مرورگر انجام دهید. در حالت دوم عامل موجود در مرورگر می تواند به یک یا تعداد بیشتری نمودار Use case افزوده شود.

یک عامل abstract عاملی است که هیچ مصداق واقعی ندارد به عبارت دیگر کاردینالیتی عامل، دقیقا صفر است به طور مثال ممکن است چندین عامل داشته باشید: کارمند ساعتی، کارمند ثابت و کارمند موقتی. تمامی اینها نوعی از عامل چهارم هستند که عامل کارمند می باشد. با وجود این هیچ کس در شرکت فقط یک کارمند نیست. هر کسی یا کارمند ساعتی است یا کارمند ثابت است و یا کارمند موقتی. دلیل وجود عاملی با نام کارمند این است که رابطه معمول استخدام ساعتی، استخدام با حقوق ثابت و استخدام موقتی نشان داده شود. هیچ مرحله و مصداق واقعی برای عامل کارمند وجود ندارد پس آن یک عامل abstractخواهد بود.

برگه relations موجود در پنجره actor specification تمام رابطه های عامل های شرکت کننده را لیست می کند. این برگه دارای تمام رابطه هایی است که یک عامل با Use case ها و یا عامل های دیگر دارد لیست شامل نام رابطه و نام Use case یا عامل های مرتبط می باشد.

UML از انواع متعددی از رابطه ها برای Use case ها و عامل ها پشتیبانی می کند. این شامل رابطه های communication رابطه های uses رابطه های extend و رابطه های generalization برای عامل می باشد. رابطه های uses, extend رابطه های بین Use case ها را تعریف می کنند. رابطه های actor generalization رابطه بین عامل ها را تعریف می کند.

به رابطه بین Use case و عامل، رابطه communication می گویند. در UML رابطه های اطلاعاتی با استفاده از فلش به حالت نمودار در می آیند:

رابطه uses به یک Use case اجازه استفاده از عملیات مهیا شده توسط یک Use case دیگر را می دهد. رابطه های Use case برای مدل سازی برخی عملیاتی که بین دو یا تعداد بیشتری Use case استفاده می شوند، به کار می روند.

رابطه های extend

یک رابطه extend به یک Use case اجازه می دهد که بطور دلخواه عملیات مهیا شده توسط دیگر Use case ها را بسط دهد که بسیار مشابه رابطه uses عمل می کند. در هر دو نوع این رابطه ها برخی عملیات معمول را در Use case های مجزای خودشان قرار می دهید.

Actor generalization برای نشان دادن همانندی چندین عامل به کار می رود.

در حین ساخت نمودارهای خود افزودن یادداشت هایی به Use case و یا عامل ها کمک زیادی به شما خواهد کرد.

دو نوع یادداشت توضیحی برای افزودن وجود دارد،یادداشت و کادر متن.

در uml آیتم هایی چون عامل ها، Use case ها کلاسها، component ها می توانند به صورت بسته هایی نرم افزاری گروه بندی شده تا سازماندهی شوند. ممکن است هنگام مشاهده Use case بخواهید Use case ها و عامل ها را به صورت بسته بندی شده گروه بندی نمایید.

نمودارهای interaction

یک نمودار interaction روندی در یک Use case را مرحله به مرحله نشان می دهد.

دو نوع نمودار interaction وجود دارند که آنها را بررسی خواهید نمود:

نمودارهای sequence و نمودارهای collaboration.

هر دو نمودار sequence, collaboration اطلاعات یکسانی را نشان خواهند داد با وجود این چند تفاوت کوچک بین نمودارهای بالا وجود دارد. نمودارهای sequence نشان دهنده مرکز کنترل هستند نمودارهای collaboration نشان دهنده یک روند داده ای هستند.

آبجکت آن چیزی است که اطلاعات و روشها را در خود کپسوله می کند. روشی است که برخی چیزهای عینی در دنیای واقعی را نشان می دهد. مثالهایی از آبجکت به صورت زیر می باشد:

- حساب joe
- خانه ای در ۷۶۳۸ main street
- گل زردی که در بیرون از پنجریه خانه منحنی قرار دارد.

بخش های اطلاعاتی که توسط آبجکت نگهداری می شوند، صفات attribute آن می باشند. با وجود اینکه مقادیر صفات برای آبجکت ها تغییر خواهند کرد خود صفات هرگز تغییر نخواهند کرد.

رفتارهای یک آبجکت به عنوان عملیات آن شناخته می شوند. در این مثال عملیات برای حساب joe شامل این موارد است : تنظیم موجودی حساب برای برداشت و یا واریز پول و بررسی اینکه آیا از حساب قرار است بیشتر از موجودی پول برداشته شود یا خیر.

در rose آبجکت ها به نمودارهای interaction افزوده می شوند. هنگام کشیدن یک عامل یا کشیدن دیگر کلاس ها به نمودار interaction یک نمونه ابجکت از آن کلاس به طور خودکار ساخته می شود در rose حذف یک آبجکت از یک نمودار کلاس را از کل مدل حذف خواهد نمود.

طرح کلی برای یک آبجکت را کلاس آن فراهم می کند. به عبارت دیگر یک کلاس تعیین کننده اطلاعاتی است که یک آبجکت می تواند نگهداری کند و نشان دهنده رفتارهایی است که می تواند داشته باشد.

یک راه برای یافتن برخی آبجکت ها این است که نام ها را در جریان رخدادها در نظر بگیرید. یک جای خوب دیگر برای بدست آوردن آنها سناریوی اسناد می باشند. یک سناریو حالت خاصی از جریان رخدادها می باشد. جریان رخدادها برای Use case مربوط به برداشت پول از حساب درباره فردی در ATM صحبت می کند که در حال برداشت پول از حساب است. یکی از سناریوها برای این مورد می تواند برداشت joe از حساب به مقدار ۲۰ دلار باشد سناریوی دیگر می تاند سعی jane در برداشت ۲۰ دلار از حساب باشد در حالی که او pin را اشتباه وارد کرده است.

یک نمودار sequence و collaboration یکی از این سناریوها را شرح می دهد.

هنگامی که در سناریوی خود به اسامی نگاه می کنید برخی از اسامی عامل خواهند بود برخی از آنها آبجکت خواهند بود و برخی صفات برای یک آبجکت خواهند بود.

همه آبجکت ها در جریان رخدادها وجود نخواهند داشت. به طور مثال form ها ممکن است در روند رویدادها ظاهر نشوند، ولی باید بر نمودار ظاهر شوند تا به عامل اجازه دهد که اطلاعات را وارد کرده و یا ببیند. آبجکت های دیگری که در جریان رخداد ها ظاهر نمی شوند. آبجکت های کنترل هستند اینها آبجکت هایی هستند که تناوب روند در Use case را کنترل می کنند.

نمودارهای collaboration زمانی مفید واقع می شوند که بخواهید به تاثیر تغییرات دست یابید. اینکه بفهمید چه آبجکت هایی با چه آبجکت های دیگری تبادل اطلاعاتی انجام می دهند. به راحتی با نگاه به نمودارهای collaboration قابل انجام است. اگر نیاز دارید که یک آبجکت را تغییر دهید می توانید به راحتی ببینید که چه آبجکتهای دیگری ممکن است در ارتباط با آن باشند.

نمودارهای sequence موارد زیر را در بر می گیرند:

Objects: یک نمودار interaction می تواند از نام ابجکت ها نام کلاس ها و یا هر دوی آنها استفاده کند.

Messages: با استفاده از یک پیغام یک آبجکت یا کلاس می تواند از یک آبجکت یا کلاس دیگر،

در هنگام ساختن نمودار sequence باید به این نکته توجه داشته باشید که در حال تخصیص مسئولیت به ابجکت ها می باشید. وقتی پیغامی را به یک نمودار interaction می افزایید، در حقیقت به ابجکت در حال دریافت پیغام یک مسئولیت را واگذار می کنید.

نمودارهای sequence نمودارهای interaction هستند که بر مبنای زمان تنظیم می شوند. شما نمودار ار از بالا به پایین مشاهده می کنید.

هر ابجکت برای خودش یک خط عمر دارد که به صورت خطوط عمومی خط چین در زیر آبجکت کشیده می شود یک پیام بین دو خط عمر موجود بین دو آبجکت قرار داده می شود

تا ارتباط بین آبجکت ها را نشان دهد. هر پیغامی نشان دهنده یک آبجکت است که توسط تابع ابجکت دیگر صدا زده می شود.

برای الصاق فایل به نمودار sequence:

۱- در مرورگر بر روی نمودار sequence کلیک راست کنید.

۲- از منوی new گزینه file را انتخاب کنید.

۳- با استفاده از کادر محاوره ای open فایلی را که می خواهید الصاق نمایید انتخاب کنید.

٤- Open را انتخاب کنید تا فایل را الصاق نمایید.

نمودار collaboration برای نشان دادن جریان در سناریوی مشخص یک Use case استفاده می شوند. نمودارهای sequence برحسب زمان منظم می شوند، نمودارهای collaboration بیشتر بر روی رابطه بین آبجکت ها متمرکز می شوند. هر نمودار sequence, collaboration باید دارای آبجکت عامل باشد. آبجکت عامل یک محرک خارجی است که به سیستم اعلام می کند تا یک عملیات را راه اندازی کند. آبجکت های عامل برای نمودار interaction عامل هایی که در نمودار Use case یا use Case ارتباط دارند را نشان می دهند.

نگاشت یک آبجکت به یک کلاس

در یک نمودار sequence یا نمودار collaboration هر آبجکتی که ممکن است به یک کلاس شود. به طور مثال حساب joe ممکن است به یک کلاس به نام account

نگاشت شود. در پنجره object specification می توانید از فیلد class برای تنظیم کلاس آبجکت استفاده کنید. به طور پیش فرض کلاس به unspecified تنظیم شده است.

یک پیغام برقرار کننده ارتباط بین آبجکت ها است که در آن آبجکت از آبجکت دیگر تقاضای انجام کار می کند. زمانی که در حال کدنویسی هستید یکه پیغام می تواند به یک فراخوانی تابع تبدیل شود.

در یک نمودار Sequence می توانید به طور دلخواه مرکز کنترل را نشان دهید. که به شما نشان می دهد کدام آبجکت در یک زمان مشخص کنترل را در دست گرفته است. این یکی از تفاوت هایی بین نمودار های collaboration یا sequence است مرکز کنترل فقط در نمودارهای sequence نشان داده می شود.

نمودارهای collaboration جریان داده ای را نشان می دهند در صورتی که نمودارهای sequence چنین کاری نمی کنند.

جریان های داده ای برای نشان دادن اطلاعات برگشتی، وقتی که آبجکتی به آبجکت دیگر پیغام ارسال می کند استفاده می شوند. عموما به هر پیغام موجود بر روی نمودار collaboration جریان داده ای اضافه نکنید زیرا باعث ازدحام و شلوغی نمودار می شود، آن هم به وسیله یک سری اطلاعاتی که ارزش زیادی ندارند.

نام پیغام و نام آبجکت اطلاعات زیادی را در نمودار interaction در اختیار شما قرار می دهند. به هر حال ممکن است در زمان هایی بخواهید اطلاعات مضاعفی را به نمودار بیفزایید. می توانید این کار را با استفاده از یادداشت و یا اسکریپت انجام دهید.

یادداشت ها برای متصل کردن تفاسیر و توضیحات به آبجکت روی نمودار استفاده می شوند. به طور مثال برای روشن شدن هدف یک ابجکت استفاده می شوند.

اسکریپت ها برای افزودن توضیح به پیغام اضافه می شوید همان گونه که می خواهید در نمودار sequence می توانید از اسکریپت برای افزودن شرایط منطقی استفاده کنید.

در Rose یادداشت ها معمولا برای افزودن توضیح به آبجکت استفاده می شوند. از طرف دیگر اسکریپت ها معمولا برای افزودن توضیح پیغام استفاده می شوند. اسکریپت ها فقط در نمودار sequence استفاده می شوند. آنها در سمت چپ نمودار و رو به روی پیغامی که به آن ارجاع می شوند ظاهر می شوند. می توانید از یک اسکریپت برای نشان دادن معنی یک پیغام استفاده کنید.

نمودارهای interaction آبجکت نشان می دهد چگونه آبجکت ها برای پیاده سازی عملکرد یک use case با یکدیگر کار می کنند. دو نوع از نمودارهای interaction وجود دارند: نمودارهای sequence و collaboration. هر دو نوع این نمودارها اطلاعات یکسان ولی با زوایای متفاوتی را نشان می دهند.

نمودارهای sequence اطلاعات را به ترتیب زمانی نشان می دهند. نمودار sequence برای مسیرهای متناوب به یک use case ساخته شده اند. آنها برای مشاهده پیشرفت عملیات یک use case مفید می باشند. نمودارهای collaboration روند اطلاعات را نشان می دهند ولی در اینجا ترتیب زمانی در نظر گرفته نشده است. نمودارهای collaboration رابطه بین آبجکت ها و پیغام های بین آبجکت ها را شرح می دهد. یک طراح سیستم توسط نمودار sequence می تواند

ببیند که کدام آبجکت ها حساس هستند و کدام آبجکت ها نیاز به برقراری ارتباط مستقیم با یکدیگر دارند. نمودارهای collaboration هم می توانند جریان داده ای را بین آبجکت ها نشان دهند نمودارهای sequence و نمودارهای collaboration قابل تغییر هستند وقتی تغییراتی روی یکی انجام شود آن یکی هم تغییر خواهد کرد.

یک نمودار class برای نمایش تعدادی از کلاس ها و بسته های کلاس در سیستم شما استفاده شده است این نمودار یک تصویر ایستا از قطعات سیستم و ارتباطات بین آنها را به شما می دهد.

به طور پیش فرض یک نمودار class وجود دارد که main نامیده شده و مستقمیا زیر نظر نمای logical است این نمودار class بسته های کلاس های موجود در مدلتان را نشان می دهد. داخل هر بسته ای نمودار دیگری است که main نامیده می شود. که شامل همه کلاس های داخل آن بسته است در rose با دوبار کلیک بر روی یک بسته در یک نمودار class بطور خودکار نمودار main class باز خواهد شد.

نمودارهایclass یک ابزار طراحی خوب برای تیم می باشند. آنها به برنامه نویسان کمک می کنند تا ساختار سیستم را قبل از اینکه کدی نوشته شود ببینند و طراحی کنند و کمک می کنند تا مطمئن شوند که سیستم از ابتدا خوب طراحی شده است.

در بالاترین بخش کلاس نام کلاس قرار دارد و به طور اختیاری stereotype آن را نگه می دارد. بخش میانی صفات یا اطلاعاتی که یک کلاس دارد را نگهداری می کند بخش پایین صفات و یا عملگرهای یک کلاس را نگه می دارد تا نمودارهای شما را توضیح دهد.

همچنین می توانید visibility هر صفت و عملگر نوع داده ای هر صفت و علامت مشخصه هر عملگر روی این نمودارها را نشان دهید.

یک آبجکت نمونه ای از یک کلاس است.

یک محل خوب برای پیدا کردن کلاس ها جریان رخدادهای use case شماست. نگاه به آسامی در جریان رخدادها به شما اجازه خواهد داد تا بدانید چه تعدادی از کلاس ها وجود دارند. وقتی به اسامی نگها می کنید. آنها یکی از چهار چیز خواهند بود:

- یک عامل
- یک کلاس
- یک صفت از یک کلاس
- یک اصطلاح که یک عامل کلاس یا صفت نیست.

کار با کلاس

اولین باری که نمودارهای Class خود را ایجاد می‌نمایید، قدم بعدی افزودن کلاس‌ها به مدل است. چند نوع کلاس وجود دارد که شما می‌توانید اضافه کنید. کلاسهای معمولی، (Rwgular) کلاس‌ها پارامتری شده (Parameterized)، کلاس‌هایی که به عنوان نمونه آورده شده (instantiated)،کلاسهای Utility، کلاسهای Utility پارامتری شده، کلاسهای Utility که به عنوان نمونه آورده شده، و کلاسهای متا (meta).

نسبت دادن یک Stereotype به کلاس

یک Stereotype مکانیسمی است که شما می‌توانید با استفاده از آن، کلاسهایتان را طبقه‌بندی کنید. در UML سه نوع Stereotype اساسی برای کلاس وجود دارد :

Control .Entity , Boundary

کار با بسته‌ها

یک بسته برای گروه‌بندی کلاسهایی استفاده می‌شود که توضیحات یکسانی دارند. در هنگام بسته‌بندی کلاس‌ها، روشهای مشترکی وجود دارد، اما شما می‌توانیدهر جوری که دوست دارید کلاس‌ها را گروه‌بندی کنید. یک روش این است که کلاس‌ها را به وسیله Stereotype گروه‌بندی کنید. با این روش، شما یک بسته با کلاس های entity یک با کلاس های Boundary و یکی با کلاس های control و غیره دارید. این می تواند روش مفید برای برنامه نویسان باشد. همه کلاس های Boundary که روش ماشین سرویس گیرنده می روند در حال حاضر با هم بسته بندی شده اند. یک روش مفید دیگر برای گروه بندی کلاس ها بر اساس کارایی می باشد. مثلا ممکن است شما بسته ای داشته باشید که security نامیده می شود. که همه کلاس هایی را نگه می دارد که با امنیت برنامه ارتباط دارد.

اگر شما با دقت کلاس هایتان را گروه بندی کنید به بسته هایی رسیده اید که نسبتا وابسته به یکدیگر هستند.

بسته ها می توانند در داخل بسته های دیگر جا بگیرند تا بیشتر کلاس های شما را سازماندهی کنید. در بالاترین سطح ممکن است کلاس های خود را بر اساس کارایی گروه

بندی کنید تا بسته security شما را بسازید. از طریق این بسته شما می توانید بسته های دیگری داشته باشید کلاس های امنیتی را بر اساس کارایی یا stereotype گروه بندی کنید.

یک صفت قطعه اطلاعاتی مرتبط با یک کلاس است به طور مثال کلاس company ممکن است صفاتی به این شکل داشته باشد نام آدرس، تعداد کارکنان.

یک جای مناسب برای یافتن صفات نگاه کردن به سند نیازمندیها می باشد.

آخرین راه بررسی ساختار بانک اطلاعاتی است اگر بانک اطلاعاتی شما به طور کامل تعریف شده است. فیلدهای موجود در جداول درباره اینکه چه صفاتی خواهید داشت ایده خوبی به شما می دهند.

به هر حال وقتی صفات را تعریف می کنید از اینکه هر کدام از آنها راه برگشتی تبدیل به یک نیاز را داشته باشد. مطمئن شوید این موضوع می تواند در حل مشکلات مربوط به قدیمی بودن یک برنامه کاربردی که اطلاعات زیادی که هیچ کس مورد استفاده قرار نمی دهد را تعقیب می کند کمک کند.

هر نیاز باید به یک جریان رخداد متعلق به یک use case به یک نیاز مشخص و یا یک جدول بانک اطلاعاتی موجود برگردانده شود. اگر بتوانید یک نیاز را ردیابی کنید از مورد استفاده بودن آن برای مشتری مطمئن نخواهید شد./ این موضوع ممکن است باعث کمی انحراف از برخی روشهای قدیمی تر شود. اینکه ابتدا ساختار بانک اطلاعاتی را ساخته و سپس سیستم را در اطراف آن قرار دهید سیستم و بانک اطلاعاتی را در یک زمان و در حالی که هر دوی آنها را با نیازهای یکسان وفق می دهید می سازید.

در حالی که صفات را تعریف می کنید با احتیاط آنها را به کلاس های مربوطه تخصیص دهید یک صفت باید دارای اطلاعات مربوط به کلاس باشد به طور مثال کلاس کارمند ممکن است دارای نام و ادرس باشد ولی نباید تولیداتی که شرکت مربوط به این کارمند تولید می کند را در برداشته باشد. کلاس تولیدات مکان بهتری برای ذخیره اطلاعات مربوط به تولیدات خواهد بود./

مواظب کلاس هایی که صفات زیادی دارند باشید . اگر کلاسی را بیابید که دارای تعداد زیادی صفات باشد نشان دهنده این خواهد بود که کلاس کوچکتری شکسته شود. اگر کلاس با بیش از ۱۰ یا ۱۵ صفت داشته باشید نگاه دقیقتری به کلاس بیندازید ممکن است کلاس کاملا منطقی و موجه به نظر رسد فقط از اینکه تمامی صفات مورد نیاز باشند و حقیقتا به همان کلاس تعلق داشته باشند اطمینان حاصل کنید. برای کلاس هایی که صفات بسیار کمی دارند نیز به طور مشابه و با احتیاط عمل کنید. ممکن است باز هم کلاس از نظر شما منطقی و موجه باشد به طور مثال ممکن است کلاس های control گرایش بیشتری به صفات کم داشته باشند . همچنین ممکن است نشان دهنده این باشد که یک یا دو کلاس باید با هم ترکیب شوند. اگر کلاسی با یک یا دو صفت داشته باشید ممکن است ارزش این را داشته باشد که یک نگاه دقیقتری به این مساله بیاندازید.

گاهی ممکن است به اطلاعات و داده هایی برسید که ندانید آیا صفات هستند و یا کلاس.

سه بخش اصلی از اطلاعاتی که می توانید به هر صفت منتقل کیند بدین شرحند: نام صفت، نوع داده ای و مقدار اولیه. قبل از اینکه برای مدل خویش کدسازی کنید باید به هر صفت یک نام و یک نوع داده ای عرضه کنید. قراردادن مقادیر اولیه دلخواه و انتخابی است.

تنظیم visibility صفت

یکی از مرکزی ترین و اصلی ترین مفاهیم در برنامه نویسی شی گرا کپسوله سازی است هر کلاس دارای صفات و عملیات مقداری از اطلاعات و مقداری از رفتارها و عملکردها را در خود کپسوله می کند. یکی از مزیتهای این روش داشتن قطعات کد کوچک موجود در خود است. به طور مثال کلاس employee تمام اطلاعات و رفتارهای یک کارمند را در برخواهد داشت.

صفات درون کلاس ها و به صورت مخفی از دیگر کلاس ها قرار دارند به دلیل کپسوله بودن صفات درون یک کلاس باید تعریف کیند که چه کلاسهایی برای مشاهده و تغییر صفتها اجازه دسترسی دارند که به عنوان visibility صفت شناخته می شود.

تنظیم محدودیتهای صفت

محدودیتهای صفت نشان می دهد که صفت چگونه در کلاس ذخیره شده است سه انتخاب برای محدودیت وجود دارد.

BY VALUE توسط مقدار

By refrence توسط مرجع

Unspecified تعریف نشده

هنگامی که صفتی به یک کلاس افزوده می شود هر نمونه از کلاس صفت خود را دریافت خواهد کرد.

یک صفت ایستا صفتی است که توسط همه نمونه های یک کلاس بتواند استفاده شود.

یک صفت مشتق شده صفتی است که از یک یا چند صفت دیگر ساخته شده باشد. به طور مثال کلاس rectangle داریا صفات widthe و height می باشد. همچنین ممکن است صفتی با نام area داشته باشد که با انجام محاسبات بر روی طول و عرض به دست می آید. Area از دو صفت دیگر یعنی width و height مشتق شده است و به عنوان یک صفت مشتق شده در نظر گرفته می شود.

عملیات رفتار و عملکرد مربوط به یک کلاس می باشد. یک عملیات سه قسمت دارد: نام عملیات پارامترهای عملیات و نوع برگشتی عملیات، پارامترها، آرگومان هایی هستند که عملیات آنها را به عنوان ورودی دریافت می کند. نوع برگشتی، خروجی عملیات است.

در یک نمودار class می توانید یا نام عملیات و یا نام عملیات که در ادامه آن پارامترها و نوع برگشتی قرار دارد را ببینید. برای کم کردن بی نظمی و بهم ریختگی در نمودار class داشتن برخی نمودارهای class فقط با نام عملیات و برخی دیگر با علائم عملیاتی کامل شامل پارامترها و نوع داده ای کمک بزرگی خواهد بود.

یافتن عملیاتها

یافتن عملیاتها حقیقتا بسیار آسان است. در حالی که نمودارهای sequence و collaboration را می سازید. بیشترین کارهای مورد لزوم برای یافتن عملیات را انجام داده اید.

چهار نوع متفاوت از عملیاتها وجود دارند:

- عملیاتهای مجری
- عملیاتهای مدیریتی
- عملیاتهای دسترسی
- عملیاتهای امدادرسانی

آرگومان ها و یا پارامترهای عملیات داده های ورودی هستند که عملیات ها دریافت می کنند. به طور مثال عملیات `add` ممکن است دو آرگومان `x` و `y` را گرفته و آنها را با همدیگر جمع بزنید.

دو قطعه از اطلاعات برای عرضه به هر آرگومان وجود دارند اولین آن نام آرگومان است و دومین نوع داده ای است آرگومان ها و انواع داده ای در نمودار `class` درون پرانتزهای بعد از نام عملیات قرار می گیرند.

تعریف `protocol` عملیات

پروتکل عملیات تعریف کننده این است که یک در خواست کننده ممکن است چه عملیاتی را بر روی آبجکت انجام دهد. و عملیات در چه روندی باید اجرا شوند.

تعریف `qualifications` عملیات

این فیلد به شما اجازه تعریف شروط مختص به زبان را برای عملیات می دهد. هر چیزی که در اینجا وارد می کنید به صورت یک توضیح به کد تولید شده اضافه می گردد. به هر حال در کد تولید شده برای عملیات تاثیری ندارد.

تعریف exeptions استثنائات عملیات

فیلد استثنائات عملیات مکانی است که می توانید موارد استثنایی که ممکن است عملیات با آن روبه رو شود را لیست کنید.

تعریف اندازه عملیات size

فیلد اندازه مکانی است که در آنجا معین می کنید این عملیات در زمان اجرا به چه مقدار حافظه نیاز دارد.

تعریف time زمان عملیات

زمان عملیات مقدار زمان تخمینی است که عملیات برای اجرا نیاز دارد. هر اطلاعاتی که در اینجا وارد کنید به صورت یک تفسیر در کد ظاهر می شود.

تعریف concurrency همروندی عملیات

فیلد همروندی تعیین می کند که در صورت همزمانی چند پردازش کنترلی چه رخ خواهد دارد.

تعریف precondition پیش شرایط عملیات

پیش شرایط شرایطی هستند که قبل از اینکه عملیات بتواند اجرا شود باید true باشد.

تعریف postconditions شرایط پسین عملیات

شرایط پسین شرایطی هستند که همیشه باید بعد از خاتمه اجرای عملیات true باشند.

یک رابطه relationship یک ارتباط معنایی بین کلاس ها است که به یک کلاس اجازه می دهد در باره صفتها عملیات و ارتباطات کلاس دیگر چیزهایی بداند. برای اینکه یک کلاس از طریق یک نمودار sequence یا collaboration پیغامی را به کلاس دیگر بفرستد. یک رابطه باید بین آن دو کلاس وجود داشته باشد.

چهار نوع رابطه وجود دارد که می توانید آنها را بین کلاس ها برقرار کنید.

Generalization,. Aggregation, dependency, association.

Associatoion ها رابطه های معنایی بین کلاس ها هستند که در نمودار کلاس به وسیله یک خط ساده کشیده می شوند.

وقتی یک association دو کلاس را به هم وصل می کند هر کلاس می تواند از طریق یک نمودار sequence یا یک نمودار collaboration به کلاس دیگر پیغامهایی را بفرستد.

Association ها می توانند دو طرفه یا یک طرفه باشند. در uml،

association های دو طرفه می توانند هم توسط فلش دو طرفه و هم توسط یک خط ساده کشیده شوند. Association های یک طرفه که دارای یک فلش یک طرفه می باشند مسیر هدایت را نشان می دهند.

Dependency ها نیز دو کلاس را به هم وصل می کند اما با یک تفاوت کوچک نسبت به association ها. Dependency ها همیشه یک طرفه هستند و نشان می دهند که یک کلاس به تعاریف dependency کلاس دیگر بستگی دارد.

Aggregation ها یک فرم قویتر از association هستند یک aggregation یک رابطه بین یک واحد کل whole و بخشهای parts آن می باشد. برای مثال ممکن است یک کلاس ماشین داشته باشید به علاوهه یک کلاس موتور یک کلاس لاستیک و کلاسهایی برای سایر بخشهای یک ماشین در این مورد یک آبجکت ماشین از یک آبجکت ماشین از یک آبجکت موتور چهار آبجکت لاستیک و غیره تشکیل می شود.

Aggregation ها مانند یک خط با یک لوزی در کنار کلاسی که واحد کل whole را نمایش می دهد نشان داده می شوند.

Generalization برای نمایش یک رابطه وراثتی بین دو کلاس استفاده می شوند اکثر زبانها شی گرا مستقیما از مفهوم وراثت پشتیبانی می کنند. وراثت به یک کلاس امکان می دهد تا تمام صفتها عملکردها و رابطه های کلاس دیگر را به ارث ببرد. در uml یک رابطه وراثتی به عنوان یک generalization شناخته می شود و مانند یک فلش از کلاس فرزند به کلاس والد نشان داده می شود.

یک آبجکت مشخص شده می تواند در یک یا چند حالت وجود داشته باشد برای مثال یک کارمند می تواند استخدام شده باشد اخراج شده باشد، در دوره آزمایشی باشد در مرخصی باشد و یا بازنشته شده باشد. یک آبجکت کارمند در هر کدام از این حالات ممکن است رفتار و عملکرد متفاوتی داشته باشد.

یک نمودار تغیر حالت چرخه زندگی یک آبجکت منفرد را نشان می دهد این نمودارها روش خوبی برای مدل کردن رفتار و عملکرد دینامیکی یک آبجکت می باشند. در یک پروژه

معمولی برای هر کلاس یک نمودار تغییر حالت ایجاد نمی کنید. در واقع خیلی از پروژه ها اصلا از آنها استفاده نمی کنند.

اگر یک کلاس دارید که چند رفتار وعلکرد مهم دینامیکی دارد. ایجاد یک نمودار تغییر حالت برای آن مفید است یک کلاس با رفتارها و عملکردهای دینامیکی قابل ملاحظه کلاسی است که می تواند در حالات زیادی وجود داشته باشد در مثال درس یک آبجکت درس می تواند باز بسته ملغی یا تمام شده باشد.

برای اینکه تعیین کنید که آیا یک کلاس رفتارها و عملکردهای دینامیکی مهم دارد یا خیر، به صفتهای آن توجه کنید. بررسی کنید که چگونه یک نمونه از کلاس ممکن است با مقدارهای متفاوت در یک صفت متفاوت عمل کند اگر یک صفت به نام وضعیت دارید این می تواند نشانه خوبی مبنی بر حالات متنوع باشد یک آبجکت چگونه با مقدارهای متفاوت که به جای صفت قرار گرفته می شوند رفتاری متفاوتی ارائه می دهد.

همچنین می توانید رابطه های یک کلاس را بررسی کنید به دنبال رابطه هایی باشید که multiplicity آنها صفر است. صفرها نشانگر این هستند که رابطه انتخابی است.

آیا یک نمونه از کلاس وقتی رابطه وجود دارد یا ندارد. عملکرد و رفتار متفاوتی دارد؟ اگر تغییر می کند ممکن است چندین حالت داشته باشید برای مثال اجازه دهید به یک رابطه بین یک شخص یا یک شرکت نگاهی داشته باشید. اگر یک رابطه وجود داشته باشد شخص در وضعیت استخدام می باشد. اگر هیچ رابطه ای وجود نداشته باشد شخص ممکن است اخراج یا بازنشسته شده باشد.

این نمودار هابه درد مستندکردن رفتار و عملکرد دینامیکی یک کلاس می خورند تا برنامه نویسها و تحلیل گران یک شناخت و درک واضح و روشن از این رفتار و عملکرد داشته باشند در نهایت برنامه نویسها مسئول به کار گیری منطق ارائه شده در این نمودار هستند.

مانند سایر نمودارهای UML نمودار های تغییر حالت به گروه این امکان را می دهد تا درباره منطق کار قبل از اینکه کد شود بحث و بررسی کرده و آن را مستند سازی کنند.

یک component یک ماژول فیزکی کد است. component ها می تواند هم حاوی کتابخانه کد منبع و هم فایل های زمان اجرا باشند. برای مثال اگر از c++ استفاده می کنید هر کدام از فایلهای H و cpp شما یک component مجزا می باشند. فایل exe که بعد از کامپایل شدن کد ایجاد می شود نیز یک component است.

وقتی component ها ایجاد می شوند به یک نمودار component اضافه می شوند و رابطه هایی بین آنها کشیده می شود. تنها رابطه ایی که می تواند بین component ها وجود داشته باشد یک dependency است. یک dependency بیان می کند که یک component باید قبل از دیگری کامپایل شود.

یک نمودار component یکی از نمودار های uml است که component های موجود در سیستم و رابطه های dependency بین آنها را نمایش می دهد.

با استفاده از این نمودار کارمندان مسئول ترجمه و ایجاد سیستم خواهند دانست که چه کتابخانه های کدی وجود دارند و چه فایل های اجرایی به هنگام کامپایل کد ایجاد خواهند شد. برنامه نویسها خواهند دانست که چه کتابخانه های کدی وجود دارد و چه رابطه هایی بین آنها وجود دارد. رابطه های dependency بین component ها به کسانی که

مسئول کامپایل هستند. امکان می دهد که بدانند . component ها به چه ترتیبی باید کامپایل شوند.

نمای dependency از استقرار فیزیکی برنامه کاربردی متاثر می شود که شامل موضوعاتی از قبیل طراحی و آرایش شبکه و مکان component ها بر روی شبکه می باشد.

نمای deployment شامل پردازنده process ابزارهاdevice فرایندها و ارتباطهای بین پردازنده ها و ابزارها می باشد. همه اینها در یک نمودار deployment نشان داده می شوند. برای هر سیستم فقط یک نمودار deployment برای هر مدل rose وجود دارد.

یک نمودار deployment تمام گره های موجود در شبکه ارتباطهای بین آنها و فرآیندهایی که بر روی هر کدام از آنها اجرا خواهند شد را نشان می دهد.

یک پردازنده هر ماشینی است که قدرت پردازش دارد. مثل سرویس دهنده ها ، ایستگاههای کاری و سایر ماشینهایی که پردازنده ها دارند.

می توانید اطلاعاتی را درباره stereotype ، خصیصه و زمان بندی متعلق به پردازنده اضافه کنید.

Stereotype مانند عملکرد آن با سایر عناصر مدل برای طبقه بندی پردازنده استفاده می شود.

یک خصیصه پردازنده توضیحات فیزیکی پردازنده می باشد برای مثال می تواند شامل سرعت پردازنده یا مقدار حافظه آن باشد. فیلد scheduling نوع زمانبندی فرایند استفاده شده توسط پردازنده را نشان می دهد. انتخاب ها به قرار زیرند:

Preemptive نشان می دهد که فرآینده های با اولویت بالاتر می توانند پیش از فرایندهای با اولویت پایین اجرا شوند.

Non preemptive نشان می دهد که فرآیندهای هیچ اولویتی ندارند فرایند جاری اجرا می شود تا در زمانی که پردازش بعدی آغاز می شود پایان یابد.

Cyclic کنترل حلقه ایی بین فرایندها را نشان می دهد به هر فرایند یک مقدار زمان داده می شود تا اجرا گردد و سپس کنترل برای اجرا به فرایند بعدی داده می شود.

Executive نشان می دهد که گونه هایی از الگو در سیستم های محاسباتی وجود دارد که زمانبدی را کنترل می کنند.

Manual نشان می دهد که فرایندها توسط خود کاربر زمانبندی می شوند.

یک ابزار device یک ماشین یا یک قطعه سخت افزاری بدون قدرت پردزاش می باشد. ابزارها شامل عناصری از قبیل پایانه های نامفهوم چاپگرها و اسکنرها می باشد. در uml ابزارها با این نشانه ظاهر می شوند.

پردازنده ها و ابزارها می توانند به عنوان گره هایی در شبکه در نظر گرفته شوند.

فهرست

کار با Use case ها ۲٤
مشاهده شرکت کنندگان در یک Use case ۲٥
ساختن Use case های Abstract (مجرد) ۲٦
مشاهده رابطه های متعلق به یک Use case ۲٦
افزودن عامل ها ۲۷
رابطه های extend ۲۸
نمودارهای interaction ۲۹
کار با کلاس ۳۷
نسبت دادن یک Stereotype به کلاس ۳۸
کار با بسته‌ها ۳۸
تنظیم visibility صفت ٤۱
تنظیم محدودیتهای صفت ٤۱
یافتن عملیاتها ٤۲
تعریف protocol عملیات ٤۳
تعریف qualifications عملیات ٤۳
تعریف exeptions استثنائات عملیات ٤٤

Rational Rose چیست؟

Rationa Rose یک ابزار قدرتمند است که به تجزیه و تحلیل و طراحی سیستم های نرم افزاری شیء گرا کمک می‌کند این برنامه به شما کمک می‌کند تا قبل از اینکه کدی را بنویسید، سیستم خود را مدل نمایید، بنابراین می‌توانید مطمئن شود که سیستم از ابتدا معماری معتبری دارد. Rational Rose با کمک به تجزیه و تحلیل و طراحی سیستم ها شما را قادر می‌سازد تا use case و نمودارهای use case را برای نشان دادن عملیات سیستم، طراحی نمایید به شما اجازه خواهد داد تا نمودارهای Interaction را برای نشان دادن اینکه آبجکت ها چگونه باهم کاری می‌کنند تا عملیات مورد نیاز را فراهم کنند، طراحی نمایید کلاس ها و نمودارهای class ایجاد شده اند تا آبجکت های یک سیستم را نشان دهید و اینکه چگونه آنها با یکدیگر رابطه دارند. نمودارهای component می‌توانند ایجاد شوند تا نشان دهند که چگونه کلاس ها به اجزای اسبابی و ابزاری نگاشت می‌شوند. در نهایت، یک نمودار در Deployment می‌تواند تولید شود تا طراح شبکه ای سیستم را نشان دهد. مدل Rose تصویری در سیستم است. این مدل شامل همه نمودارهای UMLعامل ها، use case، آبجکت ها، کلاس ها، component ها و گره های deployment یک سیستم جریان این مدل با جزئیات بیشتری توضیح می‌دهد که چه سیستمی ایجاد خواهد شد و چگونه کار خواهد کرد، بنابراین برنامه نویسان می‌توانند از مدل به عنوان یک طراح کلی برای ساخت سیستم استفاده کنند. این به کاهش یک مشکل قدیمی کمک می‌کند گروه با مشتریان صحبت کرده اند و درخواستها را مستند سازی نموده اند، حال برنامه نویسان آماده هستند تا کد

نویسی را شروع کنند. به برنامه نویسان درخواستهای یکسانی داده شده که ممکن است سیستم های مختلفی را کد کنند. مشکل زمانی پیش می‌آید که شخصی نیاز دارد تا سیستمی را بفهمند یا نگهداری می‌کند. بدون اجرای مصاحبه های مفصل با هر یک از برنامه نویسان، برای هر شخص مشکل است تا ببیند که چه تصمیمات طراحی اتخاذ شده قطعات سیستم چه هستند، همان سیستمی است که کاربران در ذهن داشته اند مثلا یکی از برنامه نویسان (Bob) طرحی در سر دارد، و درخواستها مستند سازی شده اند، بنابراین کس دیگری به غیر از Bob، یک دید خوب نسبت به ساختمان سیستم ندارد. اگر (Bob) برود، اطلاعات هم با او می‌روند اگر شما تا به حال یکی از کارهای Bod را گرفته باشید می‌توانیم بفهمید که چقدر درک یک سیستم با مستندات کم مشکل می‌باشد.

Requirements object model code

- مشتریان و مدیران پروژه از نمودارهای use case استفاده خواهند کرد تا دید سطح بالایی را نسبت به سیستم به دست آورند بر روی محدوده پروژه موافقت کنند.
- مدیران پروژه از نمودارهای use case استفاده خواهند تا پروژه را به تکۀهای قابل مدیریت بشکنند.
- تحلیلگران و مشتریان به مستندات use case نگاه خواهند کرد تا ببینند که سیستم چه عملیاتی را فراهم خواهد کرد.
- نویسندگان تکنیکی به مستندات use case نگاه خواهند کرد تا شروع به نوشتن طرحهای آموزشی و دستی کاربران کنند.

- تحلیلگران و برنامه نویسان به نمودارهای sequence , collaboration نگاه خواهند کرد تا ببینند که چگونه منطق؟ آبجکت های سیستم، پیغام های بین آبجکت ها جریان خواهند داشت.
- کارمندان تضمین کیفیت از اسناد use case و نمودارهای sequence , collaboration خواهند کرد تا اطلاعاتی را به دست آورند که برای تست script نیاز دارند.

تا دید جزیی نسبت به قطعات سیستم و چگونگی ارتباط آنها را به دست آورند.

- کارمندان Dep loyment از نمودارهای component , Deployment استفاده خواهند کرد تا ببیند که چه فایلهای اجرایی، فایلهای DDL یا component های دیگری ایجاد خواهند شد و این component در کجا بر روی شبکه قرار خواهند گرفت.
- کل تیم از مدل استفاده خواهند کرد تا مطمئن شوند که درخواستها به کد تبدیل شده اند و آن کد می‌تواند به درخواستها تبدل شود.

بنابراین، Rose ابزاری است که به وسیله کل تیم پروژه استفاده می‌شود. منبع فرصت و آزادی عمل واطلاعات طراحی است که هر عضو تیم می‌تواند استفاده کند تا اطلاعاتی که نیاز دارد را به دست آورد.

همچنین Rotional Rose با تولید کد اولیه به برنامه نویسان کمک خواهد کرد و می‌تواند این کار را برای تعدادی از زبانهای متفاوت در بازار شامل c++، جاوا، power Builder , visual Basic انجام دهد.

داشتن یک مدل در Rose برای یک برنامه کاربردی موجود بسیار سودمند است وقتی که تغییر در مدل ایجاد می‌شود، Rose می‌تواند کد را اصلاح کند تا تغییر در بر گیرد. وقتی که

تغییری در کد ایجاد می‌شود، شما می‌توانید آن تغییر را در مدل به صورت خود کار ثبت نمایید. این ویژگی ها به شما کمک می‌کنند تا مدل را حفظ کرده و کد را هماهنگ نمایید و خطر داشتن یک مدل قدیمی را کاهش می‌دهد.

همچنین Rose می‌تواند با استفاده از Rose script (یک زبان برنامه نویسی است که با Rose بسته بندی شده است) گسترش داده شود. شما می‌توانید با استفاده از این زبان برنامه نویسی، کدی بنویسید که به طور خودکار تغییراتی را در مدل تان ایجاد نماید، یک گزارش ایجاد کنید یا کارهای دیگری را با مدل Rose خود اجرا نمایید در حال حاضر سه نسخه متفاوت از Rose در دسترس است.

Rose Modeler که به شما اجازه می‌دهد تا مدلی را برای سیستم خود ایجاد کنید اما ازتولید و مهندسی معکوس کد پشتیبانی نمی کند.

Rose Professionae که به شما اجازه می‌دهد تا کدی را در C++ و , visual Basic Java ایجاد کند مانند Oracle ۸ علاوه بر این، Rose ۹۸i، ارتقا (بهبودی) جدید Rose، بر روی یکپارچه کردن Rose با دیگر ابزارها مانند Visual C++, Team test, Rational Requisite pro وغیر متمرکز می‌شود. همچنین Rose ۹۸i به شما اجازه می‌دهد تا مدل خود را روی وب منتشر نکنید Rose ۹۸i مانند Rose ۹۸ در نسخه های , Enterprise Professional , Modeler موجود می‌باشد.

نصب Rose ۹۸

از طریق CD بر روی یک کامپیوتر با windows ۹۵, NT, windows ۹۸ نصب می‌شود هنگام نصب، اگر کامپوتر شما از فایلهایی خود اجرای روی CD پشتیبانی نمی کند باید

فایل setup. Exe را از فهرست ریشه روی CD اجرا کنید. مکان پیش فرش به این صورت است:

C:\prog ram files\Rational\Rational Rose ۹۸ Enterprise Edition

نصب Rose ۹۸i نیز به همین صورت انجام می‌شود. با این تفاوت که هنگام اجرای آن باید یک کلید مجوز معتبر وارد کنید که از طریق پنجره License key Admin strator در هنگام نصب این اطلاعات داده شده و برای هر بار اجرای Rational Rose باید آنها را وادار کنیم.

Rose عمدتا یک برنامه نویسی (menu- driven) است و یا از نوارهای ابزاری که به اشکال استفاده شده عمومی کمک می‌کند، استفاده می‌نماید Rose از هفت نوع متفاوت از نمودارهای jml پتشیبانی می‌کند نمودارهای collaboration , sequence, use case Rose. Deployment , component state transition, class برای هر یک از این نمودارها یک نوار ابزار متفاوت را نمایش خواهد داد.

یکی از آسان ترین راهها برای پرداختن به Rose استفاده از مرورگر است با مرورگر، به سرعت و به آسانی، نمودارها و عناصر دیگری از مدل به دست می‌آیند.

بخشهای صفحه نمایش

پنج قسمت مهم واسط کاربر Rose عبارتند از مرورگر (Browser)، پنجره مستند سازی (Documentation) نوار ابزار (Toolbar)، پنجره نمودار Log, Diagram window اهداف آنها به طور مختصر عبارتند از:

Browser (مرورگر): حرکت نمودن در سراسر مدل با سرعت بیشتر

Docament window (پنجره مستندسازی): دستیابی به عناصر مدل

Toolbar(نوار ابزار): دستیابی سریع به زبانهایی که عموما استفاده می‌شوند.

Diagram sinsow (پنجره نمودار): نمایش و ویرایش یک یا چند نمودار uml

Log: دیدن خطاها و گزارش نتایج فرمانهای مختلف

Browser (مرورگر)

مرورگر یک ساختار سلسله مراتبی است که با آن می‌توان در سراسر مدل Rose حرکت کرد و هر چیزی که به مدل اضافه کنید- عاملها، use case، کلاسها، component ها (اجزاء) و غیره- در مرورگر نمایش داده خواهد شد.

با استفاده از مرورگر شما می‌توانید: عناصر مدل را اضافه کنید، ببینید، روابط موجود بین عناصر را ببینید.

عناصر مدل را انتقال دهید، نام عناصر مدل را تغییر دهید، یک عنصر مدل را به یک نمودار اضافه کنید، یک فایل یا URL را به یک عنصر ضمیمه کنید، عناصر را در بسته هایی گروهبندی کنید، به مشخصات جزئی تر یک عنصر دستیابی پیدا کنید، و یک نمودار را باز کنید. چهار نما در مرورگر وجود دارد: نمای use case، نمای Logical، نمای componenation window و نمای Deployment.

Documentation window (پنجره مستند سازی)

برای مستند کردن عناصر مدل Rose شما، استفاده می‌شود. وقتی شما مستندی را به یک کلاس اضافه می‌کنید، هر چیزی را که در پنجره مستند سازی تایپ می‌کنید، به عنوان فرمان در کد تدبیر شده ظاهر شد و نیاز به رفتن به آینده و فرمان روی کد سیستم را کاهش

می‌دهد. همچنین مستند سازی در گزارشهایی ظاهر می‌شود که شما می‌توانید از Rose تولید کنید. هنگامی که شما عناصر مختلفی را از مرورگر یا از روی نمودار انتخاب می‌کنید، پنجره مرورگر به طور خود کار بروز رسانی می‌شود تا مستندات عنصر انتخاب شده را نمایش دهد.

Toolbar (نوار ابزار)

نوار ابزار، دستیابی سریعی به فرمان هایی که عموما استفاده می‌شوند را فراهم می‌کند. در Rose، دو نوار ابزار وجود دارد نوار ابزار استاندارد (standard) نوار ابزار نمودار (Diagram)

Diagram window (پنجره نمودار)

می توانید یک یا چند نوار UML را در مدلتان ببینید. وقتی شماه تغییراتی را در عناصر یک نمودار ایجاد می‌کنید، در مواقع ضروری Rose به طور خودکار مرورگر را بروز رسانی می‌کند. به همین ترتیب، وقتی که شما با استفاده از مرورگر تغییراتی را در یک عنصر ایجاد می‌کنید، Rose به طور خودکار نمودارهای مناسب را بروز رسانی خواهد نمود. با انجام این کارها، Rose به شما کمک می‌کند یک مدل را به طور ثابت (سازگار) نگهداری کنید (حفظ کنید).

Log هنگامی که روی مدل Rose کار می‌کنید،

اطلاعات مشخصی در پنجره log درج خواهند شد. هیچ راهی برای بستن پنجره log وجود ندارد اما اندازه این پنجره را می‌توان مبینیمم کرد.

چهار نمای وجود در یک مدل Rose:

use case

نمای use case شامل هم عامل ها use case و نمودارهای use case سیستم می‌باشد و ممکن است شامل تعدادی نمودارهای sequence , collaboration باشد. نمای use case یک نگاه وابسته به پیاده سازی در سیستم است. این نما بر روی تصویری سطح بالا متمرکز می‌شود که سیستم چه (what) کاری انجام خواهد داد بدون اینکه نگران جزئیات چگونگی کار در سیستم باشد.

نمای use case شامل Actors (موجودیت های خارجی که با سیستم ساخته شده ارتباط برقرار می‌کنند), use case (قطعات سطح بالایی از عملیاتی هستند که سیستم فراهم خواهد کرد) use case documentationمستندات use case, نمودارهای Interaction, بسته ها (گروه هایی از use case ها و عاملها) است. هنگامی که پروژه به پیش می‌رود, همه اعضای گروه می‌توانند به نمای use case نگاه کنند تا درک سطح بالایی نسبت به سیستم ساخته شده را به دست آورند. مستند سازی use case جریان رخدادها را در طول یک use case شرح خواهد داد.

نمای منطقی (logica view)

این نما, بر روی این متمرکز می‌شود که چگونه سیستم, رفتار در use casپیاده سازی می‌کند. این نما, تصویر مفصلی از قطعات سیستم را فراهم می‌کند, چگونگی ارتباط قطعات

را شرح می‌دهند نمای منطقی شامل چیزهایی میانی دیگر, کلاس مشخصی که ضرورت پیدا خواهد کرد, نمودارهای class و نمودارهای state transiton است. برنامه نویسان, می‌توانند با این عناصر مفصل, طرح مفصلی را برای سیستم بسازند.

اغلب, گروهها در نمای منطقی از یک روش دو گذره(Two- pass) استفاده می‌کنند. در اولین روش, آنها کلاسهای تجزیه و تحلیل (Malysis classes) را تعیین می‌کنند. کلاسهای تجزیه و تحلیل, کلاس های وابسته به زبان اند. گروه در ابتدا با متمرکز شدن نمای منطقی (Logical) روی ساختار منطقی سیستم است. در این نما, شما قطعات سیستم را تعیین می‌کنید. اطلاعات و رفتار سیستم را تست کرده و ارتباطات بین قطعات را امتحان کنید. Reuse (استفاده مجدد) در اینجا یکی از ملاحظات اصلی است. اگر اطلاعات و رفتارها را با دقت به کلاس ها نسبت دهید, کلاسهای خود گروهبندی کنید, ارتباطات بین کلاسها و بسته ها را امتحان کنید می‌توانید کلاس ها بسته هایی را تعیین کنید که می‌تواند مجددا استفاده شوند. وقتی شما پروژه های بیشتر و بیشتری را کامل کنید, می‌توانید کلاس ها و بسته های جدیدی را به reuse اضافه کنید.

نمای component

نمای component شامل اطلاعاتی درباره کتابخانه های کد, فایلهای قابل اجرا کتابخانه های زمان ارجاع و component های دیگر مدل شما می‌باشد. یک component جز معیار فیزیکی از کد است. نمای componentشامل: component, نمودارهای آن, وابسته است (گروههایی از component های برنامه هستند. تعدادی از component ها, کتابخانه های کد هستند. ما بقی, کامپوننت های زمان اجرا خواهند بود مانند فایلهای زمان اجرای فایلهای

DDL (کتابخانه پیوندی پویا). برنامه نویسان از نمای component استفاده می‌کنند تا بینند که چه کتابخانه های کدی ایجاد شده اند و هر کتابخانه که شامل کدام کلاسها هستند.

نمای Deployment

نمای نهایی در Rose نمای Deployment, مربوطه به گسترش فیزیکی سیستم است که ممکن است به سبک معماری منطقی سیستم متفاوت باشد. مثلا ممکن است سیستم یک سبک معماری سه طبقه ای منطقی داشته باشد. به عبارت دیگر, ممکن است واسط کاربر از منطق تجاری و موضوعی جدا شده باشد, که از منطق پایگاه داده جدا شده است.

اگر چه Deployment ممکن است دو طبقه ای باشد. ممکن است واسط کار بر روی یک ماشین در گرفته باشد در حالی که منطق پایگاه داده و تجاری (موضوعی) روی ماشین دیگری واقع شده اند. همچنین موضوعات دیگر همانند تحمل پهنای باند شبکه, بهبود خطا, زمان پاسخ, با استفاده از نمای Deployment قابل دسترسی شده است نمای Deployment شامل: Processes پردازشها که thread ها (نخهای پردازشی) هستند که در فضای حافظه خودشان قابل اجرا می‌باشند. Processor (پردازنده ها)، pevices (وسایل ها)، یک Deployment است. مجددا، کل تیم از اطلاعات نمای Deployment استفاده خواهند کرد تا بفهمند که سیستم چگونه تولید شده است اگر چه، کاربران اصلی در نهایت مسئولین توزیع کردن برنامه هستند.

کاربرد برنامه Retional Rose

هر کاری که در Rose انجام دهید وابسته به یک مدل است.

ایجاد مدلها اولین مرحله در کار با Rose، ایجاد یک مدل است مدلهامی توانند از طریق موقتی و یا با استفاده از مدل چار چوب کاری موجود، ایجاد شده باشند. یک مدل Rose، شامل همه نمودارها آبجکت ها دیگر عناصر مدل است که در یک فایل جداگانه با DML (مدل) ذخیره شده اند.

کلیک ۰k انتخاب چارچوب framework wizard نصبNew file (ایجاد فایل)

نام فایل Log file save log Log دخیره Log eile save ذخیره مدلها

وارد کردن و ارسال مدلها

یکی از امتیازات اصلی الگوی شی گرایی reuse (استفاده مجدد) است. Rose از وارد کردن و ارسال مدلها و عناصر مدل پشتیبانی می‌کند. شما می‌توانید یک مدل یا بخشی از یک مدل را ارسال کرده و آن را وارد مدلهای دیگر نمایید.

نام فایل ارسال شده را وارد کنید File Export Model ارسال یک مدل (Export)

نام فایل خارج شده را وارد کنید File Export <package> انتخاب بسته ارسال یک بسته کلاسها

نام فایل ارسالی را وارد کنید File Export <class> انتخاب کلاس ارسال یک کلاس

انتخاب فایل File Imoprt model وارد کردن یک مدل، بسته یا کلاس

انوع فایلهای قابل قبول (MDL) model, (PIL) patal, (CAP) category یا (SUB) subsystem هستند.

می توان مدلها را با استفاده از Rational Rose ۹۸i بر روی وب منتشر کرد ب؟ اینکه جزء کاربران Rose باشند و بدون اینکه بسیاری از مستندات مدل را چاپ کنند.

Tools → Web publisher → انتخاب نمادها و بسته ها → انتخاب سطح دلخواهی از جزئیات → انتخاب برای انتشار → آیتم های به ارث برده شده منتشر شوند یا نه ؟

Publish → نوع فرمت گرافیکی → اگر فایل یا فرمت گرافیکی است انتخاب دکمه Diagrams → نام فایل رشدی HTML را وارد کنید → خاصیتها منتشر شوندیا نه؟

کار با واحدهای کنترل شده

Rose از چند کاربری (multi- user) توسعه ؟ با استفاده از واحدهای کنترل شده پشتیبانی می‌کند، یک واحد کنترل شده در Rose می‌تواند هر بسته ای در داخل نمای use case نمای Logcal یا نمای component باشد واحدهای نمای model properties Deployment می‌توانند تحت کنترل قرار گیرند.

وقتی که یک واحد کنترل شده است، در یک فایل جدا از بقیه مدل ذخیره می‌شود بدین روش، فایل مجزا را می‌توان با استفاده از یک نسخه فرمانبردار scc ابزاری مانند Rotional crearcase Microsoft saurcesafe یا حداقل Rose به طور مستقیم کنترل کرد.

واحدهای کنترل شده می‌توانند از طریق مدل دیده شده بازگذاری (load) و یا حذف (un load) شوند یا اگر از یک نسخه کنترل ابزار استفاده می‌شود، آنها را انتخاب نمود (checked in) یا از انتخاب خارج کرد (check out)

نام فایل را برای واحد کنترل شده وارد کنید ← Control <package> ← units ← روی unit کلیک راست ← منوی ایجاد یک واحد کنترل شده در Rose ۹۸i

آیکن درون مرورگر، دارای یک نشانه صفحه ای بر روی پوشه می‌شباشد تا بسته ای که کنترل شده است را نشانه گذاری می‌کند.

Unload < package ← منوی Unit ← کلیک راست روی unit ← Un load یک واحد کنترل شده

Unload sobunits of < view> ← منوی Units ← کلیک راست روی نما ← Unload هم واحدهای کنترل شده در یک نما

انتخاب واحد کنترل شده ← Load ← Load < package> ← منوی units ← کلیک راست روی unti ← Load یک واحد کنترل شده

Write protect ← انتخاب بسته ← کلیک راست روی units ← حفاظت در مقابل نوشتن

Write Enable ← انتخاب بسته ← Units ← منوی Browse ← فعال کردن نوشتن بر روی یک واحد کنترل شده