فرادرس
فراتر از یک کلاس درس
www.faradars.org

# آموزش مفهوم UML با کمک نرم افزار Rational rose

مدرس:
سمیه توکلی
کارشناسی ارشد معماری سیستم های کامپیوتری

# فهرست مطالب

# فصل اول

## مقدمه ای بر Uml و نرم افزار Rational Rose

۱-۱: آشنایی با مفهوم Uml

۱-۲: آشنایی با محیط نرم افزار rational rose

# آشنایی با مفهوم UML

# مقدمه

زبان مدل سازی یکنواخت ( Unified Modeling Language) یا UML یک زبان مدلسازی است که برای تحلیل و طراحی سیستم های شی گرا به کار می‌رود. UML اولین بار توسط شرکت Rational ارائه شد و پس از آن از طرف بسیاری از شرکت های کامپیوتری و مجامع صنعتی و نرم افزاری دنیا مورد حمایت قرار گرفت.

**UML زبانی است برای مدلسازی یا ایجاد نقشه تولید نرم افزار.**

**UML یا زبان مدلسازی یکنواخت، زبانی است برای مشخص کردن (Specify)، مصورسازی (Visualize)، ساخت (Construction) و مستندسازی (Documenting) محصولات میانی یک سیستم نرم افزاری می باشد.**

# چرا مدل و مدلسازی؟

ایجاد یک مدل برای سیستمهای نرم افزاری قبل از ساخت یا بازساخت آن، به اندازه داشتن نقشه برای ساختن یک ساختمان ضروری و حیاتی است. مدلهای خوب و دقیق در برقراری یک ارتباط کامل بین افراد پروژه، نقش زیادی می توانند داشته باشند.

شاید علت مدل کردن سیستمهای پیچیده این باشد که تمامی آن را نمی توان یک باره مجسم کرد، بنابراین برای فهم کامل سیستم و یافتن و نمایش ارتباط بین قسمتهای مختلف آن، به مدلسازی می‌پردازیم.

یک زبان مدلسازی، زبانی است که فرهنگ لغات و قواعد آن بر نمایش فیزیکی و مفهومی آن سیستم متمرکزند. برای سیستمهای نرم افزاری نیاز به یک زبان مدلسازی داریم که بتواند دیدهای مختلف معماری سیستم را در طول چرخه تولید آن، مدل کند.

UML شامل تعدادی عنصر گرافیکی است که از ترکیب آنها نمودارهای UML شکل می گیرند. هدف استفاده از نمودارهای مختلف در UML، ارائه دیدگاه های گوناگون از سیستم است.

نکته ای که باید حتما به آن توجه کنید این است که : مدل UML آنچه که یک سیستم باید انجام دهد را توضیح می دهد، ولی چیزی درباره نحوه پیاده سازی سیستم نمی گوید.

# ویژگی های زبان UML

UML یک زبان مدلسازی است اما چیزی فراتر از چند نماد گرافیکی است. به طوریکه در ورای این نمادها، یک سمانتیک (معناشناسی) قوی وجود دارد، به طوریکه یک تولیدکننده می‌تواند مدلهایی تولید کند که تولیدکننده های دیگر و یا حتی یک ماشین آن را بخواند و بفهمد. بنابراین یکی دیگر از نقش های مهم UML "تسهیل ارتباط" بین اعضای پروژه و یا بین تولیدکنندگان مختلف می باشد. این ارتباط بسیار مهم است. شاید دلیل اصلی اینکه تولید نرم افزار به صورت فریبنده ای دشوار است، همین عدم ارتباط مناسب بین اعضای پروژه باشد و اگر در تولید نرم افزار، بین اعضای پروژه گزارشهای هفتگی و مداوم وجود داشته باشد، بسیاری از این دشواریها برطرف خواهد شد.

البته این را هم باید در نظر گرفت که UML کمی پیچیده است و این به خاطر آن است که سعی شده است نمودارهایی فراهم شود که در هر موقعیتی و با هر ترتیبی قابل استفاده باشند. دلیل دیگر پیچیدگی از آنجا ناشی می شود که UML ترکیبی است از زبانهای مختلف، که برای حفظ سازگاری و جمع کردن خصوصیات مثبت آنها، ناگزیر از پذیرش این پیچیدگی می باشد.

# ویژگی های زبان UML

UML موفقیت طرح را تضمین نمی کند، اما در عین حال خیلی چیزها را بهبود می‌بخشد. به عنوان مثال استفاده از UML، تا حد زیادی، هزینه های ثابتی نظیر آموزش و استفاده مجدد از ابزارها را در هنگام ایجاد تغییر در سازمان و طرحها کاهش می دهد.

مساله دیگر اینکه، UML یک زبان برنامه نویسی بصری (visual) نیست، اما مدلهای آن را می‌توان مستقیماً به انواع زبانهای مختلف ارتباط داد. یعنی امکان نگاشت از مدلهای UML به کد زبانهای برنامه نویسی مثل Java و C++ وجود دارد که به این عمل "مهندسی رو به جلو" می گویند.

عکس این عمل نیز ممکن است؛ یعنی این امکان وجود دارد که شما بتوانید از کد یک برنامه زبانی شی گرا، مدلهای UML معادل آن را به دست آورید. به این عمل "مهندسی معکوس" می گویند. مهندسی رو به جلو و معکوس از مهمترین قابلیت های UML به شمار می روند، البته نیاز به ابزار Case مناسبی دارید که از این مفاهیم پشتیبانی کنند.

# ویژگی های زبان UML

یکی دیگر از ویژگی های مهم UML این است که مستقل از متدولوژی یا فرایند تولید نرم افزار می باشد و این بدان معنی است که شما برای استفاده از UML، نیاز به استفاده از یک متدولوژی خاص ندارید و می توانید طبق متدولوژی های قبلی خود عمل کنید با این تفاوت که مدلهایتان را با UML نمایش می دهید. البته مستقل بودن از متدولوژی و فرایند تولید، یک مزیت برای UML می‌باشد؛ زیرا بسیاری از انواع پروژه ها و سیستمها نیاز به متدولوژی خاص خود دارند. اگر UML در پی پیاده کردن همه اینها بر می آمد، یا بسیار پیچیده می شد و یا استفاده خود را محدود می کرد. البته متدولوژیهایی براساس UML در حال شکل گیری می باشند.

# ویژگی های زبان UML

1. مدل UML آنچه که یک سیستم باید انجام دهد را توضیح می دهد، ولی چیزی درباره نحوه پیاده سازی سیستم نمی گوید.
2. مدل UML باعث "تسهیل ارتباط" بین اعضای پروژه و یا بین تولیدکنندگان مختلف می شود.
3. به دلیل اینکه UML ترکیبی است از زبانهای مختلف، به منظور حفظ سازگاری و جمع کردن خصوصیات مثبت آنها، ناگزیر کمی پیچیدگی می باشد.
4. استفاده از UML، تا حد زیادی، هزینه های ثابتی نظیر آموزش و استفاده مجدد از ابزارها را در هنگام ایجاد تغییر در سازمان و طرحها کاهش می دهد.
5. مهندسی رو به جلو و مهندسی معکوس از مهمترین قابلیت های UML به شمار می روند.
6. مدل UML مستقل از متدولوژی یا فرایند تولید نرم افزار می باشد.

# آشنایی با محیط نرم افزار
# Rational Rose

# فصل دوم

# مفاهیم پایه ای در Rational Rose

۲-۱: آشنایی با مفاهیم Actor و Use Case

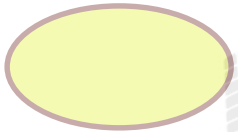
۲-۲: انواع Stereotype

۲-۳: انواع viwe در نرم افزار رشنال رز

# آشنایی با مفاهیم Actor و Use Case

# Use Case

Use Case در واقع عبارت است از هر سرویسی که سیستم در اختیار کاربران قرار می دهـد. شـکل آن در نمودارهـای UMLماننـد شکل زیر است.

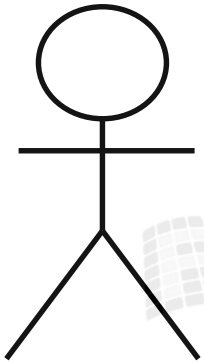


Use Case

سناریو: سناریو در واقع متنی است که فعالیتهای Use Case را بطور کامل شرح می دهد.

# Actor

هر کس که با Use Case کار می کند یک Actorاست. Actorها کسانی هستند که اطلاعـاتی از Use Caseدریافـت مـی کننـد و اینکه اطلاعاتی به آن تزریق می نمایند. شکل Actorدر نمودارهای UMLمانند شکل زیر است.



Actor

# Actor

- یک Actorدر واقع کسی است که Use Caseدر جهت سرویس دادن به آن عمل می کند. معمولاً Actorها از این بابت مهم می باشند که سیستم ساخته شده باید جوابگوی نیازهای آنها باشد. می توان گفت که شناسایی Actorها اولین قدمی است که برای رسم Use Caseدیاگرام برداشته می شود.
- **از دیدگاه کلی انواع Actorها عبارتند از :**

۱) Actorهای اصلی
۲) Actorهای فرعی

- **از دیدگاه دیگر می توان گفت که Actorها عبارتند از :**

۱) کاربران سیستم
۲) سیستم های دیگر
۳) زمان

# Actor

- **Actorهای اصلی:** Actorهای اصلی معمولاً کسانی هستند که از سیستم اطلاعات می گیرند یا به آن اطلاعات تزریق می کنند.

- **Actorهای فرعی:** Actorهایی هستند که مستقیماً با سیستم در ارتباط نیستند، ولی وجودشان برای فعال نگه داشتن سیستم ضروری می باشد.

- ✓ مشخص نمودن اینکه کدام Actorاصلی و کدام Actorفرعی است از بابت مورد توجه است که یک سیستم در نهایت جهت استفاده Actorهای اصلی تهیه می شود و نیازمندیهای آنها بالاترین اولویت را برای تولید سیستم برای ما ایجاد می کند.

- **زمان:** از زمان می توان به عنوان یک Actorاستفاده کرد ؛ مثل زمانبند سیستم عامل که ما مشخص می کنیم در یک زمان خاص عملی صورت بپذیرد،

# انواع Stereotype

- Actor :یا عامل که قبلاً درباره آن بحث نموده ایم.
- Boundary :به معنای User Interface هستند. یعنی هرکجا خواستیم بگوئیم واسط کاربر از ایـن Stereotype اسـتفاده می شود.
- Control :این آبجکت ها همان اشیاء کنترلی هستند یعنی هرکجا در تحلیل قصد نمایش اشیاء کنترلی را داشته باشیم از این Stereotype استفاده می شود.
- Entity: اشیای هستند که در سیستم وجود دارند. مثلاً شی بلیط را در سیستم صدور بلیط با ایـن Stereotype نمـایش مـی شود.
- Table: اگر از میان اشیاء از جدولی از پایگاه داده استفاده شود می توان برای نمایش آن از این Stereotype استفاده شود.

## انواع viwe در نرم افزار رشنال رز:

- Use Case viwe
- Logical view
- Component view
- Deployment view

# Use Case viwe

- Use Case viwe نمای مورد استفاده سیستم است، که به تشریح رفتار سیستم از دیدگاه کاربر می پردازد و در آن actorها و Use Case ها و رفتار متقابل آن ها نمایش داده می شود.

# Logical view

- Logical viewنقشه اصلی سیستم یا طرحی است که باید پیاده سازی شود. یعنی آنچه در Use Case viewنمایش داده شده است اکنون کمی تخصصی تر شده. به این معنی که ما در Logical view با مفاهیم برنامه نویسی بیشتر سرکار خواهیم داشت. عناصر اصلی Logical view همان کلاسها و اتریبیوتها و متدها هستند. عمده کار Logical view نمایش ارتباط واقعی میان کلاسها در سیستم است.

# Component view

- در Component view باید کلاسهای ایجاد شده در مرحله قبل به Componentتبدیل شوند. در این viewهمچنین باید ارتباط میان این Componentها مشخص شود. این ارتباط می تواند از نوع ارتباط زمان اجرا یا زمان ترجمه یا حتی ارتباطات زمان کامپایل باشد. این Componentها همچنین می توانند زیر سیستمها باشند. این زیر سیستمها معمولاً مجموعه ای از کلاسهای هستند و از لحاظ منطقی با هم سازگاری داشتند و در کنار هم قرار گرفته اند. در این viewعمده فعالیتهای طراحی باید به صورت کدهای برنامه نویسی درآید.

# Deployment view

- Deployment view یا نمای پیاده سازی ، این نما شامل نگاشتی از فعالیت های موجود با سخت افزار سیستم است. و زمانی مهم است که برنامه ها و سرور ها در قسمت های مکانی مختلف موجود باشند، و لازم است که ساختار توزیعی محیط نشان داده شود.

# فصل سوم

# رسم نمودار Use Case

# رسم نمودار Use Case

- ما در نمودارهای Use Caseبه دنبال نیازمندیهای کاربران هستیم پس ابتدا باید کاربران را تشخیص دهیم در قدم بعد باید بدانیم هر کاربر از سیستم چه می خواهد. در Use Caseدیاگرام **هدف ما مستند سازی همه آن چیزی است که سیستم به کاربران خود ارائه می دهد**. در واقع Use Caseدیاگرام نقطه ورود یک سیستم است، از آنجا مسئله تحلیل می شود و سپس در اختیار طراحان و برنامه نویسان قرار می گیرد. Use Caseدیاگرام در عین سادگی یک دید کلی نسبت به آن چه در سیستم انجام می شود را نشان می دهد به همین خاطر قابل درک برای کاربران خواهد بود. لذا تحلیل گر سیستم می تواند برای بیان شناخت خود از سیستم و انتقال مفاهیم به کاربران از این نمودار استفاده کنند. در نمودارهای Use Caseما باید Actorها و Use Caseها و همچنین روابط مابین آنها را مشخص کنیم، اینها در واقع سه عنصر تشکیل دهنده Use Caseدیاگرام می باشند.

# تعیین رابطه در نمودار Use Case

# فصل چهارم

# نمودار های تعاملی (Interaction)

# مقدمه

**یک آبجکت جیست؟** آبجکت ها در اطراف ما قرار دارند. آبجکت آن چیزی است که اطلاعات و روشها را در خود کپسوله(نگهداری) می کند.

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

# نمودار توالی

نمودار توالی برای نشان دادن جریان عملیات در یک Use Caseبر حسب زمان استفاده می شود. این نمودار موقعی مفید است که کس بخواهد روند منطقی یک سناریو را بازدید کند.

نمودار توالی موارد زیر را در بر می گیرد :

- Objects : یک نمودار Interactionمی تواند از نام آبجکت ها، نام کلاسها و یا از هر دوی آنها استفاده کنند.
- Messages: با استفاده از یک پیغام، یک آبجکت یا کلاس می تواند از یک آبجکت یا کلاس دیگر،برخی عملیات خاص را در خواست نماید.

# نمودار همکاری

یکی دیگر از نمودارهای Interaction، نمودار همکاری می باشد. نمودار همکاری شباهات بسیاری به نمودار توالی دارد، اصلی ترین تفاوت آنها در شمای ظاهری آنها می باشد. دیاگرام همکاری بیشتر بر روی رابطه بین آبجکت ها متمرکز می شود، در حالی که یک دیاگرام توالی اعمال آبجکت ها را در یک توالی زمانی نشان می دهد و بر حسب زمان منظم می شود.

## تفاوت میان نمودارهای توالی و نمودارهای همکاری

نمودارهای توالی اطلاعات را به ترتیب زمانی نشان می دهند و  برای مسیرهای متناوب به یک Use Case ساخته شده اند. آنها برای مشاهده پیشرفت عملیات یک Use Case مفید می باشند. نمودارهای همکاری، روند اطلاعات را نشان می دهند ولی در اینجا ترتیب زمانی در نظر گرفته نشده است. نمودارهای همکاری رابطه بین آبجکت ها و پیغام های بین آبجکت ها را شرح می دهند.

# فصل پنجم

## انواع رابطه در Rational Rose

۵-۱: رابطه Association

۵-۲: رابطه Dependency

۵-۳: رابطه Generalization

۵-۴: رابطه Aggregation

# رابطه Association

• Associationرابطه های معنایی بین کلاسها هستند که در نمودار کلاس بوسیله یک خط ساده به هم متصل می شوند. وقتی یک Association دو کلاس را به هم وصل می کند، هر کلاس می تواند از طریق یک نمودار توالی یا همکاری به کلاس دیگر پیغام بفرستد. Association می توانند دو طرفه یا یک طرفه باشند.

# رابطه Dependency

• Dependency شبیه به Association ها هستند با یک تفاوت که همیشه یک طرفه هستند. Rose در یک رابطه Dependency صفتها را برای کلاسها تولید نمی کند. Dependencyها با فلش خط چین نشان داده می شوند. به بیان دیگر از این رابطه بیشتر به منظور تعریف مفهوم یک کلاس استفاده می شود، تا با مشاهده نمودار کلاس به درک بهتری از کلاس ها برسیم.

•رابطه Dependency می تواند بین دو بسته نیز وجود داشته باشد. یعنی می توانید در نمودار package نیز از این رابطه استفاده کنید.

# رابطه Generalization

- Generalization ها برای نشان دادن یک رابطه وراثتی بین کلاسها استفاده می شوند. این رابطه بین دو کلاس که اصطلاحا کلاس پدر و کلاس فرزند نامیده می شوند تعریف می گردد.

# رابطه Aggregation

• Aggregationها یک فرم قویتر از Associationها هستند. رابطه Aggregation نشان دهنده یک ارتباط ساختاری میان دو کلاس است. یک رابطه Association به این معنی است که هر دو کلاس از نظر ادراکی در یک سطح قرار دارند و یکی از دو کلاس نسبت به دیگری مهمتر یا برتر نمی باشد. ولی رابطه Aggregation نشان دهنده یک رابطه کل به جزء است.

# فصل ششم

# کلاس ها و عملیات روی آن ها



۶-۱: نمودارهای کلاس

۶-۲: بررسی خصوصیات کلاس

۶-۳: انواع کلاس در نرم افزار Rational Rose

۶-۴:تعریف صفات در کلاس ها

۶-۵:تعریف عملیات های یک کلاس

۶-۶:تعیین نحوه نمایش یک کلاس

# نمودارهای کلاس

- یک کلاس دیاگرام یک دیاگرام برای توصیف یک سیستم است. کلاس دیاگرام شامل آیکون هایی است که کلاسها و نمونه ها و ارتباط بین آنها را نشان می دهد. شما می توانید یک یا بیش از یک کلاس دیاگرام را برای شرح کلاس ها در مدلی سطح بالا بسازید.
- معمولاً برای یک سیستم واحد چندین نمودار کلاس را ایجاد می شود، که برخی از آن ها زیر مجموعه ای از کلاس ها و روابط آنها را نمایش خواهند داد. شما می توانید بسیاری از نمودارهای کلاس را که نیاز دارید، بسازید تا یک تصویر کاملی را از سیستم خود بدست آورید. نمودار کلاس 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.

# بررسی خصوصیات کلاس

# انواع کلاس در نرم افزار Rational Rose

هفت نوع کلاس مختلف در نرم افزار رشنال رز قابل رسم هستند که عبارتند از:

1. Class
2. ParameterizedClass
3. InstantiatedClass
4. ClassUtility
5. ParameterizedClassUtility
6. InstantiatedClassUtility
7. MetaClass

# تعریف صفت ها در کلاس ها

# تعریف عملیات های یک کلاس

# تعیین نحوه نمایش یک کلاس

# فصل هفتم

## نمودار حالت یا State Diagram



۷-۱: مقدمه

۷-۲: انواع فعالیت های یک حالت (State)

۷-۳: نحوه ایجاد نمودار حالت

# مقدمه

- نمودار حالت نسبت به نمودارهای توالی و همکاری کمتر مورد استفاده قرار می گیرد. این نمودار همانطور که از نامش پیداست حالت های مختلفی که یک شی در آن قرار می گیرد را مدل می کند. در واقع این نمودار تصویری از چرخه حیات شی ( Object life cycle ) را به نمایش می گذارد.

**موارد استفاده از نمودار حالت ( State Chart Diagram)**

اشیائی که دارای تعداد زیادی حالت هستند.

اشیائی که برای Updateکردن صفات خاصه خود شروط متنوعی دارند.

اشیائی که معمولاً به صورت سخت افزاری هستند.

اشیائی که عملکرد بعدی آنها به عملکرد قبلی شان بستگی دارد.

- اشیا گفته شده بالا اشیائی هستند، که رسم نمودار حالت معمولاً برای آنها مفید خواهد بود. بررسی نمودارهای حالت به این دلیل است که نمودارها قابلیت این را دارند که یک سری از رفتارهای کلاسها را برای ما مشخص نمایند.

# انواع فعالیتهای یک حالت (State)

- Entry: مجموعه فعالیتهایی که در زمان ورود شی به یک حالت (State) باید انجام گیرند را در Entry قرار می دهند.
- Exit: مجموعه فعالیتهایی که در زمان خروج شی از یک حالت (State) باید انجام شوند را در Exit قرار می دهند.
- Do: این فعالیت برای انجام یک سری عملیات ایجاد شده است یعنی شی به یک حالت (State) می رود تا یک سری عملیات را انجام دهد. این فعالیت ها با نام Doدر حالت (State) قرار می گیرند.

# نحوه ایجاد نمودار حالت

# فصل هشتم

# نمودار مولفه یا Component



۸-۱: مقدمه

۸-۲: ابزارهای رسم یک نمودار Component

۸-۳:رسم و بررسی یک نمودار Component به همراه یک مثال

# مقدمه

- نمودار مولفه یا Component بر روی سازماندهی فیزیکی سیستم متمرکز می باشد. در این قسمت Component های سیستم را مشخص می کنیم. یک Component یک ماژول فیزیکی کد است که می تواند حاوی Source کد و فایل های زمان اجرا باشد.برای مثال در زبان C++ هر کدام از فایل های h. و cpp.یک Component جداگانه هستند. یک فایل EXE که بعد از کامپایل ایجاد می شود نیز یک مولفه جداگانه است.
- در نمودار Component، Component های مختلف و روابط میان آن ها نمایش داده می شود.

# ابزارهای رسم یک نمودار Component

# رسم و بررسی یک نمودار Component به همراه یک مثال

# فصل نهم

# نمودار Deployment



۹-۱: مقدمه

۹-۲: رسم یک نمودار Deployment

۹-۳:تعیین ویژگی های هر پردازشگر

۹-۴:تغییر نحوه نمایش نمودار

# مقدمه

- نمودار Deployment پردازش ها، دستگاه ها و ارتباطات میان آن ها را بیان می کند. به معنای دیگر وظیفه این نمودار نمایش نحوه استقرار فیزیکی برنامه کاربردی می باشد. هر سیستم دارای یک نمودار Deployment است که در آن ارتباط بین پردازش ها، دستگاه ها و نحوه تخصیص پردازش ها به پردازشگرها نمایش داده می شود.

# رسم یک نمودار Deployment

# تعیین ویژگی های هر پردازشگر

# تغییر نحوه نمایش نمودار

# مراحل رسم نمودارهای uml

1. محدوده سیستم را با استفاده از Use Caseها و actor ها تعریف و نمودار Use Case را رسم می شود.
2. یک مساله با استفاده از Use Case ها و مستندات آن ها تحلیل می شود.
3. توضیح اشیاء موجود در سیستم و گردش سیستم را در یک نمودار Sequence یا Collabroation رسم می شود.
4. کلاس هایی که برای پیاده سازی نیاز هستند، در نمودار کلاس رسم می شود.
5. صفات، عملیات و رابطه بین کلاس ها تعریف و نمودار کلاس را تکمیل می شود.
6. با ایجاد نمودار حالت رفتار داینامیکی سیستم ترسیم می شود.
7. با گروه بندی کلاس ها در بسته ها و بررسی رابطه های بین کلاس ها و بسته ها یک ارزیابی ساختاری انجام می شود.
8. ساختار شبکه و نحوه استقرار ابزارها و پردازشگرها در یک نمودار Deployment نمایش داده می شود.

فصل دهم

# تولید کد

۱۰-۱ نحوه تولید کد در محیط visual

۱۰-۲مهندسی معکوس

# نحوه تولید کد در محیط visual

# مهندسی معکوس

این اسلاید ها بر مبنای نکات مطرح شده در فرادرس

«آموزش مدل سازی UML با نرم افزار Rational Rose»

تهیه شده است.

برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید

faradars.org/fvsft9503