



شماره مستند: UML-EDU-1405-001

نسخه: ۰.

تاریخ تهیه: ۱۶ شهریور ۱۴۰۵ – ساعت ۱۰:۳۰

طبقه‌بندی: آموزشی / عمومی

EDUCATIONAL DOCUMENTATION

مستند آموزشی جامع نمودارهای UML

راهنمای کامل با مثال‌های عملی و کاربردی

همکاران گرامی و علاقه‌مندان به مهندسی نرم‌افزار،

این مستند آموزشی به‌عنوان مرجعی جامع برای یادگیری و درک عمیق هشت نمودار اصلی UML (Unified Modeling Language) تهیه شده است. هر فصل به تفصیل یک نوع نمودار را با ذکر المان‌ها، روابط، کاربردها و مثال‌های عملی توضیح می‌دهد. هدف از این مستند، ارائه درکی کاربردی و آماده‌سازی شما برای مدل‌سازی حرفه‌ای سیستم‌های نرم‌افزاری است.

فهرست مطالب

۱ نمودار مورد کاربرد (Use Case Diagram)

۲ نمودار فعالیت (Activity Diagram)

۳ نمودار ماشین حالت (State Machine Diagram)

۴ نمودار کامپوننت (Component Diagram)

۵ نمودار استقرار (Deployment Diagram)

نمودار کلاس (Class Diagram)

٧ نمودار توالی (Sequence Diagram)

٨ Swimlane نمودار

نمودار Use Case یکی از پرکاربردترین نمودارهای UML است که برای نمایش تعامل بین کاربران (Actors) و سیستم استفاده می‌شود. این نمودار دیدگاهی بیرونی از سیستم ارائه داده و نیازمندی‌های عملکردی را بدون ورود به جزئیات فنی نشان می‌دهد.



Actor (بازیگر)

نماینده کاربر، سیستم خارجی یا موجودیتی که با سیستم تعامل دارد. به صورت شکل انسان نمایش داده می‌شود.



Use Case (مورد کاربرد)

یک عملکرد یا سرویس خاص که سیستم ارائه می‌دهد. به صورت بیضی نمایش داده می‌شود.

System Boundary (مرز سیستم)

مستطیلی که محدوده سیستم را مشخص کرده و Use Case ها درون آن قرار می‌گیرند.



Association (ارتباط)

خط ساده که Actor را به Use Case مرتبط وصل می‌کند.

انواع روابط در Use Case:

Include (شامل): رابطه‌ای که نشان می‌دهد یک Use Case حتماً شامل Use Case دیگری است. با خط چین و stereotype <<include>> نمایش داده می‌شود.

Extend (گسترش): رابطه‌ای اختیاری که نشان می‌دهد یک Use Case می‌تواند در شرایط خاصی Use Case دیگر را گسترش دهد. با خط چین و stereotype <<extend>> نمایش داده می‌شود.

می‌شود.

▲ **Generalization (تعمیم):** رابطه ارث‌بری بین Actors یا بین Use Cases. با فلش مثلثی توخالی نمایش داده می‌شود.

مثال عملی: سیستم بانکداری آنلاین

Actors: مشتری، کارمند بانک، مدیر سیستم

:Use Cases

- مشتری → ورود به سیستم، مشاهده موجودی، انتقال وجه، پرداخت قبض
- کارمند بانک → ثبت مشتری جدید، تأیید تراکنش‌های بزرگ
- مدیر سیستم → مدیریت کاربران، مشاهده گزارش‌ها
- «انتقال وجه» <<include>> «بررسی موجودی»
- «ورود به سیستم» <<extend>> «احراز هویت دو مرحله‌ای» (در صورت فعال بودن)

💡 **نکته کلیدی:** نمودار Use Case در فاز تحلیل نیازمندی‌ها استفاده شده و به درک مشترک بین تیم فنی و ذینفعان کسب‌وکار کمک می‌کند.

نمودار Activity برای نمایش جریان کار (Workflow)، فرآیندهای کسب و کار و الگوریتم ها استفاده می شود. این نمودار شبیه فلوچارت است اما قدرت بیشتری در نمایش فعالیت های موازی و تصمیم گیری های پیچیده دارد.



Initial Node (گره شروع)

دایره توپر سیاه که نقطه شروع فرآیند را نشان می دهد.



Final Node (گره پایان)

دایره توپر با حاشیه ضخیم که پایان فرآیند را نشان می دهد.



Action (فعالیت)

مستطیل با گوشه های گرد که یک عملیات یا کار را نشان می دهد.



Decision/Merge (تصمیم/ادغام)

لوزی که برای انشعاب شرطی یا ادغام جریان ها استفاده می شود.



Fork/Join (انشعاب/هم زمانی)

خط ضخیم افقی یا عمودی برای نمایش فعالیت های موازی.



Control Flow (جریان کنترل)

فلش که ترتیب اجرای فعالیت‌ها را نشان می‌دهد.

مثال عملی: فرآیند ثبت سفارش آنلاین

جریان فرآیند:

- شروع: مشتری وارد سایت می‌شود
- فعالیت‌ها: انتخاب محصول → افزودن به سبد خرید → ورود/ثبت‌نام → انتخاب آدرس → انتخاب روش پرداخت
- Decision: آیا موجودی کافی است؟ (بله/خیر)
- Fork: هم‌زمان: ۱) کاهش موجودی انبار ۲) ارسال ایمیل تأیید
- Join: انتظار برای تکمیل هر دو فعالیت موازی
- پایان: ثبت نهایی سفارش

کاربردهای کلیدی: مدل‌سازی فرآیندهای کسب‌وکار، الگوریتم‌های پیچیده، گردش کار (Workflow)، و تحلیل سناریوهای مختلف

نمودار State Machine برای نمایش وضعیت‌های مختلف یک شیء و رویدادهایی که باعث تغییر وضعیت می‌شوند استفاده می‌شود. این نمودار برای مدل‌سازی چرخه حیات (Lifecycle) موجودیت‌ها ایده‌آل است.



Initial State (وضعیت شروع)

دایره توپر کوچک که وضعیت اولیه را نشان می‌دهد.

Final State (وضعیت پایانی)

دایره توپر با حاشیه ضخیم که وضعیت نهایی را نشان می‌دهد.



State (وضعیت)

مستطیل با گوشه‌های گرد که یک وضعیت خاص را نشان می‌دهد.

Transition (گذار)

فلش که تغییر از یک وضعیت به وضعیت دیگر را نشان می‌دهد. می‌تواند برچسب (رویداد/شرط/اقدام) داشته باشد.

ساختار Transition:

فرمت: eventName [guardCondition] / action

مثال: pay [amount > 0] / updateInventory()

وضعیت‌ها:

- ایجاد شده (Created)
- در انتظار پرداخت (Pending Payment)
- پرداخت شده (Paid)
- در حال پردازش (Processing)
- ارسال شده (Shipped)
- تحویل داده شده (Delivered)
- لغو شده (Cancelled)

گذارها:

- Created → Pending Payment: رویداد «ثبت سفارش»
- Pending Payment → Paid: رویداد «پرداخت موفق»
- Paid → Processing: رویداد «تأیید پرداخت»
- Processing → Shipped: رویداد «ارسال کالا»
- Shipped → Delivered: رویداد «تحویل به مشتری»
- هر وضعیت → Cancelled: رویداد «لغو» (با شرایط خاص)

⚠ **نکته مهم:** State Machine برای اشیایی استفاده می‌شود که چرخه حیات مشخص و وضعیت‌های گسسته دارند. برای فرآیندهای پیچیده با فعالیت‌های موازی، از Activity Diagram استفاده کنید.

نمودار Component برای نمایش ساختار فیزیکی سیستم، ماژول‌ها، کتابخانه‌ها و وابستگی‌های بین آن‌ها استفاده می‌شود. این نمودار دیدگاهی از معماری نرم‌افزار در سطح کامپوننت ارائه می‌دهد.



Component (کامپوننت)

مستطیل با آیکون کامپوننت (دو مستطیل کوچک در گوشه) که یک ماژول یا جزء سیستم را نشان می‌دهد.



Interface Provided (اینترفیس ارائه‌شده)

دایره (Lollipop) که سرویس‌هایی که کامپوننت ارائه می‌دهد را نشان می‌دهد.



Interface Required (اینترفیس مورد نیاز)

نیم‌دایره (Socket) که سرویس‌هایی که کامپوننت نیاز دارد را نشان می‌دهد.



Package (بسته)

پوشه‌ای که کامپوننت‌های مرتبط را گروه‌بندی می‌کند.



Dependency (وابستگی)

فلش خط‌چین که وابستگی بین کامپوننت‌ها را نشان می‌دهد.

Connector (اتصال)

خط جامد که اتصال بین Interface Provided و Required را نشان می‌دهد.

مثال عملی: سیستم مدیریت محتوا (CMS) 📄

کامپوننت‌ها:

- **User Management**: مدیریت کاربران و احراز هویت
- **Content Engine**: مدیریت محتوا و ویرایشگر
- **Media Library**: مدیریت فایل‌های چندرسانه‌ای
- **Search Module**: موتور جستجو
- **Notification Service**: ارسال اعلان‌ها

وابستگی‌ها:

- Content Engine → User Management (برای بررسی مجوزها)
- Search Module → Content Engine (برای ایندکس محتوا)
- Notification Service → User Management (برای دریافت اطلاعات کاربران)

اینترفیس‌ها:

- User Management: `IAuthentication`, `IUserProfile`
- Content Engine: `IContentCRUD`, `IVersioning`

✓ مزایا: درک معماری سیستم، مدیریت وابستگی‌ها، برنامه‌ریزی برای ماژول‌ریزاسیون، و تسهیل توسعه موازی تیم‌ها

نمودار Deployment برای نمایش سخت افزارها، نودها، محیط های اجرا و نحوه استقرار نرم افزار بر روی زیرساخت فیزیکی استفاده می شود. این نمودار دیدگاهی از معماری فیزیکی سیستم ارائه می دهد.



Node (گره)

مکعب سه بعدی که یک سخت افزار یا محیط اجرا (مانند سرور، کامپیوتر، کانتینر) را نشان می دهد.



Device (دستگاه)

نود فیزیکی مانند سرور، روتر، یا دستگاه موبایل.



Execution Environment (محیط اجرا)

نود نرم افزاری مانند Docker، JVM، OS، یا Kubernetes.



Artifact (مصنوع)

فایل های اجرایی، کتابخانه ها، پایگاه داده، یا فایل های پیکربندی که بر روی نودها استقرار می یابند.



Communication Path (مسیر ارتباطی)

خط جامد که ارتباط فیزیکی بین نودها را نشان می دهد.



Deployment Relationship (رابطه استقرار)

فلش خطچین که استقرار Artifact بر روی Node را نشان می‌دهد.

مثال عملی: سیستم وب اپلیکیشن میکروسرویس 📄

نودها:

- **Load Balancer:** Nginx (Device)
- **Application Servers:** Kubernetes Cluster (Execution Environment)
 - User Service Pod
 - Order Service Pod
 - Payment Service Pod
- **Database Server:** PostgreSQL Cluster (Device)
- **Cache Server:** Redis Cluster (Device)
- **Message Broker:** RabbitMQ Server (Device)

:Artifacts

- User Service → user-service.jar
- Order Service → order-service.jar
- Database → schema.sql, data.migration

ارتباطات:

- Load Balancer ⇌ Application Servers (HTTP/HTTPS)
- Application Servers ⇌ Database Server (JDBC)
- Application Servers ⇌ Cache Server (Redis Protocol)
- Application Servers ⇌ Message Broker (AMQP)

💡 **کاربرد:** برنامه‌ریزی زیرساخت، مستندسازی محیط‌های Development/Staging/Production، و تحلیل

Scalability و Performance

نمودار Class مهم‌ترین و پرکاربردترین نمودار UML است که ساختار ایستایی (Static Structure) سیستم را نشان می‌دهد. این نمودار کلاس‌ها، ویژگی‌ها، متدها و روابط بین آن‌ها را مدل‌سازی می‌کند.



Class (کلاس)

مستطیل سه‌بخشی: ۱) نام کلاس ۲) Attributes (ویژگی‌ها) ۳) Methods (متدها)



Interface (اینترفیس)

دایره یا مستطیل با `<<interface>>` stereotype که قراردادهای تعریف می‌کند.



Abstract Class (کلاس انتزاعی)

کلاس با نام ایتالیک که نمی‌توان نمونه مستقیم از آن ساخت.



Association (ارتباط)

خط جامد که ارتباط بین دو کلاس را نشان می‌دهد. می‌تواند Multiplicity (تعداد) داشته باشد.



Aggregation (تجميع)

لوزی توخالی که رابطه «has-a» ضعیف را نشان می‌دهد (جزء می‌تواند مستقل وجود داشته باشد).



Composition (ترکیب)

لوزی توپر که رابطه «has-a» قوی را نشان می‌دهد (جزء بدون کل نمی‌تواند وجود داشته باشد).



Inheritance (ارث‌بری)

فلش مثلثی توخالی که رابطه «is-a» یا Generalization را نشان می‌دهد.



Dependency (وابستگی)

فلش خط‌چین که استفاده موقت یک کلاس از کلاس دیگر را نشان می‌دهد.

Visibility (سطح دسترسی):

public | - private | # protected | ~ package +

مثال عملی: سیستم مدیریت کتابخانه 📖

کلاس‌ها:

• Book

Attributes: -isbn: String, -title: String, -author: String, -isAvailable: Boolean ◦

()Methods: +borrow(), +return(), +reserve ◦

• Member

Attributes: -memberId: String, -name: String, -email: String ◦

()Methods: +searchBook(), +borrowBook(), +returnBook ◦

• Loan

Attributes: -loanId: String, -loanDate: Date, -dueDate: Date ◦

◦ Methods: +calculateFine(), +extendLoan()

• **Librarian**: (ارث‌بری از Member)

◦ Methods: +addBook(), +removeBook(), +generateReport()

روابط:

• Loan «*» – «1» Member (یک عضو می‌تواند چندین امانی داشته باشد)

• Loan «*» – «1» Book (یک کتاب می‌تواند در چندین امانی باشد)

• Loan ♦ Book (Composition): امانی بدون کتاب معنی ندارد)

• Librarian $\triangle$ Member (ارث‌بری)

• Book Notification (وابستگی برای ارسال اعلان)

نکته طلایی: نمودار Class پایه و اساس طراحی شیء‌گرا است و مستقیماً به کد تبدیل می‌شود. طراحی دقیق این نمودار، کیفیت کد نهایی را تضمین می‌کند.

نمودار Sequence برای نمایش تعاملات بین اشیاء در طول زمان استفاده می‌شود. این نمودار نشان می‌دهد که پیام‌ها به چه ترتیبی بین اشیاء مبادله می‌شوند و برای درک سناریوهای اجرایی ایده‌آل است.

**Participant/Lifeline (خط حیات)**

مستطیل در بالا با خط‌چین عمودی که حضور یک شیء در طول زمان را نشان می‌دهد.

**Activation (فعال‌سازی)**

مستطیل باریک روی Lifeline که مدت زمان اجرای یک متد را نشان می‌دهد.

**Synchronous Message (پیام هم‌زمان)**

فلش جامد که فراخوانی متد را نشان می‌دهد (فرستنده منتظر پاسخ می‌ماند).

**Asynchronous Message (پیام ناهم‌زمان)**

فلش خط‌چین که ارسال پیام بدون انتظار برای پاسخ را نشان می‌دهد.

**Return Message (پیام بازگشت)**

فلش خط‌چین که مقدار بازگشتی یا پایان اجرا را نشان می‌دهد.



Self Message (پیام خودی)

فلش منحنی که فراخوانی متد روی خود شیء را نشان می‌دهد.



Combined Fragment (قطعه ترکیبی)

مستطیل با برجسب (alt, opt, loop, break) برای نمایش ساختارهای کنترلی.

انواع Combined Fragments:

alt (شرطی): معادل if-else (چندین گزینه با شرایط مختلف)

opt (اختیاری): معادل if (یک گزینه در صورت برقراری شرط)

loop (حلقه): معادل for/while (تکرار چندین بار)

break (شکست): معادل break (خروج از سناریو در صورت خطا)

مثال عملی: فرآیند خرید آنلاین 📄

Participants: Customer, ShoppingCart, PaymentGateway, Inventory, EmailService

ترتیب پیام‌ها:

1. Customer → ShoppingCart: addItem(product)
2. Customer → ShoppingCart: checkout()
3. ShoppingCart → Inventory: checkAvailability(product)
4. Inventory → ShoppingCart: return true
5. ShoppingCart → PaymentGateway: processPayment(amount)
6. alt [payment success]
 - PaymentGateway → ShoppingCart: paymentConfirmed()

```
ShoppingCart → Inventory: reduceStock(product) ◦  
ShoppingCart → EmailService: sendConfirmation(customerEmail) ◦  
EmailService → Customer: send email ◦  
    .7 [else payment failed]  
PaymentGateway → ShoppingCart: paymentFailed ◦  
ShoppingCart → Customer: showError ◦
```

✓ کاربرد: درک سناریوهای پیچیده، دیباگ کردن، مستندسازی API ها، و ارتباط بین تیم های Frontend و Backend

نمودار Swimlane (یا Activity Diagram با Swimlanes) نسخه پیشرفته‌ای از Activity Diagram است که فعالیت‌ها را بر اساس مسئولیت (Actor، Department، یا System) در Lane‌های جداگانه سازماندهی می‌کند. این نمودار برای نمایش «چه کسی چه کاری انجام می‌دهد» عالی است.



Swimlane (لاین شنا)

ناحیه افقی یا عمودی که تمام فعالیت‌های یک مسئول یا بخش را در خود جای می‌دهد.



Action (فعالیت)

مستطیل با گوشه‌های گرد که یک کار را در Swimlane مربوطه نشان می‌دهد.



Decision (تصمیم)

لوزی که می‌تواند بین Swimlane‌ها جریان ایجاد کند.



Cross-Lane Flow (جریان بین لاین‌ها)

فلش که انتقال کار از یک مسئول به مسئول دیگر را نشان می‌دهد.

مثال عملی: فرآیند استخدام نیروی جدید

Swimlanes: متقاضی، منابع انسانی (HR)، مدیر فنی، مدیرعامل

جریان فرآیند:

- **متقاضی:**

- ارسال رزومه → HR
- شرکت در مصاحبه فنی
- دریافت پیشنهاد کاری

- **HR:**

- بررسی اولیه رزومه
- هماهنگی مصاحبه → مدیر فنی
- بررسی سوابق (Background Check)
- ارسال پیشنهاد کاری (در صورت تأیید مدیرعامل)

- **مدیر فنی:**

- مصاحبه فنی
- ارزیابی مهارت‌ها
- ارسال نتیجه → HR

- **مدیرعامل:**

- تأیید نهایی استخدام (Decision)

Decision Points

- آیا رزومه مناسب است؟ (HR) → خیر: رد درخواست | بله: ادامه
- آیا مصاحبه فنی موفق بود؟ (مدیر فنی) → خیر: رد درخواست | بله: ادامه
- آیا مدیرعامل تأیید می‌کند؟ → خیر: رد درخواست | بله: ارسال پیشنهاد

💡 **مزیت کلیدی:** Swimlane Diagram به وضوح نشان می‌دهد که هر Actor یا بخش چه مسئولیتی دارد و کجاها نیاز به هماهنگی و Handoff وجود دارد. این نمودار برای بهینه‌سازی فرآیندهای کسب‌وکار و شناسایی گلوگاه‌ها عالی است.

🌟 **تفاوت با Activity Diagram ساده:** Activity Diagram فقط «چه کاری» را نشان می‌دهد، اما Swimlane Diagram «چه کسی چه کاری» را مشخص می‌کند.

هر یک از نمودارهای UML هدف خاصی دارند و در فازهای مختلف چرخه حیات توسعه نرم‌افزار استفاده می‌شوند:

نمودار	هدف اصلی	فاز استفاده	سطح
Use Case	نیازمندی‌ها و تعامل کاربران	تحلیل	بالا (Business)
Activity	فرآیندها و الگوریتم‌ها	تحلیل/طراحی	متوسط
State Machine	چرخه حیات اشیاء	طراحی	پایین (Technical)
Class	ساختار ایستایی سیستم	طراحی	پایین (Technical)
Sequence	تعاملات زمانی بین اشیاء	طراحی	پایین (Technical)
Component	معماری ماژولار	طراحی معماری	بالا (Architecture)
Deployment	زیرساخت فیزیکی	استقرار	بالا (Infrastructure)
Swimlane	مسئولیت‌ها در فرآیندها	تحلیل کسب‌وکار	بالا (Business)

توصیه‌های کلیدی

- از ساده شروع کنید: ابتدا Use Case و Activity Diagram برای درک نیازمندی‌ها
- سپس به جزئیات بروید: Class و Sequence Diagram برای طراحی فنی
- مستندسازی زنده: نمودارها را هم‌زمان با توسعه به‌روز نگه دارید
- ابزار مناسب: از ابزارهایی مانند ابزار UML استفاده کنید که امکان ویرایش آسان و Export به فرمت‌های مختلف را فراهم می‌کنند

💡 نتیجه نهایی: تسلط بر UML نه تنها مهارت مدل‌سازی شما را ارتقا می‌دهد، بلکه ارتباط مؤثر با ذینفعان، کاهش خطاهای طراحی و افزایش کیفیت نرم‌افزار را به همراه خواهد داشت.

منابع پیشنهادی برای مطالعه بیشتر:

- کتاب «UML Distilled» توسط Martin Fowler
- استاندارد OMG UML 2.5
- الگوهای طراحی (Design Patterns) توسط Gang of Four

این مستند آموزشی صرفاً جهت استفاده داخلی و آموزشی تهیه شده است. هرگونه کپی برداری یا افشا بدون اجازه کتبی ممنوع است.

طراحی، تحلیل و توسعه: هادی خزاعی اصل

تاریخ تهیه: ۱۶ شهریور ۱۴۰۵ – ساعت ۱:۳۰ | نسخه: ۱.۰ | طبقه بندی: آموزشی / عمومی