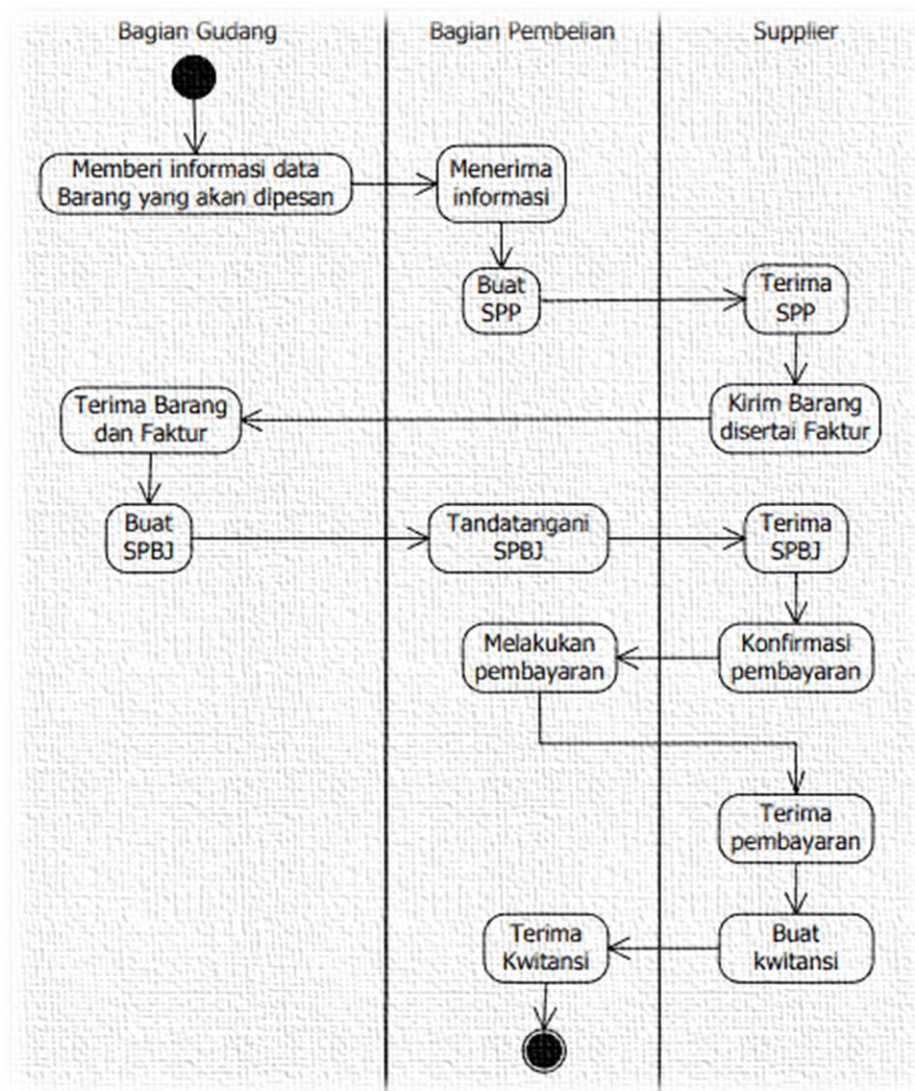


## نمودار فعالیت یا Activity diagram در UML



تهیه و تنظیم: پیمان مالکی



## فهرست مطالب

3	مقدمه
3	چه زمانی و به چه دلایلی تهیه این نمودار اهمیت خواهد داشت؟
4	activity
6	Activity Partition
9	Activity Edge
10	Object Flow Edge
10	Interrupting Edge
12	Actions
14	Object Action
14	Variable Action
15	Invocation Action
16	Call Behavior Action
17	Send Signal Action
17	Structural Feature Action
18	Link Action
19	Event Action
20	Accept Event Action
21	Accept Signal Action
21	Wait Time Action
22	Controls
22	Initial Node
23	Flow Final Node
23	Activity Final Node
23	Decision Node
25	Merge Node
26	Fork Node
26	Join Node
28	Objects
28	Object Node
29	Pin
29	Central Buffer

29	..... Data Store
31	..... خلاصه و چکیده
41	..... لغت نامه تصویری
41	..... Activity:
41	..... Partition:
41	..... Action:
42	..... Action Type:
44	..... Initial Node:
44	..... Flow Final Node:
44	..... Activity Final Node:
44	..... Edge:
45	..... Decision and Merge Node:
45	..... Decision Input Flow:
46	..... Decision Input:
46	..... Fork and Join Node:
46	..... Pin:
47	..... Data Store:

نمودار فعالیت یک نمودار رفتاری UML است که جریان کنترل یا جریان داده شیء را با تأکید بر توالی و شرایط جریان نشان می‌دهد. فعالیت ترتیبی بیان شده توسط این مدل فعالیت، یا به واسطه خاتمه فعالیت دیگری، آغاز شوند، زیرا اشیاء و داده‌های مورد نیاز در محدوده این مدل فعالیت، در دسترس خواهند بود، یا به واسطه برخی از رویدادهای خارج از جریان مورد نظر رخ می‌دهند.

گره‌ها و یال‌های زیر معمولاً در نمودارهای فعالیت UML ترسیم می‌شوند:

- activity
- activity partition
- activity edge
- action
- control
- object

چه زمانی و به چه دلایلی تهیه این نمودار اهمیت خواهد داشت؟

نمودار فعالیت برای مدلسازی جریان فعالیت سیستم مناسب است. یک برنامه کاربردی می‌تواند چندین سیستم داشته باشد. نمودار فعالیت نیز این سیستم‌ها را به تصویر می‌کشد و جریان از یک سیستم به سیستم دیگر را توصیف می‌کند. این کاربرد خاص در نمودارهای دیگر موجود نیست.

یک نمودار فعالیت، فرآیندهای کسب‌وکار و نرم‌افزار را به صورت مرحله‌ای از انجام کار، نشان می‌دهد. این اقدامات یا Action ها می‌توانند توسط افراد، اجزای نرم‌افزاری (متودها، سرویس‌ها یا سیستم‌ها) یا رایانه‌ها انجام شود. نمودارهای فعالیت برای توصیف فرآیندهای کسب‌وکاری یا BPD و Use Case و همچنین برای مستندسازی اجرای فرآیندهای سیستم استفاده می‌شود.

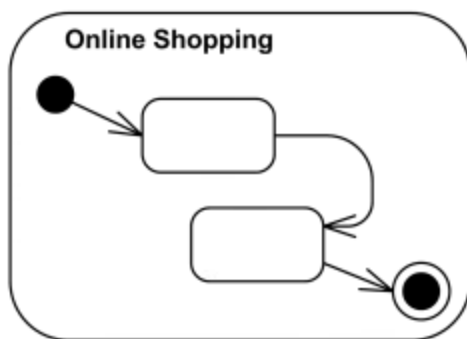
**activity** یا فعالیت یک رفتار پارامترپذیر است که به عنوان جریان هماهنگ شده **Action** ها یا اقدامات، نشان داده می‌شود. یک جریان اجرایی به شکل، **activity node** ها یا گره‌های فعالیت که توسط **edge** های فعالیت متصل می‌شوند، مدل‌سازی می‌شود. یک گره می‌تواند اجرای یک رفتار فرعی مانند محاسبات حسابی، فراخوانی یک عملیات یا دستکاری محتوای یک شیء را شامل شود. گره‌های فعالیت همچنین شامل جریان سازه‌های کنترلی مانند **synchronization** یا همگام‌سازی، **decision** یا تصمیم‌گیری و **concurrency control** یا کنترل همزمان هستند. فعالیت‌ها ممکن است سلسله مراتب فراخوانی را تشکیل دهند که فعالیت‌های دیگر را فراخوانی می‌کند و در نهایت به اقدامات منحصر به فرد منتج می‌شود. در یک مدل شی‌گرا، فعالیت‌ها معمولاً به صورت غیرمستقیم به عنوان متوذهای متصل به عملیاتی که مستقیماً فراخوانی می‌شوند، فراخوانی می‌شوند. فعالیت شامل گره‌های فعالیت است که می‌تواند:

- action
- control
- object

فعالیت‌ها ممکن است شامل انواع مختلف زیر باشد:

- وقوع توابع اولیه، مانند توابع حسابی.
- فراخوانی رفتار، مانند فعالیت‌ها.
- اقدامات ارتباطی، مانند ارسال سیگنال.
- دستکاری اشیاء، مانند خواندن یا نوشتن مشخصات یا وابسته‌ها.

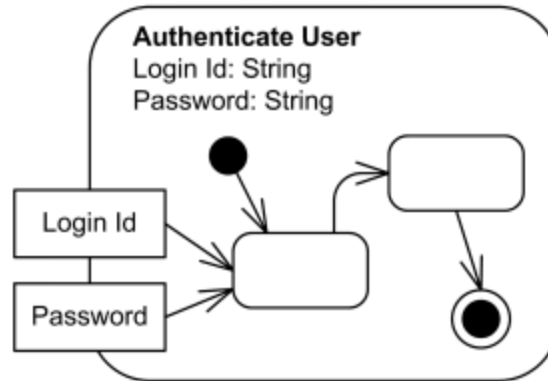
اقداماتی وجود دارند که فعالیت‌ها را فراخوانی می‌کنند، یا مستقیماً از اقدام فراخوانی رفتار استفاده می‌کنند یا به‌طور غیرمستقیم با اقدام فراخوانی عملیات این کار را انجام می‌دهند. فعالیت را می‌توان به صورت مستطیل گوشه گرد با نام فعالیت در گوشه سمت چپ بالا و نودها و **edge** های فعالیت در داخل این محدوده ارائه کرد. نمونه‌های مشخصات **UML 2.4** نام فعالیت را به صورت پررنگ نشان می‌دهند.



فعالیت خرید آنلاین

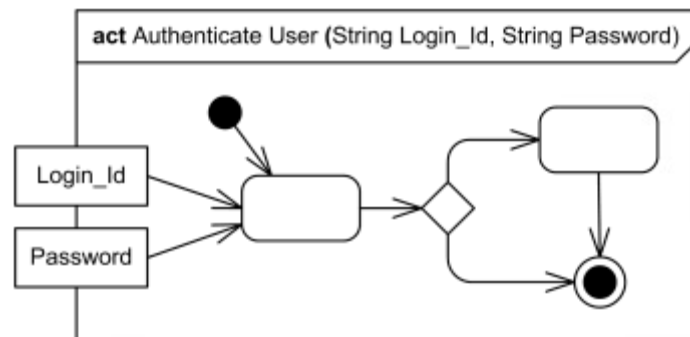
پارامترهای فعالیت، درون محدوده و در زیر نام آن به صورت فهرست با نگارشی مشابه نگارش زیر، نمایش داده می‌شوند:

parameter-name :parameter-type



فعالیت، احراز هویت کاربر، با دو پارامتر، Login Id یا شناسه ورود و Password یا رمز عبور نمایش داده شده است

مانند هر رفتاری، یک فعالیت، می‌تواند دارای پیش شرط یا پس شرط باشد. در صورت وجود، این موارد به ترتیب با کلمات کلیدی «precondition» و «postcondition» نشان داده می‌شوند. کلمه کلیدی «singleExecution» برای فعالیت‌هایی استفاده می‌شود که به صورت یک اجرای اشتراکی (singleton) اجرا می‌شوند، در غیر این صورت، هر فراخوانی در فضای خاص خود اجرا می‌شود. حاشیه فعالیت گوشه گرد ممکن است با نماد frame یا قاب در نمودارها جایگزین شود. نوع قاب در این مورد act است. پارامترهای فعالیت در صورت وجود روی قاب نمایش داده می‌شوند.



قاب فعالیت شناسایی هویت کاربر به همراه دو پارامتر Login Id یا شناسه ورود و Password یا رمز عبور

نماد کلاس‌ها با کلمه کلیدی «activity» می‌تواند برای نشان دادن مشخصات یک فعالیت بازتابی، برای نشان دادن آن یک کلاس فعالیت استفاده شود. در صورت لزوم می‌توان از نمادهای State Machine یا Association نیز استفاده کرد. UML به رفتارها اجازه می‌دهد تا توکن‌هایی تولید کنند که فعالیت هستند و به نوبه خود می‌توانند در زمان اجرا، اجرا شوند.

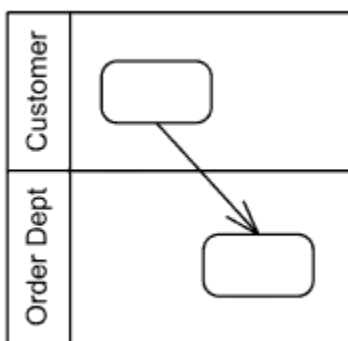
**Activity Partition** یک گروه فعالیت برای اقداماتی است که دارای برخی مشخصات مشترک هستند. پارتیشن‌ها اغلب با واحدهای سازمانی یا نقش‌های کسب‌وکاری در یک مدل کسب‌وکاری مطابقت دارند. پارتیشن‌ها یک دید محدود در مورد رفتارهای فراخوانی شده در فعالیت‌ها ارائه می‌دهند. محدودیت‌ها را می‌توان با توجه به نوع عنصری که پارتیشن مورد نظر نشان می‌دهد انتخاب کرد. محدودیت‌های زیر در UML 2.4 استاندارد شده‌اند:

- classifier یا طبقه‌بندی
- instance یا نمونه، مثال
- part یا بخش
- attribute and value یا مشخصه و مقدار

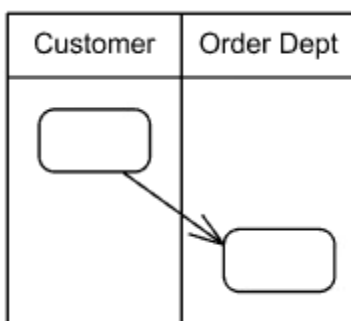
به عنوان مثال، پارتیشن‌ها می‌توانند طبقه‌بندی کننده‌های خاصی را نشان دهند. در این حالت، اقدامات در هر پارتیشن باید عملیات یا سیگنال‌هایی باشند که اشیایی را که نمونه‌هایی از طبقه‌بندی کننده مربوطه هستند، هدف قرار دهند.

یک پارتیشن ممکن است یک مشخصه و پارتیشن‌های فرعی آن، مقادیر خاص آن مشخصه را نشان دهد. به عنوان مثال، یک پارتیشن ممکن است شامل مشخصه مکانی باشد که یک رفتار در آن انجام می‌شود، و پارتیشن/پارتیشن‌های فرعی مقدار خاصی را برای آن مشخصه، مانند نیویورک، نشان دهند.

**Activity Partition** ممکن است با استفاده از نماد swimlane نشان داده شود، با دو خط معمولاً موازی، افقی یا عمودی، و نامی که پارتیشن را در یک کادر در یک انتهای آن به صورت یک برجسب نمایش می‌دهد. هر گره فعالیت، به عنوان مثال. یک Action و edge هایی که بین این خطوط قرار می‌گیرند در داخل پارتیشن در نظر گرفته می‌شوند.

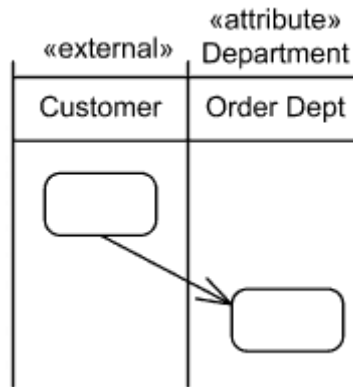


پارتیشن‌های فعالیت Customer و Order Dept به شکل swimlane افقی



## پارتیشن‌های فعالیت Order Dept و Customer به شکل swimlane عمودی

پارتیشن بندی سلسله مراتبی با استفاده از swimlanes برای پارتیشن‌های فرعی همانطور که در زیر نشان داده شده است، نمایش داده می‌شود.

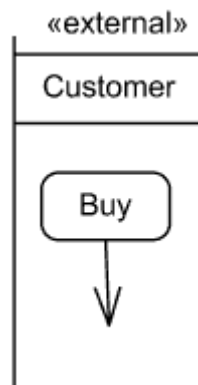


پارتیشن بندی سلسله مراتبی با پارتیشن‌های فرعی

یک پارتیشن ممکن است به عنوان یک محدوده برای پارتیشن‌های فرعی آن در نظر گرفته شود تا پارتیشن‌های فرعی را در امتداد محدوده (گروه) قرار دهد. به عنوان مثال، یک فعالیت ممکن است درون یک پارتیشن دارای محدوده قرار گیرد که رفتارهای مرتبط با یک مکان را انجام دهد و دیگری برای مقدار انجام آنها. پارتیشن‌های دارای محدوده را نمی‌توان درون هیچ پارتیشن دیگری قرار داد. نمودارها همچنین می‌توانند به صورت چند بعدی تقسیم شوند، که در آن هر سلول swim یک تقاطع از پارتیشن‌های متقاطع است. پارتیشن‌های موجود در هر محدوده ممکن است در یک پارتیشن فعالیت محصور با `isDimension=true` گروه بندی شوند که نام آن نام محدوده است. با این حال، به جای اینکه به عنوان یک پارتیشن نشان داده شود، محدوده با قرار دادن نام آن در کنار مجموعه‌ای از پارتیشن‌ها درونی، مشخص می‌شود.

پارتیشن می‌تواند یک موجودیت خارجی را نشان دهد که ساختار پارتیشن بندی برای آن اعمال نمی‌شود. پارتیشن‌های خارجی استثنای عمده از قوانین ساختار پارتیشن هستند. برای مثال، یک محدوده ممکن است دارای پارتیشن‌هایی باشد که بخش‌هایی از یک طبقه بندی ساخت یافته را نشان می‌دهد. می‌توان یک پارتیشن خارجی داشت که یکی از قسمت‌ها را نشان نمی‌دهد، بلکه یک طبقه بندی کننده کاملاً مجزا دارد. در مدل سازی کسب و کار، پارتیشن‌های خارجی را می‌توان برای مدل سازی موجودیت‌های خارج از یک کسب و کار استفاده کرد.

وقتی در نظر گرفته می‌شود که فعالیت‌ها خارج از دامنه یک مدل خاص رخ می‌دهند، پارتیشن را می‌توان با کلمه کلیدی «external» برچسب گذاری کرد. هر زمان که یک فعالیت در یک swimlane با برچسب «external» علامت گذاری می‌شود، این علامت روی swimlane و محدوده نیز قرار داده می‌شود.





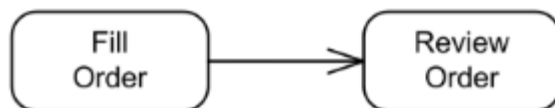
### اقدام Buy در پارتیشن خارجی Customer رخ می‌دهد

در شرایطی که نمی‌توان از swimlanes برای نمایش پارتیشن‌ها استفاده کرد، می‌توان از نماد متنی جایگزین با نام متناظر با اقدام، استفاده کرد. در این حالت نام پارتیشن در پرانتز بالای نام اقدام قرار می‌گیرد. لیست نام پارتیشن‌های محدود شده که با کاما از هم جدا شده‌اند، به این معنی است که گره مورد نظر در بیش از یک پارتیشن قرار دارد. یک دونقطه دوتایی در نام پارتیشن، نشان می‌دهد که پارتیشن تودرتو است و پارتیشن‌های بزرگتر در نام قبلی قرار می‌گیرند.



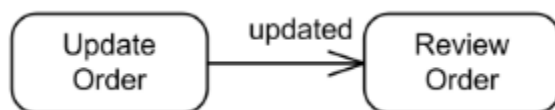
### اقدام Buy در پارتیشن خارجی Customer رخ می‌دهد

Activity Edge یک کلاس انتزاعی برای اتصالات جهت‌دار است که بین توکن‌ها یا اشیاء داده و نودهای فعالیت، هستند. این کلاس‌ها شامل control edges و object flow edges هستند. ابتدا و انتهای یک edge باید از نوع فعالیت باشد. Activity Edge با یک خط پیکان باز که دو نود فعالیت را به هم متصل می‌کند، مشخص می‌شود.



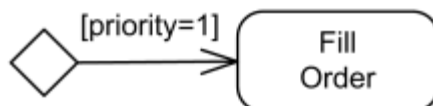
Activity edge به Fill Order و Review Order متصل می‌شود

Edge ها را می‌توان نامگذاری کرد، با این حال، Edge ها نیازی به داشتن نام‌های منحصر به فرد مجزای از نام فعالیت ندارند. اگر Edge نام دارد، در نزدیکی فلش (انتهای Edge) قرار خواهد گرفت.



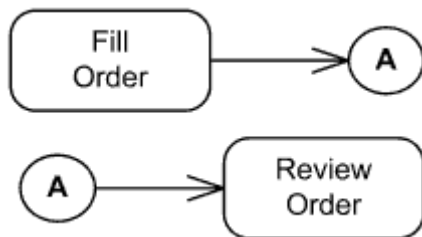
Activity Edge به نام "update" دو نود فعالیت را به هم متصل می‌کند

Activity Edge می‌تواند دارای یک عبارت محدود کننده باشد، (مشخصاتی که در زمان اجرا ارزیابی می‌شود) تا مشخص شود که آیا می‌توان از این edge عبور کرد یا خیر. عبارت محدود کننده باید برای هر توکن که بخواهد از edge عبور کند، ارزیابی شود و مقدار true در نتیجه ارزیابی حاصل شود. عبارت محدود کننده Activity Edge باید در علامت براکت قرار گیرد.



وقتی priority برابر 1 است فعالیت Fill Order اجرا می‌شود

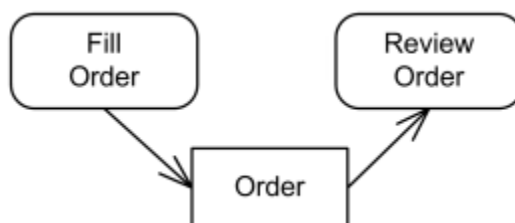
Activity Edge را می‌توان با استفاده از یک اتصال‌دهنده، که یک دایره کوچک با نامی در داخل آن است، نمایش داد. اگرچه مشخصات UML 2.4 آن را نام edge می‌نامد، اما نمادهای اتصال و مثال‌ها نشان می‌دهند که اتصال‌دهنده نام خاص خود را دارد که همچنان label نامیده می‌شود. اتصال‌دهنده‌ها معمولاً برای جلوگیری از ترسیم edge های بلند استفاده می‌شوند. این کاملاً نمادین است. استفاده از این نماد تأثیری بر مدل اصلی ندارد. دایره‌ها و خطوط ارتباطی به یک Activity Edge در مدل نگاشت می‌شوند. هر اتصال‌دهنده با یک Label مشخص باید دقیقاً با یکی دیگر با همان Label در نمودار فعالیت یکسان جفت شود. یک از اتصال‌دهنده‌ها باید دقیقاً یک edge ورودی و دیگری دقیقاً یک edge خروجی داشته باشند که هر کدام دارای یک نوع object flow یا control باشند.



اتصال دهنده A دو edge را بین Fill Order و Review Order متصل می کند

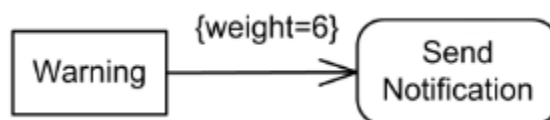
### Object Flow Edge

Object Flow Edge یال های فعالیتی هستند که برای نشان دادن جریان داده ای شیء و داده ای توکن ها بین نودهای فعالیت استفاده می شوند. یک Object Flow با یک خط پیکان مشخص می شود.



Object Flow به نام Order بین فعالیت Fill Order و Review Order قرار دارد

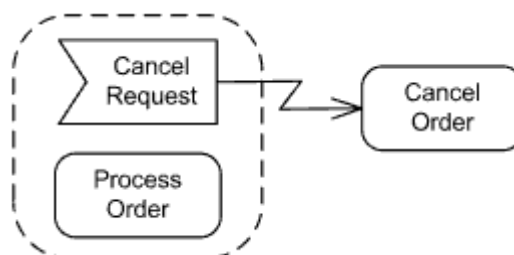
هر تعداد توکن می تواند در امتداد edge، به صورت گروهی در یک زمان یا به صورت جداگانه در زمان های مختلف عبور کند. مشخصه وزن، حداقل تعداد توکن هایی را که باید همزمان از edge عبور کنند را تعیین می کند. هنگامی که حداقل تعداد توکن ها ارائه می شود، همه توکن های موجود در ابتدای جریان به یکباره به انتهای جریان ارائه می شوند. وزن edge ممکن است در علامت آکولاد که حاوی مشخصه weight است، نشان داده شود. وزن، مشخصه ای است عددی که می تواند مقدار ثابتی داشته باشد که به ممکن است به شکل یک عدد طبیعی نامحدود غیرصفر نیز نمایش داده شود. وزن نامحدود به شکل "\*" نشان داده می شود.



وقتی تعداد اخطارها به 6 رسید، اعلامی ارسال کنید

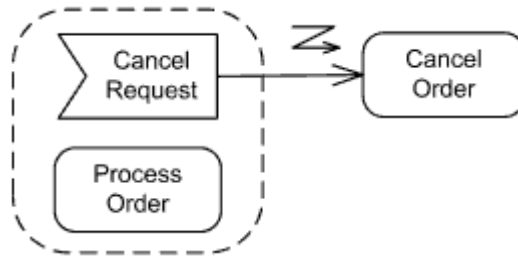
### Interrupting Edge

Interrupting Edge، یک Activity Edge برای مناطق دارای دستور وقفه می باشد. این edge به شکل یک رعد و برق نمایش داده می شود.



سیگنال Cancel Request باعث وقفه و در نتیجه فعالیت Calcel Order می شود

یک گزینه برای نمایش Interrupting Edge، ترسیم یک رعد و برق کوچک روی یک Activity Edge است.



سیگنال Cancel Request باعث وقفه و در نتیجه فعالیت Calcel Order می شود

**Action** یک عنصر نامگذاری شده است که یک مرحله اتمیک واحد را در فعالیت یا **activity** نشان می‌دهد و در داخل یک فعالیت نمی‌توان آن را به عناصر بیشتری تجزیه نمود. به عبارت دیگر فعالیت یا **activity**، نشان دهنده ترکیبی از رفتار عناصر منفردی است که **Action** نام دارند. توجه داشته باشید که فرخوانی رفتار **Action** ممکن است به فراخوانی یک فعالیت یا **activity** منجر شود. این فراخوانی‌ها برای فعالیت یا **activity** درونی آن ساده است، اما ممکن است تأثیر پیچیده‌ای داشته باشد. یک فعالیت یا **activity** رفتاری را تعریف می‌کند که می‌تواند قابل استفاده مجدد باشد. عملیات مدل‌سازی **Action** ها، در قالب فعالیت‌ها یا **activity** در UML 2.0 جایگزین **call state**، **action state**، و **subactivity state** در UML 1.5 می‌شوند. **Action** ها به صورت مستطیل‌های گوشه گرد مشخص می‌شوند. نام یا شرح **Action** در داخل مستطیل قرار می‌گیرد.

Process  
Order

### Action به نام Process Order

نام **Action** معمولاً فعل یا اسم عمل مورد نظر است که با توضیحی همراه می‌شود. از نام **State** ها به عنوان نام **Action** استفاده نکنید. چند نمونه نام **Action** در زیر آورده شده است:

- Fill Order (پر کردن درخواست سفارش)
- Review Document (بازبینی و بررسی سند)
- Enroll in Course (ثبت نام در دوره)
- Checkout (پرداخت/دریافت یا به عبارت دیگر نهایی‌سازی)
- Show Error Page (نمایش صفحه خطا)

**Action** همچنین می‌تواند در برخی از زبان‌های عمل‌گرای وابسته به برنامه، به صورت درونی بیان شود.

```
for (Account a: accounts)
    a.verifyBalance();
end_for
```

نمونه‌ای از **Action** که با زبان عمل‌گرای وابسته به ابزار بیان شده است

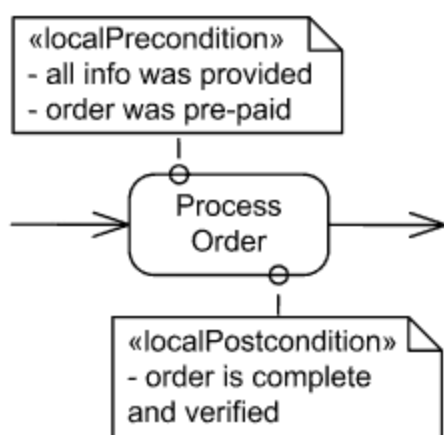
یک **Action** ممکن است مجموعه‌ای از **edge** های ورودی و خروجی داشته باشد که جریان کنترل و جریان داده شیء از و به سایر نودها را مشخص می‌کند. یک **Action** تا زمانی که تمام شرایط ورودی آن برآورده نشود، اجرا نمی‌شود. تکمیل اجرای یک **Action** ممکن است امکان اجرای مجموعه‌ای از نودها و **Action** های در جریان را فراهم کند که ورودی‌های خود را از خروجی‌های **Action** مورد نظر می‌گیرند.

اگر در حین اجرای یک **Action**، استثنایی رخ دهد، اجرای آن **Action** رها می‌شود و خروجی قابل انتظاری از این **Action** ایجاد نمی‌شود. ولی اگر **Action** دارای یک کنترل کننده استثنا باشد، شیء استثنا را به عنوان یک توکن دریافت می‌کند. اگر **Action** کنترل کننده استثنا نداشته باشد، استثنا به اولین لایه نود محصور کننده منتشر می‌شود و به همین ترتیب اگر لایه مورد نظر نیز کنترل کننده استثنا نداشته باشد تا زمانی که توسط یکی از این لایه‌ها، مدیریت نشود، به لایه محصور کننده بیرونی‌تر منتشر می‌شود. اگر یک استثنا از یک نود تودرتو منتشر

شود (Action، نود فعالیت یا activity ساخت یافته، یا فعالیت یا activity)، همه توکن‌ها در نود تودرتو خاتمه می‌یابند. داده‌ای که یک استثناء را توصیف می‌کند به عنوان یک شیء توکن از هر کلاس داده معرفی می‌شود.

پیش شرط‌ها و پس شرط‌های محلی (pre-conditions و post-conditions) محدودیت‌هایی هستند که به ترتیب باید هنگام شروع و تکمیل اجرای یک Action، محقق شده باشند. آنها فقط در نقطه‌ای از جریان که مشخص شده، محقق شوند، بررسی می‌شوند، و اگر Action مورد نظر در مکان‌های دیگر و یا در نمودارهای دیگر استفاده شود، فراخوانی آن نیاز به ارزیابی این پیش شرط‌ها و پس شرط‌های محلی ندارد.

نحوه اجرای پیش و پس شرایط محلی در زمان اجرا تعیین می‌شود. به عنوان مثال، violation ها ممکن است در زمان کامپایل یا زمان اجرا شناسایی شوند. اثر مورد نظر ممکن است خطایی باشد که اجرا را متوقف کند یا فقط یک هشدار و یا مورد دیگر باشد. پیش‌شرط‌های محلی و شرایط پس‌شرط محلی به ترتیب با کلیدواژه‌های «localPrecondition» و «localPostcondition» به عنوان یادداشت‌های پیوست به فراخوان نشان داده می‌شوند.



پیش و پس شرایط محلی به عنوان یادداشت‌های پیوست شده به Action به نام Process Order نشان داده شده است

زیر کلاس‌های Action در زیر فهرست شده‌اند. توجه داشته باشید که Action های شیء و Action های رویداد اضافه شده‌اند که به صراحت در مشخصات UML 2.4 تعریف نشده‌اند:

- **object action** یا Action شیء (در استاندارد UML به صراحت ذکر نشده است)
- **variable action** یا Action متغیر
- **invocation action** یا Action فراخوانی
- **raise exception action** یا Action ایجاد استثناء
- **structural feature action** یا Action امکان ساختاری
- **link action** یا Action پیوند
- **event action** یا Action رویداد (در استاندارد UML به صراحت ذکر نشده است)
- **opaque action** یا Action غیر شفاف

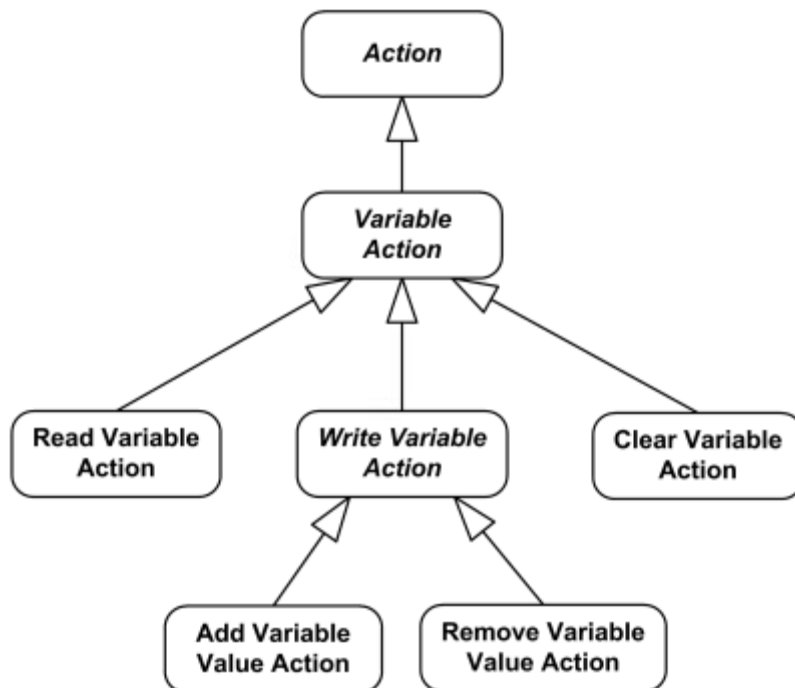
Object Action شامل اعمال مختلفی بر روی اشیاء می‌شود، به عنوان مثال، ایجاد (create) و نابود کردن (destroy) شیء، تست شناسایی (identity) شیء، تعیین مقدار، و غیره. Object Action به صراحت توسط استاندارد UML تعریف نشده است. در استاندارد UML، تمام Object Action ها بی واسطه، زیر کلاس‌های Action هستند. Object Action شامل موارد زیر هستند:

- create object action (ایجاد شیء)
- destroy object action (از بین بردن شیء)
- test identity action (آزمایش هویت)
- read self action (فراخوانی خود)
- value specification action (تعیین مقدار)
- start classifier behavior action (آغاز رفتار طبقه‌بندی)
- read is classified object action (خواندن یک شیء طبقه‌بندی شده)
- reclassify object action (طبقه‌بندی مجدد شیء)
- read extend action (خواندن گسترش)

## Variable Action

Variable Action شامل موارد زیر هستند:

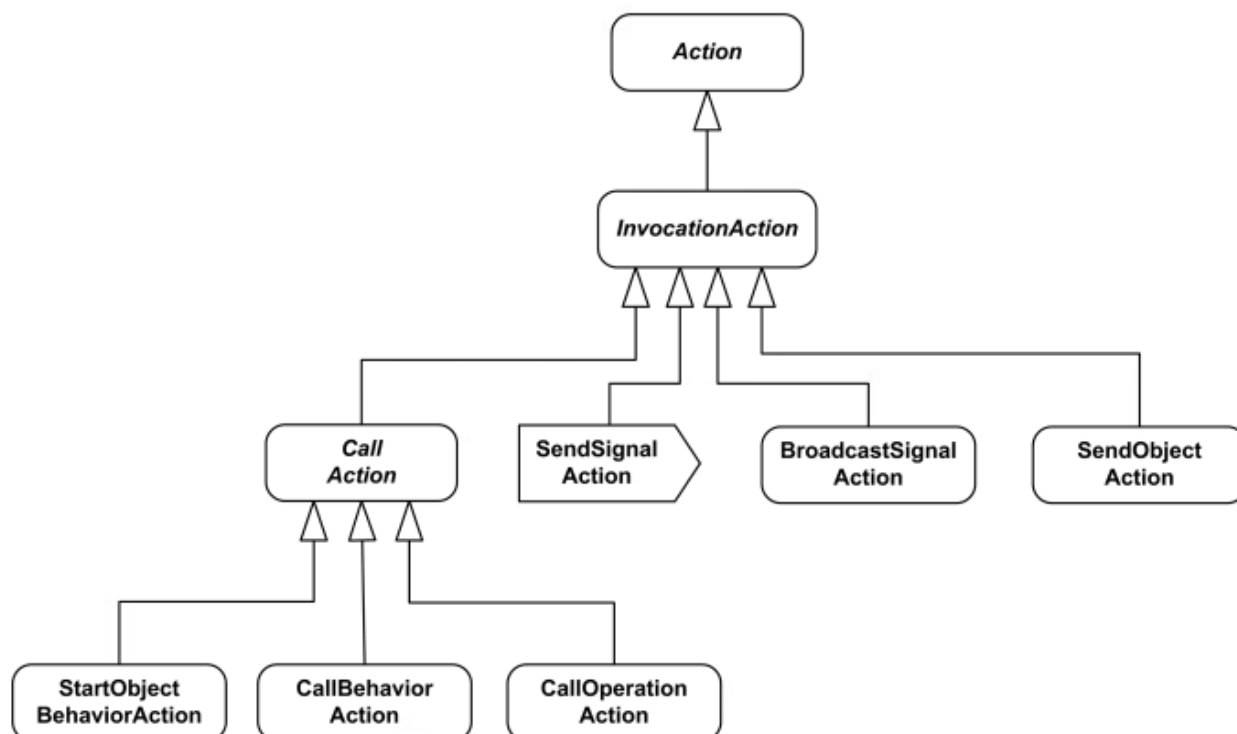
- read action (خواندن متغیر)
- write action (نوشتن متغیر)
- add action (افزودن متغیر)
- remove action (حذف متغیر)
- clear action (پاک کردن متغیر)



نمای کلی از Variable Action ها

## Invocation Action

Invocation Action شامل چند نوع *call action* (عملیات فراخوانی)، *signal send* (ارسال سیگنال) و *broadcast* (پخش سیگنال) و *send object* (ارسال شیء) می‌باشد.





## نمای کلی Invocation action ها

### Call Behavior Action

Call Behavior Action یک عمل فراخوانی است که به جای فراخوانی Action که رفتار را فراخوانی می‌کند، مستقیماً یک رفتار را فراخوانی می‌کند. پارامترها را می‌توان به واسطه Action که رفتار را فراخوانی می‌کند، منتقل کرد. تعداد پین‌های آرگومان (argument pin) و تعداد پارامترهای ورودی و خروجی رفتار مورد نظر باید برابر باشد. همچنین باید تعداد پین‌های نتیجه (result pin) و تعداد پارامترهای بازگشتی رفتار، خارج و در خارج برابر باشد.

برای فراخوانی همزمان یا synchronous call، اجرای Call Behavior Action، منتظر می‌ماند تا اجرای رفتار فراخوانی شده کامل شود. اجرای Call Action تا زمانی که پاسخی دریافت نکند مسدود می‌شود. پاسخ دریافت شده شامل مقادیری پارامترهای بازگشتی، خروجی یا ورودی است. مقادیر نتیجه، روی پین‌های نتیجه (result pin) Call Behavior Action قرار می‌گیرند و پس از آن اجرای Action به پایان می‌رسد. اگر اجرای رفتار فراخوانی شده یک استثناء ایجاد کند، استثناء به Call Behavior Action ارسال می‌شود تا جستجو برای کنترل کننده استثناء آغاز شود. اگر فراخوانی ناهمزمان یا asynchronous باشد، عمل فراخوانی بلافاصله پس از شروع رفتار، کامل می‌شود. هر مقدار بازگشتی یا خروجی از رفتار فراخوانی شده پس داده نمی‌شود.

Call Behavior Action به صورت عملی با نام رفتار مورد نظر، نشان داده می‌شود که با عمل یا توصیف رفتاری که در داخل مستطیل گوشه گرد Action قرار می‌گیرد، انجام می‌شود. اگر نام نود با نام رفتار متفاوت باشد، به جای آن در نماد ظاهر می‌شود.

Checkout

### Call behavior action برای رفتار Checkout

توجه داشته باشید، از آنجایی که دقیقاً شبیه به عملیات رایج (common action) است، راهی وجود ندارد که فقط با نگاه کردن به نمودار بگوییم که آیا نام مورد نظر، نام عملیات رایج (common action) است، یا نام call behavior action یا نام behavior است.

همانطور که می‌دانید، برخی از زیر کلاس‌های behavior عبارتند از:

- interaction
- state machine
- activity

call behavior action با قرار دادن یک نماد شنکس در نماد Action نشان داده می‌شود. نماد شنکس شبیه یک شکل سلسله مراتب مینیاتوری است، که نشان می‌دهد این فراخوانی، Action دیگری را آغاز می‌کند. توجه داشته باشید که اگرچه UML این نماد را ارائه می‌دهد، هیچ فعالیت فراخوانی را به صورت رسمی در مشخصات UML معرفی نمی‌نماید.

User Authentication

### نمایشی از فراخوانی فعالیت User Authentication

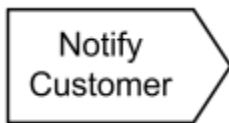
یک نماد جایگزین برای فعالیت فراخوانی شده، نشان دادن محتوای فعالیت فراخوانی شده در یک مستطیل بزرگ با گوشه گرد است. edge هایی که در فراخوانی جریان دارند به object node ها پارامتر در فعالیت فراخوانی شده متصل می‌شوند. object node ها پارامتر در مرز فعالیت فراخوانی شده نشان داده می‌شوند. پیش شرط‌ها و پس شرط‌های رفتار مورد نظر را می‌توان با استفاده از کلمات کلیدی «precondition» و «postcondition» نشان داد.

### Send Signal Action

Send Signal Action یک عمل فراخوانی است که سیگنالی را به واسطه ورودی‌های خود ایجاد می‌کند و آن را به شیء هدف مورد نظر، که ممکن است باعث آغاز یک state machine و یا اجرای یک Activity شود، ارسال می‌کند. هنگامی که تمام پیش نیازهای اجرای Action برآورده شد، از آرگومان‌ها، یک سیگنال تولید می‌شود و به شیء هدف مورد نظر، منتقل می‌شود. شیء مورد نظر ممکن است محلی باشد یا یک شیء از راه دور باشد. اگر ورودی Action از قبل یک سیگنال است، باید از send object action به جای آن استفاده شود. فرستنده سیگنال (با نام مستعار "requestor" یا درخواست کننده) بلافاصله اجرا را ادامه می‌دهد، بدون اینکه منتظر هیچ پاسخی باشد. Send Signal Action هیچ پاسخی از رفتار فراخوانی شده دریافت نمی‌کند. هر گونه تلاش برای پاسخ، به سادگی نادیده گرفته می‌شود و هیچ انتقالی به فرستنده انجام نمی‌شود. نحوه ارسال سیگنال، مدت زمان لازم برای ارسال آن، ترتیب رسیدن ارسال‌ها به اهداف مختلف و مسیر رسیدن به اهداف در نمونه سیگنال ممکن است در حین انتقال کپی شود، بنابراین ممکن است هویت حفظ نشود.

هنگامی که یک انتقال به یک شیء هدف می‌رسد، ممکن است رفتاری را در شیء مورد نظر فراخوانی کند. اثر دریافت یک شیء سیگنال در رفتارهای رایج مشخص شده است. چنین اثراتی شامل اجرای Activity ها و آغاز به کار انتقال state machine می‌شود.

Send Signal Action به صورت پنج ضلعی محدب مشخص می‌شود. توجه داشته باشید که نام Action با نام کلاس سیگنالی که ارسال می‌کند، مطابقت دارد. شیء هدف با این نماد مشخص نشده است.



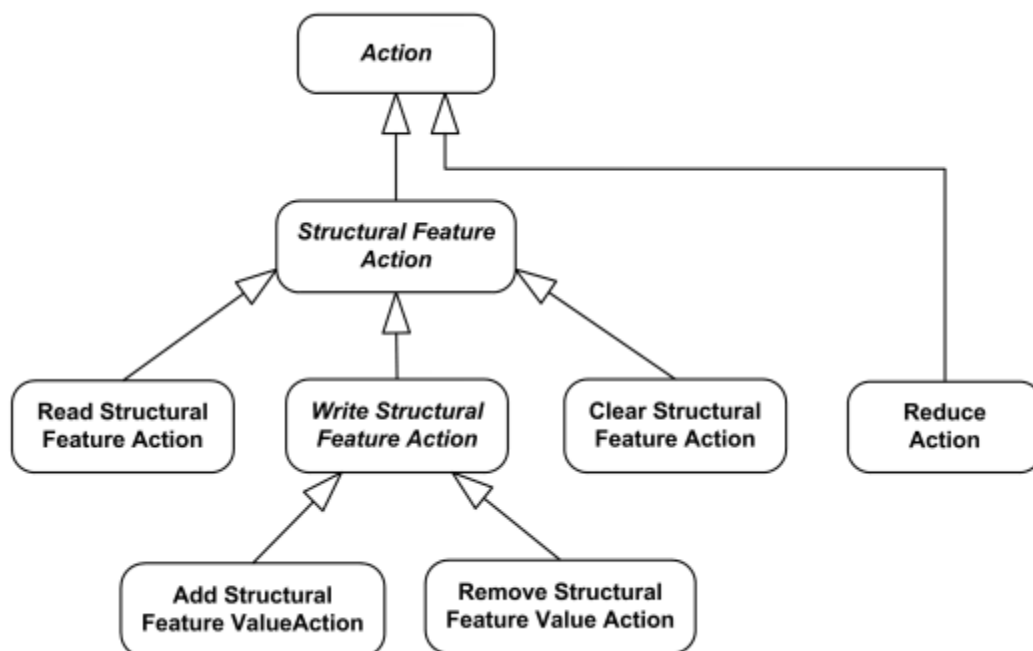
Notify Customer به نام Send Signal Action

سیگنال Notify Customer را ایجاد و ارسال می‌کند

### Structural Feature Action

Structural Feature Action شامل چندین عمل است که روی ساختارهای ترکیبی کار می‌کنند. این اعمال شامل موارد زیر می‌باشند:

- **read** یا خواندن
- **write** یا نوشتن
- **add** یا اضافه کردن
- **remove** یا حذف کردن
- **clear** یا پاک کردن
- **reduce** یا کاهش دادن



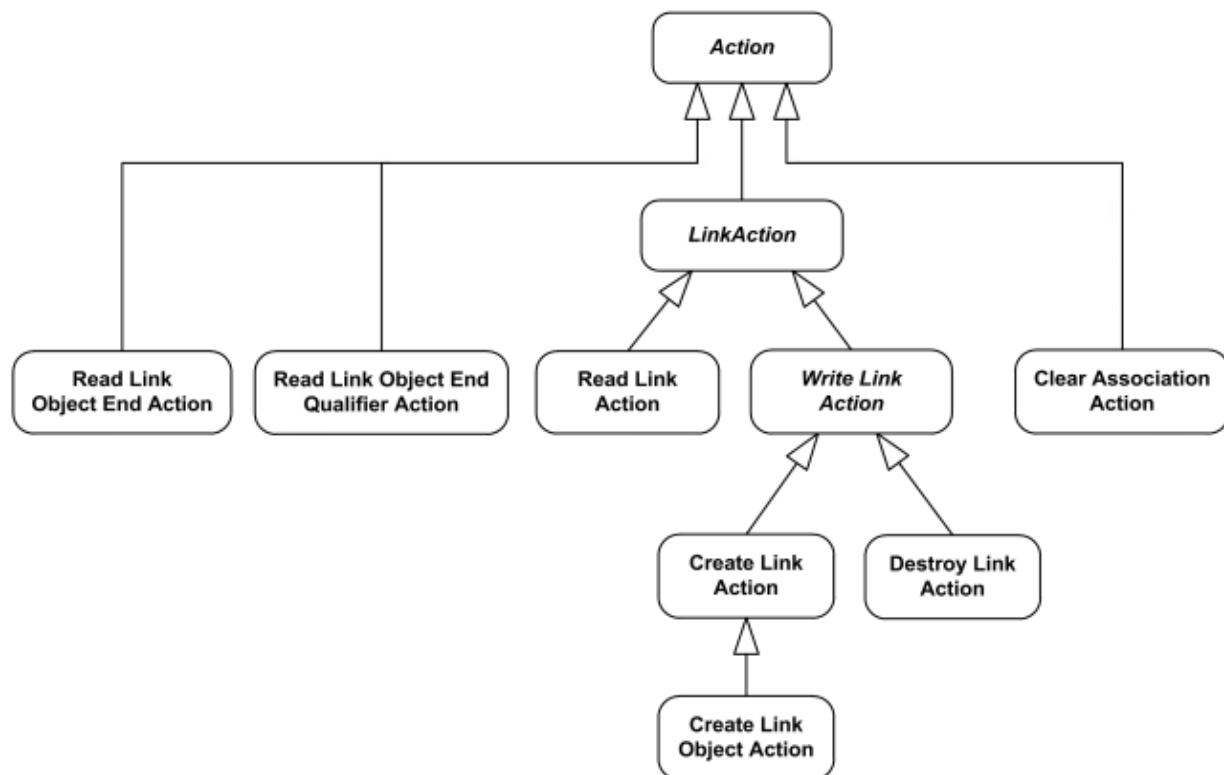
نمای کلی Structural Feature Action ها

توجه داشته باشید که در مشخصات UML 2.4 عمل **reduce** یا کاهش، مستقیماً از زیرکلاس Action مشتق شده است

### Link Action

Link Action شامل چندین Action است که روی پیوندها کار می‌کنند. این Action ها شامل موارد زیر است:

- read link یا خواندن پیوند
- write link یا نوشتن پیوند
- create link یا ایجاد پیوند
- destroy link یا تخریب کردن پیوند
- clear association یا پاک کردن اجتماع



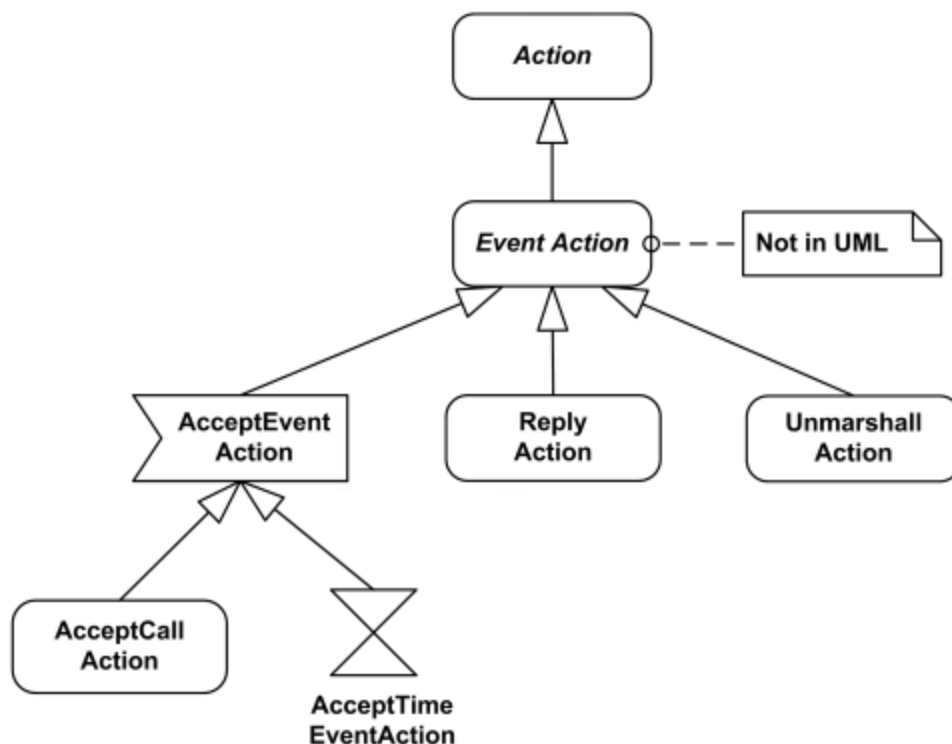
نمای کلی Link Action ها

## Event Action

Event Action شامل موارد زیر است:

- accept event action یا پذیرش رویداد
- accept call action یا پذیرش فراخوانی
- accept time event action یا پذیرش رویداد زمانی
- reply action یا پاسخ
- unmarshall action یا هدایت کننده خودکار سازگاری

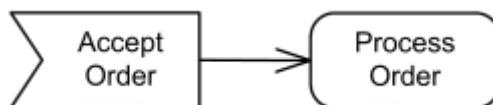
توجه داشته باشید که Event Action بخشی صریح از مشخصات UML نیست، در UML 2.4 همه Event Action ها زیر کلاس های مستقیم Action هستند. برای توضیح Event Action در اینجا اضافه شده اند.



نمای کلی Event Action ها

#### Accept Event Action

Accept event action، Action است که منتظر وقوع یک رویداد خاص است. این Action پیام‌های ناهمزمان (asynchronous message)، شامل فراخوانی‌های ناهمزمان (asynchronous call) را مدیریت می‌کند. نمی‌توان از آن برای فراخوانی‌های همزمان (synchronous call) استفاده کرد (به جز Action فراخوانی پذیرش). Accept event action در UML 2.0 معرفی شد. event action رویدادهای شناسایی شده توسط شیء صاحب رفتار اجرا را کنترل می‌کند. رویدادها، در برخی از صف‌ها توسط شیء، ذخیره می‌شوند. ترتیب رویدادهای شناسایی شده در مشخصات UML تعریف نشده است، اما می‌تواند در برنامه‌های extension یا profile ها مشخص شود. اگر Accept event action اجرا شود و رویداد شناسایی شده شیء مطابق با یکی از محرک‌های Action باشد، Accept event action مقداری را که توصیف کننده رویداد است، در خروجی ارائه می‌دهد. اگر رویداد مورد نظر با رویداد مورد انتظار مطابقت نداشته باشد، Action منتظر رویداد بعدی می‌ماند. در یک سیستم همزمان، چندین Action یا رفتارهای دیگر ممکن است برای یک رویداد موجود، در رقابت باشند. فقط یک Action، رویداد را می‌پذیرد، مگر اینکه توسط یک برنامه extension یا profile به‌طور دیگری مشخص شده باشد، حتی اگر این رویداد چندین Action را به طور همزمان اجرا کند. یک Accept event action به عنوان یک پنج ضلعی مقعر نشان داده می‌شود. به‌طور پیش فرض، نام Accept event action با نام رویدادی که این عمل می‌پذیرد مطابقت دارد، به‌طور مثال. عمل پذیرش سفارش، Accept event action رویداد پذیرش سفارش است.



پذیرش رویداد "پذیرش درخواست سفارش" توسط Accept event action به نام Accept Order باعث عمل فراخوانی یک Action به نام Process Order می‌شود.

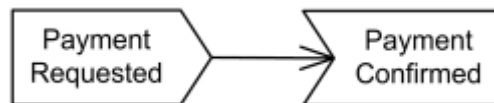
### Accept event action پس از ورود به فعالیت حاوی آن فعال می‌شود

اگر یک Accept event action فاقد edge های ورودی باشد، در آن صورت، زمانی شروع می‌شود که Activity حاوی مورد نظر یا نود ساخت یافته، هر کدام که فوراً حاوی Action باشد، زودتر اقدام کنند. علاوه بر این، یک Accept event action بدون edge های ورودی پس از پذیرش یک رویداد فعال شده، باقی می‌ماند. پس از پذیرش یک رویداد و خارج کردن یک مقدار، کار خاتمه نمی‌یابد، و همچنان منتظر رویدادهای دیگر است. این معنایی از یک استثناء در قوانین اجرای عادی در Activity ها است. یک Accept event action بدون edge های ورودی و حاوی یک نود ساخت یافته، زمانی خاتمه می‌یابد که دربرگیرنده آن خاتمه یابد.

### Accept Signal Action

Accept signal action نام غیر رسمی برای accept event action است که فعال کننده آن یک رویداد سیگنال است و وابسته است به ارسال سیگنال.

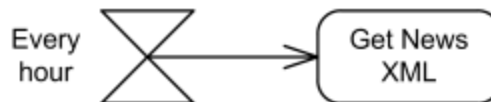
یک Accept signal action مانند Accept event action، به شکل یک پنج ضلعی مقعر نمایش داده می‌شود. به طور پیش‌فرض، نام Accept signal action با نام سیگنالی که این Action می‌پذیرد، مطابقت دارد.



سیگنال Payment Requested ارسال می‌شود سپس Activity منتظر می‌ماند تا سیگنال Payment Confirmed را دریافت کند. سیگنال Payment Confirmed تنها پس از ارسال Payment Requested فعال می‌شود.

### Wait Time Action

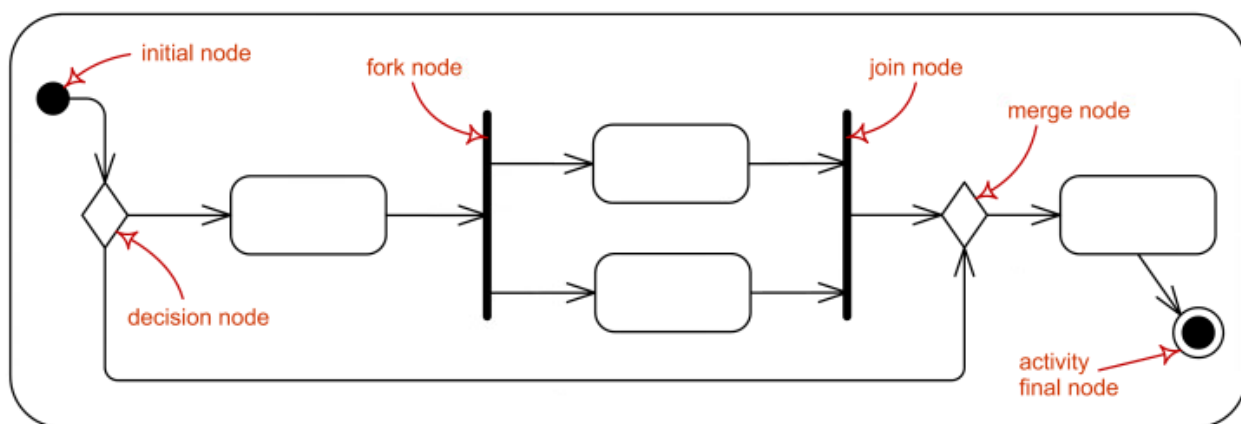
اگر رویداد مورد نظر یک رخداد رویداد زمانی باشد، نتیجه آن شامل زمانی است که در آن رخداد، رخ داده است. چنین عملی به طور غیررسمی، Wait time action نامیده می‌شود. Accept time event action (معرفی غیررسمی: Wait time action) با نمادی شبیه به یک ساعت شنی مشخص شده است.



Accept time event action به نام Every Hour هر ساعت خروجی تولید می‌کند. هیچ edge ورودی برای این time event action وجود ندارد، بنابراین تا زمانی که Activity دربرگیرنده Action مورد نظر یا نود ساخت یافته آن فعال باشد، فعال است.

نود Control یک نود Activity است که برای هماهنگ کردن جریان‌ها بین نودهای دیگر استفاده می‌شود. آن نودها شامل موارد زیر هستند:

- **initial node** یا نود آغازگر
- **flow final node** یا نود پایان جریان
- **activity final node** یا نود پایان فعالیت
- **decision node** یا نود تصمیم‌گیری
- **merge node** یا نود ادغام‌کننده
- **fork node** یا نود چند شاخه‌کننده
- **join node** یا نود هم‌بند‌کننده



نمای کلی نودهای کنترل Activity

نودهای کنترل Activity را می‌توان هم در activity diagram و هم در interaction overview diagram استفاده کرد.

### Initial Node

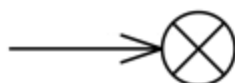
**Initial Node** یک نود کنترلی است که با فراخوانی Activity، جریان درون آن شروع می‌شود. هنگامی که Activity شروع می‌شود، یک توکن کنترلی در Initial Node قرار می‌گیرد، اما در Initial Node ها باقی نمی‌مانند بلکه در نودهای ساخت یافته Activity قرار می‌گیرند. توکن‌های یک Initial Node بلافاصله به تمام edge های خروجی آن ارائه می‌شوند. برای راحتی کار اینگونه تصور کنید، Initial Node ها یک استثنا از قاعده پذیرفتن توکن هستند و به عبارت دیگر نودهای کنترلی نمی‌توانند توکن‌ها را نگه دارند، مگر آنکه به‌عنوان مثال توسط عبارت‌های محدود کننده از حرکت در پایین‌دست مسدود شوند. به عبارت دیگر در یک جریان از یک Activity، نودهای کنترلی مثل Initial Node یک state تصور نمی‌شوند. Activity ها ممکن است بیش از یک Initial Node داشته باشند. در این حالت، فراخوانی این گونه Activity، جریان‌های متعددی را که هر کدام دارای یک Initial Node هستند، آغاز می‌کند. توجه داشته باشید که جریان‌ها می‌توانند از نودهای دیگر نیز شروع شوند، بنابراین Initial Node ها برای شروع یک Activity الزامی نیستند. Initial Node ها به صورت یک دایره کوچک توپر نشان داده می‌شوند.



Activity در یک Initial Node

### Flow Final Node

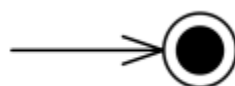
**Flow Final Node** یک نود نهایی کنترلی است که یک جریان را خاتمه می‌دهد. تمام توکن‌هایی که به آن می‌رسند از بین می‌روند اما هیچ تاثیری بر سایر جریان‌های **Activity** ندارد. **Flow final** در **UML 2.0** معرفی شد. نماد **Flow Final Node** دایره کوچکی است که یک علامت X داخل آن است.



Flow Final Node

### Activity Final Node

**Activity Final Node** یک نود نهایی کنترلی است که تمام جریان‌های یک **Activity** را متوقف می‌کند. **Activity Final** در **UML 2.0** معرفی شد. یک **Activity** ممکن است بیش از یک **Activity Final Node** داشته باشد. اولین موردی که جریان به آن می‌رسد، تمام جریان‌های فعالیت را متوقف می‌کند. توکنی که به **Activity Final Node** می‌رسد، **Activity** را خاتمه می‌دهد. به طور خاص، تمام **Action** های در حال اجرا در **Activity** را متوقف می‌کند و تمام توکن‌های درون نودهای شی، را به جز نودهای پارامتر فعالیت خروجی، از بین می‌برد. خاتمه، اجرای **Action** های فراخوانی شده همزمان را با هر رفتاری که منتظر بازگشت آن هستند، خاتمه می‌دهد. هر رفتاری که به طور ناهمزمان توسط **Action** فراخوانی شود تحت تأثیر قرار نمی‌گیرد. اگر نمی‌خواهید همه جریان‌ها را در فعالیت خاتمه دهید، به جای **activity final** از **flow final** استفاده کنید. **Activity Final Node** به صورت یک دایره توپر با یک دایره توخالی در داخل نشان داده می‌شوند. می‌توان آن را به شکل یک سیبیل هدفگیری در نظر گرفت.

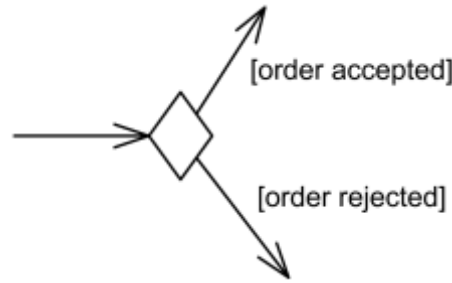


Activity Final Node

### Decision Node

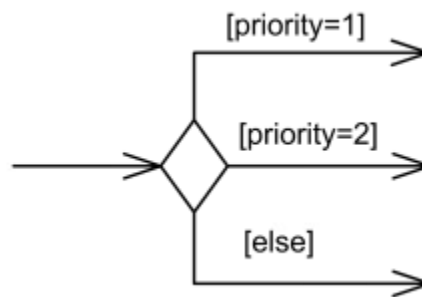
**Decision Node** یک نود کنترلی است که توکن‌ها را در یک یا دو **edge** ورودی می‌پذیرد و یک **edge** خروجی را از یک یا چند جریان خروجی انتخاب می‌کند. **Decision Node** در **UML** برای پشتیبانی از شرط‌ها در **Activity** ها معرفی شدند. **edge** هایی که به یک **Decision Node** وارد می‌شوند و یا از آن خارج می‌شوند، به غیر از جریان ورودی تصمیم‌گیری (در صورت وجود)، باید همه یا جریان‌های شیء باشند یا همه جریان‌های کنترلی باشند. هر توکنی که به یک **Decision Node** می‌رسد می‌تواند تنها یک **edge** خروجی را طی کند. توکن‌ها نمی‌توانند تکراری باشند. هر توکن ارائه شده توسط **edge** ورودی به **edge** های خروجی ارائه می‌شود. اینکه کدام یک از **edge** ها ناقل جریان خواهند بود، بستگی به ارزیابی عبارات محدود کننده در **edge** های خروجی دارد. ترتیب ارزیابی عبارات محدود کننده، تعریف نشده است، یعنی نباید به هیچ ترتیب توصیف تصویری یا متنی اعتماد کنیم. نماد **Decision Node** یک نماد الماس شکل است.





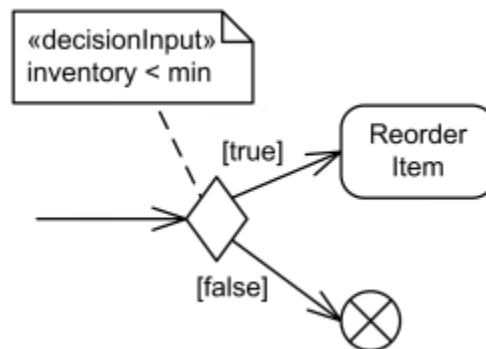
Decision Node با دو edge خروجی به همراه عبارات محدود کننده

مدلساز باید ترتیبی دهد که هر توکن فقط برای عبور از یک edge خروجی انتخاب شود. برای نقاط تصمیم‌گیری یا decision point ها، عبارت محدوده کننده از پیش تعریف شده "else" ممکن است حداکثر برای یک edge خروجی تعریف شود.



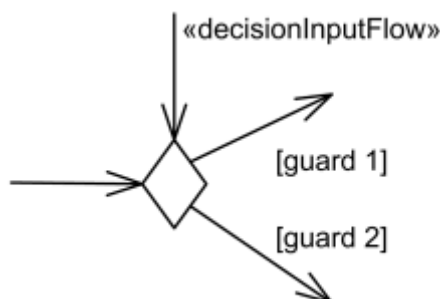
Decision Node با سه edge خروجی دارای عبارت محدود کننده که یکی از عبارات محدود کننده [else] است

تصمیم‌گیری یا Decision می‌تواند رفتار ورودی تصمیم‌گیری داشته باشد. رفتارهای ورودی تصمیم‌گیری در UML برای جلوگیری از محاسبات مجدد اضافی در عبارات محدود کننده، معرفی شدند. در این مورد، قبل از ارزیابی عبارات محدود کننده در edge های خروجی، هر داده توکن به رفتار ارسال می‌شود. رفتار مورد نظر، بدون ورودی برای توکن‌های کنترل، فراخوانی می‌شود. خروجی رفتار مورد نظر، برای هر عبارت محدود کننده در دسترس است. از آنجایی که این رفتار در طول فرآیند ارائه توکن‌ها به edge های خروجی استفاده می‌شود، ممکن است قبل از پذیرش توکن توسط آن edge ها، بارها روی همان توکن اجرا شود. این بدان معناست که این رفتار نمی‌تواند عوارض جانبی داشته باشد. رفتار ورودی تصمیم‌گیری با کلمه کلیدی «DecisionInput» مشخص می‌شود و برخی رفتار یا شرایط تصمیم‌گیری در نماد یادداشت (note) قرار می‌گیرند و به Decision Node مناسب متصل می‌شوند.



Decision Node با رفتار ورودی تصمیم‌گیری

تصمیم‌گیری ممکن است جریان ورودی تصمیم‌گیری نیز داشته باشد. در این مورد، توکن‌های ارائه شده در جریان ورودی تصمیم‌گیری که در هر edge خروجی در دسترس عبارت محدود کننده قرار می‌گیرند، تعیین می‌کنند که آیا پیشنهاد در edge ورودی معمولی در امتداد آن edge خروجی ارسال می‌شود یا خیر. یک جریان ورودی تصمیم‌گیری با کلمه کلیدی «decisionInputFlow» مشخص می‌شود که آن جریان را توضیح می‌دهد.

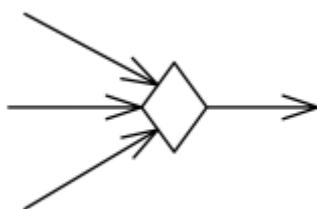


Decision Node با جریان ورودی تصمیم‌گیری

اگر هم یک رفتار ورودی تصمیم‌گیری و هم جریان ورودی تصمیم‌گیری وجود داشته باشد، توکن ارائه شده در جریان ورودی تصمیم‌گیری به رفتار ورودی تصمیم‌گیری منتقل می‌شود (این در حالی است که آرگومان اول یک edge ورودی معمولی از نوع جریان کنترل است، و آرگومان دوم یک جریان شیء خواهد بود). Decision Node ها با جریان ورودی تصمیم‌گیری اضافی، تنها زمانی توکن‌ها را به edge های خروجی ارائه می‌کنند که یک توکن در هر edge ورودی ارائه شود.

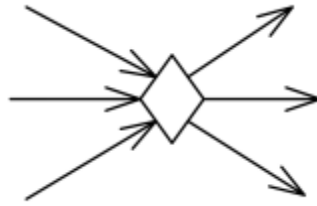
## Merge Node

Merge Node یک نود کنترلی است که چندین جریان متناوب ورودی را برای پذیرش جریان خروجی واحد گرد هم می‌آورد. هیچ پیوستن توکن‌ها وجود ندارد. ادغام نباید برای همگام‌سازی جریان‌های همزمان استفاده شود. به عنوان مثال، اگر تصمیم‌گیری پس از یک چندشاخه کننده (fork) استفاده شود، دو جریانی که از تصمیم‌گیری خارج می‌شوند باید قبل از رفتن به یک اتصال در همدیگر ادغام شوند. در غیر این صورت، اتصال برای هر دو جریان، منتظر می‌ماند تا فقط یکی از آنها برسد. تمام edge هایی که وارد و خارج به/از یک Merge Node می‌شوند باید یا جریان‌های شیء یا جریان‌های کنترلی باشند. نماد Merge Node نمادی به شکل الماس است که دو یا چند edge وارد آن می‌شود و یک edge فعالیت واحد از آن خارج می‌شود.



Merge Node سه edge ورودی را برای یک edge خروجی ادغام می‌کند

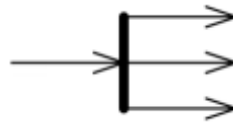
عملکرد Merge Node و Decision Node را می‌توان با استفاده از همان نماد نود، همانطور که در زیر نشان داده شده است، ترکیب کرد. این مورد، مدلی است که شامل یک Merge Node با تمام edge های ورودی و یک edge خروجی که به یک Decision Node که تمام edge های خروجی در آن ترسیم شده است، نگاشت می‌شود.



Decision Node و Merge Node که مشترکا در یک نماد قرار دارند

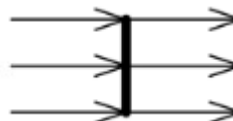
## Fork Node

**Fork Node** یک نود کنترلی است که دارای یک **edge** ورودی و چندین **edge** خروجی است و برای تقسیم جریان ورودی به چند جریان همزمان استفاده می‌شود. **Fork Node** ها برای پشتیبانی از موازی سازی در **Activity** ها معرفی شده‌اند. در مقایسه با UML 1.5، **fork** های **Activity** در UML 2.0 موازی سازی نامحدود را مدل می‌کنند. توکن‌هایی که به یک **fork** می‌رسند برای **edge** های خروجی کپی می‌شوند. اگر حداقل یک **edge** خروجی، توکن را بپذیرد، کپی‌هایی از توکن ساخته می‌شود و یک کپی از هر **edge** که توکن را می‌پذیرد، عبور می‌کند. **edge** های خروجی که به دلیل عدم پذیرش توکن توسط اهداف خود، آن را قبول نکردند، کپی خود را در یک صف ضمنی FIFO نگه می‌دارند تا زمانی که مورد پذیرش قرار گیرد. بقیه **edge** های خروجی، توکن دریافت نمی‌کنند. نماد **Fork Node** یک پاره خط است که یک **edge** از **Activity** وارد آن می‌شود و دو یا چند **edge** از آن خارج می‌شوند.



**Fork Node** با یک **edge** مربوط به **Activity** که وارد آن می‌شود و سه **edge** که از آن خارج می‌شود

عملکرد **join node** و **fork node** را می‌توان با استفاده از یک نماد نود مشابه، ترکیب کرد. این مورد، مدلی است که شامل یک **join** **node** با تمام **edge** های ورودی و یک **edge** خروجی که به یک **fork node** با تمام **edge** های خروجی موجود آن متصل شده است، نگاشت می‌شود.

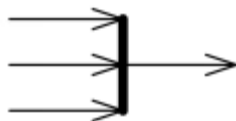


ترکیبی از **join node** و **fork node**

اگر عبارات محدود کننده در **edge** های خروجی از **fork** استفاده می‌شوند، مدل‌سازان باید اطمینان حاصل کنند که هیچ اتصال پایین‌دستی به ورود توکن‌هایی که از **edge** با عبارت محدود کننده عبور می‌کنند، بستگی ندارد. اگر نمی‌توان از آن اجتناب کرد، باید یک **Decision Node** برای داشتن یک عبارت محدود کننده معرفی شود تا در صورت رد این عبارت محدود کننده، توکن را به اتصال پایین‌دستی منتقل کند.

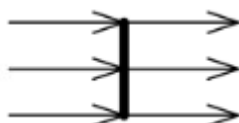
## Join Node

**Join Node** یک نود کنترلی است که دارای چندین **edge** ورودی و یک **edge** خروجی است و برای همزمان سازی جریان‌های همزمان ورودی استفاده می‌شود. **Join Node** برای پشتیبانی از موازی سازی در فعالیت‌ها معرفی شده‌اند. نماد یک **Join Node** یک پاره خط است که چندین **edge** مربوط به **Activity** وارد آن می‌شود و تنها یک **edge** از آن خارج می‌شود.



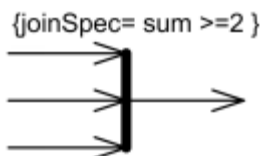
Join Node که سه edge مربوط به Activity وارد آن می‌شود و یک edge از آن خارج می‌شود

عملکرد Join Node و Fork Node را می‌توان با استفاده از یک نماد نود مشابه، ترکیب کرد. این مورد، مدلی است که شامل یک join node با تمام edge های ورودی و یک edge خروجی که به یک fork node با تمام edge های خروجی موجود آن متصل شده است، نگاشت می‌شود.



ترکیبی از join node و fork node

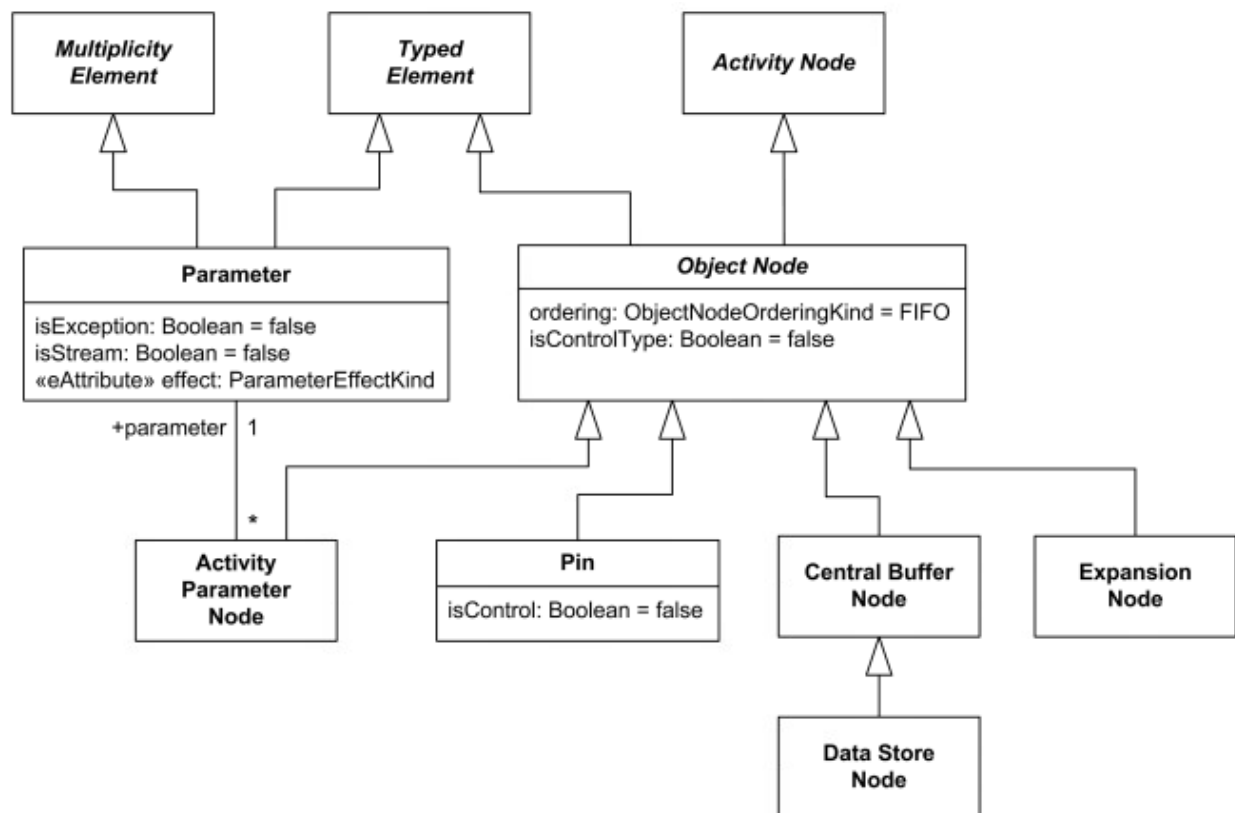
Join Node دارای یک مشخصه اتصال Boolean یا منطقی است که با استفاده از نام edge های ورودی می‌تواند ارزیابی شرایطی که در آن مشخص شده است را انجام داده و یک توکن منتشر کند. هر زمان که یک توکن جدید در هر edge ورودی ارائه شود، مشخصه اتصال ارزیابی می‌شود. ارزیابی با هیچ توکن جدیدی که در طول ارزیابی ارائه می‌شود قطع نمی‌شود، همچنین ارزیابی‌های همزمان با ارائه توکن‌های جدید در طول ارزیابی آغاز نمی‌شوند. مشخصه اتصال پیش‌فرض، رشته رزرو شده "and" است. این حداقل مشخصه‌ای است که به یک توکن ارائه شده در هر لبه ورودی نیاز دارد. مشخصه اتصال در آکولدها، نزدیک Join Node به صورت joinSpec=... نشان داده شده است.



Join Node با مشخصه اتصال نشان داده شده در براکت‌ها

object node یک نود Activity انتزاعی است که برای تعریف جریان‌های شیء در یک Activity استفاده می‌شود. object node ها شامل موارد زیر هستند:

- **pin** یا پین
- **central buffer** یا بافر مرکزی
- **parameter** یا پارامتر
- **expansion nodes** یا نودهای توسعه

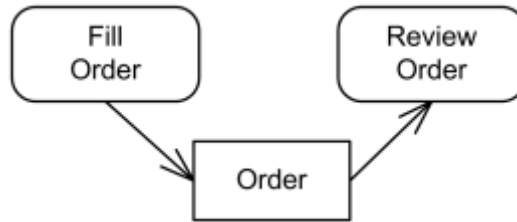


object node های Activity شامل parameter, pin, central buffer, expansion nodes هستند

کمی عجیب است که گرچه object node یک نود Activity انتزاعی است، اما مستقیماً در object flow با استفاده از نماد خود استفاده می‌شود (به زیر مراجعه کنید). منطقی است که متاکلاس UML جداگانه‌ای برای نودهای داده داشته باشیم.

## Object Node

Object Node یک نود Activity انتزاعی است که برای تعریف object flow در یک Activity استفاده می‌شود. این نود نشان می‌دهد که نمونه‌ای از یک طبقه‌بندی‌کننده خاص، احتمالاً در یک حالت خاص، ممکن است در نقطه خاصی از Activity در دسترس باشد. Object Node ها را می‌توان به روش‌های مختلفی استفاده کرد، بسته به اینکه اشیاء از کجا و به کجا جریان دارند. Object Node ها به شکل مستطیل مشخص می‌شوند. نامی که نود مورد نظر را برچسب گذاری می‌کند در داخل نماد قرار می‌گیرد. در این نود نام نشان دهنده نوع Object Node است یا به صورت الگوی نام و نوع نود به صورت "name:type" قرار می‌گیرد.



object flow به نام Order بین Action هایی به نام Fill Order و Review Order قرار دارد

این نام همچنین می‌تواند توسط یک state یا state ها مشخص شود، که باید در داخل پرانتز زیر قالب نام و نوع، نوشته شود. حداقلها، ترتیب، و نوع کنترل به غیر از موارد پیش‌فرض در پرانتزهای زیر Object Node مشخص می‌شوند.

#### Pin

pin یک Object Node برای ورودی و خروجی Action ها است. pin معمولاً به صورت یک مستطیل کوچک متصل به مستطیل Action نشان داده می‌شود. نام pin را می‌توان در نزدیکی pin نمایش داد.



Item، pin ورودی به Action به نام Add to Shopping Cart است



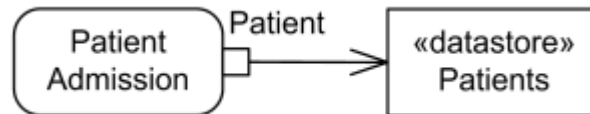
Invoice، pin خروجی از Action به نام Create Invoice است

#### Central Buffer

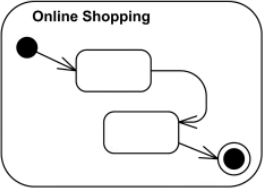
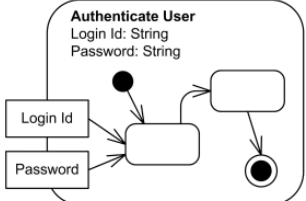
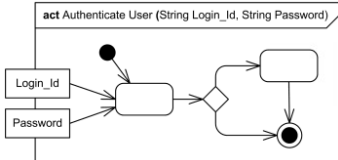
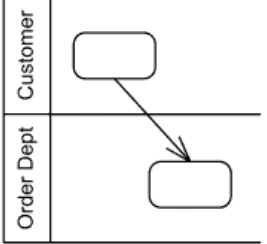
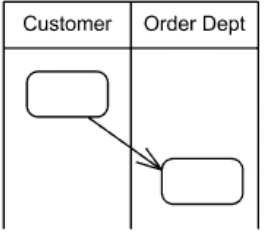
یک نود Central Buffer یک Object Node برای مدیریت جریان‌های متعدد از ابتدا تا انتها است. یک نود Central Buffer توکن‌ها را از چندین شیء در جریان می‌پذیرد، آن‌ها را بافر می‌کند و آن‌ها را در امتداد جریان‌های خارج ارسال می‌کند. Central Buffer ها مستقیماً به Action ها متصل نمی‌شوند. توجه داشته باشید که تمام Object Node دارای قابلیت داخلی بافر کردن داده‌ها هستند. pin ها می‌توانند داده‌ها را برای Action ها بافر کنند، مانند پارامترهای Activity، برای Activity Central Buffer از این جهت متفاوت است که به یک Action یا یک Activity وابسته نیست. این پشتیبانی اضافی برای صف بندی و رقابت بین object flow ها فراهم می‌کند. Central Buffer به عنوان یک Object Node با کلمه کلیدی اختیاری «centralBuffer» مشخص شده است که به شما امکان می‌دهد آن را از pin مستقل متمایز کنید.

#### Data Store

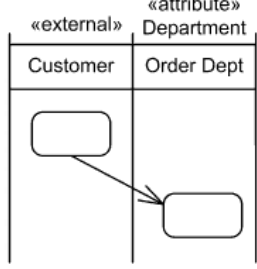
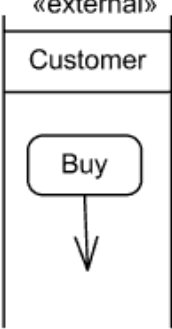
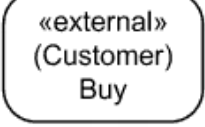
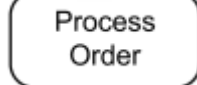
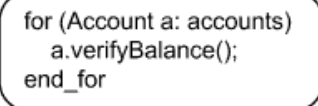
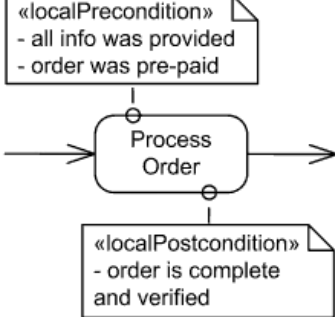
Data Store یک نود Central Buffer برای اطلاعات غیر گذرا است. تمام توکن‌های دریافتی توسط Data Store ذخیره می‌شوند. یک قانون خاص این است که اگر توکن ورودی حاوی یک شیء خاص باشد که قبلاً در Data Store ذخیره شده است، توکن جدید جایگزین هر توکن در Object Node حاوی آن شیء می‌شود. توکن‌هایی که برای خروج ("حرکت به پایین دست") از Data Store انتخاب شده‌اند، در واقع کپی می‌شوند در حالی که توکن اصلی نگهداری می‌شود و از Data Store حذف نمی‌شود. Data Store به‌عنوان یک Object Node با کلمه کلیدی «datastore» مشخص می‌شود.



توکن Patient ورودی توسط Data Store به نام Patients ذخیره می شود

شرح	Notation
<b>Activity</b>	
<p>Activity یک رفتار پارامتریک است که به عنوان جریانی الگوریتمیک از Action ها نشان داده می شود. Activity می تواند به صورت مستطیل گوشه گرد که نام Activity در گوشه سمت چپ بالا و نودها و edge ها در داخل آن قرار دارند، ارائه شود.</p>	 <p>Online Shopping به نام Activity</p>
<p>پارامترهای Activity در حاشیه آن نمایش داده می شوند و در زیر نام Activity به صورت فهرست در قالبی مانند قالب زیر نمایش داده می شوند:</p> <p>parameter-name : parameter-type</p>	 <p>Activity به نام Authenticate User، با دو پارامتر، Login Id و Password</p>
<p>حاشیه Activity گوشه گرد ممکن است با قاب نمودار جایگزین شود. نوع قاب در این مورد Activity یا به صورت خلاصه act است. پارامترهای Activity در صورت وجود روی قاب نمایش داده می شود.</p>	 <p>قاب Activity به نام Authenticate User به همراه دو پارامتر Login Id و Password</p>
<b>Partition</b>	
<p>Partition مربوط به Activity نماینده بخشی از یک Activity است که شامل Action هایی است که دارای برخی مشخصه های مشترک (مثل نقش کاربری یا فاز) هستند. Partition مربوط به Activity ممکن است با استفاده از نماد swimlane با دو خط معمولاً موازی افقی یا عمودی نشان داده شود، که نام پارتیشن در یک کادر در یک انتهای آن برجسته گذاری شده باشد.</p>	 <p>Partition های Activity به نام های Customer و Order Dept به شکل افقی swimlane</p>  <p>Partition های Activity به نام های Customer و Order Dept به شکل عمودی swimlane</p>



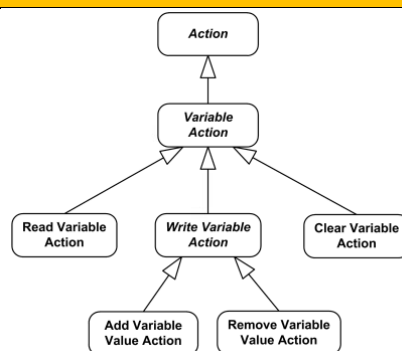
<p>پارتیشن بندی سلسله مراتبی با استفاده از swimlane ها برای Partition های فرعی نشان داده می شود.</p>	 <p>پارتیشن بندی سلسله مراتبی با Partition های فرعی</p>
<p>زمانی که Activity ها خارج از دامنه یک مدل خاص در نظر گرفته می شوند، Partition را می توان با کلمه کلیدی «external» برچسب گذاری کرد.</p>	 <p>Action به نام Buy در Partition خارجی Customer رخ می دهد</p>
<p>در شرایطی که نتوانیم از swimlane ها برای نمایش Partition ها استفاده کنیم، می توان از نماد متنی جایگزین با نام Action معادل استفاده کرد. در این حالت نام Partition در پرانتز بالای نام Action قرار می گیرد. اگر لیستی از نام Partition های محدود شده با کاما وجود داشته باشد، به این معنی است که نود مورد نظر در بیش از یک Partition قرار دارد. دو دونقطه (:) در نام Partition نشان می دهد که Partition مورد نظر تودرتو است و Partition والد در نام قبل از دو دونقطه قرار می گیرند.</p>	 <p>Action به نام Buy در Partition خارجی Customer رخ می دهد</p>
<p align="right"><b>Action</b></p>	
<p>Action ها به صورت مستطیل های گوشه گرد مشخص می شوند. نام عمل یا سایر توضیحات مربوط به آن ممکن است در نماد ظاهر شود.</p>	 <p>Action به نام Process Order</p>
<p>Action را می توان با زبان شبه کد نزدیک به یک زبان واقعی برنامه نویسی بیان کرد.</p>	 <p>نمونه ای از Action که با زبان شبه کد بیان شده است</p>
<p>پیش شرط ها و پس شرط های محلی به ترتیب با کلیدواژه های «localPrecondition» و «localPostcondition» به عنوان یادداشت های پیوست شده به فراخوان نشان داده می شوند.</p>	

	پیش و پس شرایط محلی به عنوان یادداشت‌های پیوست شده به نام <b>Action</b> به نام <b>Process Order</b> نشان داده شده است
در شرایطی که نتوانیم از swimlane ها برای نمایش <b>Partition</b> ها استفاده کنیم، می‌توان از نماد متنی جایگزین با نام <b>Action</b> معادل استفاده کرد. در این حالت نام <b>Partition</b> در پراتز بالای نام <b>Action</b> قرار می‌گیرد. اگر لیستی از نام <b>Partition</b> های محدود شده با کاما وجود داشته باشد، به این معنی است که نود مورد نظر در بیش از یک <b>Partition</b> قرار دارد. دو دونقطه (::) در نام <b>Partition</b> نشان می‌دهد که <b>Partition</b> مورد نظر تودرتو است و <b>Partition</b> والد در نام قبل از دو دونقطه قرار می‌گیرند.	<div style="border: 1px solid black; border-radius: 10px; padding: 10px; text-align: center;"> <b>«external»</b>  <b>(Customer)</b>  <b>Buy</b> </div> <p><b>Action</b> به نام <b>Buy</b> در <b>Partition</b> خارجی <b>Customer</b> رخ می‌دهد</p>

### Object Action

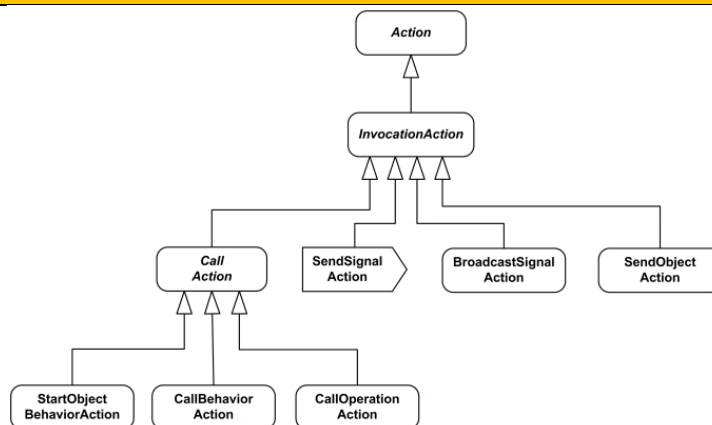
<p><b>Object Action</b> شامل <b>Action</b> های مختلف بر روی اشیاء می‌باشد. <b>Object Action</b> به صراحت در استاندارد <b>UML</b> وجود ندارد، برای وضوح در اینجا اضافه شده است. در استاندارد <b>UML</b>، تمام <b>Object Action</b> ها به صورت مستقیم زیر کلاس‌های <b>Action</b> هستند. <b>Object Action</b> ها شامل موارد زیر هستند:</p> <ul style="list-style-type: none"> <li>• <b>create object action</b> (عملیات ایجاد شیء)</li> <li>• <b>destroy object action</b> (عملیات ازبین بردن شیء)</li> <li>• <b>test identity action</b> (عملیات مقایسه هویت اشیاء)</li> <li>• <b>read self action</b> (عملیات خود خوانده)</li> <li>• <b>value specification action</b> (عملیات تعیین ارزش)</li> <li>• <b>start classifier behavior action</b> (عملیات آغاز رفتار طبقه بندی کننده)</li> <li>• <b>read is classified object action</b> (عملیات تعیین طبقه‌بندی کننده شیء)</li> <li>• <b>reclassify object action</b> (عملیات تغییر طبقه‌بندی کننده شیء)</li> <li>• <b>read extent action</b> (عملیات بازایی اشیاء یک طبقه‌بندی کننده)</li> </ul>	
---	--

### Variable Action

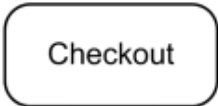
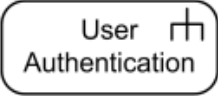

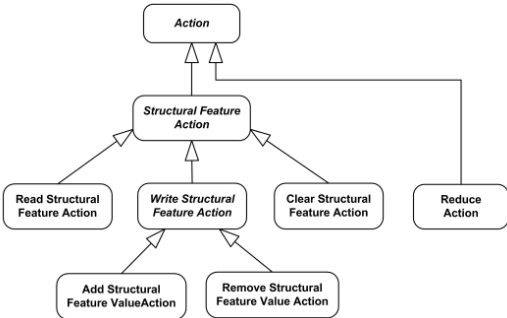
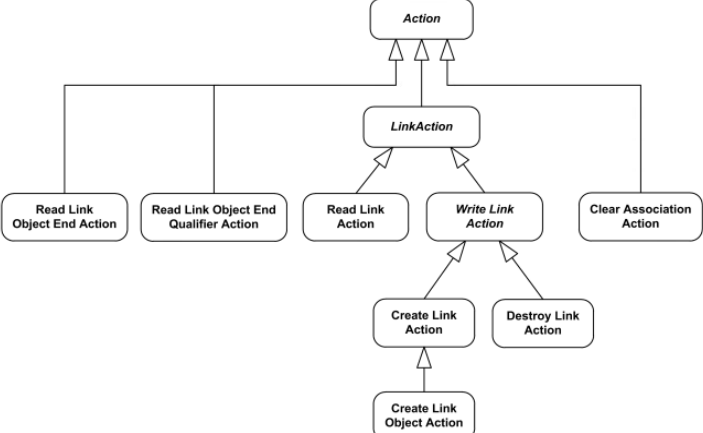


Variable actions overview diagram

### Invocation Action

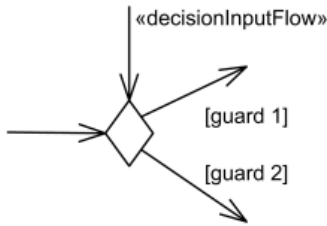
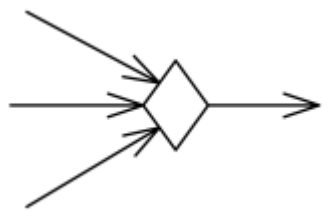
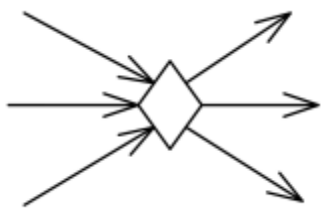
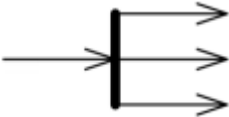
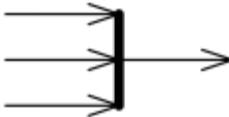


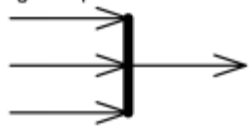
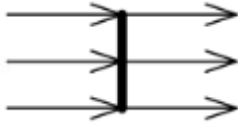
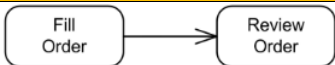
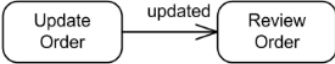
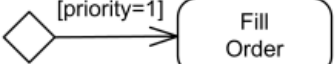
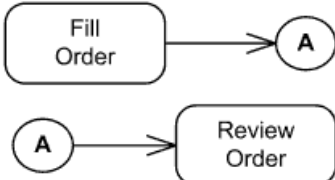
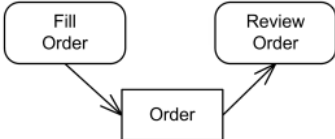
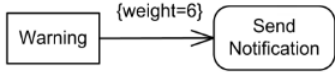
Invocation actions overview diagram

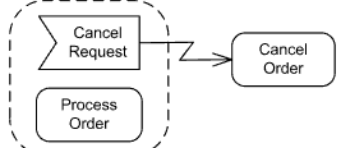
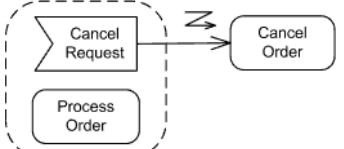
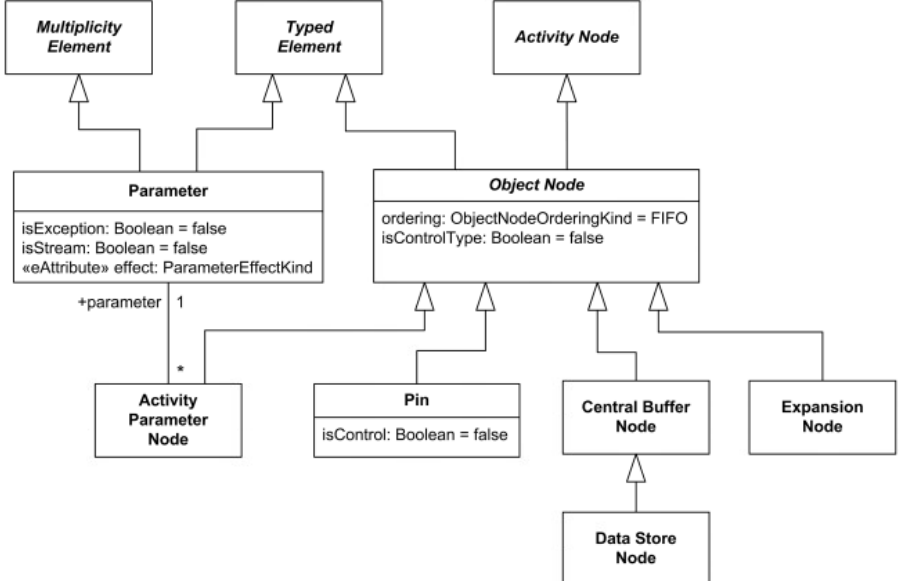
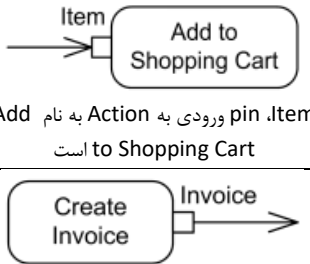
Call Behavior Action	
<p>Call behavior action یک Action فراخوانی است که به جای فراخوانی عملیاتی که رفتار را فراخوانی می‌کند، مستقیماً یک رفتار را فراخوانی می‌کند. این Action با نام عملیات و یا توصیف رفتار در داخل مستطیل گوشه گرد مربوط به Action نشان داده می‌شود. اگر نام نود مورد نظر با نام رفتار آن متفاوت باشد، به جای این حالت، به شکل یک نماد معرفی می‌شود.</p> <p>توجه داشته باشید، از آنجایی که دقیقاً شبیه به Action معمولی است، هیچ راهی وجود ندارد که فقط با نگاه کردن به نمودار بگوییم که آیا نام داخل این Action، نام Action به صورت معمول است، یا نام رفتار یا نام رفتار مشابه Action است.</p>	 <p>Checkout</p> <p>Call behavior action برای رفتار Checkout</p>
<p>Call activity action با یک نماد شبیه به شنکش در نماد Action نشان داده می‌شود. توجه داشته باشید که اگرچه مشخصات UML 2.4 این نماد را ارائه می‌دهد، هیچ Action رسمی برای Call activity در مشخصات UML وجود ندارد.</p>	 <p>User Authentication</p> <p>User Call activity action به نام Authentication</p>
Send Signal Action	
<p>Send signal action یک Action فراخوانی خاص است که سیگنالی را از ورودی‌های خود ایجاد می‌کند و آن را به شیء هدف مشخص شده، ارسال می‌کند. این کار ممکن است باعث تحرک و یک انتقال در state machine یا اجرای یک Activity شود.</p> <p>هنگامی که تمام پیش نیازهای اجرای Action برآورده شد، یک سیگنال از جنس آرگومان تولید می‌شود و به شیء هدف مشخص شده منتقل می‌شود. فرستنده سیگنال (با نام مستعار "requestor") بلافاصله پس از انتقال سیگنال، اجرا را ادامه می‌دهد، بدون اینکه منتظر هیچ پاسخی باشد.</p> <p>Send signal action به صورت پنج ضلعی محدب مشخص می‌شود. توجه داشته باشید که نام Action با نام کلاس سیگنالی که ارسال می‌کند مطابقت دارد. شیء هدف در این نماد مشخص نشده است.</p>	 <p>Notify Customer</p> <p>Send Signal Action به نام Notify Customer</p> <p>سیگنال Notify Customer را ایجاد و ارسال می‌کند</p>
Structural Feature Action	
 <pre> graph TD     Action --&gt; StructuralFeatureAction[Structural Feature Action]     StructuralFeatureAction --&gt; ReadStructuralFeatureAction[Read Structural Feature Action]     StructuralFeatureAction --&gt; WriteStructuralFeatureAction[Write Structural Feature Action]     StructuralFeatureAction --&gt; ClearStructuralFeatureAction[Clear Structural Feature Action]     StructuralFeatureAction --&gt; ReduceAction[Reduce Action]     WriteStructuralFeatureAction --&gt; AddStructuralFeatureValueAction[Add Structural Feature Value Action]     WriteStructuralFeatureAction --&gt; RemoveStructuralFeatureValueAction[Remove Structural Feature Value Action] </pre> <p>Structural feature actions overview diagram</p>	
Link Action	
 <pre> graph TD     Action --&gt; LinkAction[Link Action]     LinkAction --&gt; ReadLinkObjectEndAction[Read Link Object End Action]     LinkAction --&gt; ReadLinkObjectQualifierAction[Read Link Object End Qualifier Action]     LinkAction --&gt; ReadLinkAction[Read Link Action]     LinkAction --&gt; WriteLinkAction[Write Link Action]     LinkAction --&gt; ClearAssociationAction[Clear Association Action]     WriteLinkAction --&gt; CreateLinkAction[Create Link Action]     WriteLinkAction --&gt; DestroyLinkAction[Destroy Link Action]     CreateLinkAction --&gt; CreateLinkObjectAction[Create Link Object Action] </pre> <p>Link actions overview diagram</p>	

Event Action	
<p>Event actions overview diagram</p>	
Accept Event Action	
<p>Accept Event Action با یک پنج ضلعی مقعر علامت گذاری می‌شود. اگر یک Accept Event Action دارای edge های ورودی نباشد، در آن صورت این Action زمانی شروع می‌شود که Activity دربرگیرنده یا نود ساخت یافته اجرا شود، البته هر کدام که فوراً حاوی Action باشد. علاوه بر این، یک Accept Event Action بدون edge های ورودی پس از پذیرش یک رویداد، فعال و در دسترس باقی می‌ماند. پس از پذیرش یک رویداد و خارج شدن یک مقدار، خاتمه نمی‌یابد، اما همچنان منتظر رویدادهای دیگر است.</p> <p>Action که فعال کننده آن یک رویداد سیگنال است به طور غیررسمی accept signal action نامیده می‌شود. آن مطابق send signal action است.</p>	<p>پذیرش سیگنال Accept Order باعث فراخوانی یک Action به نام Process Order می‌شود. Accept Event Action به نام Accept Order با ورود به Activity دربرگیرنده آن، فعال می‌شود، بنابراین هیچ پیکان ورودی به آن نشان داده نمی‌شود.</p>
<p>Accept Event Action می‌تواند edge های ورودی داشته باشد. در این حالت Action پس از اتمام عمل قبلی شروع می‌شود.</p>	<p>سیگنال Payment Requested ارسال می‌شود. سپس Activity منتظر می‌ماند تا سیگنال Payment Confirmed را دریافت کند. پذیرش Payment Confirmed تنها پس از ارسال Payment Requested فعال می‌شود. تا آن زمان هیچ تائیدیای پذیرفته نمی‌شود.</p>
Wait Time Action	
<p>اگر رویداد مورد نظر وقوع یک رویداد زمانی باشد، مقدار حاصل، شامل زمانی است که در آن زمان، رخ داده است یا خواهد داد. چنین Action به طور غیررسمی، wait time action نامیده می‌شود.</p> <p>Accept time event action (به صورت غیررسمی wait time action نام دارد) با نمادی شبیه به یک ساعت شنی مشخص شده است.</p>	<p>Action رویداد پذیرش زمانی Every Hour در هر ساعت یک خروجی تولید می‌کند. هیچ edge ورودی برای این Action رویداد زمانی وجود ندارد، بنابراین تا زمانی که Activity دربرگیرنده یا نود ساخت یافته آن فعال باشد، فعال است.</p>
Control Nodes	

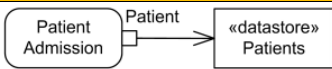
<p>نمای کلی نودهای کنترل Activity</p>	
<b>Initial Node</b>	
<p>Initial node یک نود کنترلی است که با فراخوانی Activity، جریان در آن شروع می‌شود. Activity ممکن است بیش از یک Initial node داشته باشد. Initial node ها به صورت یک دایره کوچک توپر نشان داده می‌شوند.</p>	<p>Initial Node در یک Activity</p>
<b>Flow Final Node</b>	
<p>Flow Final Node یک نود نهایی کنترلی است که یک جریان را خاتمه می‌دهد. نماد نود نهایی جریان دایره کوچکی است که یک علامت X داخل آن است.</p>	<p>Flow Final Node</p>
<b>Activity Final Node</b>	
<p>Activity Final Node یک نود نهایی کنترلی است که تمام جریان‌های یک Activity را متوقف می‌کند. نهایی کننده Activity در UML 2.0 جدید است. Activity Final Node ها به صورت یک دایره توپر با یک دایره توخالی در حاشیه آن نشان داده می‌شوند. می‌توان آن را به عنوان یک سیبل توصیف کرد و یا با عنوان "چشم درشت" یا هدف ذکر کرد.</p>	<p>Activity Final Node</p>
<b>Decision</b>	
<p>Decision node یک نود کنترلی است که توکن‌ها را در یک یا دو edge ورودی می‌پذیرد و یک edge خروجی را از یک یا چند جریان خروجی انتخاب می‌کند. نماد Decision node یک نماد الماس شکل است.</p>	<p>Decision Node با دو edge خروجی به همراه عبارت محدود کننده کننده</p>
<p>برای decision point ها، یک عبارت محدود کننده از پیش تعریف شده به نام "else" ممکن است حداکثر برای یک edge خروجی تعریف شود.</p>	<p>Decision Node با سه edge خروجی دارای عبارت محدود کننده که یکی از عبارات محدود کننده [else] است</p>
<p>هر Decision می‌تواند decision input behavior داشته باشد. decision input behavior ها در UML برای جلوگیری از محاسبات مجدد اضافی در عبارات محدود کننده معرفی شدند. در این مورد، قبل از ارزیابی عبارات محدود کننده در Edge های خروجی، هر توکن دارای داده به رفتار مورد نظر (decision input behavior) ارسال می‌شود. خروجی رفتار برای هر عبارت محدود کننده در دسترس خواهد بود. decision input behavior با کلمه کلیدی «DecisionInput» مشخص می‌شود و برخی رفتار یا شرایط تصمیم‌گیری در نماد note قرار می‌گیرند و به نود تصمیم‌گیری مناسب متصل می‌شوند.</p>	

	Decision Node با رفتار ورودی تصمیم‌گیری یا decision input behavior
Decision ممکن است decision input flow نیز داشته باشد. در این مورد، توکن‌های ارائه شده در decision input flow که در هر edge خروجی در دسترس عبارت محدود کننده قرار می‌گیرند، تعیین می‌کنند که آیا عرضه در edge ورودی در امتداد edge خروجی ارسال می‌شود یا خیر. یک decision input flow با کلمه کلیدی «decisionInputFlow» مشخص می‌شود که آن جریان را توضیح می‌دهد.	 <p>decision input با Decision Node flow</p>
<b>Merge</b>	
Merge Node یک نود کنترلی است که چندین جریان مختلف ورودی را دریافت می‌کند تا جریان خروجی واحد را پذیرش نماید. هیچیک از توکن‌ها با هم پیوند داده نمی‌شوند. Merge Node نباید برای همگام‌سازی جریان‌های همزمان (synchronize concurrent flows) استفاده شود. نماد Merge Node نمادی به شکل الماس است که دو یا چند edge وارد آن می‌شود و یک edge واحد از آن خارج می‌شود.	 <p>Merge Node سه edge ورودی را برای یک edge خروجی ادغام می‌کند</p>
<b>Merge and decision combined</b>	
عملکرد Merge Node و Decision node را می‌توان با استفاده از نماد نود مشابه، ترکیب کرد.	 <p>Merge Node و Decision Node که مشترکا در یک نماد قرار دارند</p>
<b>Fork Node</b>	
Fork Node یک نود کنترلی است که دارای یک edge ورودی و چندین edge خروجی است و برای تقسیم جریان ورودی به چند جریان همزمان (concurrent flows) استفاده می‌شود. نماد Fork Node یک پاره خط است که یک edge وارد آن می‌شود و دو یا چند edge از آن خارج می‌شوند.	 <p>Fork Node با یک edge که وارد آن می‌شود و سه edge که از آن خارج می‌شود</p>
<b>Join Node</b>	
Join Node یک نود کنترلی است که دارای چندین edge ورودی و یک edge خروجی است و برای همگام‌سازی جریان‌های همزمان (synchronize concurrent flows) ورودی استفاده می‌شود. نماد Join Node یک پاره خط است که چندین edge وارد آن می‌شود و تنها یک edge از آن خارج می‌شود.	 <p>Join Node که سه edge وارد آن می‌شود و یک edge از آن خارج می‌شود</p>

<p>join specification در آکولادهای نزدیک Join Node با قالبی مثل نگارش زیر</p> <p>joinSpec = ❶ ❶ → expression</p> <p>نشان داده شده است.</p>	<p>{joinSpec= sum &gt;=2 }</p>  <p>Join Node با join specification که در براکت‌ها قرار دارد</p>
Join and fork combined	
<p>عملکرد join node و fork node را می‌توان با استفاده از نماد نود مشابه ترکیب کرد.</p>	 <p>ترکیبی از join node و fork node</p>
Activity Edge	
<p>Activity edge می‌تواند edge کنترل یا edge جریان داده (با نام مستعار object flow edge) باشد. هر دو توسط یک خط پیکان باز که نودهای داخل یک Activity را به هم متصل می‌کند، مشخص می‌شوند.</p>	 <p>Activity edge که Fill Order و Review Order را به هم متصل می‌کند</p>
<p>Activity edge را می‌توان نامگذاری نیز نمود، با این حال، edge ها الزامی برای داشتن نام‌های اختصاصی در یک Activity ندارند. اگر edge نام دارد، در نزدیکی فلش، (در انتهای edge) ترسیم می‌شود.</p>	 <p>Activity Edge به نام "update" دو نود داخل Activity را به هم متصل می‌کند</p>
<p>عبارت محدود کننده edge درون Activity در براکت‌ها نشان داده شده است که حاوی عبارت محدود کننده است. عبارت محدود کننده باید برای هر توکنی که برای عبور از edge ارائه می‌شود، true ارزیابی شود.</p>	 <p>وقتی priority برابر 1 است Action به نام Fill Order اجرا می‌شود</p>
<p>یک edge درون Activity را می‌توان با استفاده از یک اتصال‌دهنده یا connector، که یک دایره کوچک با یک نام (label نامیده می‌شود) در آن است، علامت‌گذاری کرد. اتصال‌دهنده‌ها معمولاً برای جلوگیری از کشیدن edge های بلند، استفاده می‌شوند. این کاملاً نمادین است. هر اتصال‌دهنده با یک برچسب مشخص باید دقیقاً با یکی دیگر با همان برچسب در نمودار Activity جفت شود. دایره‌ها و خطوط درگیر به یک edge درون یک Activity در مدل نگاشت می‌شوند.</p>	 <p>اتصال‌دهنده A دو edge را بین Fill Order و Review Order متصل می‌کند</p>
Object Flow Edge	
<p>Object Flow Edge، edge های درون Activity هستند که برای نشان دادن جریان داده بین نودهای Action استفاده می‌شوند. Object Flow Edge علامت خاصی ندارند. در این مثال Order یک شیء داده است که در جریان داده یک Activity تبادل می‌شود.</p>	 <p>جریانی از داده‌های Order بین Action های Fill Order و Review Order</p>
<p>وزن لبه ممکن است در آکولادهایی که وزن را مشخص می‌کند، نشان داده شود. وزن یا weight یک مقدار توصیفی است که ممکن است یک مقدار عدد طبیعی (مقدار عددی بین 1 تا بی‌نهایت) باشد. اگر بخواهیم وزن نامحدود را نمایش دهیم مقدار weight را "*" نشان می‌دهیم.</p>	 <p>وقتی تعداد اخطارها به 6 رسید، Action به نام Send Notification اجرا می‌شود</p>
Interrupting Edge	

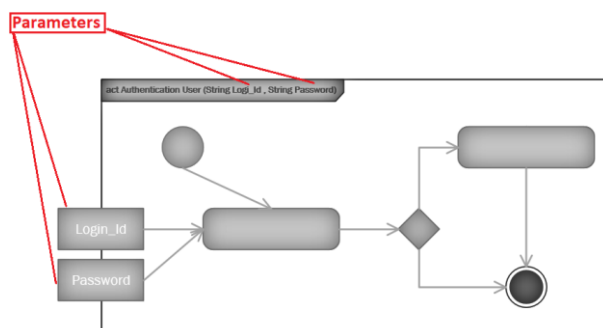
<p>Interrupting Edge یک edge در یک Activity است که وقایع یا Action های خارج از جریان اصلی را برای مناطق دارای وقفه بیان می‌کند. این نماد به صورت یک edge به شکل رعد و برق ارائه شده است.</p>	 <p>سیگنال Cancel Request که یک Action فارق از جریان اصلی می‌تواند باعث وقفه و در نتیجه اجرای Action به نام Cancel Order می‌شود</p>
<p>یک گزینه برای علامت گذاری Interrupting Edge. یک نشانک زیگ زاگ روی یک edge معمولی است.</p>	 <p>سیگنال Cancel Request که یک Action فارق از جریان اصلی می‌تواند باعث وقفه و در نتیجه اجرای Action به نام Cancel Order می‌شود</p>
<p style="text-align: right;"><b>Object Nodes</b></p>	
 <p>نودهای شیء درون Activity شامل Parameter, Pin, Central Buffer, Expansion هستند</p>	
<p style="text-align: right;"><b>Pin</b></p>	
<p>Pin یک نود شیء ورودی و خروجی برای Action ها است. Pin معمولاً به صورت یک مستطیل کوچک متصل به مستطیل Action نشان داده می‌شود. نام Pin را می‌توان در نزدیکی Pin نمایش داد.</p>	 <p>Item ورودی به Action به نام Add to Shopping Cart است</p> <p>Invoice pin خروجی از Action به نام Create Invoice است</p>



Data Store	
<p>Data Store یک نود Central Buffer برای اطلاعات پایدار است.</p> <p>Data Store به عنوان یک نود شیء است که با کلمه کلیدی «datastore» مشخص می شود.</p>	 <p>توکن ورودی Patient توسط Data Store به نام Patients دریافت شده و ذخیره می شود</p>

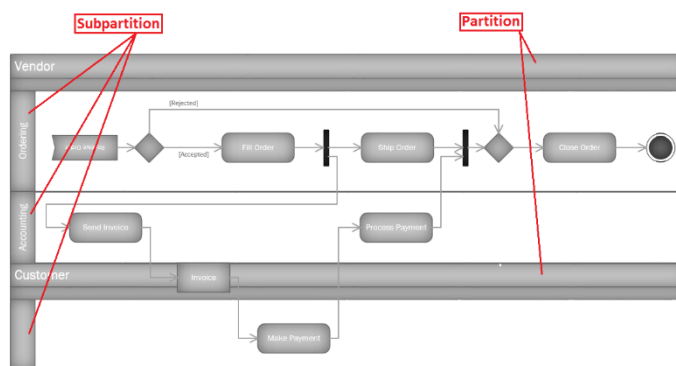
## Activity:

محدوده‌ای الگوریتمیک که ترکیبی از عملیات را داخل خود انجام می‌دهد و پارامتریک است.



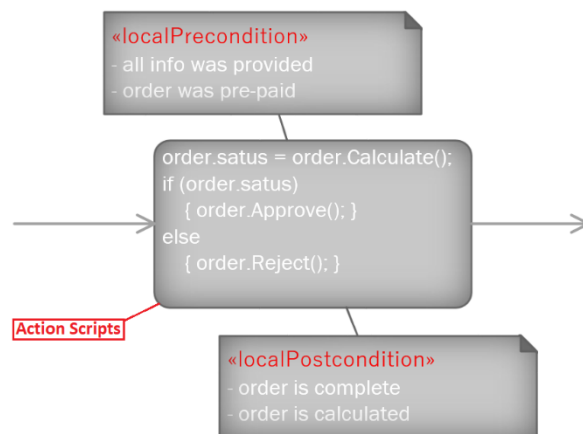
## Partition:

نوعی طبقه‌بندی برای بخش‌ها یا عملیات‌هایی که مشخصات مشترک دارند. تقریباً معادل swimlane در فرآیند است، فقط swimlane روشی برای طبقه‌بندی عملیاتی است که نقش خاصی در یک فرآیند انجام می‌دهد یا به عهده دارد. در واقع در فرآیند، مشخصه مشترک در این خصوص، همان نقش کاربری است ولیکن در خصوص Activity ها این مشخصات مشترک، می‌تواند به پارامترها، مقادیر خروجی و یا متغیرهایی که عملیات متوالی یا موازی هر طبقه یا Partition درون الگوریتم به آن/آنها مرتبط خواهند بود، باشند. این طبقه‌بندی‌ها برخلاف swimlane های فرآیند که می‌توانند مسیر جریان داده یا شیء را تغییر دهند، هیچ تأثیر بر جریان داده و شیء ندارند. به عبارت دیگر در فرآیندها، برای شناسایی الگوریتم جریان داده و شیء باید ابتدا swimlane ها یا همان مشخصه نقش کاربری شناسایی شود ولیکن در Activity برای شناسایی الگوریتم جریان داده و شیء نیازی به شناسایی طبقه‌بندی‌ها یا Partition ها نیست و طبقه‌بندی‌ها یا Partition ها درک الگوریتم را بهتر می‌کنند.



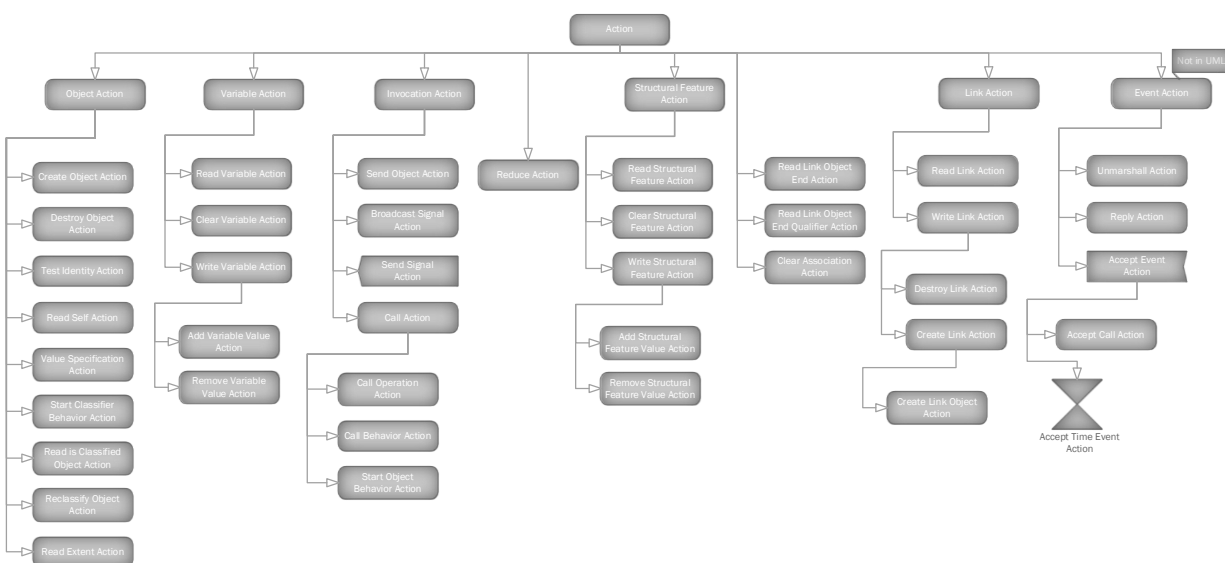
## Action:

عملی است که در یک روال یا فرآیند الگوریتمیک می‌تواند اجرا شود. هر عملی سه ماهیت وابسته دارد: پیش‌شرط‌های آغاز اجرا - دستور/دستورات اجرایی - پس‌شرط‌های خاتمه اجرا



## Action Type:

انواع عملیات که در یک نمودار فعالیت قابل تعریف است.



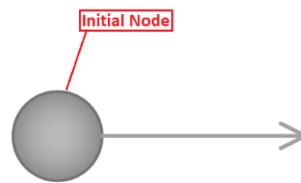
عملیات:

1. عملیات مربوط به شیء (Object Action)
  - 1.1. عملیات مربوط به ایجاد شیء (Create Object Action)
  - 1.2. عملیات مربوط به رهاسازی شیء (Destroy Object Action)
  - 1.3. عملیات مربوط به تست هویت شیء (Test Identity Action)
  - 1.4. عملیات مربوط به خواندن شیء توسط خود شیء (Read Self Action)
  - 1.5. عملیات مربوط به مشخص کردن مقدار شیء (Value Specification Action)
  - 1.6. عملیات مربوط به آغاز رفتار طبقه‌بندی کننده شیء (Start Classifier Behavior Action)
  - 1.7. عملیات مربوط به خواندن شیء طبقه‌بندی شده (Read is Classified Object Action)
  - 1.8. عملیات مربوط به طبقه‌بندی مجدد شیء (Reclassify Object Action)
  - 1.9. عملیات مربوط به محدوده قابل خواندن (Read Extent Action)

2. عملیات مربوط به متغیر (Variable Action)
  - 2.1. عملیات مربوط به خواندن مقدار متغیر (Read Variable Action)
  - 2.2. عملیات مربوط به رهاسازی متغیر (Clear Variable Action)
  - 2.3. عملیات مربوط به نوشتن مقدار متغیر (Write Variable Action)
    - 2.3.1. عملیات مربوط به افزایش مقدار متغیر (Add Variable Value Action)
    - 2.3.2. عملیات مربوط به حذف مقدار متغیر (Remove Variable Value Action)
3. عملیات مربوط به تبادلات (Invocation Action)
  - 3.1. عملیات مربوط به ارسال شیء (Send Object Action)
  - 3.2. عملیات مربوط به پخش سیگنال (Broadcast Signal Action)
  - 3.3. عملیات مربوط به ارسال سیگنال (Send Signal Action)
  - 3.4. عملیات مربوط به فراخوانی (Call Action)
    - 3.4.1. عملیات مربوط به فراخوانی عملکرد (Call Operation Action)
    - 3.4.2. عملیات مربوط به فراخوانی رفتار (Call Behavior Action)
    - 3.4.3. عملیات مربوط به آغاز رفتار شیء (Start Object Behavior Action)
4. عملیات کاهش دهنده (Reduce Action)
5. عملیات مربوط به ویژگی‌های ساختاری (Structural Feature Action)
  - 5.1. عملیات مربوط به خواندن ویژگی‌های ساختاری (Read Structural Feature Action)
  - 5.2. عملیات مربوط به رهاسازی ویژگی‌های ساختاری (Clear Structural Feature Action)
  - 5.3. عملیات مربوط به نوشتن ویژگی‌های ساختاری (Write Structural Feature Action)
    - 5.3.1. عملیات مربوط به افزودن مقدار به ویژگی ساختاری (Add Structural Feature Value Action)
    - 5.3.2. عملیات مربوط به حذف مقدار از ویژگی ساختاری (Remove Structural Feature Value Action)
6. عملیات مربوط به پیوندها (Link Action)
  - 6.1. عملیات مربوط به خواندن پیوند (Read Link Action)
  - 6.2. عملیات مربوط به نوشتن پیوند (Write Link Action)
    - 6.2.1. عملیات مربوط به رهاسازی پیوند (Destroy Link Action)
    - 6.2.2. عملیات مربوط به ایجاد پیوند (Create Link Action)
      - 6.2.2.1. عملیات مربوط به ایجاد شیء پیوند (Create Link Object Action)
7. عملیات مربوط به خواندن شیء اشاره شده در پیوند (Read Link Object End Action)
8. عملیات توصیفی مربوط به خواندن شیء اشاره شده در پیوند (Read Link Object End Qualifier Action)
9. عملیات مربوط به پیوندهای صریح و شفاف (Clear Association Action)
10. عملیات مربوط به رویدادها (Event Action)
  - 10.1. عملیات مربوط به هدایت‌کننده خودکار سازگاری (Unmarshall Action)
  - 10.2. عملیات مربوط به گاسخگویی (Reply Action)
  - 10.3. عملیات مربوط به رویداد پذیرش (Accept Event Action)
    - 10.3.1. عملیات مربوط به فراخوانی دستور پذیرش (Accept Call Action)
    - 10.3.2. عملیات مربوط به رویداد زمان پذیرش (Accept Time Event Action)

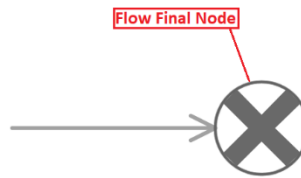
## Initial Node:

نود کنترل آغاز فعالیت



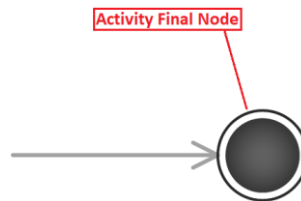
## Flow Final Node:

نود کنترل خاتمه یک جریان



## Activity Final Node:

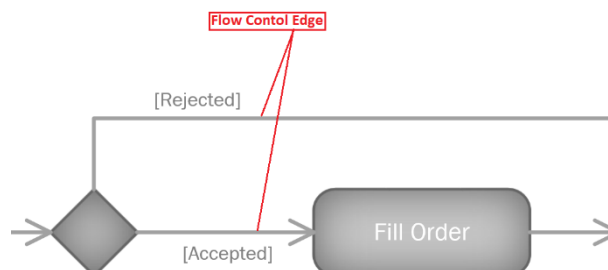
نود کنترل خاتمه یک فعالیت

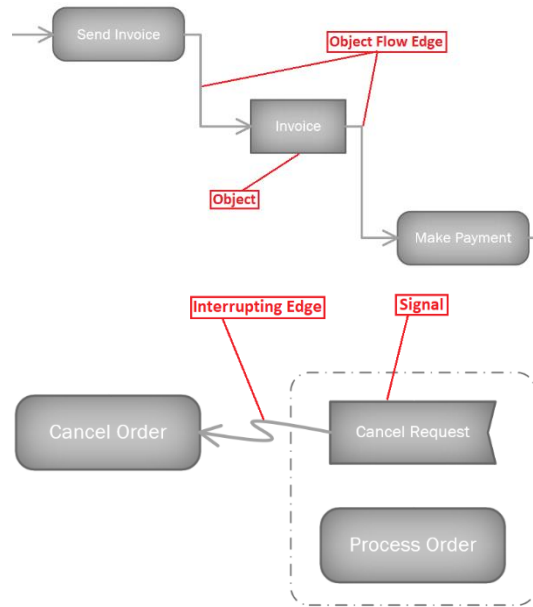


## Edge:

انتقال دهنده که انواع زیر را دارد:

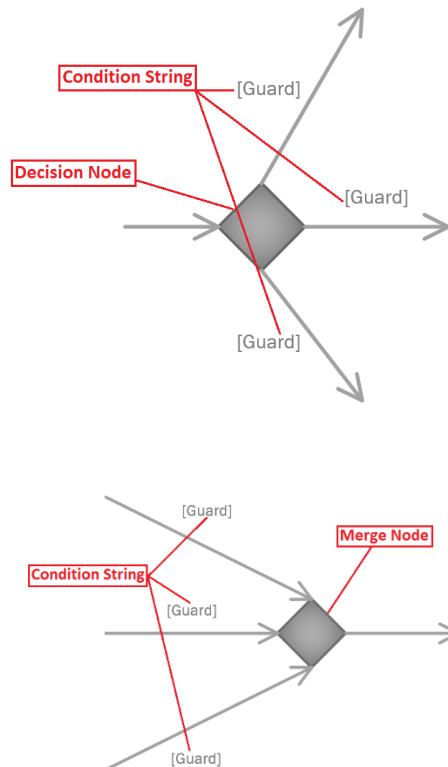
- انتقال دهنده کنترل کننده جریان (Flow Control Edge)
- انتقال دهنده شیء جریان (Object Flow Edge)
- انتقال دهنده وقفه خارج از جریان (Interrupting Edge)





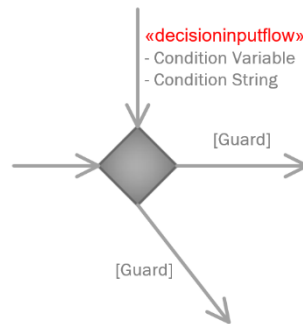
## Decision and Merge Node:

نود متفرق کننده انتقال بر اساس تصمیم‌گیری و نود هم‌بندی یا تجمع کننده انتقال بر اساس تصمیم‌گیری



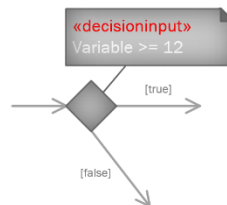
## Decision Input Flow:

جریان ورودی تصمیم‌گیری. این جریان می‌تواند دستورالعمل یا متن تصمیم‌گیری یا شرط تصمیم‌گیری را به نود تصمیم‌گیری معرفی نماید



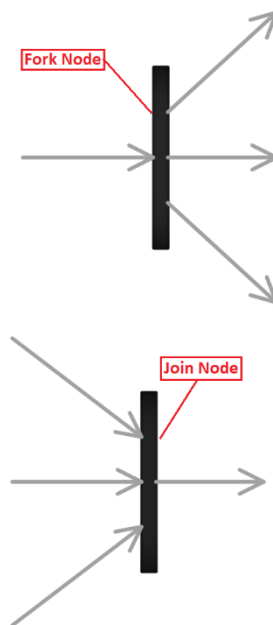
## Decision Input:

معرفی دستورالعمل یا متن تصمیم‌گیری



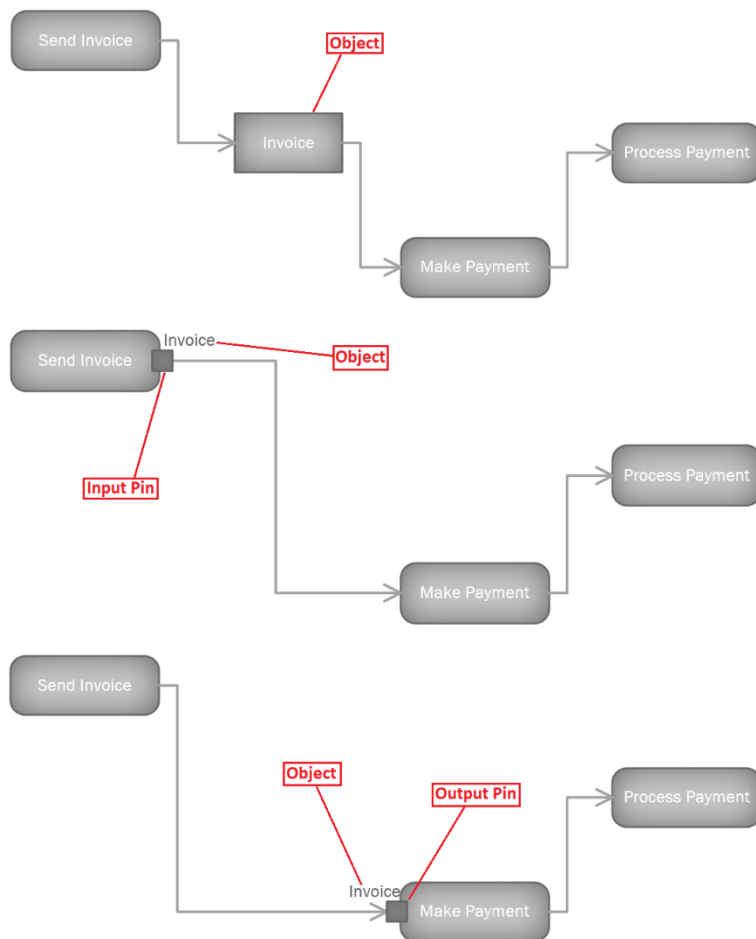
## Fork and Join Node:

نود متفرق کننده انتقال و نود هم‌بندی یا تجمع کننده انتقال



## Pin:

نودی برای معرفی شیء وارد شونده و یا خارج شونده به/از عملیات



Data Store:

محل ذخیره سازی و نگهداری بلند مدت داده ها (ذخیره سازی پایدار)

