



Plant UML

PlantUML 은 다이어그램을 빠르게 작성하기 위한 오픈 소스 프로젝트입니다.

Archimate Diagram

This is only a proposal and subject to change.

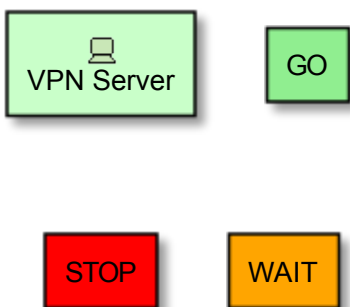
You are very welcome [to create a new discussion](#) on this future syntax. Your feedbacks, ideas and suggestions help us to find the right solution.

Archimate keyword

You can use the `archimate` keyword to define an element. Stereotype can optionally specify an additional icon. Some colors (`Business` , `Application` , `Motivation` , `Strategy` , `Technology` , `Physical` , `Implementation`) are also available.

```
@startuml
archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```



Using the `circle` keyword and the [preprocessor](#), you can also create junctions.

```

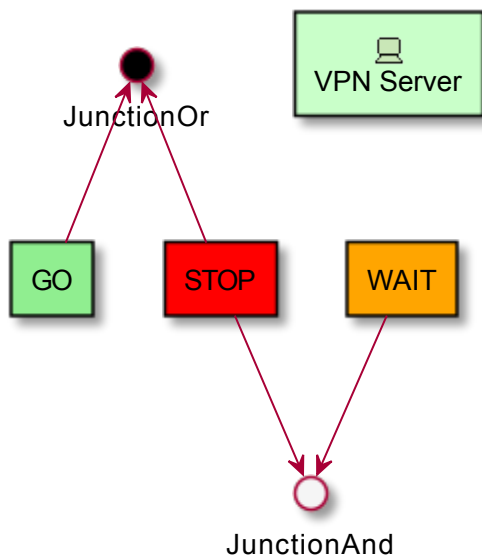
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

Junction_And JunctionAnd
Junction_Or JunctionOr

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
GO -up-> JunctionOr
STOP -up-> JunctionOr
STOP -down-> JunctionAnd
WAIT -down-> JunctionAnd
@enduml

```



Example 1

```

@startuml
skinparam rectangle<<behavior>> {
    roundCorner 25
}
sprite $bProcess jar:archimate/business-process
sprite $aService jar:archimate/application-service
sprite $aComponent jar:archimate/application-component

rectangle "Handle claim" as HC <<$bProcess>><<behavior>> #Business
rectangle "Capture Information" as CI <<$bProcess>><<behavior>> #Business
rectangle "Notify\nAdditional Stakeholders" as NAS <<$bProcess>><<behavior>> #Business
rectangle "Validate" as V <<$bProcess>><<behavior>> #Business
rectangle "Investigate" as I <<$bProcess>><<behavior>> #Business
rectangle "Pay" as P <<$bProcess>><<behavior>> #Business

HC *-down- CI
HC *-down- NAS
HC *-down- V
HC *-down- I
HC *-down- P

CI -right->> NAS
NAS -right->> V
V -right->> I
I -right->> P

rectangle "Scanning" as scanning <<$aService>><<behavior>> #Application
rectangle "Customer administration" as customerAdministration <<$aService>><<behavior>> #Application
rectangle "Claims administration" as claimsAdministration <<$aService>><<behavior>> #Application
rectangle Printing <<$aService>><<behavior>> #Application
rectangle Payment <<$aService>><<behavior>> #Application

scanning -up-> CI
customerAdministration -up-> CI
claimsAdministration -up-> NAS
claimsAdministration -up-> V
claimsAdministration -up-> I
Payment -up-> P

Printing -up-> V
Printing -up-> P

rectangle "Document\nManagement\nSystem" as DMS <<$aComponent>> #Application
rectangle "General\nCRM\nSystem" as CRM <<$aComponent>> #Application
rectangle "Home & Away\nPolicy\nAdministration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\nFinancial\nAdministration" as HFPA <<$aComponent>> #Application

```

```
DMS .up.|> scanning
DMS .up.|> Printing
CRM .up.|> customerAdministration
HAPA .up.|> claimsAdministration
HFPA .up.|> Payment
```

```
legend left
```

```
Example from the "Archisurance case study" (OpenGroup).
```

```
See
```

```
====
```

```
<$bProcess> :business process
```

```
====
```

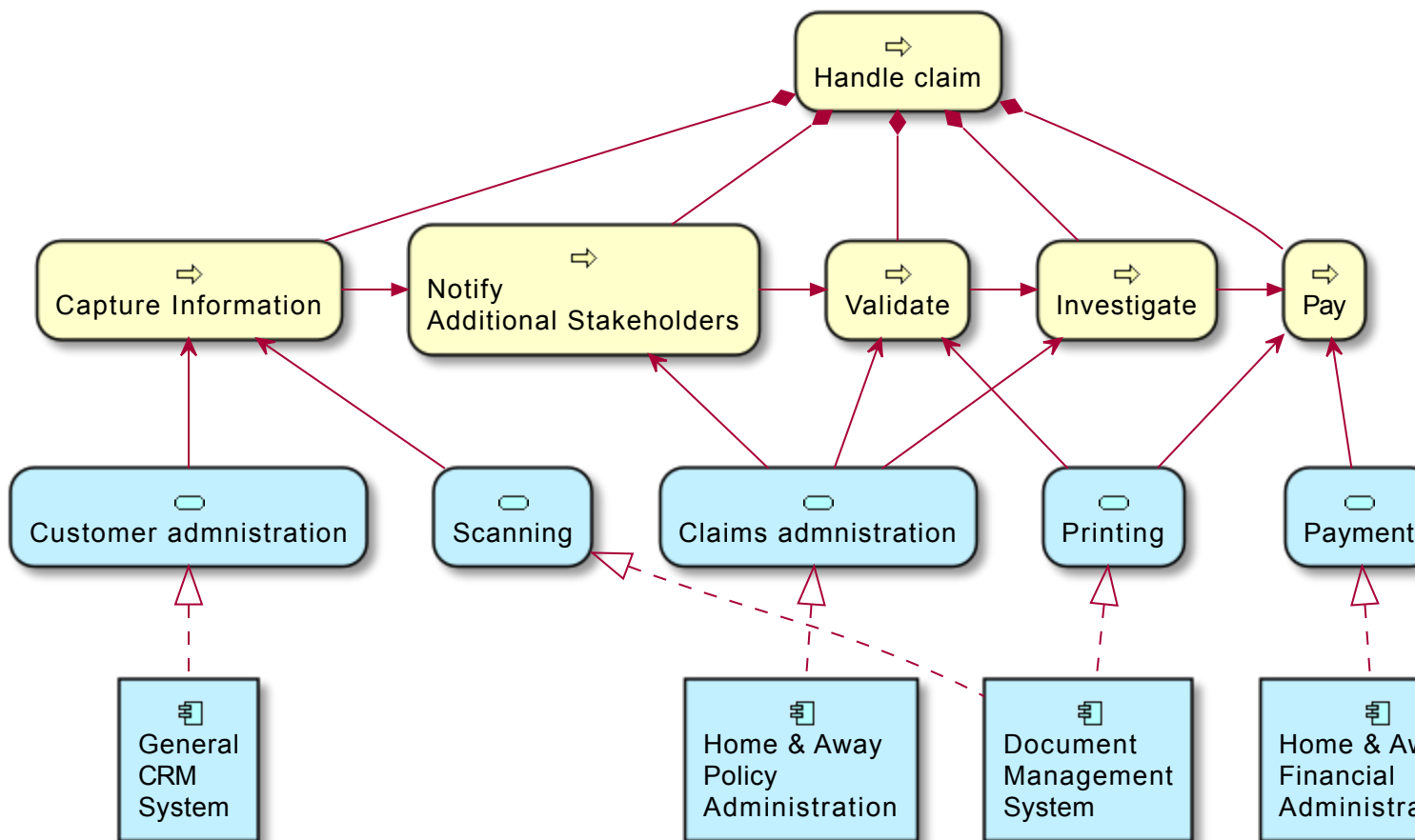
```
<$aService> : application service
```

```
====
```

```
<$aComponent> : application component
```

```
endlegend
```

```
@enduml
```



Example from the "Archisurance case study" (OpenGroup).
See

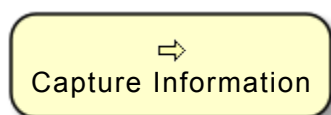
⇒ :business process

▢ : application service

▣ : application component

Example 2

```
@startuml
skinparam roundcorner 25
rectangle "Capture Information" as CI <<$archimate/business-process>> #Business
@enduml
```



List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```

List Current Sprites

Credit to

<http://www.archimatetool.com>

archimate :

- access
- activity
- actor
- aggregation
- application-collaboration
- application-component
- application-data-object
- application-event
- application-function
- application-interaction
- application-interface
- application-process
- application-service
- assessment-filled
- assessment
- assignment
- association-unidirect
- association
- business-activity
- business-actor
- business-collaboration
- business-contract
- business-event
- business-function
- business-interaction
- business-interface
- business-location
- business-meaning

- business-object
- business-process
- business-product
- business-representation
- business-role
- business-service
- business-value
- collaboration
- communication-path
- component
- composition
- constraint-filled
- constraint
- contract
- deliverable-filled
- deliverable
- device
- driver-filled
- driver
- event
- flow
- function
- gap-filled
- gap
- goal-filled
- goal
- implementation-deliverable
- implementation-event
- implementation-gap
- implementation-plateau
- implementation-workpackage
- influence
- interaction
- interface-required

- interface-symmetric
- interface
- junction-and
- junction-or
- junction
- location
- meaning
- motivation-assessment
- motivation-constraint
- motivation-driver
- motivation-goal
- motivation-meaning
- motivation-outcome
- motivation-principle
- motivation-requirement
- motivation-stakeholder
- motivation-value
- network
- node
- object
- physical-distribution-network
- physical-equipment
- physical-facility
- physical-material
- plateau
- principle-filled
- principle
- process
- product
- realisation
- representation
- requirement-filled
- requirement
- role

- service
- serving
- specialisation
- specialization
- stakeholder-filled
- strategy-capability
- strategy-course-of-action
- strategy-resource
- strategy-value-stream
- system-software
- technology-artifact
- technology-collaboration
- technology-communication
- technology-communication
- technology-device
- technology-event
- technology-function
- technology-infra-interface
- technology-infra-service
- technology-interaction
- technology-interface
- technology-network
- technology-node
- technology-path
- technology-process
- technology-service
- technology-system-software
- triggering
- used-by
- value
- workpackage-filled

ArchiMate Macros

A list of Archimate macros are defined [Archimate-PlantUML](#) here which simplifies the creation of ArchiMate diagrams, and Archimate is natively on the [Standard Library](#) of PlantUML.

ArchiMate elements

Using the macros, creation of ArchiMate elements are done using the following format:

```
Category_ElementName(nameOfTheElement, "description")
```

For example:

- To define a Stakeholder element, which is part of Motivation category, the syntax will be

```
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description") :
```

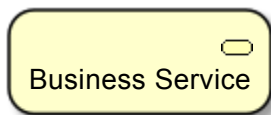
```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
@enduml
```



- To define a *Business Service* element,

```
Business_Service(BService, "Business Service") :
```

```
@startuml
!include <archimate/Archimate>
Business_Service(BService, "Business Service")
@enduml
```



Archimate relationships

The ArchiMate relationships are defined with the following pattern:

`Rel_RelationType(fromElement, toElement, "description")` and to define the direction/orientation of the two elements:

```
Rel_RelationType_Direction(fromElement, toElement, "description")
```

The `RelationTypes` supported are:

- Access
- Aggregation

- Assignment
- Association
- Composition
- Flow
- Influence
- Realization
- Serving
- Specialization
- Triggering

The **Directions** supported are:

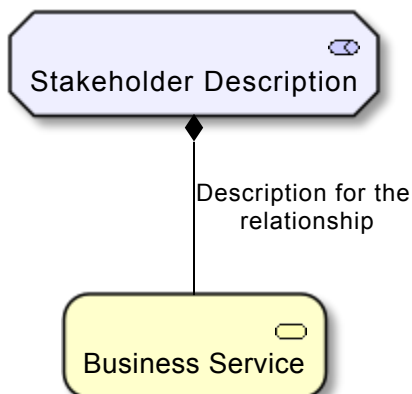
- Up
- Down
- Left
- Right

For example:

- To denote a composition relationship between the Stakeholder and Business Service defined above, the syntax will be

```
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
```

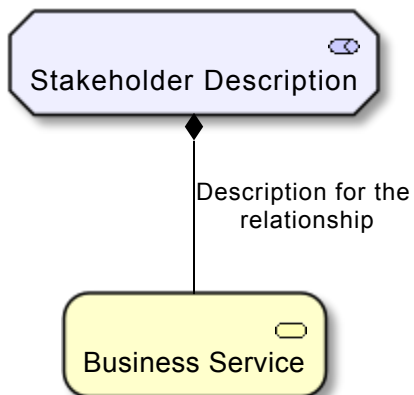
```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml
```



- Unordered List ItemTo orient the two elements in top - down position, the syntax will be

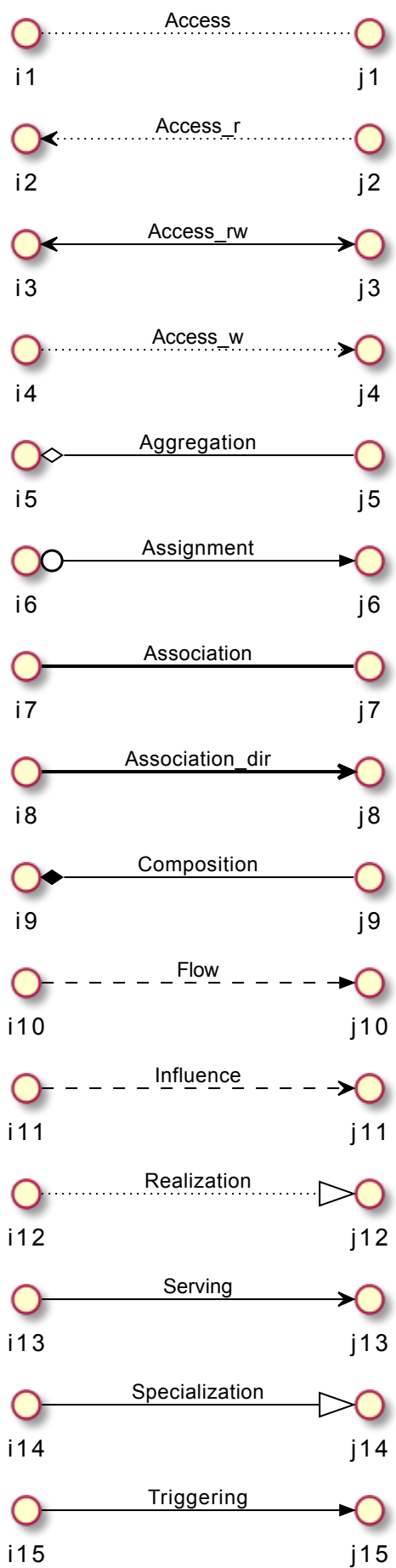
```
Rel_Composition_Down(StakeholderElement, BService, "Description for the relation
```

```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
@enduml
```



Appendice: Examples of all Archimate RelationTypes

```
@startuml
left to right direction
skinparam nodesep 4
!include <archimate/Archimate>
Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i8, j8, Association_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i3, j3, Access_rw)
Rel_Access_r(i2, j2, Access_r)
Rel_Access(i1, j1, Access)
@enduml
```



```
@startuml
title ArchiMate Relationships Overview
skinparam nodesep 5
<style>
interface {
    shadowing 0
    backgroundcolor transparent
    linecolor transparent
    FontColor transparent
}
</style>
!include <archimate/Archimate>
left to right direction

rectangle Other {
    () i14
    () j14
}

rectangle Dynamic {
    () i10
    () j10
    () i15
    () j15
}

rectangle Dependency {
    () i13
    () j13
    () i4
    () j4
    () i11
    () j11
    () i7
    () j7
}

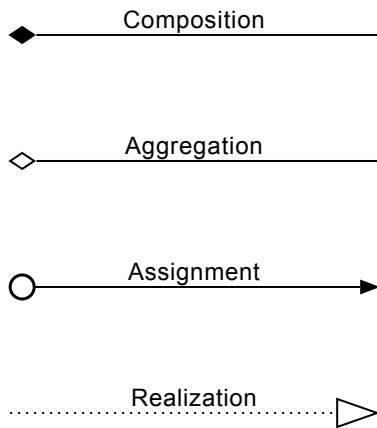
rectangle Structural {
    () i9
    () j9
    () i5
    () j5
    () i6
    () j6
    () i12
```

```
() j12  
}
```

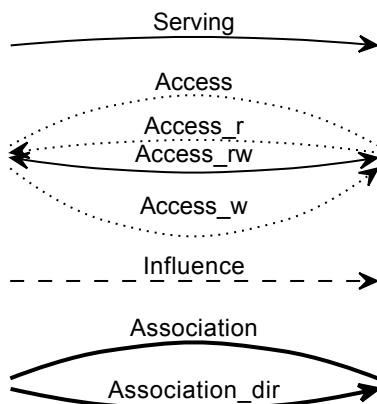
```
Rel_Triggering(i15, j15, Triggering)  
Rel_Specialization(i14, j14, Specialization)  
Rel_Serving(i13, j13, Serving)  
Rel_Realization(i12, j12, Realization)  
Rel_Influence(i11, j11, Influence)  
Rel_Flow(i10, j10, Flow)  
Rel_Composition(i9, j9, Composition)  
Rel_Association_dir(i7, j7, \nAssociation_dir)  
Rel_Association(i7, j7, Association)  
Rel_Assignment(i6, j6, Assignment)  
Rel_Aggregation(i5, j5, Aggregation)  
Rel_Access_w(i4, j4, Access_w)  
Rel_Access_rw(i4, j4, Access_rw)  
Rel_Access_r(i4, j4, Access_r)  
Rel_Access(i4, j4, Access)  
@enduml
```

ArchiMate Relationships Overview

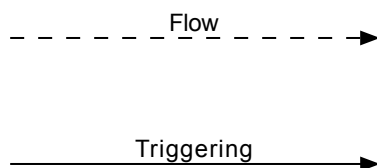
Structural



Dependency



Dynamic



Other

