SPARX SYSTEMS

Enterprise Architect

User Guide Series

Unified Modeling Language (UML)

What UML Modeling tool to use? Sparx Systems Enterprise Architect contains a wealth of tools for creating models under the Unified Modeling Language (UML), providing full compliance with the UML specification of elements, relationships and diagrams.

Author: Sparx Systems
Date: 2026-05-04
Version: 17.1

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Unified Modeling Language (UML) 13
UML Diagrams 17
UML Structural Models 19
Class Diagram 21
Composite Structure Diagram 27
Properties 32
Component Diagram 35
Deployment Diagram 39
Object Diagram 45
Package Diagram 49
Profile Diagram 54
UML Behavioral Models 57
Activity Diagram 60
Use Case Diagram 67
Example Use Case Diagram 72
StateMachines 73
Pseudostates 82
Regions 85
Create a Connection Point Reference 88
StateMachine Table 92
StateMachine Table Options 95
StateMachine Table Operations 100
Change StateMachine Table Position 102
Change StateMachine Table Size 103

Insert Trigger 104
Insert/Change Transition 106
Insert New State 109
Reposition State or Trigger Cells 112
Add Legend 113
Find Cell in StateMachine Diagram 115
StateMachine Table Conventions 116
Export State Table To CSV File 118
Example State-Trigger Table 119
Example State-Next State Table 120
StateMachine Table Simulation 121
Timing Diagram 124
Create a Timing Diagram 127
Set a Time Range 128
Edit a Timing Diagram 130
Add and Edit State Lifeline 131
Add States to a State Lifeline 134
Edit States in a State Lifeline 136
Delete States in a State Lifeline 137
Edit Transitions In State Lifeline 138
Add and Move Transitions 140
Add and Edit Value Lifeline 145
Add States In Value Lifeline 147
Edit Transitions In Value Lifeline 148
Configure Timeline - States 152
Numeric Range Generator 155
Configure Timeline - Transitions 157

Time Intervals 160
Create Time Intervals 162
Compress Time Intervals 165
Select Time Intervals 167
Time Interval Operations 169
Messages (Timing Diagram) 176
Create a Timing Message 178
Sequence Diagram 182
Denote Lifecycle of an Element 188
Layout of Sequence Diagrams 190
Sequence Elements 192
Messages (Sequence Diagram) 194
Self-Message 201
Call 203
Message Examples 204
Change the Timing Details 206
General Ordering 209
Asynchronous Signal Message 211
Co-Region Notation 212
Sequence Diagrams and Version Control 213
Sequence Element Activations 215
Lifeline Activation Levels 220
Sequence Message Label Visibility 222
Change the Top Margin 223
Inline Sequence Elements 224
Communication Diagram 226
Communication Diagrams in Color 230

Messages (Communication Diagrams) 232
Create a Communication Message 234
Re-Order Messages 236
Interaction Overview Diagram 239
UML Elements 245
Behavioral Diagram Elements 247
Action 248
Action Types 253
Variable Actions 261
Local Pre/Post Conditions 264
Class Operations in Diagrams 266
Action Pin 269
Assign Action Pins 272
Activity 274
Activity Notation 277
Activity Parameter Nodes 278
Activity Partition 281
Actor 283
Central Buffer Node 286
Choice 288
Combined Fragment 291
Create a Combined Fragment 297
Interaction Operators 299
Constraint 307
Datastore 308
Decision 310
Diagram Frame 314

Gate 318
Endpoint 320
Entry Point 322
Event 323
Exception 325
Expansion Node 327
Expansion Region 328
Exit Point 331
Final 333
Flow Final 336
Fork/Join 339
  Fork 343
  Join 345
History 347
Initial 350
Interaction 353
Interaction Occurrence 356
Interruptible Activity Region 360
Junction 362
Lifeline 365
Merge 367
Message Endpoint 369
Message Label 370
Note 372
Object Node 374
Partition 375
Receive 379

Region 381
Send 382
State 384
Composite State 386
State/Continuation 389
Continuation 390
State Invariant 393
State Lifeline 395
StateMachine 399
Structured Activity 401
Structured Node 404
Sequential Node 405
Loop Node 406
Conditional Node 412
Synch 416
System Boundary 417
System Boundary Properties 420
Terminate 431
Trigger 432
Use Case 435
Use Case Extension Points 437
Value Lifeline 440
Structural Diagram Elements 444
Artifact 445
Create File Artifacts 459
Using the Checklist and Audited Checklist Artifacts 462

Using the Reading List Artifact 471
Document Artifact 474
Custom Table Artifact 475
Class 489
Active Classes 492
Parameterized Classes (Templates) 494
Collaboration 497
Collaboration Use 499
Component 501
Data Type 504
Deployment Specification 506
Device 508
Enumeration 510
Execution Environment 511
Expose Interface 513
Information Item 515
Interface 517
Node 520
Object 522
Run-time State 524
Object State 528
Package 530
Packaging Component 532
Part 533
Add Property Value 534
Port 535
Add a Port to an Element 537

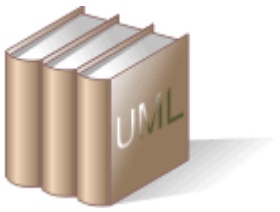
Inherited and Redefined Ports 539
Ports as Owners of Parts 540
Properties Window - Property, Redefined/Subsetted 541
Primitive 542
Signal 543
Reception 545
Properties Window for Receptions 549
UML Connectors 552
Abstraction 553
Aggregation 554
Change Aggregation Connector Form 555
Assembly 556
Association 557
Qualifiers 559
Qualifiers Dialog 562
Association Class 566
Connect New Class to Existing Association 569
Communication Path 570
Composition 571
N-Ary Association 573
Connector 575
Control Flow 576
Delegate 578
Dependency 580
Apply a Stereotype 582
Deployment 584

Extend 585
Generalization 590
Include 592
Information Flow 594
  Using Information Flows 597
  Convey Information on a Flow 603
  Realize an Information Flow 606
Interrupt Flow 610
Manifest 611
Message 613
Nesting 615
Notelink 616
Object Flow 618
  Object Flows in Activity Diagrams 620
Occurrence 622
Package Import 623
Package Merge 625
Realization 628
Recursion 630
Role Binding 631
Represents 633
Representation 634
Substitution 635
Template Binding 636
  Parameter Substitution 638
Trace 641
Transition 643

Internal Transition 648
Usage 651
Use 653
UML Stereotypes 654
Apply Stereotypes 657
Stereotype Selector 659
Stereotype Visibility 663
Standard Stereotypes 666
Stereotypes with Alternative Images 670
Custom Stereotypes 672
Extending UML 678
Using UML Profiles 680
Add Profile Objects to a Diagram 681
Tagged Values in Profiles 683
Synchronize Tagged Values and Constraints 685
Extension Stereotypes 689
Boundary 691
Create a Boundary 692
Control 694
Create a Control Element 695
Entity 697
Create an Entity 698
Hyperlink 700
Image 704
Process 705
Risk 706
Task 708

Test Element 709
Test Case 710
Design Patterns 711
Publish a Pattern 713
Save a Pattern as an Artifact 717
Import a Model Pattern 722
Use a Pattern 724
Add Pattern Dialog 728

Unified Modeling Language (UML)



Enterprise Architect provides a wealth of tools a modeler can use to create models that comply with a wide range of formal and informal modeling languages. One of these languages is the Unified Modeling Language (UML), and Enterprise Architect has comprehensive support for all the elements, relationships and diagrams specified in the language. The UML is governed by the Object Management Group (OMG), of which Sparx Systems is an active member and contributor to the process of managing and improving the language.

Facilities

Facility	Description
The Unified Modeling Language (UML)	The UML standard defines notations and rules for specifying business and software systems; the notation supplies a rich set of graphic elements for modeling object oriented systems, and the rules state how those elements can be connected and used. UML is not a tool for creating software

	systems; instead, it is a visual language for communicating, modeling, specifying and defining systems. UML is not a prescriptive process for modeling software systems; it does not supply a method or process, simply the language. You can therefore use UML in a variety of ways to specify and develop your software engineering project. This language is designed to be flexible, extendable and comprehensive, yet generic enough to serve as a foundation for all system modeling requirements. With its specification, there is a wide range of elements characterized by the kinds of diagrams they serve, and the attributes they provide. All can be further specified by using stereotypes, Tagged Values and profiles. Enterprise Architect supports many different kinds of UML elements (as well as some custom extensions); together with the connectors between elements, these form the basis of the model.
Wide Range of Applications	Although initially conceived as a language for software development, UML can be used to model a wide range of real world domains and processes (in

	business, science, industry, education and elsewhere), organizational hierarchies, deployment maps and much more. Enterprise Architect also provides additional Custom diagrams and elements, to address further modeling interests.
Extending UML for New Domains	Using UML Profiles, Patterns, Grammars, Data Types, Constraints, MDG Technologies and other extensions, UML and Enterprise Architect can be tailored to address a particular modeling domain not explicitly defined in the original UML specification. Enterprise Architect makes extending UML simple and straightforward and, best of all, the extension mechanism is still part of the UML Specification.
Recommended Reading	In addition to the UML Specification available from the OMG, two books that provide excellent introductions to UML are: • Schaum's Outlines: UML by Bennett, Skelton and Lunn (2nd Edn.) Published by McGraw Hill. ISBN: 0-07-710741-1 ISBN-13: 978-0-07-710741-3

	• Developing Software with UML by Bernd Oestereich Published by Addison Wesley. ISBN-10: 0201398265 ISBN-13: 978-0201398267

UML Diagrams

A UML diagram is a graphical representation of part of a model, typically showing a number of elements connected by relationships. Diagrams are one of the most expressive and appealing views of the repository; the diagram has a name and type and is typically constructed for a particular audience to convey an idea or to create a narrative description of part of the model. Diagrams can also be used to generate useful system Artifacts such as XML schemas, database schemas, programming code and more.

The UML specification defines fourteen types of diagram and lists elements and relationships that can be included on each diagram. These elements are conveniently provided in the Enterprise Architect default Toolboxes for each diagram type. While these Toolboxes act as a guide for the novice modeler, the experienced modeler can create highly expressive diagrams by including a wide range of element types on the same diagram.

Diagrams are created and viewed in the main workspace and are stored in Packages or other elements in the repository.

Diagram Grouping

Group	Detail
Structural Diagrams	Structural diagrams depict the structural elements composing a system or function,

	reflecting the static relationships of a structure, or run-time architectures.
Behavioral Diagrams	Behavioral diagrams show a dynamic view of the model, depicting the behavioral features of a system or business process.
Extended Diagrams	Enterprise Architect provides a set of additional diagram types that extend the core UML diagrams for domain-specific models.
Custom Diagrams	Enterprise Architect also supports diagram types specific to MDG Technologies, including integrated technologies.

UML Structural Models

UML Structural diagrams depict the elements of a system that are independent of time and that convey the concepts of a system and how they relate to each other. The elements in these diagrams resemble the nouns in a natural language, and the relationships that connect them are structural or semantic relationships. For example, a structural diagram of a vehicle reservation system might contain elements such as Car, Reservation, Drivers License and Credit Card, and connectors linking these elements. Experienced modelers will also show relationships to behavioral elements on these diagrams.

The UML defines seven types of UML structural diagram.

Structural Diagram types

Diagram Type	Detail
Class	Class diagrams capture the logical structure of the system, the Classes and objects that make up the model, describing what exists and what attributes and behavior it has.
Composite Structure	Composite Structure diagrams reflect the internal collaboration of Classes,

	Interfaces and Components (and their properties) to describe a functionality.
Component	Component diagrams illustrate the pieces of software, embedded controllers and such that make up a system, and their organization and dependencies.
Deployment	Deployment diagrams show how and where the system is to be deployed; that is, its execution architecture.
Object	Object diagrams depict object instances of Classes and their relationships at a point in time.
Package	Package diagrams depict the organization of model elements into Packages and the dependencies amongst them.
Profile	Profile diagrams are those created in a «profile» Package, to extend UML elements, connectors and components.

Class Diagram

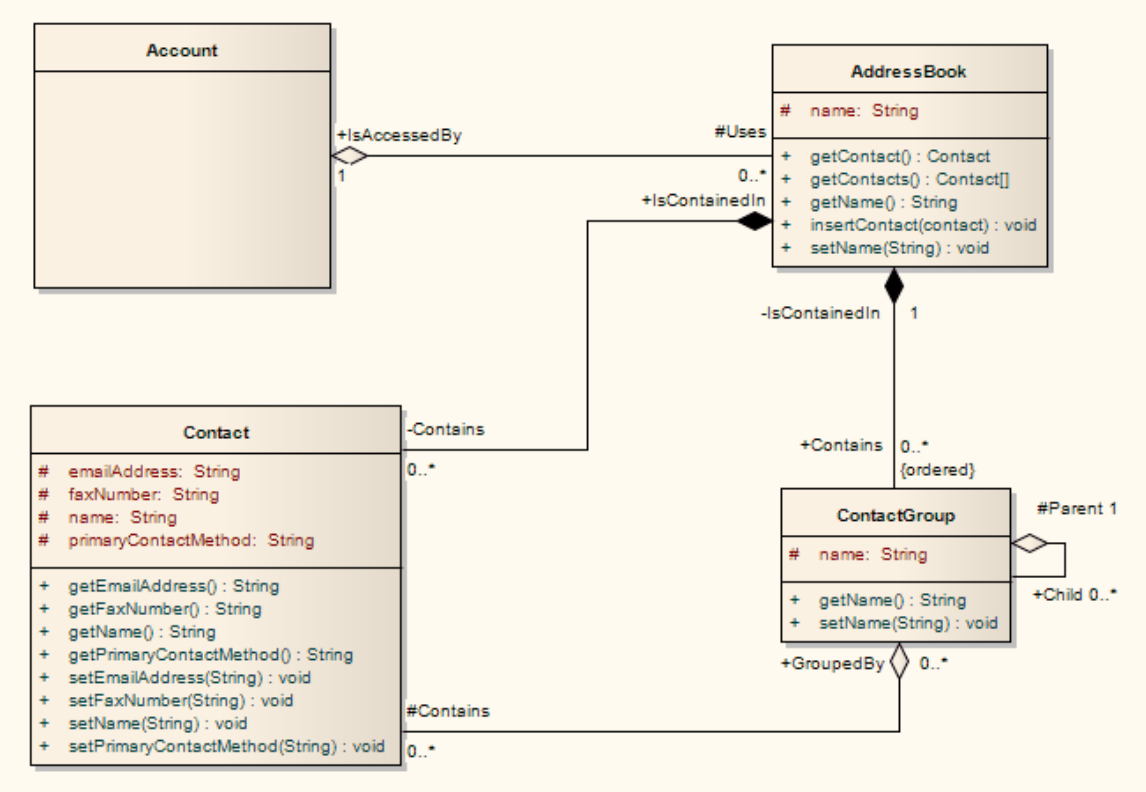
The Class diagram captures the logical structure of the system - the Classes - and things that make up the model. It is a static model, describing what exists and what attributes and behavior it has, rather than how something is done. On a Class diagram you can illustrate relationships between Classes and Interfaces using Generalizations, Aggregations and Associations, which are valuable in reflecting inheritance, composition or usage, and connections respectively.

You generate Class diagram elements and connectors from the 'Class' pages of the Diagram Toolbox.

Example Diagram

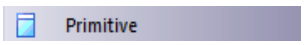
In this example Class diagram, there are two forms of the Aggregation relationship:

- The pale form indicates that the Class Account uses AddressBook, but does not necessarily contain AddressBook
- The dark Composite Aggregation form indicates ownership or containment by the target Classes (at the diamond end) of the source Classes



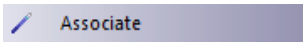
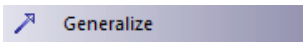
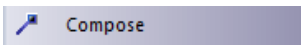
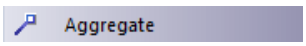
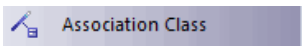
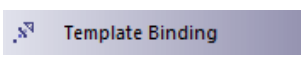
Class Diagram Element Toolbox Icons

Icon	Description
Class	A Class is a representation of a type of object that reflects the structure and behavior of such objects within the system.
Interface	An Interface is a specification of behavior (or contract) that implementers agree to meet.
Data Type	A Data Type is a specific kind of

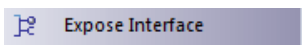
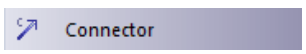
	classifier, similar to a Class except that a Data Type cannot own sub Data Types, and instances of a Data Type are identified only by their value.
	An Enumeration is a data type, whose instances can be any of a number of user-defined enumeration literals.
	A Primitive element identifies a predefined data type, without any relevant substructure (that is, it has no parts in the context of UML).
	A Signal is a specification of Send request instances communicated between objects, typically in a Class or Package diagram.
	An n-Ary Association element is used to model complex relationships between three or more elements, typically in a Class or Object diagram.

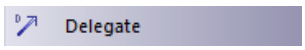
Class Diagram Connector Toolbox Icons

Icon	Description

	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	A Generalization is used to indicate inheritance.
	A Composition is used to depict an element that is made up of smaller components, typically in a Class or Package diagram.
	An Aggregation connector is a type of association that shows that an element contains or is composed of other elements.
	An Association Class is a UML construct that enables an Association to have attributes and operations (features).
	A source object implements or Realizes its destination object.
	You create a Template Binding connector between a binding Class and a parameterized Class.

Class Diagram Composite Parts

Icon	Description
	Parts are run-time instances of Classes or Interfaces.
	Ports define the interaction between a classifier and its environment.
	The Expose Interface element is a graphical method of depicting the required or supplied interfaces of a Component, Class or Part, in a Class, Component or Composite Structure diagram.
	An Assembly connector bridges a component's required interface (Component1) with the provided interface of another component (Component2), typically in a Component diagram.
	Connectors illustrate communication links between Parts to fulfill the

	structure's purpose, typically in a Class or Composite Structure diagram.
	A Delegate connector defines the internal assembly of a component's external Ports and Interfaces, on a Class diagram or Component diagram.

Class Diagram UML Standard Profile

The UML Standard Profile is a collection of stereotyped Classes, operations and relationships provided as modeling tools in compliance with the UML 2.5 Specification (Chapter 22, *Standard Profile*).

Some of these modeling elements are directly available through the 'UML Standard Profile' Toolbox page in the Class or Package Diagram Toolboxes; others can be applied as stereotypes on the base UML modeling object.

Composite Structure Diagram

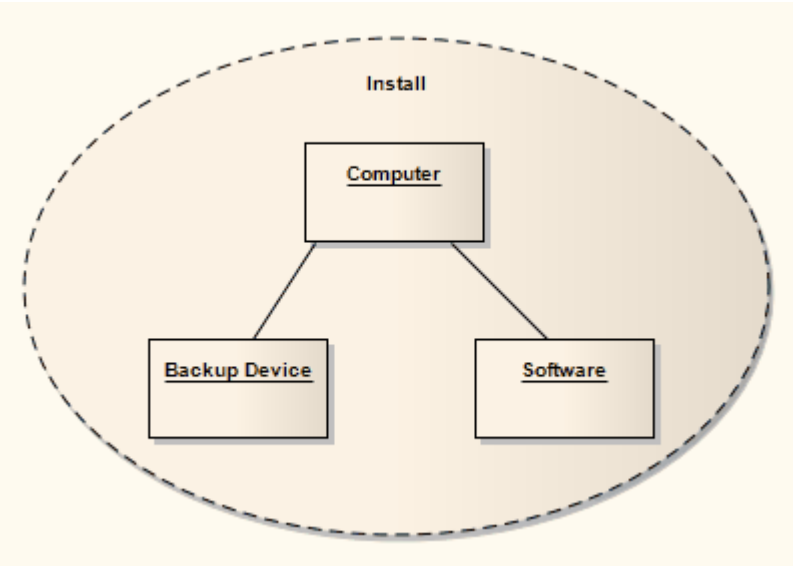
A Composite Structure diagram reflects the internal collaboration of Classes, Interfaces or Components (and their properties) to describe a functionality. Composite Structure diagrams are similar to Class diagrams, but whilst Class diagrams model a static view of Class structures, including their attributes and behaviors, Composite Structure diagrams model a specific usage of the structure. You can use them to express run-time architectures, usage patterns and the participating elements' relationships, which might not be reflected by static diagrams.

In a Composite Structure diagram, Classes are accessed as Parts or run-time instances fulfilling a particular role. These Parts can have multiplicity, if the role filled by the Class requires multiple instances. Ports defined by a Part's Class should be represented in the composite structure, so that all connecting Parts provide the required interfaces specified by the Port. There is extensive flexibility, and a consequent complexity, that come with modeling composite structures. To optimize your modeling, consider building Collaborations to represent reusable Patterns responding to your design issues.

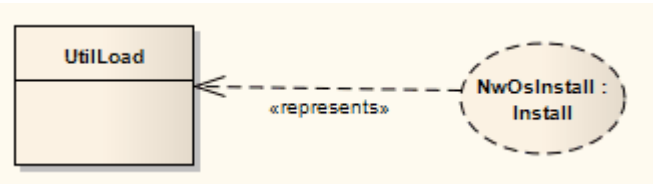
You generate Composite Structure diagram elements and connectors from the 'Composite' pages of the Diagram Toolbox.

Example Diagram

This diagram shows a Collaboration used in a Composite Structure diagram to show a relationship for performing an installation. Collaborations are often used to model common patterns.

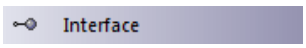
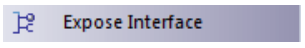


The next diagram uses this Install Collaboration in a Collaboration Use, and applies it to the UtilLoad Class via a ?represents? relationship. This indicates that the classifier UtilLoad uses the Collaboration Pattern within its implementation.

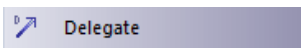
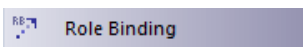


Composite Structure Diagram Element Toolbox Icons

Icon	Description

	A Class is a representation of a type of object that reflects the structure and behavior of such objects within the system.
	An Interface is a specification of behavior (or contract) that implementers agree to meet.
	Parts are run-time instances of Classes or Interfaces.
	Ports define the interaction between a classifier and its environment.
	A Collaboration defines a set of cooperating roles and their connectors.
	Use a Collaboration Use to apply a Pattern defined by a Collaboration to a specific situation, in a Composite Structure diagram.
	The Expose Interface element is a graphical method of depicting the required or supplied interfaces of a Component, Class or Part, in a Component or Composite Structure diagram.

Composite Structure Diagram Connector Toolbox Icons

Icon	Description
	Connectors illustrate communication links between Parts to fulfill the structure's purpose, typically in a Composite Structure diagram.
	An Assembly connector bridges a component's required interface (Component1) with the provided interface of another component (Component2), typically in a Component diagram.
	A Delegate connector defines the internal assembly of a component's external Ports and Interfaces, on a Component diagram.
	Role Binding is the mapping between a Collaboration Use's internal roles and the respective Parts required to implement a specific situation, typically in a Composite Structure diagram.

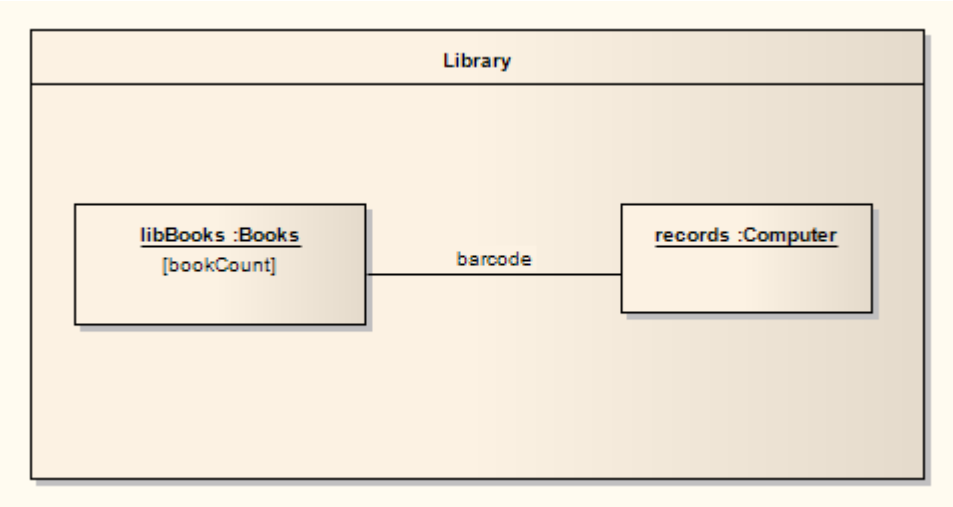
	The Represents connector indicates that a Collaboration is used in a classifier, typically in a Composite Structure diagram.
	An Occurrence relationship indicates that a Collaboration represents a classifier, in a Composite Structure diagram.

Properties

A property is a nested structure within a classifier, usually a Class or an Interface, on a Composite Structure diagram. The contained structure reflects instances and relationships reflected within the containing classifier. Properties can have multiplicity, and can be displayed as:

- Parts (preferred) or
- Association Roles

Parts



In this diagram there are two Parts, 'libBooks' and 'records', which are instances corresponding to the Classes 'Books' and 'Computer' respectively. The relationship between the two Parts is indicated by the connector, reflecting that communication between the Parts is via the barcode. This contained structure and its Parts are properties owned by the Library Class.

After dragging Parts from the Diagram Toolbox onto the

Class, right-click on a Part and select 'Advanced | Set Property Type' to connect to a classifier. If Parts disappear when dragged onto the Class, adjust the Z-order of the Class to move it behind the Parts (right-click on the Class and select the 'Z-Order' option).

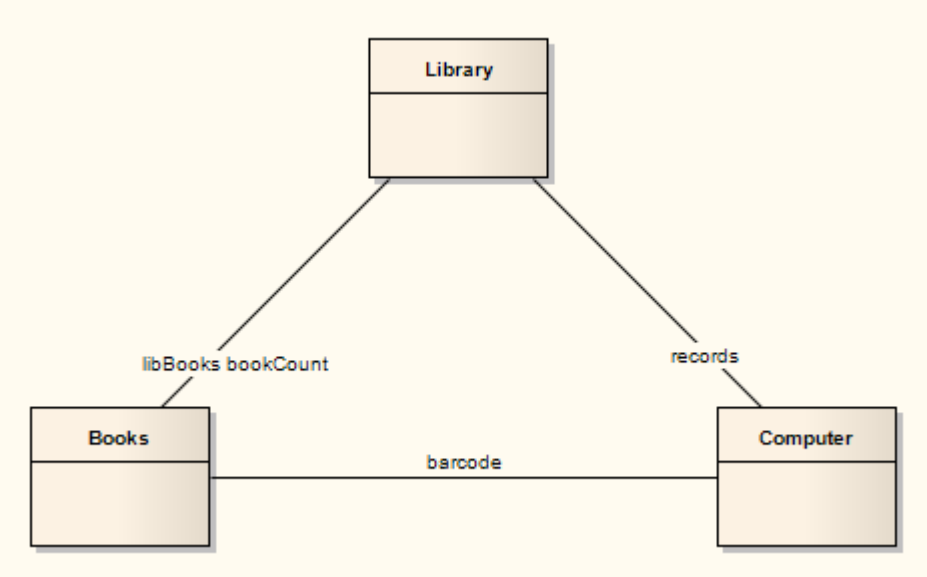
To indicate a property that is not owned by composition to the containing classifier, use a box symbol with a dashed outline, indicating association; to do this:

1. Right-click on the Part and select the 'Properties' option.
2. Select the 'Advanced' page of the 'Properties' dialog.
3. Set the 'IsReference' option to True.

Association Roles

Properties can also be reflected using a normal composite structure (without containing it in a Class), with the appropriate connectors, Parts and relationships indicated through connections to the Class.

The alternative representation is shown here; however, this representation fails to express the ownership immediately reflected by containing properties within a classifier.


Library
libBooks bookCount
records
Books
barcode
Computer

Component Diagram

A Component diagram illustrates the pieces of software, embedded controllers and such that make up a system, and their organization and dependencies.

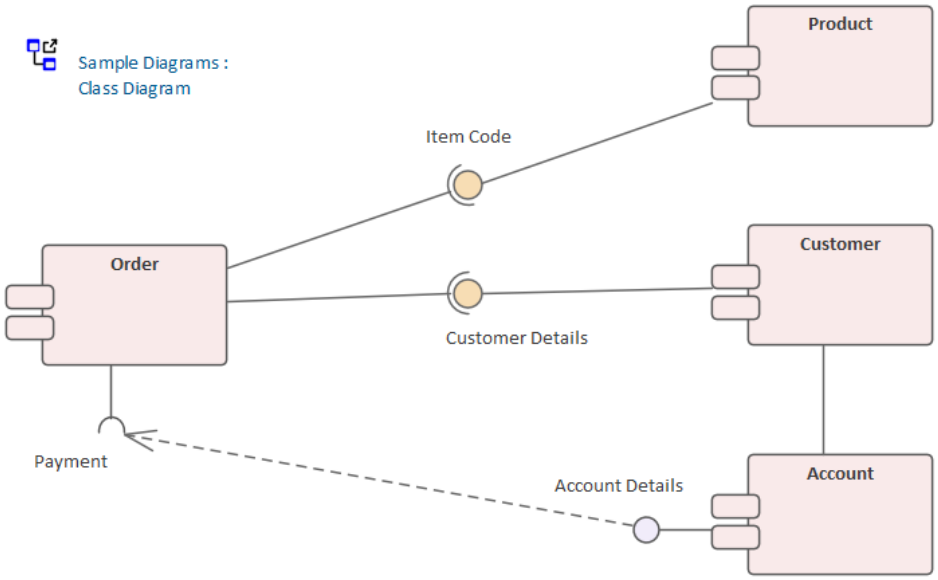
A Component diagram has a higher level of abstraction than a Class diagram; usually a component is implemented by one or more Classes (or Objects) at runtime. They are building blocks, built up so that eventually a component can encompass a large portion of a system.

You generate Component diagram elements and connectors from the 'Component' pages of the Diagram Toolbox.

Example Diagram

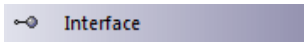
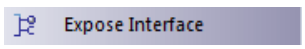
This diagram demonstrates a number of components and their inter-relationships.

Assembly connectors connect the provided interfaces supplied by Product and Customer to the required interfaces specified by Order. A Dependency relationship maps a customer's associated account details to the required interface Payment, also specified by Order.



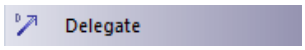
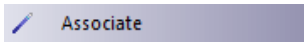
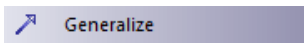
Component Diagram Element Toolbox Icons

Icon	Description
Packaging Component	A Packaging Component is an element that appears very similar to a Component in a diagram but behaves as a Package in the Browser window.
Component	A Component is a modular part of a system, whose behavior is defined by its provided and required interfaces.
Class	A Class is a representation of a type of object that reflects the structure and behavior of such objects within the system.

	An Interface is a specification of behavior (or contract) that implementers agree to meet.
	An Object is a particular instance of a Class at run time.
	Ports define the interaction between a classifier and its environment.
	The Expose Interface element is a graphical method of depicting the required or supplied interfaces of a Component, Class or Part, in a Component or Composite Structure diagram.

Component Diagram Connector Toolbox Icons

Icon	Description
	An Assembly connector bridges a component's required interface (Component1) with the provided interface of another component

	(Component2), typically in a Component diagram.
	A Delegate connector defines the internal assembly of a component's external Ports and Interfaces, on a Component diagram.
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	A source object implements or Realizes its destination object. Realize connectors are used in a Use Case, Component or Requirements diagram to express traceability and completeness in the model.
	A Generalization is used to indicate inheritance.

Deployment Diagram

A Deployment diagram shows how and where the system is to be deployed; that is, its execution architecture.

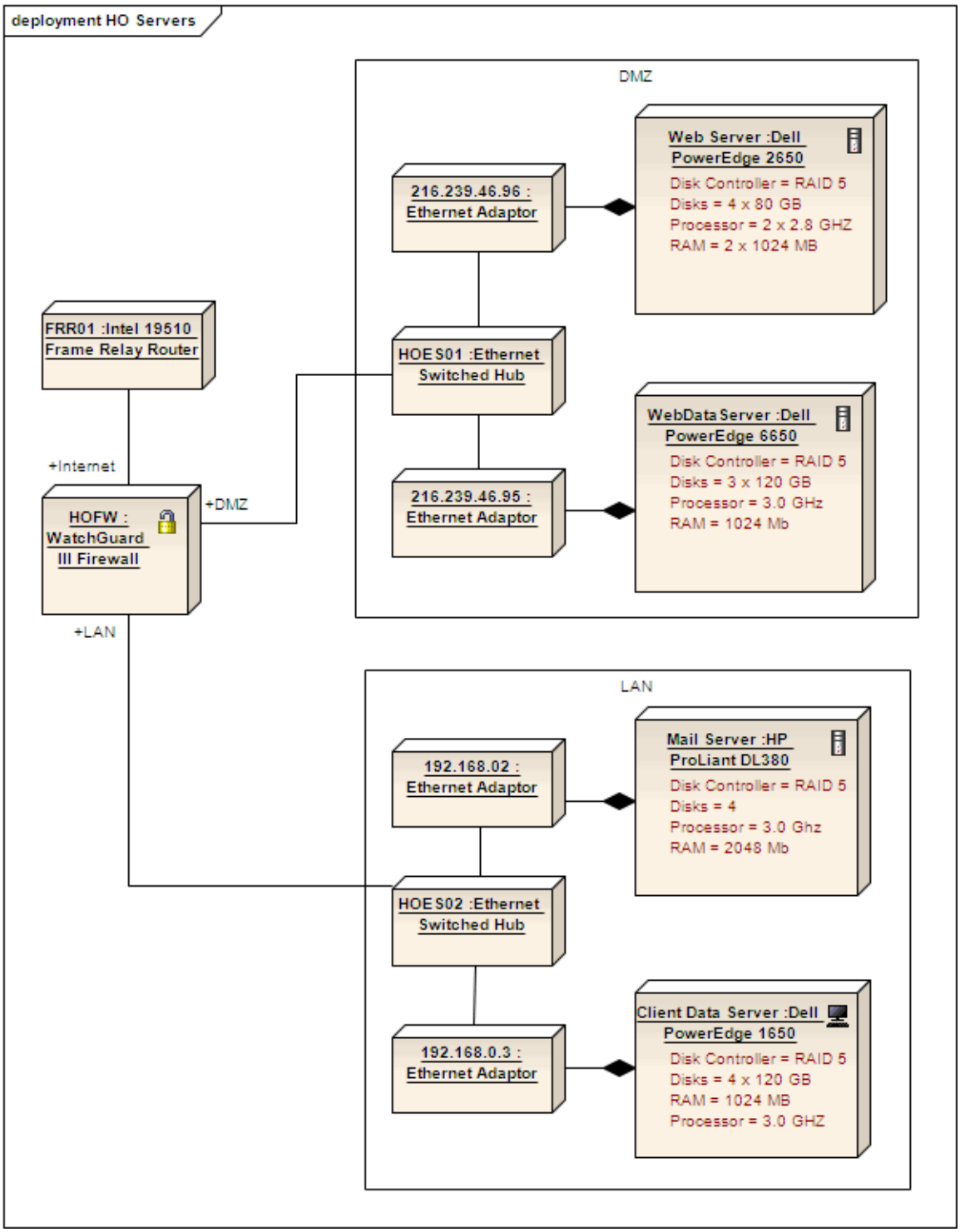
Hardware devices, processors and software execution environments (system Artifacts) are reflected as Nodes, and the internal construction can be depicted by embedding or nesting Nodes. Deployment relationships indicate the deployment of Artifacts, and Manifest relationships reveal the physical implementation of Components. As Artifacts are allocated to Nodes to model the system's deployment, the allocation is guided by the use of Deployment Specifications. A Deployment diagram can also indicate that a Node has a State, or show an instance of a Node with an actual run-time value for the state, representing a specific condition or scenario.

You generate Deployment diagram elements and connectors from the 'Deployment' pages of the Diagram Toolbox.

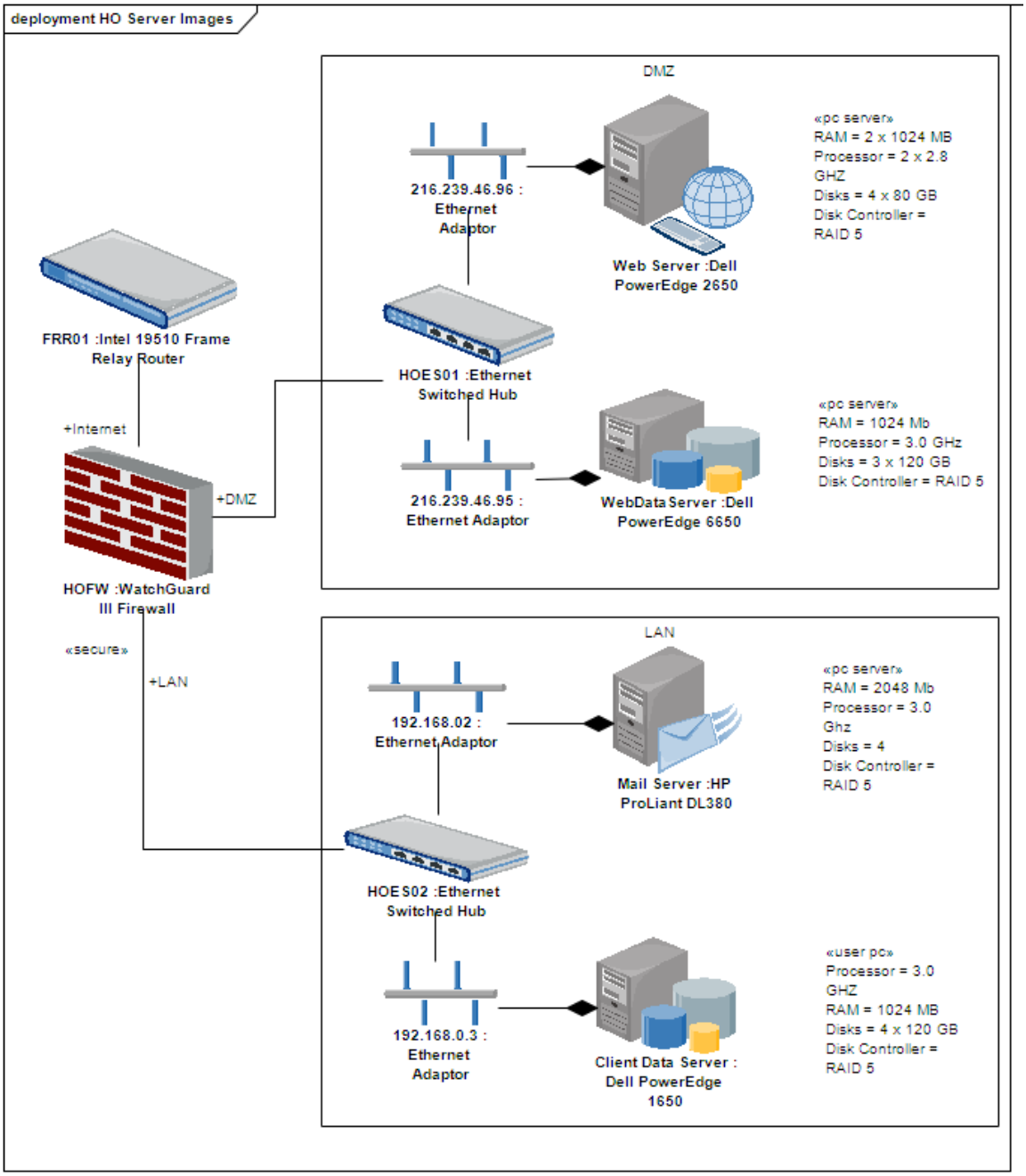
Example Diagram

This is a simple Deployment diagram, representing the arrangement of servers at a head office. The elements are instances of Nodes and show specific run-time states.

The servers are represented by Nodes linked by either simple or aggregate Association relationships.

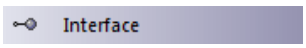
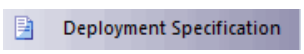


Deployment diagrams are ideal for applying alternative images to depict the objects that the elements represent. Such images can be substituted for the elements in the diagram, as shown here:

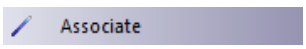
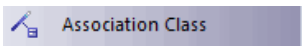
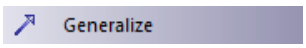
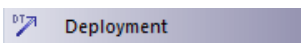


Deployment Diagram Element Toolbox Icons

Icon	Description
Node	A Node is a physical piece of equipment on which the system is deployed, such as

	a workgroup server or workstation.
	A Device is a physical electronic resource with processing capability upon which Artifacts can be deployed for execution, as represented in a Deployment diagram.
	An Execution Environment is a node that offers an execution environment for specific types of component that are deployed on it in the form of Executable Artifacts.
	A Component is a modular part of a system, whose behavior is defined by its provided and required interfaces.
	An Interface is a specification of behavior (or contract) that implementers agree to meet.
	An Artifact is any physical piece of information used or produced by a system.
	A Deployment Specification (spec) specifies parameters guiding deployment of an artifact, as is necessary with most hardware and software technologies.

Deployment Diagram Connector Toolbox Icons

Icon	Description
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	A Communication Path defines the path through which two DeploymentTargets are able to exchange signals and messages.
	An Association Class is a UML construct that enables an Association to have attributes and operations (features).
	A Generalization is used to indicate inheritance.
	A source object implements or Realizes its destination object.
	A Deployment is a type of Dependency

	relationship that indicates the deployment of an artifact onto a node or executable target, typically in a Deployment diagram.
	A Manifest relationship indicates that the Artifact source embodies the target model element, typically in Component and Deployment diagrams.

Object Diagram

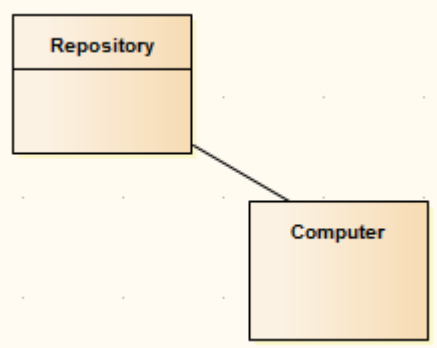
An Object diagram is closely related to a Class diagram, with the distinction that it depicts object instances of Classes and their relationships at a point in time. Object diagrams do not reveal architectures varying from their corresponding Class diagrams, but reflect multiplicity and the roles instantiated Classes could serve. They are useful in understanding a complex Class diagram, by creating different cases in which the relationships and Classes are applied

This might appear similar to a Composite Structure diagram, which also models run-time behavior; the difference is that Object diagrams exemplify the static Class diagrams, whereas Composite Structure diagrams reflect run-time architectures different from their static counterparts. An Object diagram can also be a kind of Communication diagram (which also models the connections between objects, but additionally sequences events along each path).

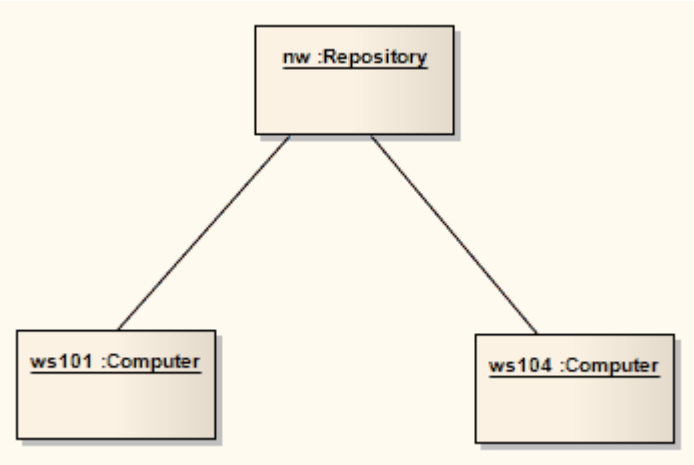
You generate Object diagram elements and connectors from the 'Object' pages of the Diagram Toolbox.

Example Diagram

This example shows a simple Class diagram, with two Class elements connected.

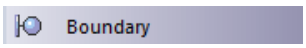


These Classes are instantiated as Objects in an Object diagram. There are two instances of Computer in this model, demonstrating the usefulness of Object diagrams in considering the relationships and interactions Classes might have in practice.

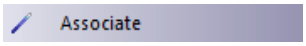


Object Diagram Element Toolbox Icons

Icon	Description
Actor	An Actor is a user of the system; user can mean a human user, a machine, or even another system or subsystem in the model.

	An Object is a particular instance of a Class at run time.
	A Collaboration defines a set of cooperating roles and their connectors.
	Use a Collaboration Use to apply a Pattern defined by a Collaboration to a specific situation, in a Composite Structure diagram.
	A Boundary is a stereotyped Object that models some system boundary, typically a user interface screen.
	A Control is a stereotyped Object that models a controlling entity or manager.
	An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system.
	An n-Ary Association element is used to model complex relationships between three or more elements, typically in a Class or Object diagram.

Object Diagram Connector Toolbox Icons

Icon	Description
	An Information Flow represents the flow of Information Items (either Information Item elements or classifiers) between two elements in any diagram.
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	Dependency relationships are used to model a wide range of dependent relationships between model elements in Use Case, Activity and Structural diagrams, and even between models themselves.

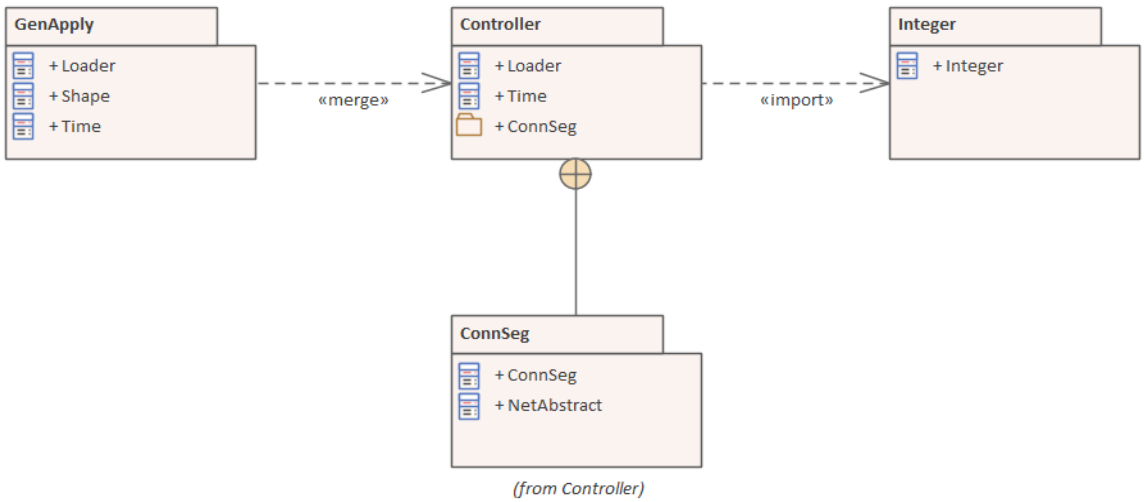
Package Diagram

Package diagrams depict the organization of model elements into Packages and the dependencies amongst them, including Package imports and Package extensions. They also provide a visualization of the corresponding namespaces.

You generate Package diagram elements and connectors from the 'Package' pages of the Diagram Toolbox.

Example Diagram

This example illustrates a basic Package diagram.



Package Diagram Element Toolbox Icons

Icon	Description
Package	Packages are used to organize your

	project contents, but when added onto a diagram they can be used to depict the structure and relationships of your model.
	Generates a Profile Package that has the stereotype «profile» in the Package diagram in your technical development model. A Profile Package is used in defining new types of structure in a model.
	Generates a Model Package with the stereotype «model», to represent the parent node in a model structure.

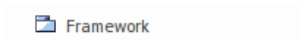
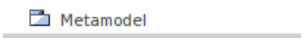
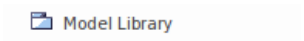
Package Diagram Relationship Toolbox Icons

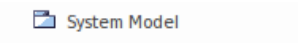
Connector	Description
c_nesting	The Nesting Connector is an alternative graphical notation for expressing containment or nesting of elements within other elements. The Nesting connector between ConnSeq and Controller reflects what the Package contents reveal. The Package contents can be listed by clicking on the diagram

	background to display the diagram's 'Properties' dialog, selecting the 'Elements' tab and selecting the 'Package Contents' checkbox in the 'Show Compartments' panel.
c_pkgmerge	In a Package diagram, a Package Merge indicates a relationship between two Packages whereby the contents of the target Package have been merged with those of the source Package. In the example diagram, the «merge» connector indicates that the Controller Package's elements have been imported into GenApply, including Controller's nested and imported contents. If an element already exists within GenApply, such as Loader and Time, these elements' definitions are expanded by those included in the Package Controller. All elements added or updated by the merge are noted by a generalization relationship back to that Package.
c_pkgimport	A Package Import relationship is drawn from a source Package to a Package whose contents have been imported. The «import» connector indicates that the

	elements within the target Integer Package, which in this example is the single Class Integer, have been imported into the Package Controller. The Controller's namespace gains access to the Integer Class; the Integer namespace is not affected.
c_profileapplication	A Profile Application relationship indicates that the source Profile has been applied to the target Package.

UML Standard Profile Toolbox Icons

Icon	Description
	Generates a Model Package with the stereotype «framework», to represent the parent node in framework structure.
	Generates a Model Package with the stereotype «metamodel», to represent the parent node in metamodel structure.
	Generates a Model Package with the stereotype «modelLibrary», to represent the parent node in model library structure.

	Generates a Model Package with the stereotype «systemModel», to represent the parent node in system model structure.

Profile Diagram

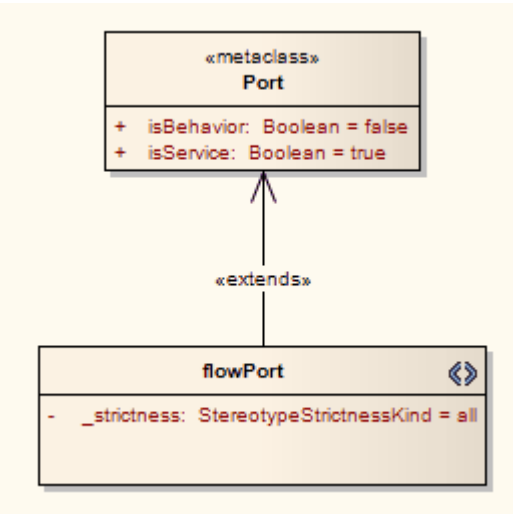
A Profile diagram is any diagram created in a ?profile? Package.

Profiles provide a means of extending the UML. They are based on additional stereotypes and Tagged Values that are applied to UML elements, connectors and their components. A Profile is a collection of such extensions that together describe some particular modeling problem and facilitate modeling constructs in that domain.

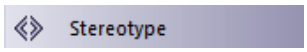
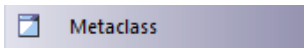
You generate Profile diagram elements and connectors from the 'Profile' pages of the Diagram Toolbox.

Example Diagram

A typical unit on a Profile diagram resembles this:

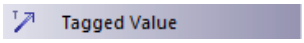


Profile Diagram Element Toolbox Icons

Icon	Description
	The first stage in creating a UML Profile is to create a Profile Package that has the stereotype «profile» in your technical development model.
	Stereotype elements represent the way in which each object is extended.
	Metaclass elements represent the types of object that you are extending in your Profile Package.
	An Enumeration is a data type, whose instances can be any of a number of user-defined enumeration literals.

Profile Diagram Connector Toolbox Icons

Icon	Description
	Connectors of type Extension represent an 'extents' relationship between two elements.
	A Generalization is used to indicate

	inheritance.
	A Tagged Value connector defines a reference-type (that is, RefGUID) Tagged Value owned by the source stereotyped element; the Tagged Value name is the name of the target role of this connector, and the Tagged Value is limited to referencing elements with the stereotype of the target element.

UML Behavioral Models

UML Behavioral Diagrams depict the elements of a system that are dependent on time and that convey the dynamic concepts of the system and how they relate to each other. The elements in these diagrams resemble the verbs in a natural language and the relationships that connect them typically convey the passage of time. For example, a behavioral diagram of a vehicle reservation system might contain elements such as Make a Reservation, Rent a Car, and Provide Credit Card Details. Experienced modelers will show the relationship to structural elements on these diagrams.

The UML defines seven types of behavioral diagram.

Diagram Types

Diagram Type	Detail
Activity Diagrams	Activity diagrams model the behaviors of a system, and the way in which these behaviors are related in an overall flow of the system.
Use Case Diagrams	Use Case diagrams capture Use Cases and relationships among Actors and the system; they describes the functional

	requirements of the system, the manner in which external operators interact at the system boundary, and the response of the system.
StateMachine Diagrams	StateMachine diagrams illustrate how an element can move between states, classifying its behavior according to transition triggers and constraining guards.
Timing Diagrams	Timing diagrams define the behavior of different objects within a time-scale, providing a visual representation of objects changing state and interacting over time.
Sequence Diagrams	Sequence diagrams are structured representations of behavior as a series of sequential steps over time. They are used to depict workflow, Message passing and how elements in general cooperate over time to achieve a result.
Communication Diagrams	Communication diagrams show the interactions between elements at run-time, visualizing inter-object relationships.

Interaction Overview Diagrams	Interaction Overview diagrams visualize the cooperation between Interaction diagrams (Timing, Sequence, Communication and other Interaction Overview diagrams) to illustrate a control flow serving an encompassing purpose.

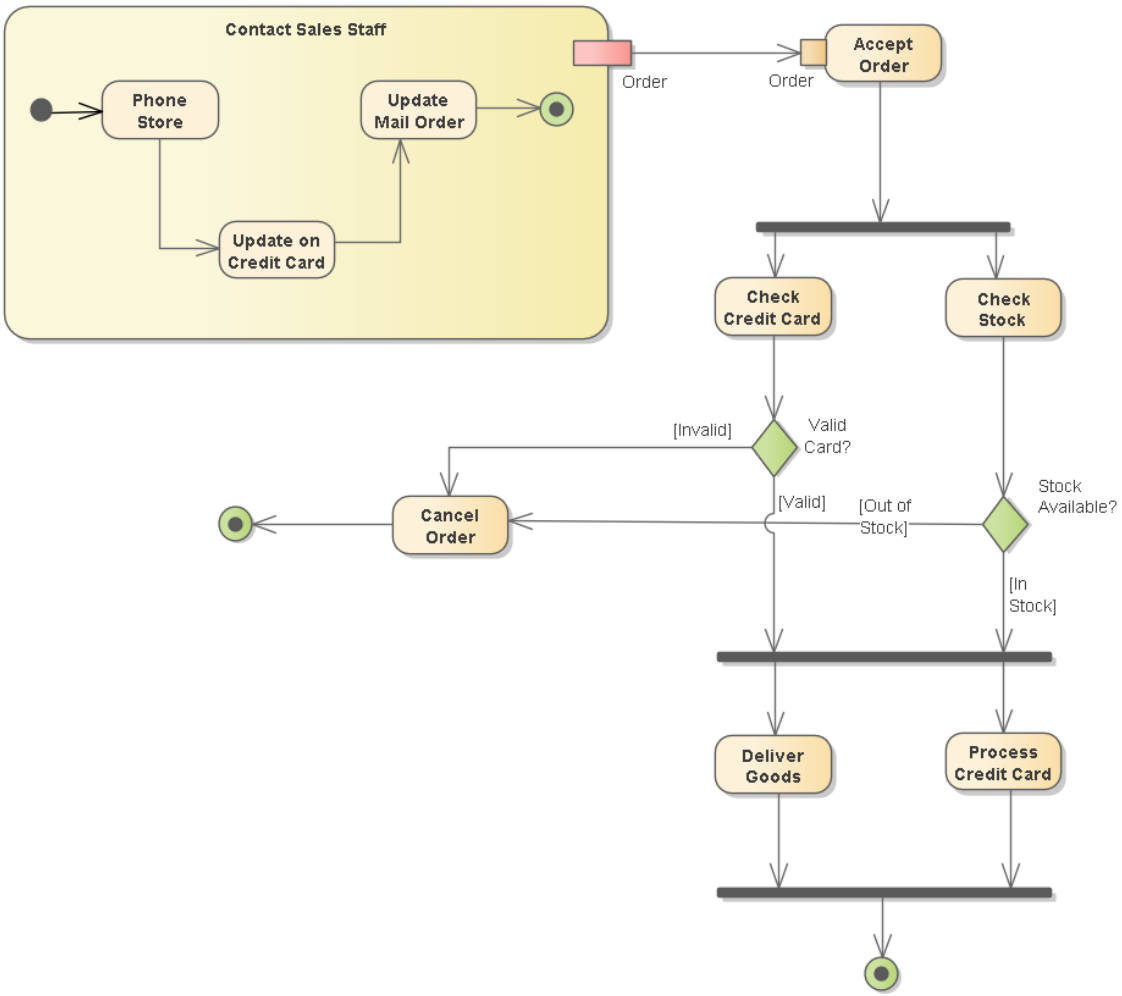
Activity Diagram

Activity diagrams are used to model system behaviors, and the way in which these behaviors are related in an overall flow of the system (that is, dynamic element interactions). The logical paths a process follows, based on various conditions, concurrent processing, data access, interruptions and other logical path distinctions, are all used to construct a process, system or procedure.

You generate Activity diagram elements and connectors from the 'Activity' pages of the Diagram Toolbox.

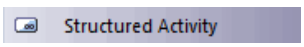
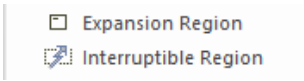
Example Diagram

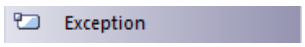
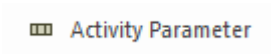
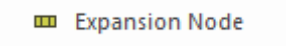
This diagram illustrates some of the features of Activity diagrams, including Activities, Actions, Start Nodes, End Nodes and Decision points.

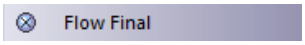


Activity Diagram Element Toolbox Icons

Icon	Description
Activity	An Activity element organizes and specifies the participation of subordinate behaviors, such as sub-Activities or Actions, to reflect the control and data flow of a process.
Action	An Action element describes a basic

	process or transformation that occurs within a system, and is the basic functional unit within an Activity diagram.
	A Partition element is used to logically organize an Activity's elements.
	The Send element depicts the action of sending a signal, in an Activity diagram.
	A Receive element defines the acceptance or receipt of a request, in an Activity diagram.
	A Structured Activity is an activity node that can have subordinate nodes as an independent Activity Group.
	Enterprise Architect supports two types of Region element: Expansion Regions and Interruptible Activity Regions. An Expansion Region surrounds a process to be imposed multiple times on the incoming data, once for every element in the input collection. An Interruptible Activity Region surrounds a group of Activity elements, all affected by certain interrupts in such a

	way that all tokens passing within the region are terminated should the interruption(s) be raised.
	The Exception Handler element defines the group of operations to carry out when an exception occurs.
	An Activity Parameter Node accepts input to an Activity or provides output from an Activity.
	An Object is a particular instance of a Class at run time.
	A Central Buffer Node is an object node for managing flows from multiple sources and destinations, represented in an Activity diagram.
	A Datastore defines permanently stored data.
	An Expansion Node is a shorthand notation to indicate that the Action/Activity consists of an Expansion Region.
	An Initial element is used to define the

	start of a flow when an Activity is invoked.
	In an Activity diagram or Interaction Overview diagram, a Decision indicates a point of conditional progression: if a condition is True, then processing continues one way; if not, then another.
	A Merge Node brings together a number of alternative flow paths in Activity, Analysis and Interaction Overview diagrams.
	A Synch state is useful for indicating that concurrent paths of a StateMachine are synchronized. It is used to split and rejoin periods of parallel processing.
	A Fork/Join element can be used to: 1) Split a single flow into a number of concurrent flows 2) Join a number of concurrent flows or 3) Both join and fork a number of incoming flows to a number of outgoing flows
	The Flow Final element depicts an exit from the system, as opposed to the

	Activity Final, which represents the completion of the Activity.
	The Activity Final element indicates the completion of an Activity; upon reaching the Final, all execution in the Activity diagram is aborted.

Activity Diagram Connector Toolbox Icons

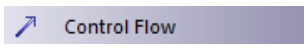
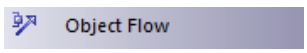
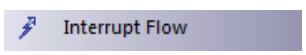
Icon	Description
	The Control Flow connects two nodes in an Activity diagram, modeling an active transition.
	An Object Flow connects two elements, with specific data passing through it, modeling an active transition.
	The Interrupt Flow defines the two UML concepts of connectors for Exception Handler and Interruptible Activity Region.

Diagram Orientation

On an Activity diagram, you can set the flow orientation to horizontal or vertical, or none (the default).

To set or clear the orientation, right-click on the diagram background and click on 'Set Diagram Flow Direction'. Then click on either:

- None (the default, no specific orientation set)
- Horizontal (diagram flows across the page, Pool and Lane elements occupy the full width of the diagram), or
- Vertical (diagram flows down the page, Pool and Lane elements occupy the full height of the diagram)

Notes

- You can create Analysis diagrams (Simplified Activity diagrams) containing the elements most useful for business process modeling, using the 'New Diagram' dialog
- You can perform model simulations on Activity models, and the model that you simulate can contain elements from more than one Package; to include the external elements in the simulation, you must create a Package diagram containing the 'parent' Package and the 'external' Packages containing the external elements, then create a Package Import connector from the parent Package to each external Package

Use Case Diagram

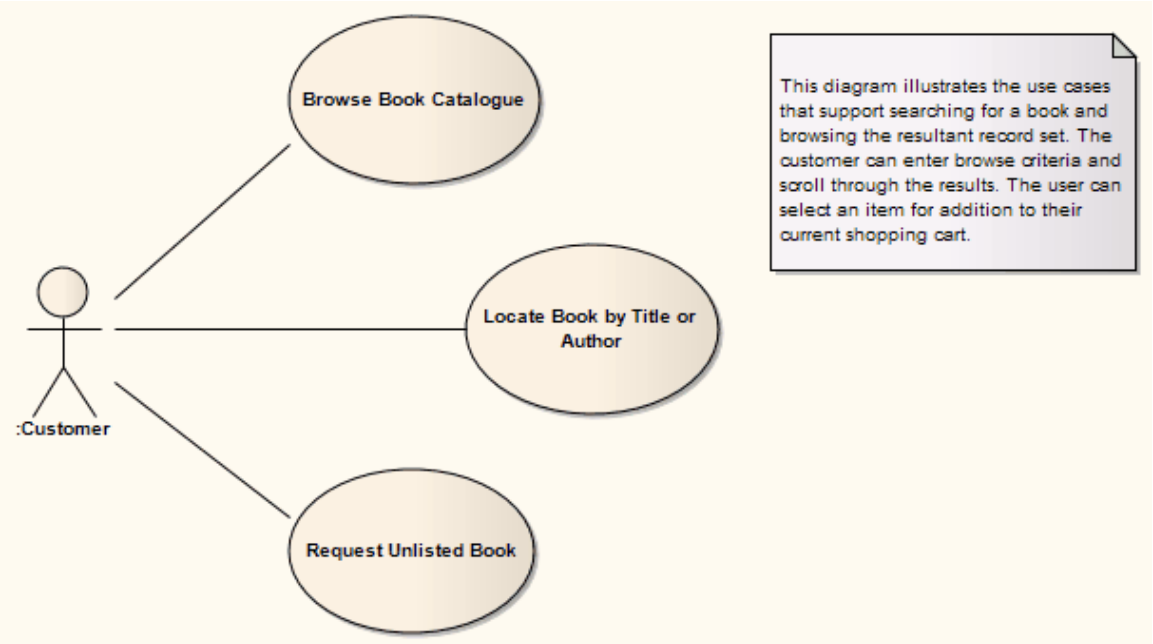
Use Case diagrams capture Use Cases and the relationships between Actors and the subject (system). You can use them to:

- Describe the functional requirements of the system
- Describe the manner in which outside things (Actors) interact at the system boundary
- Describe the response of the system

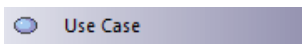
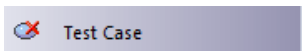
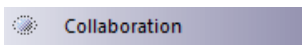
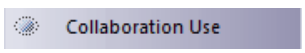
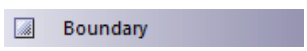
You generate Use Case diagram elements and connectors from the 'Use Case' pages of the Diagram Toolbox.

Example Diagram

This diagram illustrates some features of Use Case diagrams:

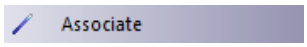
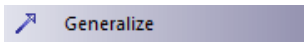


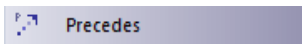
Use Case Diagram Element Toolbox Icons

Icon	Description
	An Actor is a user of the system; user can mean a human user, a machine, or even another system or subsystem in the model.
	A Use Case is a UML modeling element that describes how a user of the proposed system interacts with the system to perform a discrete unit of work.
	A Test Case is a stereotyped Use Case element which enables you to give greater visibility to tests.
	A Collaboration defines a set of cooperating roles and their connectors.
	A Collaboration Use element allows for a Pattern defined by a Collaboration to applied to a specific situation.
	A System Boundary element is a non-UML element used to define conceptual boundaries.

	Packages are used to organize your project contents, but when added onto a diagram they can be used to depict the structure and relationships of your model.

Use Case Diagram Connector Toolbox Icons

Icon	Description
	A Use relationship indicates that one element requires another to perform some interaction.
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	A Generalization is used to indicate inheritance.
	An Include connection indicates that the source element includes the functionality of the target element.
	An Extend connector is used to indicate

	that an element extends the behavior of another.
	A Realizes connector represents that the source object implements or Realizes its destination object.
	An Invokes connector indicates that source object, at some point, causes the destination object to happen.
	A Precedes connector indicates that the source object must be completed before the destination object can begin.

Notes

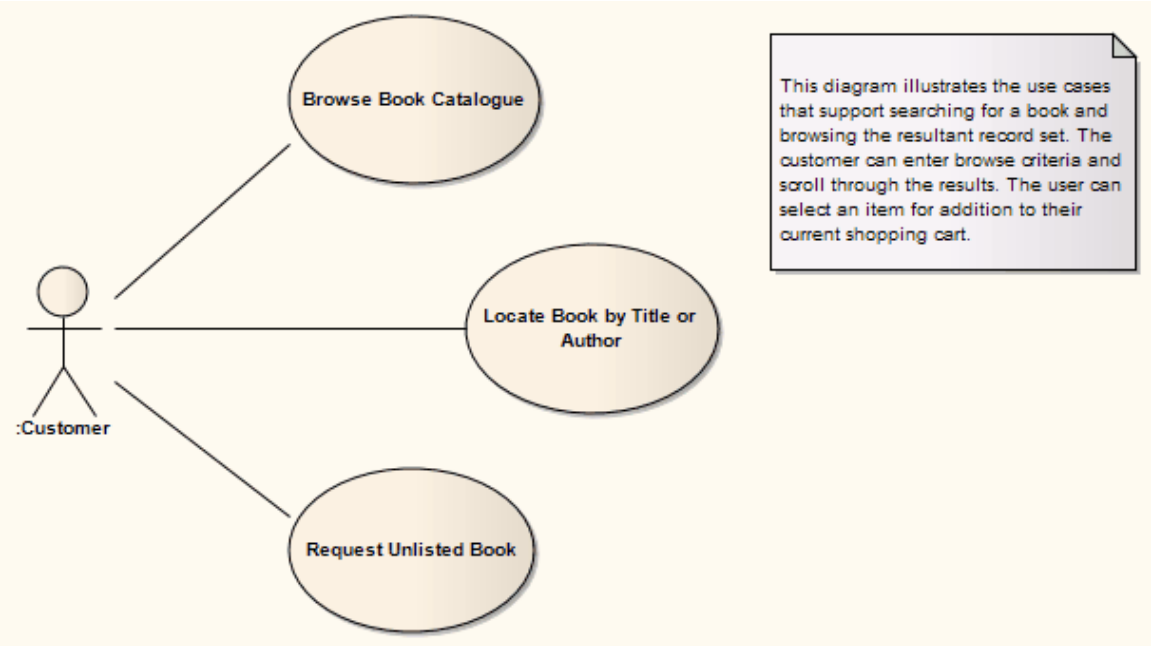
Invokes and Precedes are stereotyped Dependency relationships, defined by the OPEN Modeling Language (OML - Object-oriented Process, Environment and Notation Modeling Language - is an international de facto standard object-oriented development method developed and maintained by the OPEN Consortium). They have been incorporated into the Use Case modeling elements).

- Invokes indicates that Use Case A, at some point, causes Use Case B to happen
- Precedes indicates that Use Case C must complete before

Use Case D can begin

Example Use Case Diagram

This diagram illustrates some features of Use Case diagrams:



StateMachines

StateMachines illustrate how an element (often a Class) can move between States, classifying its behavior according to transition triggers and constraining guards.

You generate StateMachine elements and connectors from the 'State' pages of the Diagram Toolbox.

Naming

- StateMachines were formerly known as State diagrams
- StateMachine representations in UML are based on the Harel State Chart Notation and therefore are sometimes referred to as State Charts

State Tables

You can display a StateMachine as a diagram, or as a table in one of three relationship formats.

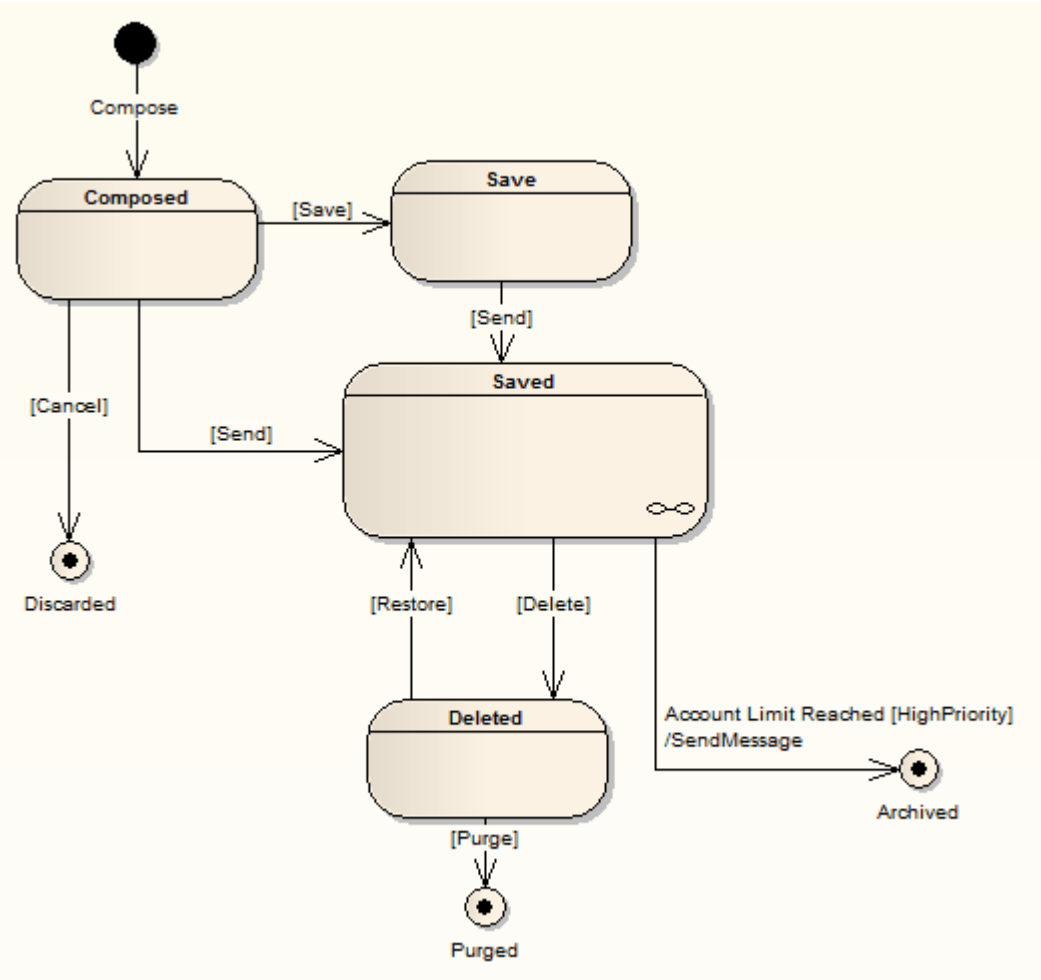
Select the display format

Step	Action
1	Right-click on the diagram background and select the 'Statechart Editor' option.

2	Select the appropriate display option: • Diagram • Table (State-Next State) • Table (State-Trigger) • Table (Trigger-State)

Example Diagram

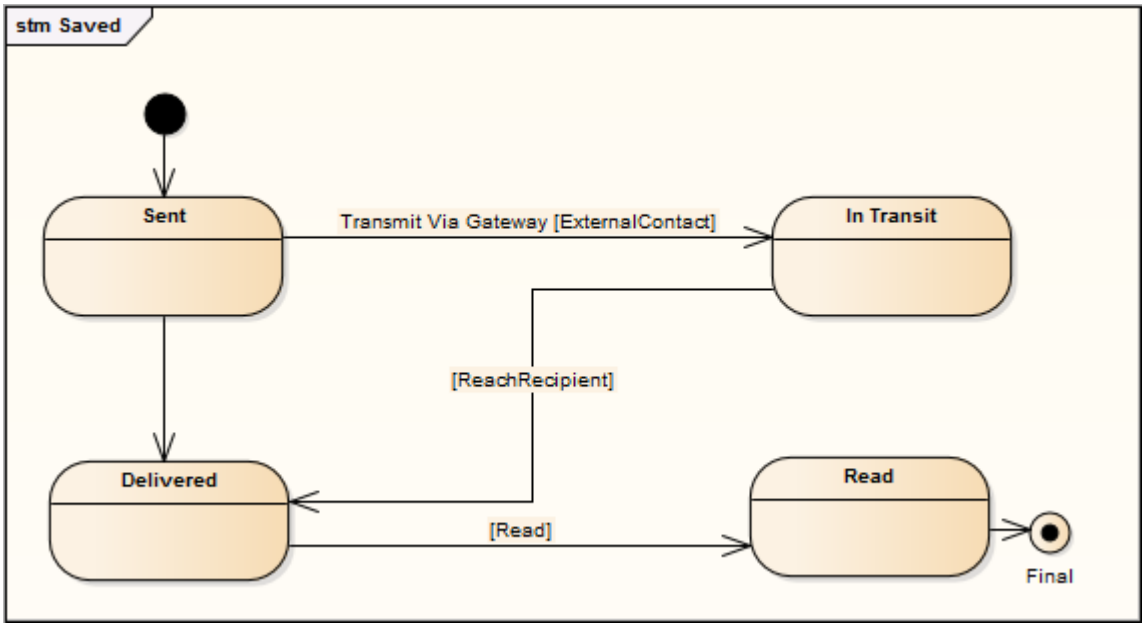
This diagram illustrates some features of StateMachines.



Composite Diagram States

The chain-link symbol in the bottom right corner of the Saved State indicates that it is a State with a Composite diagram.

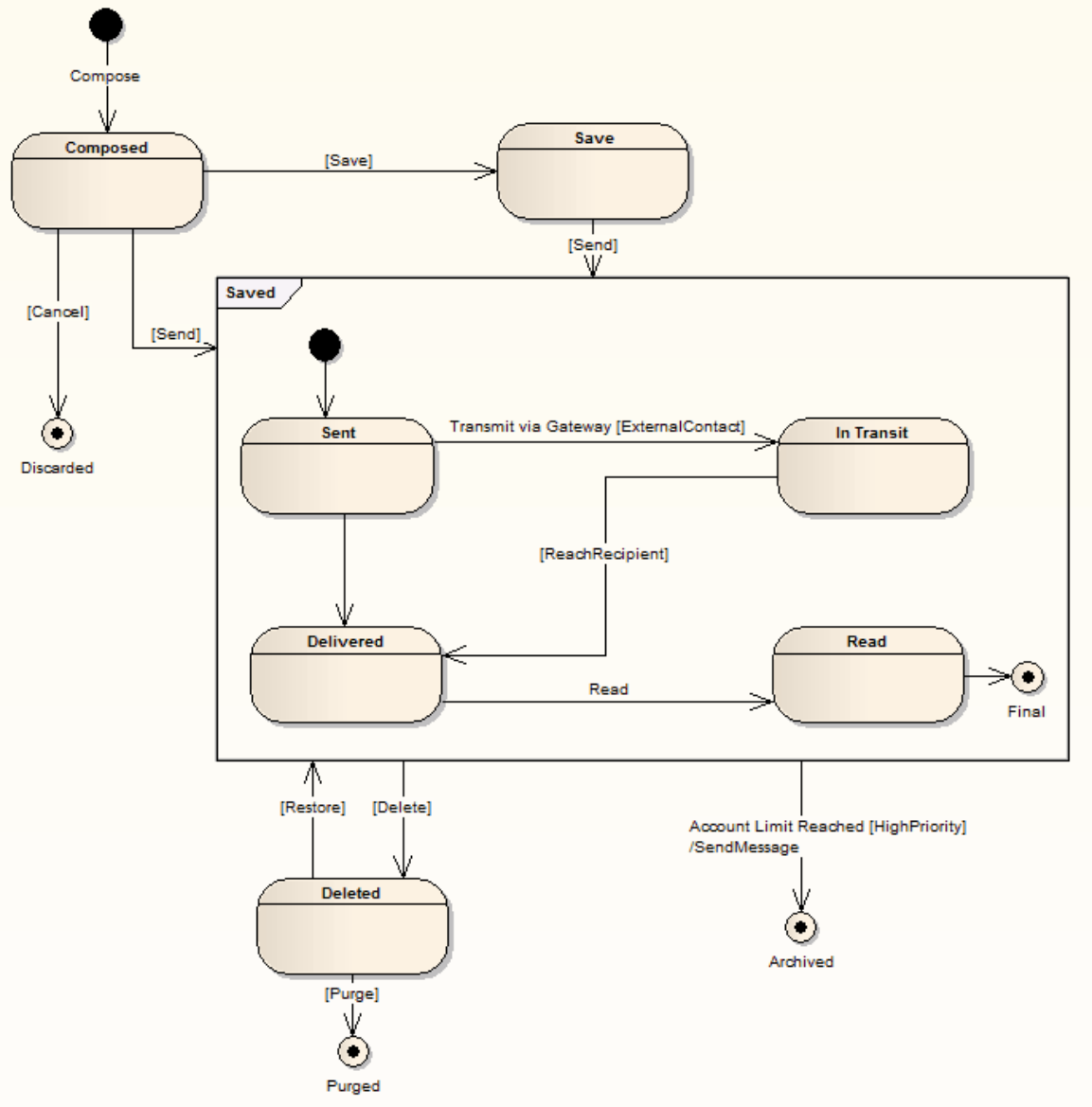
You have two options for displaying the contents of a State's Composite diagram. Firstly, you can double-click on the parent element to display its child diagram separately, as shown here:



By default, the child diagram displays within a labeled frame that represents the parent object in the context of the child diagram. You can right-click on the background and select the 'Hide Diagram Frame' option to hide the frame, and on the 'Show Diagram Frame' option to show the frame again.

Alternatively, you can right-click on the composite element on the main diagram and select the 'Advanced | Show Composite Diagram' option, which again displays the child diagram in a labeled frame, but this time within the context

of the parent diagram.



ProtocolStateMachines

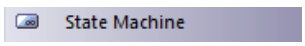
The OMG UML specification (UML Superstructure Specification, v2.5, sect. 14.4) states:

"ProtocolStateMachines are used to express usage protocols. ProtocolStateMachines express the legal sequences of Event occurrences to which the Behaviors of an associated BehavioredClassifier must conform. The StateMachine

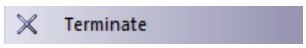
notation is a convenient way to define the order of invocations of the behavioral features of a Classifier. ProtocolStateMachines can be associated with Classifiers, Interfaces, and Ports."

To create a ProtocolStateMachine, create a StateMachine element and open the Properties window for that element. Select the 'Behavior' tab and, on that, select the 'Protocol StateMachine' checkbox. The element on the diagram now has the word <<protocol>> above the element name.

StateMachine Diagram Element Toolbox Icons

Icon	Description
	A State represents a situation where some invariant condition holds; this condition can be static (waiting for an event) or dynamic (performing a set of activities).
	A StateMachine element is a container for groups of related State elements.
	The Initial element represents a pseudostate used to denote the default state of a Composite State; there can be one Initial vertex in each Region of the

	Composite State.
	The Activity Final element indicates the completion of an Activity; upon reaching the Final, all execution in the Activity diagram is aborted.
	There are two types of History pseudostate defined in UML: shallow and deep history.
	A Synch state is useful for indicating that concurrent paths of a StateMachine are synchronized. They are used to split and rejoin periods of parallel processing.
	An Object is a particular instance of a Class at run time.
	The Choice pseudostate is used to compose complex transitional paths, where the outgoing transition path is decided by dynamic, run-time conditions.
	Junction pseudostates are used to design complex transitional paths in StateMachine diagrams. A Junction can be used to combine or merge multiple paths into a shared transition path.

	Entry Point pseudostates are used to define the beginning of a StateMachine. An Entry Point exists for each region, directing the initial concurrent state configuration.
	Exit Points are used in StateMachine elements and StateMachine diagrams to denote the point where the machine is exited and the transition sourcing this exit point.
	The Terminate pseudostate indicates that upon entry of its pseudostate, the StateMachine's execution ends.
	A Fork/Join element can be used to: 1) split a single flow into a number of concurrent flows, 2) join a number of concurrent flows or 3) both join and fork a number of incoming flows to a number of outgoing flows.
	A Fork/Join element can be used to: 1) Split a single flow into a number of concurrent flows 2) Join a number of concurrent flows or 3) Both join and fork a number of

	incoming flows to a number of outgoing flows

StateMachine Diagram Connector Toolbox Icons

Icon	Description
	A Transition connector represents the logical movement from one State to another in a StateMachine diagram.
	An Object Flow connects two elements, with specific data passing through it, modeling an active transition.

Notes

- State elements can display either with or without a line across them; the line - as shown - displays when the element has features such as operations (which could be hidden) or when the 'Show State Compartment' checkbox is selected in the 'Objects' page of the 'Preferences' dialog
- It is possible to add Entry Point and Exit Point elements to the border of a State or StateMachine element - right-click

on the element in the diagram and select the 'New Child Element| Entry Point' or 'Exit Point' option; if the element is a composite element and represented by a frame, you can also right-click on the selected frame and add the Entry Point or Exit Point elements

- If you have Entry Points and/or Exit Points on a StateMachine that is a classifier for another State, you can create Connection Point References to the classifier from the other State
- It is also possible to add Regions to a State element or StateMachine element frame; right-click on the selected frame and select the 'Define Concurrent Substates' option
- You can perform model simulations on StateMachine models, and the model that you simulate can contain elements from more than one Package; to include the external elements in the simulation, you must create a Package diagram containing the 'parent' Package and the 'external' Packages containing the external elements, and then create a Package Import connector from the parent Package to each external Package

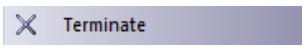
Pseudostates

Pseudostates are a UML abstraction for various types of transient vertex used in StateMachine diagrams. Pseudostates are used to express complex transition paths.

You can create a Pseudostate by dragging one of these element icons onto a diagram in Enterprise Architect.

Diagram Toolbox Icons

Icon	Description
	The Initial element represents a pseudostate used to denote the default state of a Composite State; there can be one Initial vertex in each Region of the Composite State.
	Entry Point pseudostates are used to define the beginning of a StateMachine. An Entry Point exists for each region, directing the initial concurrent state configuration.
	Exit Points are used in StateMachine elements and StateMachine diagrams to denote the point where the machine is exited and the transition sourcing this exit

	point.
	The Choice pseudostate is used to compose complex transitional paths, where the outgoing transition path is decided by dynamic, run-time conditions.
	Junction pseudostates are used to design complex transitional paths in StateMachine diagrams. A Junction can be used to combine or merge multiple paths into a shared transition path.
	There are two types of History pseudostate defined in UML: shallow and deep history.
	The Terminate pseudostate indicates that upon entry of its pseudostate, the StateMachine's execution ends.
	The Activity Final element indicates the completion of an Activity; upon reaching the Final, all execution in the Activity diagram is aborted.
	A Fork/Join element can be used to: 1) split a single flow into a number of concurrent flows, 2) join a number of

	concurrent flows or 3) both join and fork a number of incoming flows to a number of outgoing flows.

Notes

- All the listed types of pseudostate can be represented in code, and can generate code under the StateMachine code generation templates from Enterprise Architect release 11 onwards

Regions

If you are modeling an active State configuration on a StateMachine diagram, and you need to represent several States as being active concurrently, you can achieve this by firstly creating a StateMachine element or Composite State element and secondly subdividing that element into Regions. You set out the State configuration such that there is only ever one of the concurrently active States per Region. Multiple transitions can occur from a single event dispatch, so long as the similarly-triggered transitions are divided by Regions.

Regions display on an element on a diagram as subdivisions of a structured compartment, underneath other compartments such as tags, responsibilities, attributes and operations.

Access

Context Menu	Right-click on element | Advanced | Define Concurrent Substates

Create a Region in a Composite State or StateMachine element

Step	Action
1	On the 'State Regions' dialog, the 'Name' field defaults to '<anonymous>'.
2	If you want to create Regions that have no title, simply click on the Save button once for each Region to create. If you want to create named Regions, type the name and click on the Save button for each Region.
3	When you have created as many Regions as you need, click on the Close button. You can now populate the Regions with elements from the 'State' pages of the Diagram Toolbox.

Notes

- Changes to the elements in a Region are committed when the diagram is saved; if you want to undo the changes, reload the diagram without saving
- Any States, State Nodes (Pseudo-States) or Synch elements added to a Region are owned by that Region and, ordinarily, cannot be dragged into another Region; however, if you attempt to drag a State between Regions, the 'Move embedded element to region' menu option

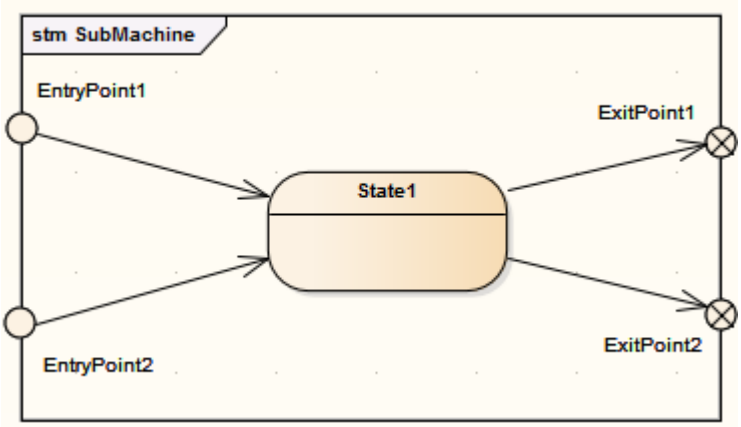
displays which - if you select it - allows the transfer to complete

Create a Connection Point Reference

A Connection Point Reference represents the use, by a Submachine State, of an Entry Point or Exit Point pseudostate defined in the State element's classifier StateMachine. You initially create the Connection Point Reference elements themselves as Entry Points or Exit Points.

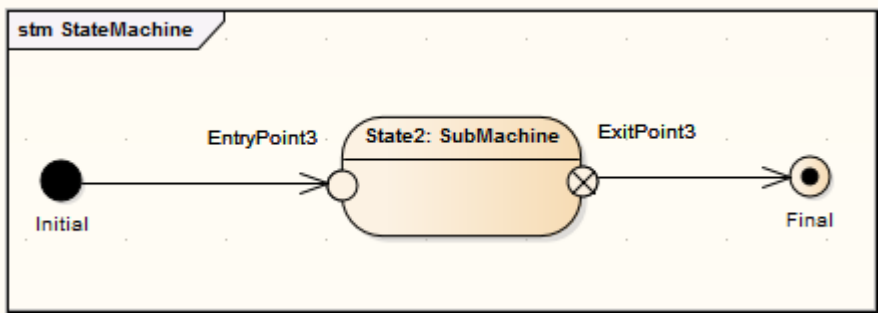
Create Entry Points and/or Exit Points

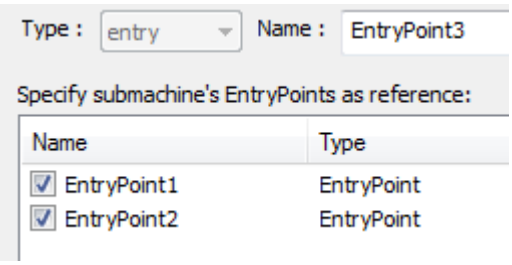
Step	Action
1	Create or open the classifier StateMachine (as a child diagram of a Class element). The StateMachine is represented by a labeled frame.
2	If the Entry Points and/or Exit Points do not already exist, right-click on the inside edge of the frame and select the 'New Element | Entry Point' or 'New Element | Exit Point' option, as necessary. The corresponding pseudostate element is immediately created on the edge of the frame. If you prefer, you can double-click on the element and give it a specific name.
3	Create as many additional Entry Point and/or Exit

	Point elements as you need.
4	If the corresponding State element does not already exist, drag a State icon from the Diagram Toolbox into the frame. Create the appropriate connectors between the State element and the Entry Point and Exit Point elements. 
5	Save the diagram.

Create Connection Point References

Step	Action
1	Create or open the calling StateMachine (as a child diagram of a Class element).

2	If the elements do not already exist, create the appropriate State and pseudostate elements and connectors in the diagram.
3	Click on the calling State element and press Ctrl+L to display the 'Select Element' dialog. Browse for and select the classifier StateMachine from the 'Create Entry Points and/or Exit Points' stage.
4	Right-click on the State element, and select the 'New Element | Entry Point' or 'New Element | Exit Point' option, as you need. The corresponding pseudostate element is immediately created on the border of the element. 
5	Double-click on the Entry Point element. The 'Edit ConnectionPointReference' dialog displays.
6	If you prefer, in the 'Name' field type a new name for the selected Entry Point.

	In the 'Specify submachine's EntryPoints as reference' panel, select the checkbox against each of the classifier's Entry Points to create a reference to. You can select more than one checkbox. 
7	Click on the OK button.
8	If necessary, repeat steps 4 to 7 for the State element's Exit Point.

StateMachine Table

A StateMachine table is one of two variants of a StateMachine (the other is the StateMachine diagram). It displays the information of the StateMachine in table form, and is a method of specifying the discrete behavior of a finite state-transition system; that is, what state the StateMachine moves to and the conditions under which the transition takes place.

Access

Context Menu	Right-click on the background of a StateMachine diagram | Statechart Editor | Table (option)

StateMachine Table Display

You can display the State transition in the table as one of two different types of relationship:

Type	Description
State - Trigger	The rows indicate the current states and the columns indicate trigger events. The cell at the intersection of a row and column identifies the target state in the

	transition if the trigger occurs, and the condition (or guard) of the transition, or the other way around if you prefer, in a Trigger - State format.
State - Next State	The rows and columns both indicate states, and the cell at the intersection of a row and column indicates: • The event that triggers a transition from the current (row) state to the next (column) state • The condition (or guard) of the event, and • The effect of the transition

Select the display format

Step	Action
1	Right-click on the diagram background and select the 'Statechart Editor' option.
2	Select the appropriate display option: • Diagram • Table (State-Next State)

	• Table (State-Trigger) • Table (Trigger-State)

StateMachine Table Options

You can choose the StateMachine table layout and set other options from the 'StateMachine Diagram: Options' dialog, which you display by either:

- Double-clicking on the StateMachine table background or
- Right-clicking on the background and selecting the 'State Table Options' option

Options

Option	Action
Table Format	Select the required table format. State - Trigger: • Rows represent States, each State name in a left edge cell • Columns represent Triggers, each Trigger name in a column header cell • The intersection of a row and column identifies the Transition (if there is one) • The Transition cell displays information about the next State and the condition (guard) of the Transition Trigger - State: as for State - Trigger, except that rows represent Triggers and columns represent States.

	State - Next State: • Both rows and columns represent States • The intersection of row and column defines the transition (if there is one) from the row State to the column State
Cell Size	Complete the next four fields.
Transition Cell Width	Specify the width of the transition cells (that is, the column width).
Transition Cell Height	Specify the height of the transition cells (that is, the row height).
Left Edge Cell Width	Specify the width of the left edge (row title) cells.
Top Edge Cell Height	Specify the height of the top edge (column title) cells.
Cell Color	Complete the next three fields.
State/Trigger Cell	Select the color of the row and column title cells.
State/Trigger Enumeration	Select the color of the enumeration (row/column numbering) cells.

	You must select at least one of the 'Enable State Enumeration' and 'Enable Event Enumeration' checkboxes to set this color.
Transition Cell	Select the color of the transition cells (in the main body of the table).
Highlight Options	
Highlight Zones Related to Selected Transition	Highlight the cells for all elements involved in a selected transition - the initial state, the target state, and the trigger.
Highlight Color	Select the color of the highlight.
Use Different Color for Target State	Highlight the cell for the target element in a transition in a different color to the cell for the source element.
Target Zone Color	Select the color of the highlight.
Display Options	

Always Display an Empty State Zone	Add an empty row (and, on a State - Next State table, an empty column) to the end of the table. The title cell contains a ... button. You can click twice (not double-click) on the button to edit the cell and identify a new state. In this case, another empty state zone is automatically added.
Enable State Enumeration	Add a cell to each state title cell, to number the state. Numbering starts at 0.
Prefix	If required, type a prefix for the state number or delete the default 'S' to have no prefix.
Enable Event Enumeration	Add a cell to each event or trigger title cell, to number the event. Numbering starts at 0.
Prefix	If required, type a prefix for the event number or delete the default E to have no prefix.
Sample State Table	Display a preview of the table format as you define it.
Advanced	Define diagram options. The

	StateMachine diagram 'Properties' dialog displays.
Restore Defaults	Reapply the State Table diagram default values.
Apply	Apply the changed options to the State Table diagram.

StateMachine Table Operations

As a StateMachine table is a variant of a StateMachine diagram, most of the operations for manipulating the data are the same as for StateMachine diagrams. The operations specific to StateMachine tables are described in these topics:

Operations

Operation
Change StateMachine Table Position
Change StateMachine Table Size
Insert New State
Insert Trigger
Insert/ChangeTransition
Reposition State or Trigger Cells
Add Legend
Locate Cell in StateMachine diagram

StateMachine Table Conventions

Export State Table To CSV File

Change StateMachine Table Position

If necessary, you can move the StateMachine table around in the Diagram View.

Change the position of the StateMachine table

Step	Action
1	Press Ctrl+A or double-click on the top left cell to select the whole StateMachine table.
2	Drag and drop the StateMachine table to the required position. Alternatively, use Shift+Right Arrow, Left Arrow, Up Arrow or Down Arrow to move the StateMachine table.

Change StateMachine Table Size

There are three ways to change the size of the StateMachine table:

- Change the cell size on the 'StateMachine Diagram: Options' dialog
- Press Ctrl+A or double-click on the top left cell to select the whole StateMachine table, then press Ctrl+ 'Left', 'Up', 'Right', or 'Down' to change the size
- Select the StateMachine table, then drag the shape handles to change the size

Insert Trigger

If the StateMachine table format is either State-Trigger or Trigger-State, you can use any of these methods to insert a new Trigger:

Methods

Step	Action
1	In the top left cell in the StateMachine table, move the cursor to the word 'Event' to display a + at the end of the word; click on the + to create a new Trigger.
2	In the top left cell in the StateMachine table, right-click and select the 'Add Trigger' option to create a new Trigger.
3	Select an existing Trigger in the StateMachine table, then press the Insert key to insert a new Trigger before the existing Trigger.
4	Click on an existing Trigger in the StateMachine table, right-click and select either the: • 'Insert New Trigger Before' option to insert a new Trigger before the current Trigger, or

	• 'Insert New Trigger After' option to insert a new Trigger after the current Trigger

Insert/Change Transition

This topic explains how you can insert or modify a transition link between two State elements.

Options

Action	Description
Insert a new Transition	You can insert a new Transition using one of these methods. Right-click on the cell in which to create a Transition: • If the StateMachine table format is State-Trigger or Trigger-State, the context menu lists the States you can choose as the target of the Transition; click on the required State name to create the Transition • If the StateMachine table format is State-Next State, click on the 'Insert Transition' context menu option to create the Transition Alternatively, in the 'State Relationships' page of the Toolbox, select the Transition element, then click on the cell in the StateMachine table in which to create the Transition; double-click on the Transition

	to define it in the 'Transition Properties' dialog.
Change the Transition	As for the State Chart diagram, to change the properties of a Transition double-click on the 'Transition' cell and edit the details on the 'Transition Properties' dialog.
Change Transition States	You can change the source and target of the Transition by right-clicking the Transition and selecting the 'Advanced | Set Source and Target' option. Alternatively, you can change the Transition source, target or Trigger by clicking on the Transition and dragging it to a different cell. If the StateMachine table format is either State-Trigger or Trigger-State, you can change the target state of a Transition by: 1. Highlighting the target state name in the Transition cell and clicking on it to display a list of the states in the table. 2. Clicking on the preferred target state name.
Highlight States and Trigger Related to	You can select options to highlight the source State, target State and Trigger cells associated with a Transition, using the 'Highlight Options' panel on the

Transition	'StateMachine Diagram: Options' dialog. When you click on the Transition cell its associated State and Trigger cells are highlighted. Alternatively, click on the Transition cell and press and hold the L key.

Insert New State

Options

Action	Description
Insert a new State in the StateMachine table	You can insert a new State in the StateMachine table, using one of these methods: 1. In the top left cell in the StateMachine table, move the cursor to the word State to display a + at the end of the word; click on the + to create a new State 2. Right-click in the top left cell in the StateMachine table and select 'Add State' 3. Right-click on an existing State cell in the StateMachine table and select: - 'Insert New State Before' to insert a new State before the current State, or - 'Insert New State After' to insert a new State after the current State 4. Click on an existing State cell in the StateMachine table, and press the Insert key to create and insert a new State

	above the selected State 5. In the Toolbox, on the 'State Elements' page, click on an element and then click on: - The diagram background to add a new State to the end of the table, or - An existing State cell to add the new State just above it From the 'State Elements' page of the Toolbox you can insert State, Initial, Final, Entry, Exit and Terminate elements.
Add a Substate to a selected State	To add a Substate to a selected State, right-click on the required State cell in the StateMachine table, and select 'Add Substate'; Enterprise Architect adds the Substate to the State. If the selected State does not allow a Substate, the 'Add Substate' option is grayed out. You can also drag one existing State over another; if the second State allows Substates, the dragged State then becomes its Substate. Similarly, you can change the parent State of a Substate by dragging the Substate from the original parent State to

	a different State.
Remove the parent relation of a Substate and make it a separate State	To remove the parent relation of a Substate and make it a separate State, right-click on the Substate in the StateMachine table and select 'Remove Parent Relation'; the Substate cell becomes a State cell. You can also drag and drop the Substate onto the top left cell of the StateMachine table; the dragged Substate again becomes a State cell.

Reposition State or Trigger Cells

You can change the position of a selected State or Trigger cell in one of these ways:

- Right-click on the State or Trigger title cell and select the appropriate 'Order | Move xxx' option
- Click on the cell and press Shift+Right Arrow, Left Arrow, Up Arrow or Down Arrow

Add Legend

You can add a simple legend to any StateMachine Table cell that has no transition. The two legend symbols are:

- I - Ignore
- N - Never Happen

Assign a legend symbol to a StateMachine Table cell

Step	Action
1	Click on the cell to which to assign the legend and press: • The I key to insert the 'Ignore' legend, or • The N key to insert the 'Never Happen' legend The required symbol displays in the center of the cell.

Alternatively

Step	Action

1	Right-click on the cell to which to assign the legend.
2	Select the appropriate context menu option: • Legend | Ignore • Legend | Never Happen The required symbol displays in the center of the cell.

Notes

- To remove a legend symbol from a cell, either:
- Click on the cell and press Delete, or
- Right-click on the cell and select Legend | Remove Legend

Find Cell in StateMachine Diagram

Locate In State Chart

On the StateMachine table, to locate a selected State or Trigger element in a StateMachine diagram:

- Select 'Find | Locate in State Chart'

Enterprise Architect switches to the StateMachine diagram and highlights the selected element.

You can locate a Transition relationship in a similar way, by selecting 'Locate in State Chart'.

A Trigger on a StateMachine table might or might not exist on the corresponding StateMachine diagram; if the Trigger does not exist on the StateMachine diagram, the 'Locate in State Chart' option is disabled.

Locate In State Table

On the StateMachine diagram, to locate a selected State or Trigger element in the corresponding StateMachine table:

- Select 'Find | Locate in State Table'

Enterprise Architect switches to the StateMachine table and highlights the selected element.

You can locate a Transition relationship in a similar way, by selecting 'Locate in State Table'.

StateMachine Table Conventions

Trigger

- Deleting a Trigger removes it completely from the model, therefore you cannot UNDO a Trigger deletion
- There is a <None> column at the end of the Event heading row; this is for Transitions that have no Trigger information

State

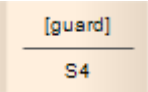
From the Toolbox you can insert these State element types only (although the StateMachine table might pick up and display other types, such as Submachine State):

- State
- Initial
- Final
- Entry
- Exit
- Terminate

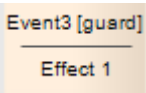
Transition

The Transition cell displays its properties in one of two ways, depending on the StateMachine table format; if the StateMachine table format is State - Trigger or Trigger -

State, the Transition cell displays the Guard and Target as shown:



If the StateMachine table format is State - Next State, then the Transition cell displays the Trigger, Guard and Effect in this format:



In the StateMachine table, you can edit the Guard and Effect in place. If the Guard or Effect is empty for your selected Transition cell, the cell displays an ellipsis (...) instead; click twice (not double-click) on the ellipsis to type in the Guard and Effect names.

Export State Table To CSV File

Export a StateMachine Table to a CSV file

Step	Action
1	Open the required StateMachine Table.
2	Right-click on the diagram background and select the 'Export Statechart to CSV file' option. The 'Save As browser' dialog displays.
3	Select the appropriate directory location and type in the .csv filename.
4	Click on the Save button.

Example State-Trigger Table

The rows indicate the current states and the columns indicate trigger events (or the other way around if you prefer, in a Trigger - State format).

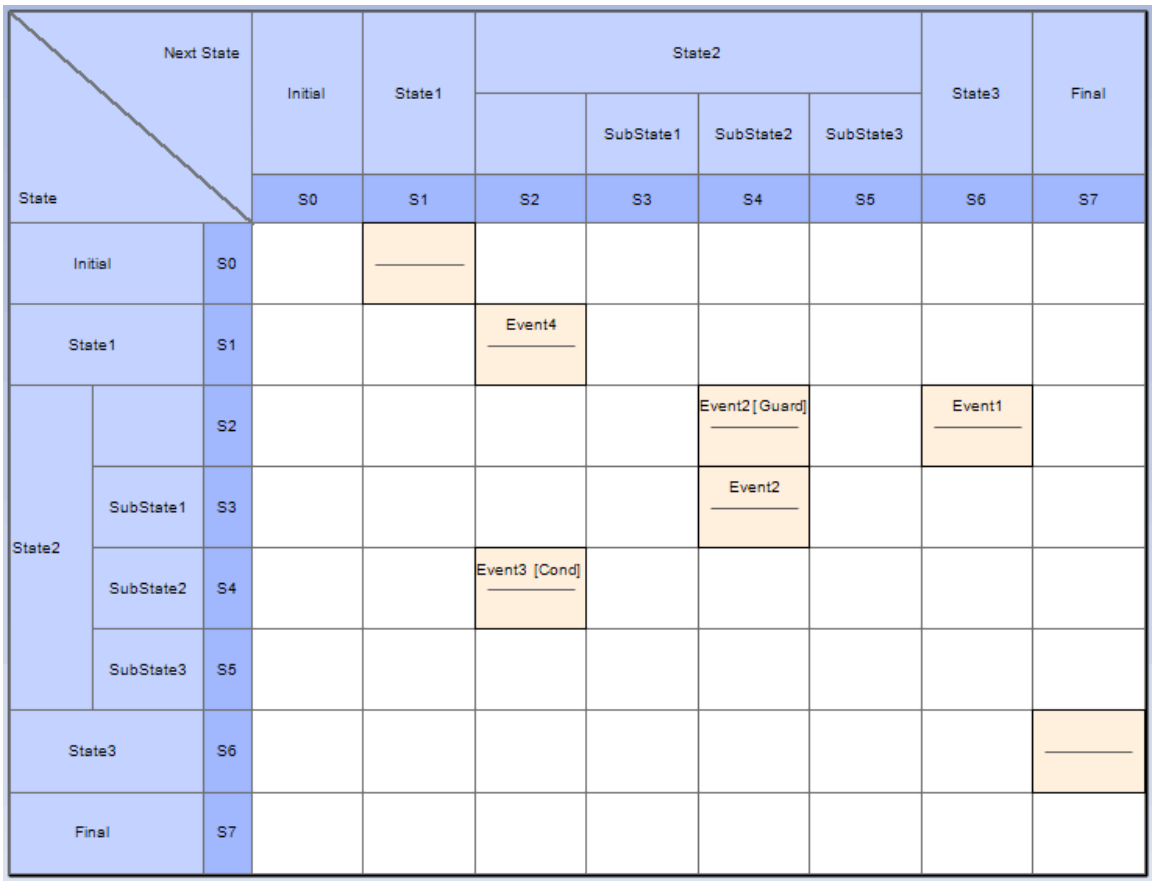
The cell at the intersection of a row and column identifies the target state in the transition if the trigger occurs, and the condition (or guard) of the transition.

State \ Trigger			Event1	Event2	Event3	Event4	<None>
			E0	E1	E2	E3	E4
Initial		S0					S1
State1		S1				S2	
State2		S2	S6	[Guard] / S4			
	SubState1	S3		S4			
	SubState2	S4			[Cond] / S2		
	SubState3	S5					
State3		S6					S7
Final		S7					

Example State-Next State Table

The rows and columns both indicate states, and the cell at the intersection of a row and column indicates:

- The event that triggers a transition from the current (row) state to the next (column) state
- The condition (or guard) of the event, and
- The effect of the transition



StateMachine Table Simulation

A StateMachine Table is a representation of a StateMachine, and can be simulated in exactly the same way as a StateMachine diagram.

Access

With a StateMachine displayed in Table form, use any of the methods outlined in this table to start the simulation.

Ribbon	Simulate > Run Simulation > Start, or Simulate > Dynamic Simulation > Simulator > Open Simulator Window > (Start icon)
Context Menu	Right-click on view background | Execute Simulation | <Interpreted or Manual>

Highlight active cells

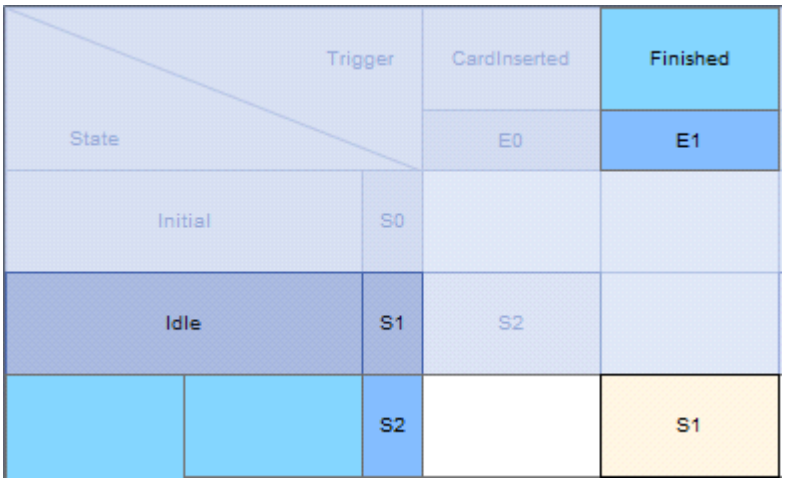
As the simulation executes, the table cells change color to indicate the:

- Currently active State(s) - the color set in the 'Highlight Color' field of the 'StateMachine Options' dialog, and a dark border
- Potential next States(s) - A variant of the color in the

'Highlight Color' field or, if the 'Use Different Color for Target State' checkbox is selected on the 'StateMachine Options' dialog, the color set in the 'Target Zone Color' field

- Active Transition(s) - the color set in the 'Transition Cell' field of the 'StateMachine Options' dialog
- Trigger(s) - the color set in the 'Highlight Color' field of the 'StateMachine Options' dialog
- Non-active States - gray

For example:



Signal Triggers

As when running a simulation as a diagram, the simulation will automatically traverse transitions with no guards or validated guards. Transitions with a Trigger will not be followed unless that Trigger has been fired. They can be fired automatically from the Simulation Events window or you can fire a Trigger manually by right-clicking on the

Transition or Trigger cell and selecting 'Signal Trigger in Simulation'.

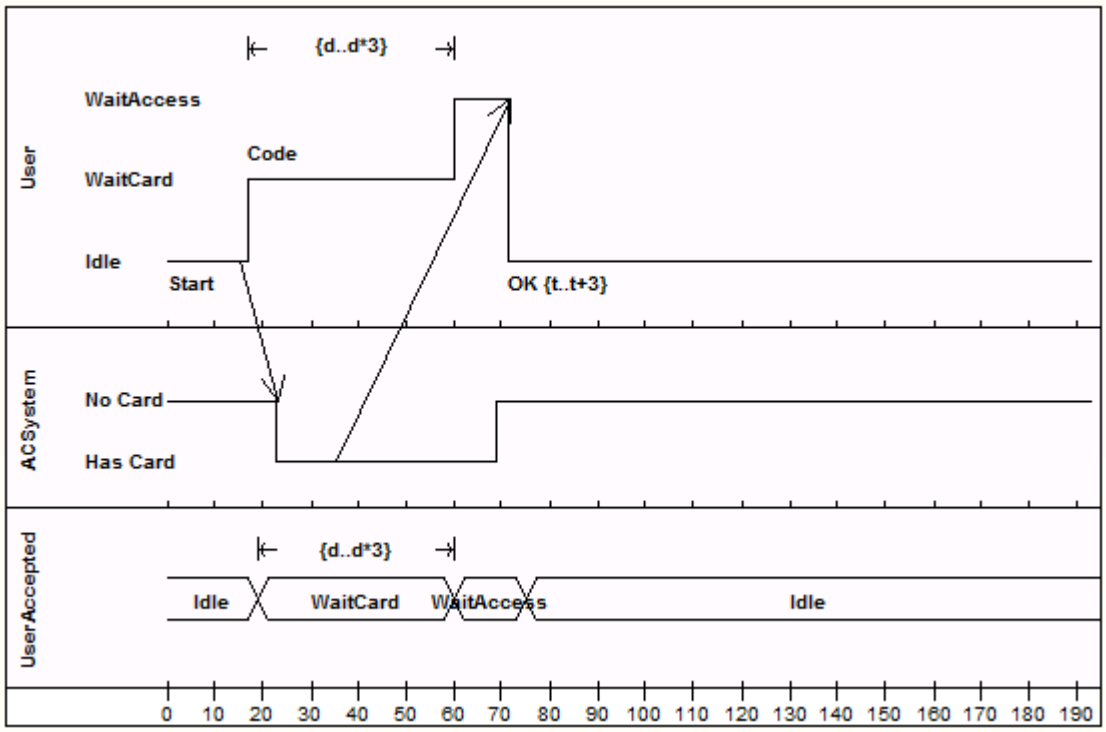
Timing Diagram

A Timing diagram defines the behavior of different objects within a time scale. It provides a visual representation of objects changing state and interacting over time. You can use it to:

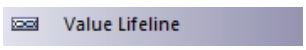
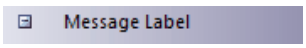
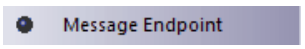
- Define hardware-driven or embedded software components; for example, those used in a fuel injection system or a microwave controller
- Specify time-driven business processes

You generate Timing diagram elements and connectors from the 'Timing' pages of the Diagram Toolbox.

Example Diagram



Timing Diagram Element Toolbox Icons

Icon	Description
	A State Lifeline element represents the state of an object across a measure of time, using changes in y-axis to represent discrete transitions between states.
	A Value Lifeline element represents the state of an object across a measure of time, using parallel lines indicating a steady state, along the x-axis.
	A Message Label is an alternative way of denoting Messages between Lifelines, which is useful for 'uncluttering' Timing diagrams strewn with messages.
	A Message Endpoint element indicates that a Message: • Terminates at an undefined point outside the State or Value Lifeline, having started at an identified point within the Lifeline, or • Originates from an undefined point outside a State or Value Lifeline, terminating at an identified point

	within the Lifeline
	A Diagram Gate element indicates that a Message: • Terminates at a defined point outside the State or Value Lifeline, having started at an identified point within the Lifeline, or • Originates at a defined point outside a State or Value Lifeline, terminating at an identified point within the Lifeline The defined point that the Diagram Gate is anchored to is the border of an Interaction Fragment, indicating that the Message issues from or delivers to that Fragment.

Timing Diagram Connector Toolbox Icon

Icon	Description
	Messages indicate a flow of information or transition of control between elements.

Create a Timing Diagram

Create a Timing diagram

Step	Action
1	Right-click on a Package in the Browser window and select 'New Diagram'. The 'Model Builder' dialog displays, with the 'Diagram Builder' tab page selected.
2	In the 'Select From' panel, select 'UML Behavioral'.
3	In the 'Diagram Types' panel, select 'Timing'.
4	Click on the OK button. The Diagram View displays, on which you create the Timing elements for the diagram.

Set a Time Range

Set a time range before adding Lifeline elements to your Timing diagram

Step	Action
1	Right-click on the diagram and select 'Set Timeline Range'. The 'Set Timeline Range' dialog displays.
2	In the 'Start Time' and 'End Time' fields, type the numeric values for the start and end points of the timeline; for example, set the range 0 to 100. The start time must be less than the end time.
3	In the 'Time Units' field, type the unit in which the time is measured; for example, seconds or minutes.
4	If it is not necessary to show the time range on the diagram, select the 'Suppress In Diagram' checkbox.
5	Click on the OK button. If you have not suppressed it, the time range displays underneath the Lifeline elements that you create on the diagram.

Edit a Timing Diagram

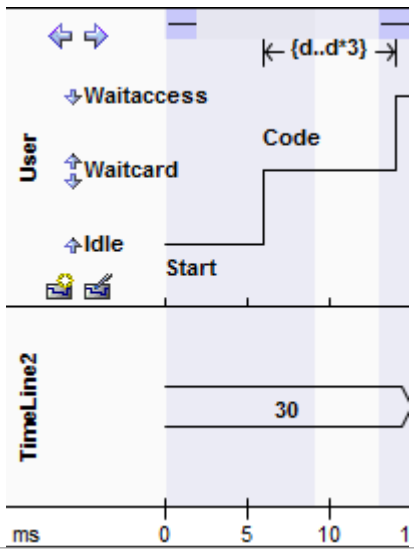
On a Timing diagram, you can add State Lifeline elements and Value Lifeline elements. You can maintain the states and transitions on these Lifeline elements either on the diagram itself or via the 'Configure Timeline' dialog.

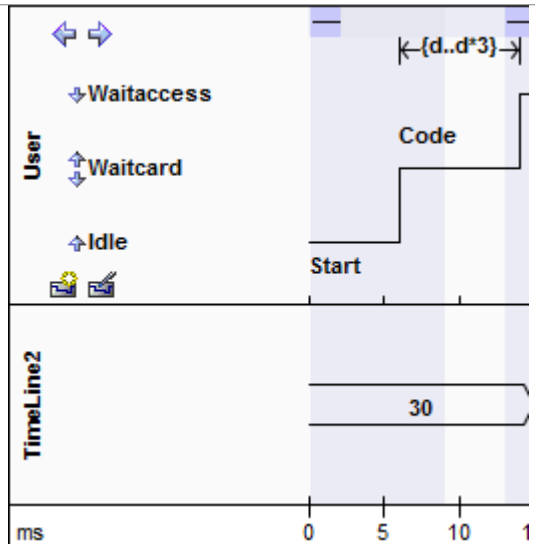
Add and Edit State Lifeline

From the 'Timing elements' page of the Toolbox drag a State Lifeline icon onto your diagram. The element displays on the diagram.

Edit Properties

Task	Action
Define the name of the State Lifeline	1. Right-click on the element and select the 'Properties | Properties' option; the 'Properties window for the element displays, showing the 'Element' tab. 2. Overtype the 'Name' field. 3. Click off the Properties window.
Sizing and Scale	In the top left corner of a selected Lifeline element are the left and right quick sizing buttons ( ). These buttons increase or decrease the width of the Lifeline element, which in turn controls the scale width of each time unit; by increasing the width of the element you increase the resolution when adding transitions, which makes them easier to edit. In order to edit the State Lifeline element,

	you must click on it to select it.
Set Timeline Start Position	You might require more space at the start of your timelines; for example, to use long state names. To insert more space in all the timelines on a diagram: 1. Right-click on the diagram background and select the 'Set Timeline Start Position' option; the 'Set Timeline Start Position' dialog displays. 2. The 'Value 80 to 300' field defaults to 80 as the minimum distance in pixels between the start of the timeline element and the start of the timeline itself; type a new value up to 300 pixels and click on the OK button to increase the space at the start of the timeline. These two diagrams have start positions of 80 pixels and 150 pixels respectively. 



You now edit the states and transitions in the State Lifeline.

Add States to a State Lifeline

Add States to a State Lifeline

Step	Description
1	Click on the State Lifeline element. The New State button ( ) and Edit States button ( ) display at the bottom left of the element.
2	Click on the New State button. The 'New State' dialog displays.
3	In the 'State' field, type the name of the state.
4	Click on the OK button. You must add at least two states; for example, 'On' and 'Off'.
5	As you add states, increase the height of the element by dragging one of the icons on the edge of the element. You can also add states using the 'States' tab of the 'Configure Timeline' dialog. Add either:

	• Discrete states to the Timeline, or • A continuous range of numeric states

Edit States in a State Lifeline

Edit States in a State Lifeline

Step	Description
1	Click on the State Lifeline element and click on the required state. The 'Edit State' dialog displays.
2	In the 'State' field, change the name as required.
3	Click on the OK button.
4	If necessary, change the order of the states by either: • Clicking on the up or down arrows ( ) beside each state name, or • Right-clicking on the state name and selecting the 'Move Up' or 'Move Down' options You can also edit the states using the 'States' tab of the 'Configure Timeline' dialog.

Delete States in a State Lifeline

Delete States in a State Lifeline

Step	Description
1	Right-click on the state name and select the 'Delete' option.

Alternatively

Step	Description
1	Click on the State Lifeline element.
2	Hold down Ctrl and move the cursor over the state name. The cursor changes form (⚡).
3	Click the mouse button. The state name is deleted.

Edit Transitions In State Lifeline

In a Timing diagram you can show the transitions (changes of state) that occur within a StateMachine over a fixed time period and at certain timing points. This is similar in many respects to an Interaction lifeline with State changes highlighted. As events and changes occur within the instance this Timing diagram represents, state changes occur and are mapped onto this Timeline. In that respect it is a record of how a particular aspect of the system behaves over time.

When building a Timeline it is necessary to define the States first, and then to add the explicit transitions between those States at particular timing points.

Edit Transitions

Task	Action
Add and Move Transitions	After you have added states, you can add transitions between states directly on the timeline using the mouse.
Change the Transition Time	Move the cursor over one or other of the vertical transition lines and drag the line left or right to change the time of the transition. While on the line, the cursor shape

	changes to the horizontal movement cursor ( ).
Merge Transitions	If necessary, you can 'push' a transition to merge it with the next or previous transition point on any Lifeline element on the diagram. Position the cursor off the appropriate side of the transition line; the cursor changes form (or). Click the mouse button; the system locates the nearest transition in the required direction, on any element on the diagram, and merges the current transition with that transition.
Delete Transitions	Transitions are automatically deleted when you move the transition to the same state as the previous transition state, and release the cursor. Alternatively, right-click on the transition line and select the 'Delete' option.

Add and Move Transitions

After you have added states, you can configure state changes (transitions) directly on the Timeline using the mouse. This is a fast and effective means of building a detailed model of state changes over time.

In order to modify the Timeline, place the mouse over the existing Timeline. As you move the cursor over the Timeline, the cursor changes to one of three shapes, described here.

Access

Context Menu	Right-click on the transition line | Edit
Other	Click directly on the appropriate transition line, after the transition begins

Modify Timeline

As you move the cursor over the vertical line of a transition, the time at which the transition occurs displays next to the line.

Task	Action
The move cursor	

	Displays when it is directly over the timeline. Hold down the mouse button and drag the line to move the timeline to a state above or below the current position; you can move the transition more than one state up or down, if necessary.
The new transition up cursor 	Displays when it is just below the timeline, and there is another state above the line. Press and hold the Alt key; the cursor changes ( ). Click to create a new transition to the state above the line. To push the transition up more than one state, move the cursor onto the line and drag it up. The transition is for one interval unit; you can make it longer by changing the transition time. If you do not hold the Alt key, the cursor does not change and the whole timeline from the transition onwards moves up.
The new transition down cursor 	Displays when it is just above the transition line, and there is another state below the line.

	Press and hold the Alt key; the cursor changes (⇩). Click to create a new transition to the state below the line. To push the transition down more than one state, move the cursor onto the line and drag it down. The transition is for one interval unit; you can make it longer by changing the transition time. If you do not hold the Alt key, the cursor does not change and the whole timeline from the transition onwards moves down.

Edit Transition

Edit the transitions as required, on the 'Edit Transition' dialog.

Option	Action
At Time	Type the point on the timescale at which the transition occurs.
Transition To	Type the name of the state to which the transition occurs.
Event	Type the name of the event that the

	transition represents. This displays on the Timeline element just above the transition line.
Duration Constraint	Type any constraint on the duration of the transition. This displays on the Timeline element, along the top of the element over the transition.
Time Constraint	Type any constraint on the start of the transition. This displays on the Timeline element at the start of the transition.
OK	Click on this button to save the changes.

Notes

- Once Event, Duration Constraint or Time Constraint are displayed on the diagram, you can edit them directly by clicking on them to display their specific dialog
- You can delete them by pressing and holding the Ctrl key as you click on them; the cursor changes form when you press the Ctrl key
- You can also edit transitions using the 'Transitions' tab of the 'Configure Timeline' dialog

Add and Edit Value Lifeline

From the Toolbox drag a 'Value Lifeline' element onto your diagram. The element displays on the diagram.

Edit the Value Lifeline name

Step	Action
1	Right-click on the element and select the 'Properties | Properties' option. The Properties window displays for the Timeline element, showing the 'Element' tab.
2	Overtype the 'Name' field.
3	Click off the Properties window.

Sizing and Scale

In the top left corner of a selected Lifeline element are the left and right quick sizing buttons ( ). These buttons increase or decrease the width of the Lifeline element, which in turn controls the scale width of each time unit. By increasing the width of the element you increase the resolution when adding transitions, which makes them

easier to edit.

Add States In Value Lifeline

Adding states to a Value Lifeline is similar to adding states to a State Lifeline element.

For a Value Lifeline, only the first state displays on the diagram. The other states are added to a list to access when creating transitions; they only display on the Lifeline element as you create transitions to those states.

You can only edit or delete states in a Value Lifeline element using the 'States' tab of the 'Configure Timeline' dialog.

Edit Transitions In Value Lifeline

Add Transitions to the states on a Value Lifeline element, via the diagram

Step	Action
1	Move the cursor above the transition line. The cursor changes form ( ).
2	Click the mouse button. The 'New Transition Event' dialog displays.
3	In the 'Transition To' field, click on the drop-down arrow and select a state from the list of available states; this displays on the Lifeline element within the transition box. The remaining fields on the dialog are optional.
4	In the 'Event' field, type the name of the event that the transition represents; this displays on the Lifeline element just below and at the start of the transition line.
5	In the 'Duration Constraint' field, type any constraint on the duration of the transition; this displays on the

	Lifeline element, along the top of the element over the transition.
6	In the 'Time Constraint' field, type any constraint on the start of the transition. This displays on the Lifeline element at the start of the transition, just after the Event name.
7	Click on the OK button to create the new transition.

Edit a Transition

Step	Action
1	Click on the state name in the transition. Alternatively, right-click on the state name and select the 'Edit' option. The 'Edit Transition' dialog displays, which is the same as the 'New Transition Event' dialog, except that the 'At Time' field is enabled.
2	If necessary, overtype the 'At Time' field to define a different start point. You cannot change the 'At Time' field for the first state in the timeline; this is always '0'.

3	Edit the remaining fields as necessary.
4	Click on the OK button to save the changes.

Change the transition time

Step	Action
1	To change the start or end time of a transition, click on the start or end point of the transition and drag it to the new position. While on the line, the cursor shape changes to the horizontal movement cursor (⇔).

Delete Transitions

Step	Action
1	To delete a transition, press and hold Ctrl and click on the transition state name. While you hold Ctrl on the transition state name, the

	cursor changes form (⚡). Alternatively, right-click on the state name and select the 'Delete' option.

Configure Timeline - States

You can manage states using the 'States' tab of the 'Configure Timeline' dialog. To display this dialog, either:

- Double-click on the Lifeline element
- Right-click on the Lifeline element and select the 'Properties' option, or
- On a Value Lifeline, click on the button

The 'Configure Timeline' dialog defaults to the 'States' tab.

All states currently defined for the Lifeline element are listed in the 'States' panel.

Add a new State

Step	Action
1	In the 'State Name' field, type the name of the first new state in the Lifeline element; for example, 'WaitState'.
2	Click on the Save button. The state is added to the 'States' panel and (for a State Lifeline Element) to the diagram.
3	Click on the New button.

4	In the 'State Name' field, type the name of the next state in the Lifeline element.
5	Repeat steps 2 to 5 until you have added all required states (you must add at least three to the Lifeline element).
6	When you have added all the required states, click on the OK button to close the 'Configure Timeline' dialog.

Edit an existing state

Step	Action
1	Click on the state in the 'States:' list.
2	In the 'State Name' field, change the name of the state.
3	Click on the Save button.

Delete an existing State

Step	Action
1	Click on the state in the 'States:' list.
2	Click on the Delete button.

Change the order of States

Step	Action
1	Click on the state in the 'States:' list.
2	Click on the or buttons to move the state up or down the sequence.

Numeric Range Generator

You can also use the 'Configure Timeline' dialog to create a range of states having numeric values to be applied to the Timeline.

Important: This operation deletes all existing states and transitions for the Timeline element.

Create a range of states having numeric values

Step	Action
1	Double-click on the Lifeline element. The 'Configure Timeline' dialog displays.
2	Click on the Create Continuous Numeric States button. The 'Numeric Range Generator' dialog displays.
3	In the 'High Value' and 'Low Value' fields, type the upper and lower values of the range.
4	In the 'Step Value' field, type the increase interval. Nonsense values do not parse; 'Low Value' must be less than 'High Value', and 'Step Value' must be a

	positive value smaller than the total range.
5	In the 'Units' field, type the name of the measurement unit; for example, 'minutes'.
6	Click on the OK button. Enterprise Architect displays a warning that existing states and transitions are to be deleted.
7	Click on the Yes button. The 'Configure Timeline' dialog redisplays, with the defined range of states listed in the 'States' panel.
8	Click on the OK button. For a: • Value Lifeline, the first state is shown on the Timeline for the full time range of the Timeline • State Lifeline, the range of states is displayed as the y-axis of the Timeline

Configure Timeline - Transitions

You can also manage transitions using the 'Transitions' tab of the 'Configure Timeline' dialog. To display this, either:

- Double-click on the Lifeline element
- Right-click on the Lifeline element and select the 'Properties' option, or
- On a Value Lifeline, click on the button

The 'Configure Timeline' dialog defaults to the 'States' tab. Click on the 'Transitions' tab.

All transitions defined for the Timeline element are listed in the 'Transition Points' panel.

Add a new transition

Step	Action
1	Click on the New button.
2	In the 'New Transition' panel, type the details of the transition.
3	Click on the Save button.

Edit a transition

Step	Action
1	Click on a transition in the list.
2	In the 'Edit Transition' panel, edit the fields for the transition as required.
3	Click on the Save button.

Delete a transition

Step	Action
1	Click on a transition in the list.
2	Click on the Delete button. The transition is removed from the dialog and the Lifeline.
3	Click on the OK button.

Time Intervals

You create and manage Time Intervals using the Interval Bar (the pale line along the top of each selected Lifeline element). With Time Intervals you can perform various operations on transitions, such as copy and paste. You can also compress sections of the timeline so that they are not visible.

Each Time Interval displays across all Timeline elements down to the last element on the diagram.

Manage Time Intervals

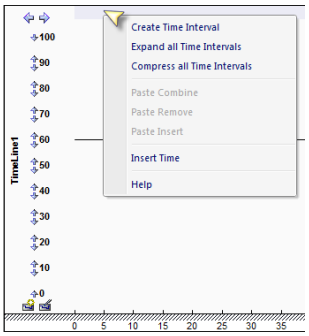
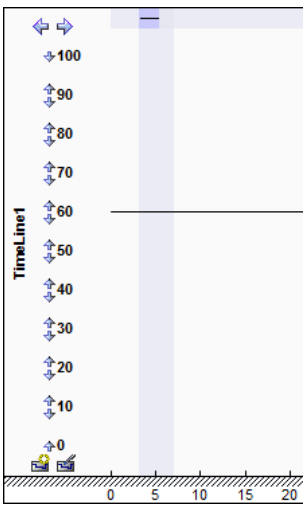
Action	Description
Create Time Intervals	You can create a Time Interval using the: • Interval Bar - context menu • Interval Bar - Shift key, or • Timeline - context menu
Compress Time Intervals	You can compress Time Intervals to conserve space on long timelines.
Select Time Intervals	There are a number of ways to select Time Intervals for performing other operations.

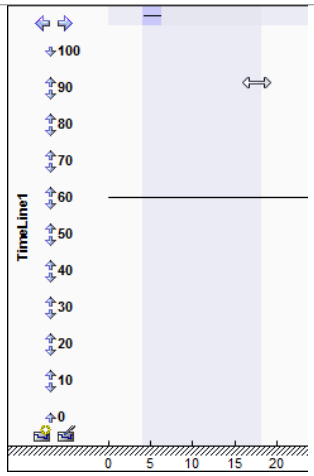
Move Time Intervals	To move a Time Interval, move the cursor over the Interval bar within the Time Interval, hold down the mouse button and drag the interval left or right. Time Intervals can meet, but cannot overlap.
Resize Time Intervals	To resize a Time Interval, move the cursor over the Interval Bar at the start or end edge of the Time Interval, hold down the mouse button and move the edge left or right. Time Intervals can meet, but cannot overlap.
Delete Time Intervals	To delete Time Intervals, select each Time Interval to be deleted and press the Delete key. Deleting the Time Interval does not delete transitions within that interval.

Create Time Intervals

You can create time intervals on Timing elements in a number of ways.

Create a Time Interval using the Interval Bar context menu

Images	Step and Action
	1. Right-click on the Interval Bar at approximately the point at which to start or finish the Time Interval, and select the 'Create Time Interval' option.
	2. The Time Interval displays down all the timeline elements, as a narrow pale band with a blue compression box at the top.
	3. Move the cursor to the edge of the Time Interval in the Interval Bar so that the cursor changes to the drag form and

	drag the edge to the correct start or end point.

Create a Time Interval using the Interval Bar and Shift key

Step	Description
1	Move the cursor over the Interval Bar and press Shift. The cursor changes shape ( ).
2	Click to create the Time Interval.
3	Move the cursor to the edge of the Time Interval in the Interval Bar so that the cursor changes to the drag form ( ) and drag the edge to the correct start or end point.

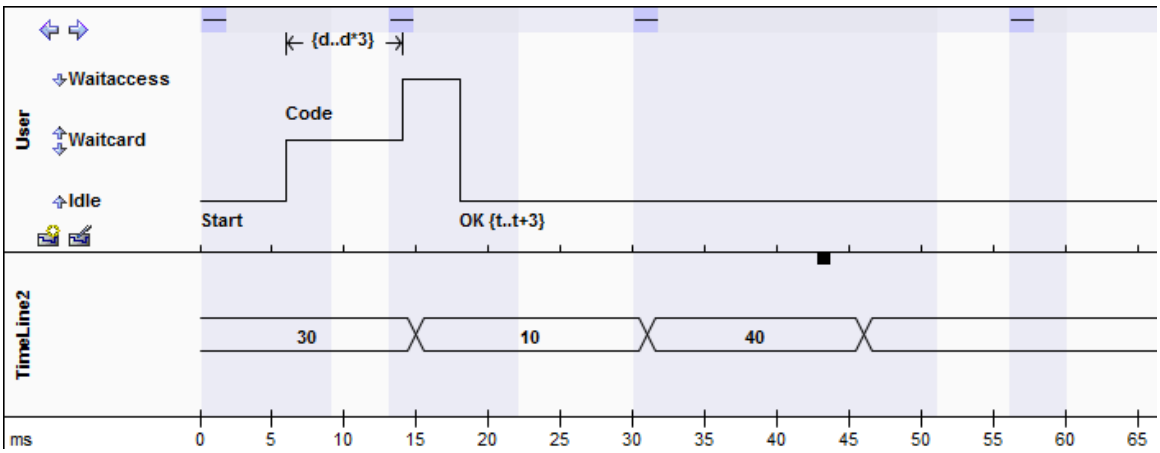
Create a Time Interval using the Timeline context menu

Step	Description
1	Right-click on the timeline just after a transition. The context menu displays.
2	Click on the 'Select' option. Enterprise Architect creates a Time Interval defining the period from the selected transition up to the next transition. If there are other Time Intervals in this period, Enterprise Architect replaces them with the single Time Interval for the transition state; you should consider this when creating the Time Interval, as it extends across the other Timeline elements in the diagram. A value of this method is that it creates a Time Interval for a period in which no transitions occur, which could be lengthy; you can then compress this Time Interval to hide the period of inactivity.

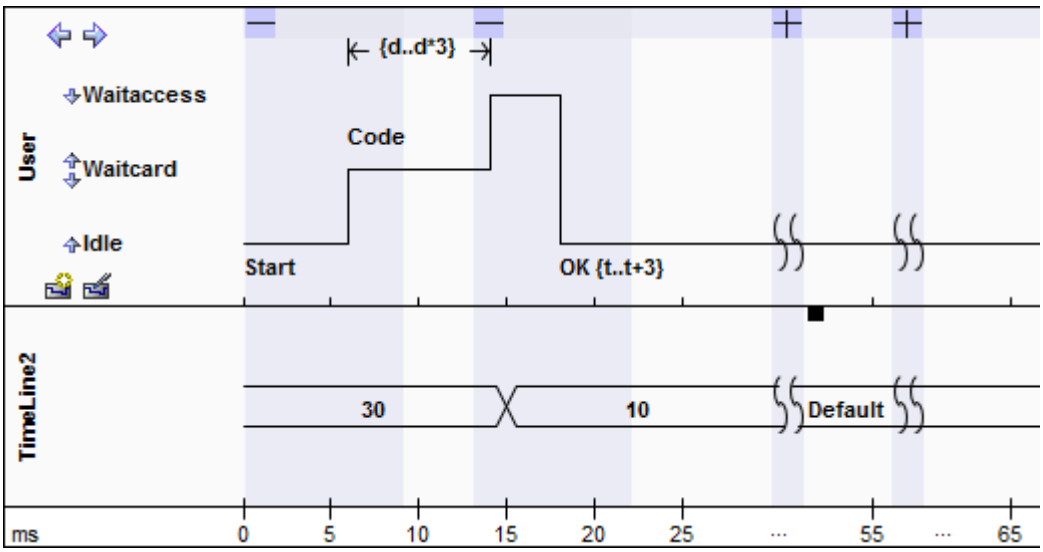
Compress Time Intervals

You can compress Time Intervals to conserve space on long timelines.

Uncompressed Time Intervals

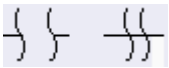
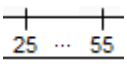


Compressed Time Intervals



Note:

You can also compress and expand Time Intervals using context menu options.

Item	Description
	The compression toggle boxes: • is expanded, click on this to compress the selected time interval • is compressed, click on this to expand the selected time interval again
	The compressed sections of the timelines themselves, in all elements. If there is space between the paired symbols, there are transitions within the compressed section. If the timeline continues through the paired symbols there are no transitions in the compressed section.
	The compressed sections in the time range underneath the elements.

Select Time Intervals

Select Intervals

Task	Action
Select a Time Interval across all elements on the diagram	Click on the Interval Bar within the Time Interval.
Select a number of individual Time Intervals	Press and hold the Ctrl key while clicking on the Interval Bar within each Time Interval.
Select all Time Intervals in a range	Click on the Interval Bar within the first Time Interval in the range, then press and hold the Shift key and click on the Interval Bar within the last Time Interval in the range. All Time Intervals between the two are selected.

Modify Intervals

After you have selected a Time Interval, you can modify it.

Task	Action
Exclude Lifeline elements from the selection	Press and hold the Ctrl key and click on any part of the selection within that element. Repeat the step to toggle the selection and re-include the element.
Select only one Lifeline element and exclude all others	Press and hold the Shift key and click on any part of the selection within that element.

Time Interval Operations

You can select and update specific Time Intervals.

Right-click on the Interval Bar within an interval. A context menu displays providing these options.

Compress Timeline

The 'Compression' toggle boxes and 'Compress Interval' menu option operate on the Time Interval and compress the timeline and all transitions within the Interval. You have an alternative option that operates on the timeline and compresses a single transition state.

1. Right-click on the timeline (rather than the Interval Bar) just after a transition, and select the 'Compress' option.
2. Enterprise Architect creates a new Time Interval spanning the period from the selected transition up to the next transition, and then compresses that Time Interval.

If there are other Time Intervals in this period, Enterprise Architect replaces them with the single Time Interval for the transition state. You should consider this when creating and compressing the Time Interval, as it extends across the other Timeline elements in the diagram.

A value of this method is that it creates a Time Interval for a period in which no transitions occur, which could be lengthy, and then compresses this Time Interval to hide the period of inactivity.

Context Menu Options

Option	Action
Select Interval Deselect Interval	Select the Time Interval or, if the interval is already selected, deselect it. You can select several Time Intervals in this way, accessing the menu separately on each interval.
Toggle Interval Selection	Switch the selection or deselection of the Time Interval within the selected Timeline element. You select or deselect a Time Interval across all Timeline elements, but the 'Toggle' option acts only on the element in which you access the menu.
Compress Interval	Compress the Time Interval, and hide all transitions within that Time Interval. This is also useful for hiding long sections of inactivity on the time line.
Remove Interval	Delete the Time Interval.
Copy	Copy the transitions for all selected Time Intervals.

Cut	Copy and delete the selected transitions from the diagram.
Cut and Remove Time	Copy and delete the transitions that lie in the selected Time Intervals from the diagram. This option also removes time from the timeline, the amount being the duration of the Time Interval. All transitions and Time Intervals to the right of the selected time interval are moved left.
Delete	Delete the selected transitions from the diagram.
Delete and Remove Time	Delete the transitions that lie in the selected Time Intervals from the diagram. This option also removes time from the timeline, the amount being the duration of the Time Interval. All transitions and Time Intervals to the right of the current Time Interval are moved left.
Insert Time	Add time to the timeline and move all transitions and time intervals to the right. Also expand the duration of the current

	Time Interval.

All Time Intervals in the Diagram

To create a new Time Interval or work across all Time Intervals in the diagram, right-click on the Interval Bar between Time Intervals. A context menu displays, providing a number of options (The 'Paste ...' menu options become active after transitions have been copied).

Menu Option	Action
Create Time Interval	Create a single Time Interval.
Expand all Time Intervals	Expand all Time Intervals over the whole diagram.
Compress all Time Intervals	Compress all Time Intervals over the whole diagram.
Paste Combine	Paste copied transitions over any existing transitions within the copied time frame. The diagram does not allow two consecutive transitions to the same state, and removes the second transition

	automatically.
Paste Remove	Delete all the transitions and then pastes the copied transition within the copied time frame.
Paste Insert	Insert time, moving all transitions and Time Intervals to the right to make room to paste in the copied transitions.
Insert Time	Add time to the timeline and move all transitions and Time Intervals to the right. This option does not change the duration of any Time Interval.

Copy and paste transitions from one timeline element to another

Step	Action
1	Press and hold the Shift key and select the Timeline element within a Time Interval to copy or cut.
2	Right-click on the Interval Bar (it doesn't matter which element you select).

	The context menu displays.
3	Copy or cut the transitions. You can also cut and remove time.
4	Select the timeline to paste transitions to and right-click on the Interval Bar. The context menu displays.
5	Select one of the paste operations. Note that states are created if they don't already exist in the timeline. Any states that don't exist in the Timeline element you are pasting to are created. Any new states created might be in the wrong order; you can change the order via the diagram 'quick' buttons.

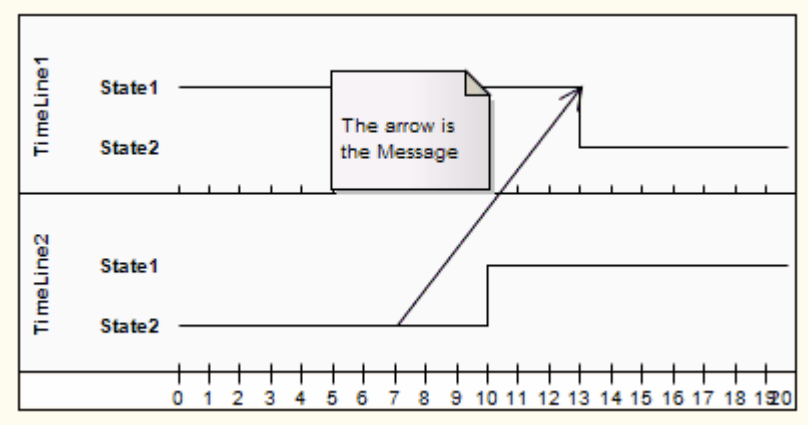
Shift transitions within a selected Time Interval or multiple selected Time Intervals

Step	Action
1	Select all the Time Intervals containing the

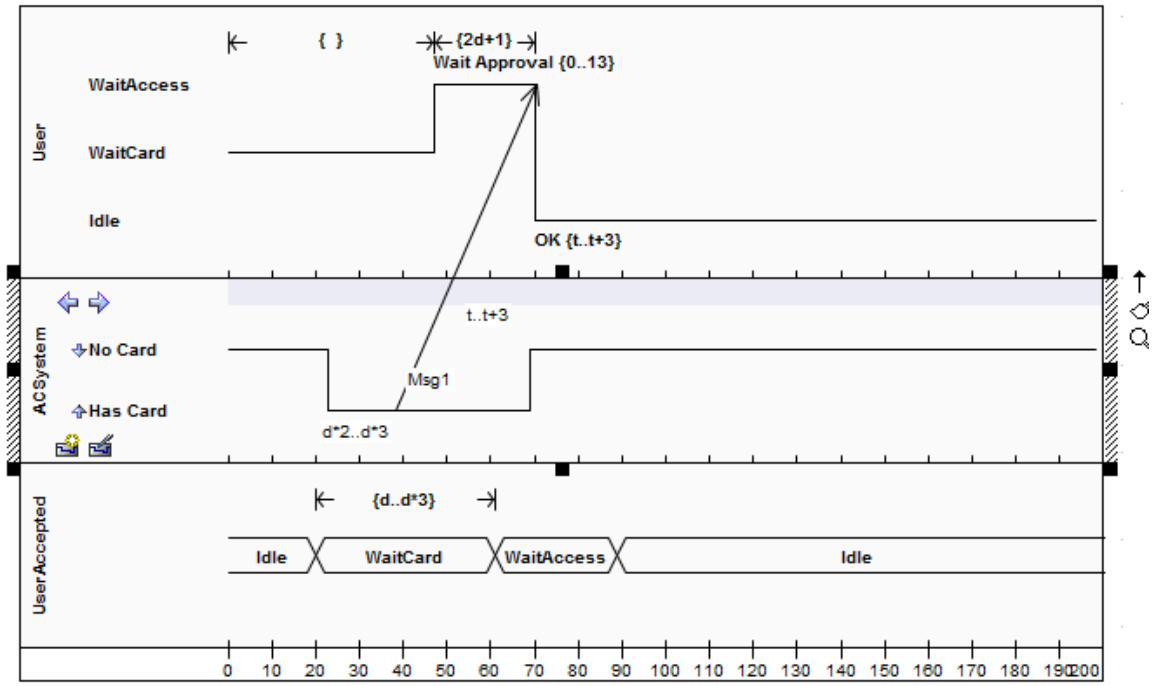
	transitions to be shifted.
2	Press and hold Shift and click on the Interval Bar (it doesn't matter which Timeline element you select), and move the transition left or right. You cannot drag transitions over other transitions; the move stops when the moved transition collides with a stationary transition. If you have collision problems, use Shift+select to shift transitions for a single Timeline element.

Messages (Timing Diagram)

Messages are the communication links between Lifelines in a Timing diagram. In the case of a Timeline, a Message is a connection between two Timeline objects.



For example:



The OMG Unified Modeling Language specification, (v2.5.1, figures 14.30 and 14.31, p.520.)

Toolbox icon

Message

Create a Timing Message

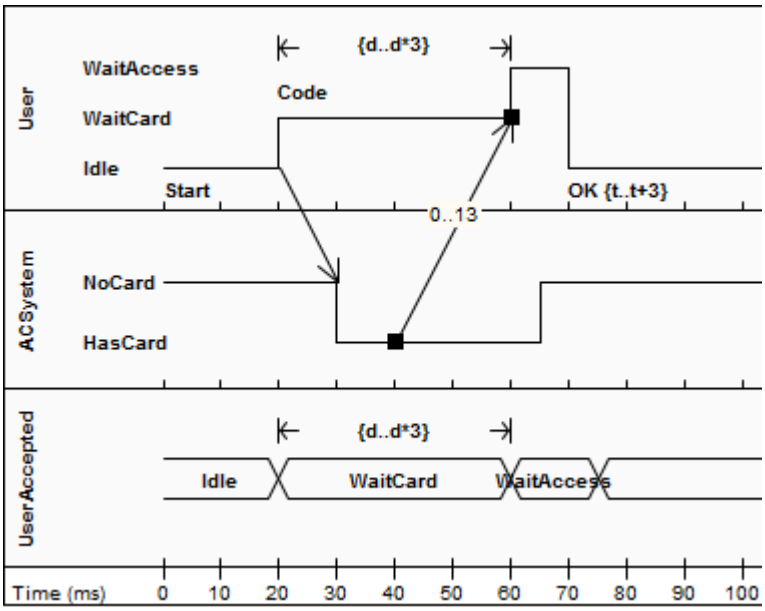
You can create a Timing Message between two Lifeline objects (State or Value) on a Timing diagram, each with existing transition points.

Create a Message between Lifelines

Step	Action
1	Click on the 'Message' icon on the 'Timing Relationships' page of the Diagram Toolbox (click on to display the 'Find Toolbox Item' dialog and specify 'Timing').
2	Click on the source Lifeline at the point at which the Message will start, and drag the cursor to the transition point on the destination Lifeline where the Message will end. A new Timing Message is created between these two points.
3	Double-click on the new Message to open the 'Timing Message' dialog. Review or complete the dialog as indicated in the 'Dialog Fields' table.

Dialog Fields

This diagram shows an example of a configured Message:



The OMG Unified Modeling Language specification, (v2.5.1, figures 14.30 and 14.31, p.520)

Field/Button	Action
Start	Identifies the Lifeline from which the Message originates.
End	Identifies the Lifeline on which the Message terminates.
Start Time	Shows the time after the timeline begins at which the Message starts. You can change this if you need to.
	Shows the time after the timeline begins

End Time	at which the Message ends. You can change this if you need to, but the time must correspond to a transition point on the target Lifeline.
Name	(Optional) Type in a name for the Message.
Time Observation	(Optional) Type any text to act as a label providing information on when the Message is sent.
Duration Observation	(Optional) Type any text to act as a label providing information on the interval of a Lifeline at a particular state, begun from receipt of the Message.
Transition To	The state in the target Lifeline that the Message terminates on. If necessary, you can click on the drop-down arrow and select a different state to transition to. The head of the Message moves accordingly.
Event	(Optional) Type in the name of any event that triggers the transition.
Time Constraint	(Optional) Type in the maximum time it can take to transmit the Message.

Duration Constraint	(Optional) Type in the maximum time the Lifeline can remain in the changed state after receipt of the Message.

Notes

- You can move the source end of the Message freely along the source timeline; however, the target end (arrow head) must attach to a transition
- If you create a new Message and do not give it a target transition, it automatically finds and attaches to the nearest transition; if you move the target end, it drags the transition with it

Sequence Diagram

A Sequence diagram is a structured representation of behavior as a series of sequential steps over time. You can use it to:

- Depict workflow, Message passing and how elements in general cooperate over time to achieve a result
- Capture the flow of information and responsibility throughout the system, early in analysis; Messages between elements eventually become method calls in the Class model
- Make explanatory models for Use Case scenarios; by creating a Sequence diagram with an Actor and elements involved in the Use Case, you can model the sequence of steps the user and the system undertake to complete the required tasks

Construction

- Sequence elements are arranged in a horizontal sequence, with Messages passing back and forward between elements
- Messages on a Sequence diagram can be of several types; the Messages can also be configured to reflect the operations and properties of the source and target elements (see the Notes in the *Message* Help topic)
- An Actor element can be used to represent the user initiating the flow of events

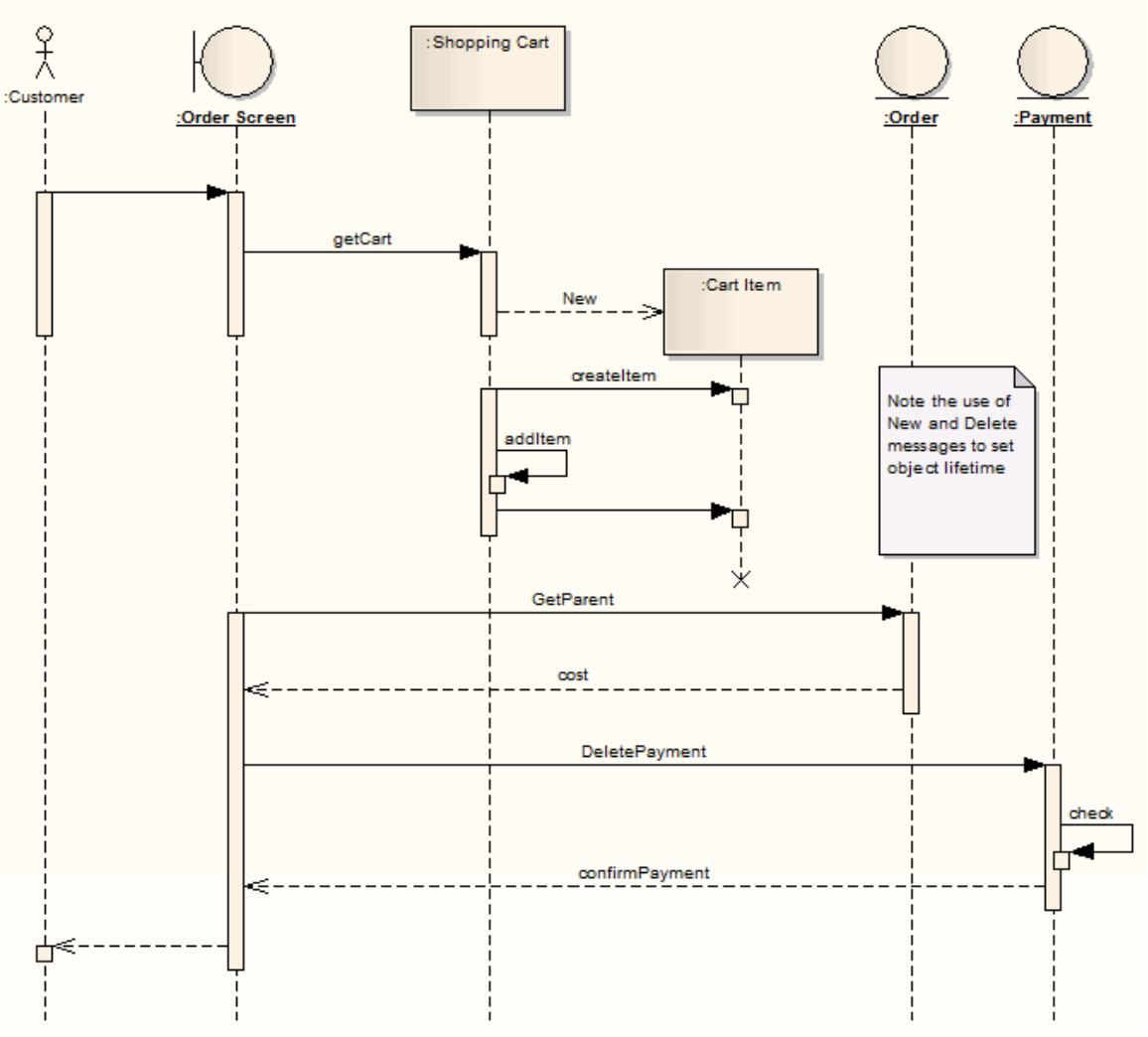
- Stereotyped elements, such as Boundary, Control and Entity, can be used to illustrate screens, controllers and database items, respectively
- Each element has a dashed stem called a Lifeline, where that element exists and potentially takes part in the interactions

To toggle the numbering of messages on a Sequence diagram, select or deselect the 'Show Sequence Numbering' checkbox on the 'Preferences' dialog.

You generate Sequence diagram elements and connectors from the 'Interaction' pages of the Toolbox.

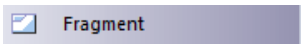
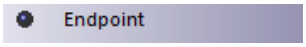
Example Diagram

This example Sequence diagram demonstrates several different elements.



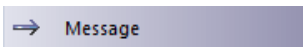
Sequence Diagram Element Toolbox Icons

Icon	Description
Actor	An Actor is a user of the system; user can mean a human user, a machine, or even another system or subsystem in the model.
Lifeline	A Lifeline represents a distinct

	connectable element and is an individual participant in an interaction.
	Boundary elements are used in analysis to capture user interactions, screen flows and element interactions.
	A Control organizes and schedules other activities and elements.
	An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system.
	A Fragment element can represents iterations or alternative processes in a Sequence diagram.
	An Endpoint is used in Interaction diagrams to reflect a lost or found Message in sequence.
	A Diagram Gate is a simple graphical way to indicate the point at which messages can be transmitted into and out of interaction fragments.
	The State/Continuation element serves

	two different purposes for Sequence diagrams, as State Invariants and Continuations.
	You can use an Interaction element to insert an Interaction diagram as a child of a Class element.

Sequence Diagram Connector Toolbox Icons

Icon	Description
	A Message indicates a flow of information or transition of control between elements.
	A Self-Message reflects a new process or method invoked within the calling lifeline's operation.
	A Recursion is a type of Message used in Sequence diagrams to indicate a recursive function.
	A Call is a type of Message connector that extends the level of activation from the previous Message.

Denote Lifecycle of an Element

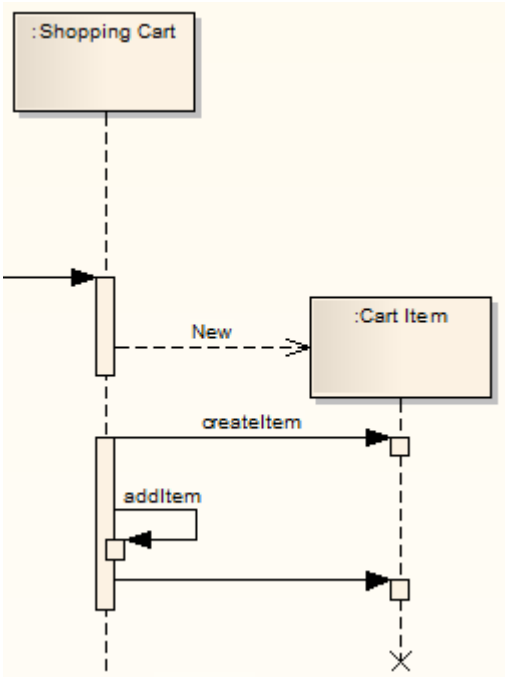
Capture element lifetimes using messages denoted as New or Delete message types

Step	Action
1	Double-click on a message within a Sequence diagram to display the Properties window for the Message.
2	In the 'Lifecycle' field, click on the drop-down arrow and select 'New' or 'Delete'.
3	Click on the OK button to save the changes.

Example Diagram

This example shows two elements that have specific

creation and deletion times.



Notes

- To show the termination X on the lifeline in the example diagram, you must switch on garbage collection: 'Start > Appearance > Preferences > Preferences > Diagram > Sequence: Garbage Collect'

Layout of Sequence Diagrams

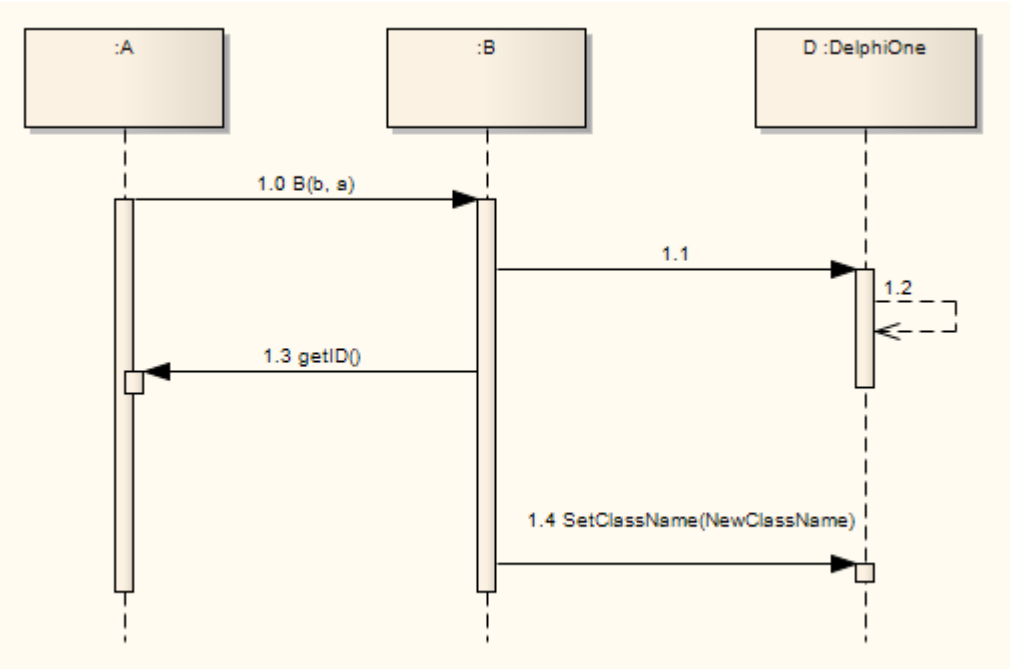
Offset the vertical separation of Sequence messages

Step	Action
1	Select the appropriate message in a Sequence diagram.
2	Use the mouse to drag the message up or down as required. As you drag a message up or down a lifeline, any messages or fragments below that message are shifted up or down the same amount. If the 'Reorder Messages' option is enabled, as you drag a message up or down past the next or previous message, the messages swap positions, rather than simply moving position. Alternatively, press and hold the Shift key as you move the message, to achieve the same result. Under Windows (but not under Linux or a Virtual Machine), you can also use the Alt key in the same way. As you move one Message past another, a tool-tip displays to remind you to 'Enable Reorder Messages from Layout | Helpers to reorder messages',

	regardless of whether or not the option is enabled. You can hide this tool-tip by deselecting the 'Enable Tooltips when reordering messages' checkbox on the 'Diagram > Sequence' page of the 'Preferences' dialog.

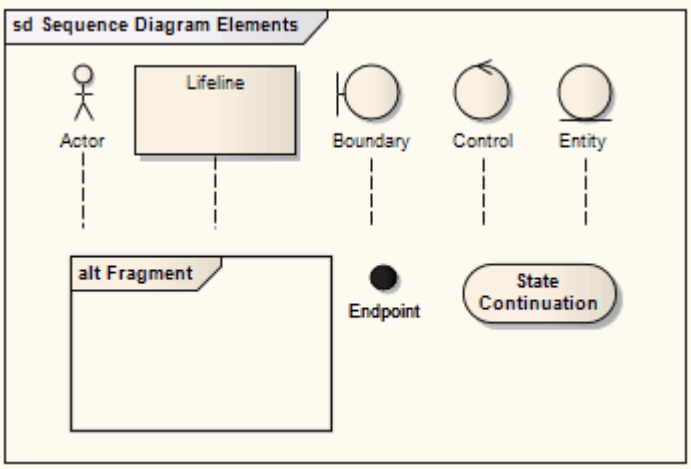
Example Diagram

This example shows an economical use of space in a Sequence diagram.



Sequence Elements

This example shows some possible elements of Sequence diagrams and their stereotyped display.

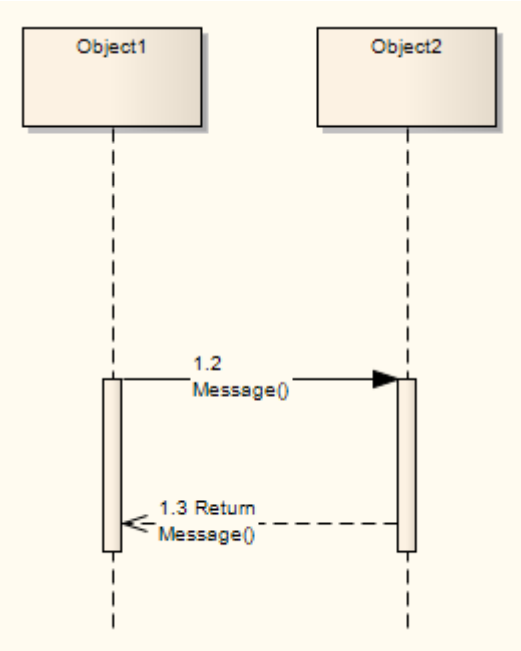


Element descriptions

Element	Description
Actor	An instance of an actor at runtime; this can be depicted either as the human figure or in rectangle notation.
Lifeline	An Object element with the stereotype Lifeline.
Boundary	Represents a user interface screen or input/output device.
Entity	A persistent element - typically

	implemented as a database table or element.
Control	The active component that controls what work gets done, when and how.

Messages (Sequence Diagram)



Sequence diagrams depict workflow or activity over time using Messages passed from element to element. In the software model. These Messages correspond to Class operations and behavior. When you display a Sequence diagram, the Diagram Toolbox automatically switches to the 'Interaction' pages of the Diagram Toolbox, containing the 'Message' icon.

Toolbox icon

Access

Diagram	Click on the 'Message' icon, click on the

Toolbox	source object and drag the cursor to the target object (If the Properties window for the Message does not display, right-click on the Message and on the 'Properties' menu option)

Create a Message on a Sequence diagram

Option	Action
Message	Type the Message name. If the Message flow is towards a Class element (dropped in from a Class diagram) or a Lifeline element having a classifier, and the destination Class has defined operations, you can click on the drop-down arrow and select an appropriate operation name; the Message then reflects the destination Class operations. You can also include operations that the element's classifier has inherited, in the list. To do this, select the 'Show Inherited Methods' checkbox.
	If the available operations on the

Operations	destination Class are not appropriate, click on this button and define a new operation in the destination element, using the 'Operations' dialog. If you create a Message without making reference to the target Class operations, no new operation is added to the target Class.
Parameters	Type any parameters that the Message has, as a comma-separated list.
Argument(s)	(Optional) Type the actual value that corresponds to each parameter, as a comma-separated list.
Return Value	If the Message has a return value or type, specify it in this field.
Show Inherited Methods	Select this checkbox to include operations that the destination element's classifier has inherited, in the drop-down list of operations available in the 'Message' field. Clear the checkbox to show only operations from the classifier itself.
Assign to	If the Message flow is from a Class element or Lifeline element with

	classifier that has defined attributes, click on the drop-down arrow and select an appropriate attribute name. The Message reflects the attributes from the source Class; you cannot add further attributes to the source Class here - if no appropriate attribute is listed, open the Class element 'Properties' dialog and add the required attribute. Otherwise, optionally type the name of the object to assign the message flow to.
Stereotype	(Optional) Type or select a stereotype for the connector (this is displayed on the diagram, if entered).
Alias	(Optional) Type an alias for the name of the Message. On the diagram, the alias displays instead of the Message name if the 'Use Alias if Available' checkbox is selected on the 'Diagram' tab of the 'Properties' dialog for the diagram.
Condition	Type any conditions that must be true in order for the Message to be sent.
Constraint	Type any constraints that might exist on when the Message is sent.

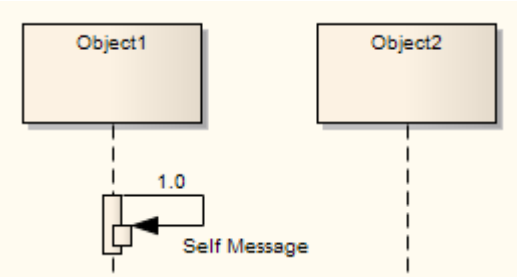
Is Iteration	Select the checkbox to indicate that the Message will iterate until the specified condition takes the value false. The condition statement on the diagram is prefixed by an asterisk (*). Clear the checkbox to indicate that the Message will only be sent once within the process cycle, if the specified condition is true.
Start New Group	(For Communication diagram Messages). Select this checkbox to reset the Message (and all subsequent Messages) to a separate group with a new initial number.
Synch	Click on the drop-down arrow and select 'Synchronous' or 'Asynchronous' as appropriate. The value 'Synchronous' disables the 'Kind' field; synchronous Messages are always Calls.
Kind	This field is enabled when the 'Synch' field is set to Asynchronous. Click on the drop-down arrow and select either 'Call' or 'Signal', as appropriate.
Lifecycle	Select 'New' to create a new element at

	the end of the Message, or 'Delete' to terminate the message flow at the end of the Message. If neither case applies, set the field to '<none>'.
Is Return	If the Message you have created is a return message, select this checkbox.
Save	Click on this Toolbar button to save the Message definition or any changes to it. • You can change the timing details of a message on the 'Timing Details' dialog, and emphasize the sequence of closely-ordered messages using General Ordering • To toggle the numbering of messages on a Sequence diagram, select or deselect the 'Show Sequence Numbering' checkbox on the 'Preferences' dialog

Notes

- You can also use the Message connector as an Information Flow, and realize information flows on the Message

Self-Message

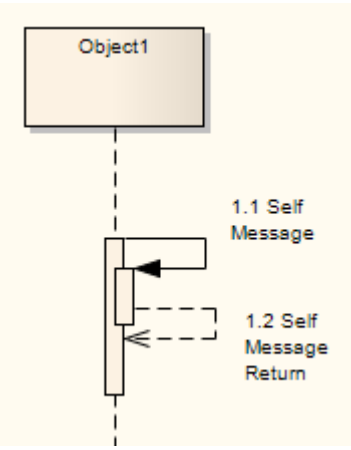


A Self-Message reflects a new process or method invoked within the calling lifeline's operation. It is a specification of a Message, typically in a Sequence diagram.

Self-Message Calls indicate a nested invocation; new activation levels are added with each Call.

Self-Message as Return

It is possible to depict a return from a Self Message call.



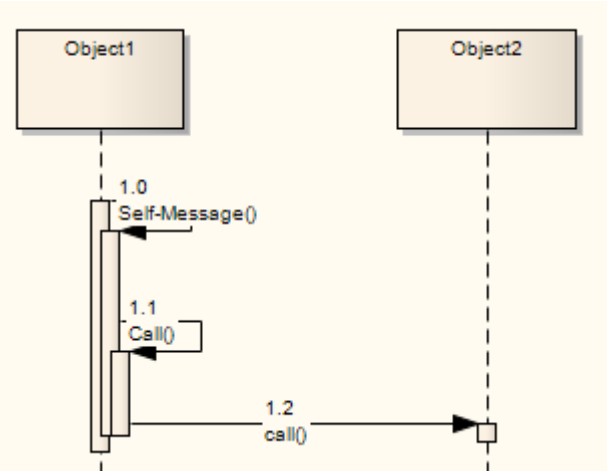
Create a Self Message return

Step	Action

1	Create a second Self Message at the end of execution.
2	Double-click on the Message name to open the Properties window for the Message.
3	Select the 'Is Return' checkbox.
4	Raise the Activation level of the return.

Toolbox icon

Call

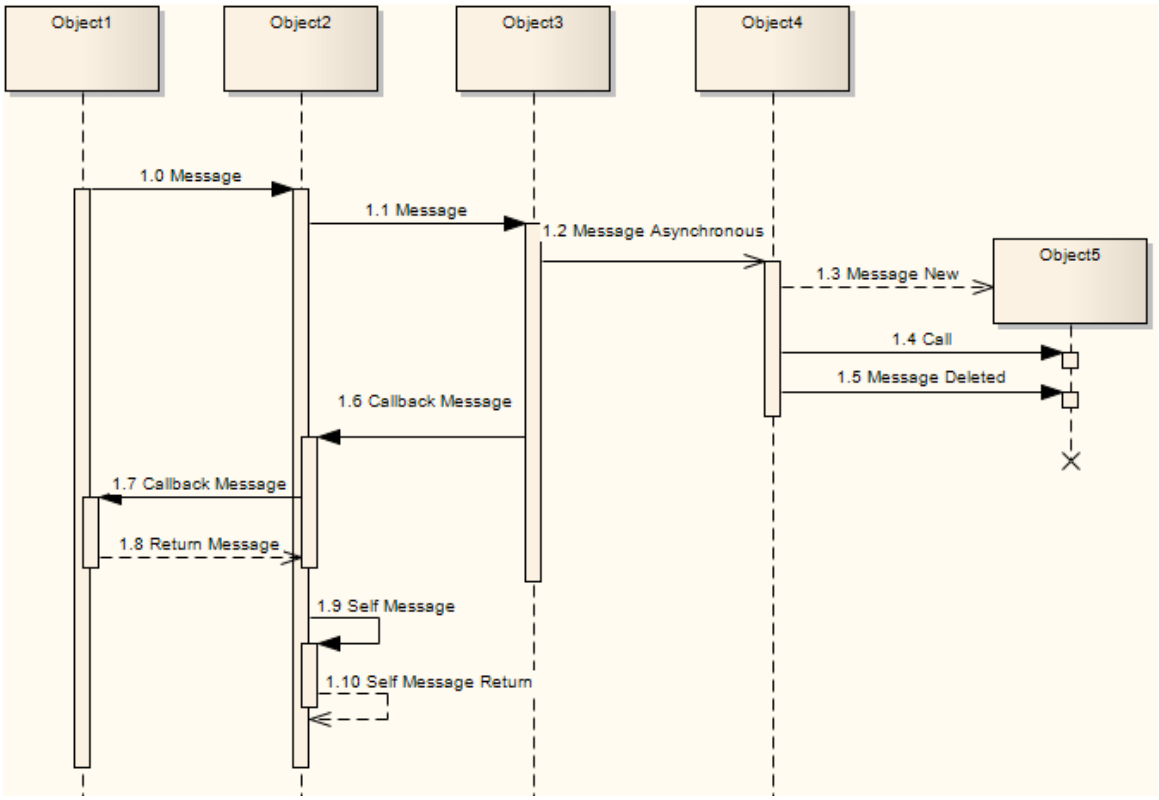


A Call is a type of Message connector that extends the level of activation from the previous Message. All Self-Messages create a new activation level, but this focus of control usually ends with the next Message (unless activation levels are manually adjusted). Self-Message Calls, as depicted in the image of the first Call, indicate a nested invocation; new activation levels are added with each Call. Unlike a regular Message between elements, a Call between elements continues the existing activation in the source element, implying that the Call was initiated within the previous Message's activation scope.

Toolbox icon

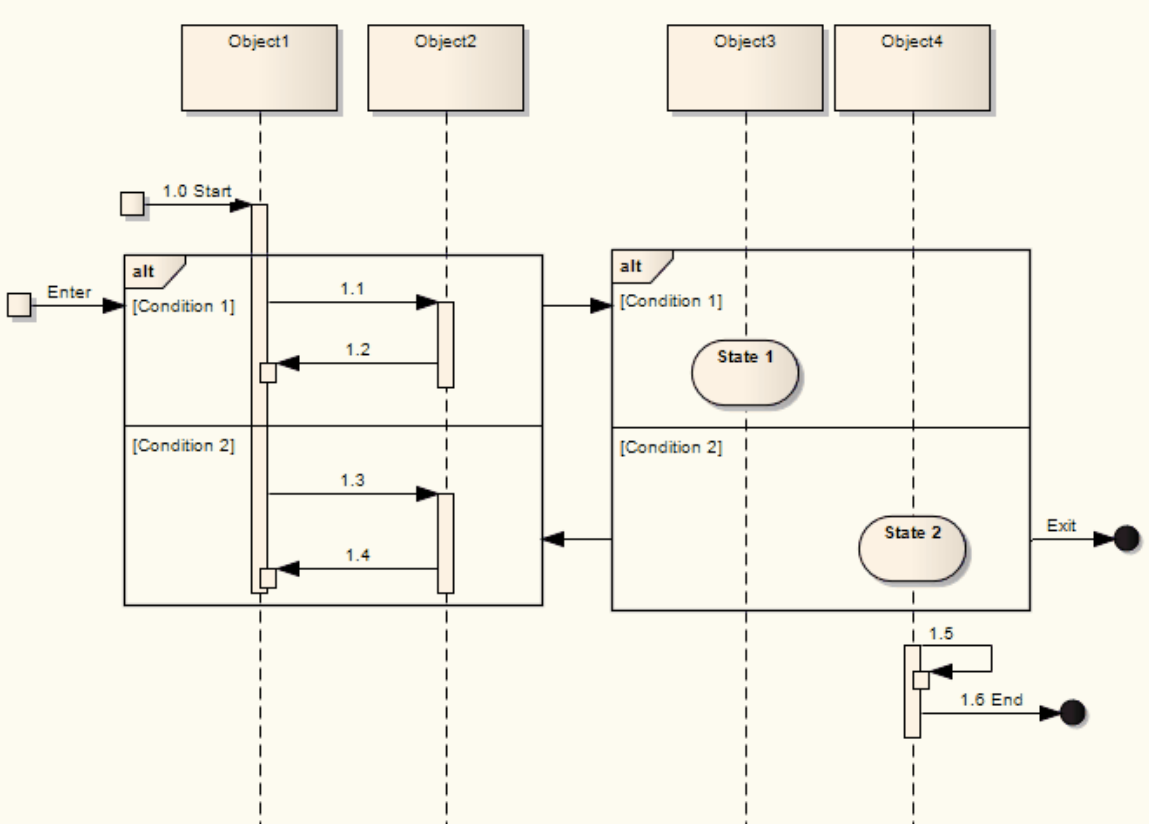
Message Examples

These are different types of Message available on Sequence diagrams. Note that Messages on Sequence diagrams can also be modified with Shape Scripts.



Other Sequence Messages

These are examples of Messages that are not part of the sequence described by the diagram.


Object1
Object2
Object3
Object4
1.0 Start
alt
Enter
[Condition 1]
1.1
1.2
[Condition 2]
1.3
1.4
alt
[Condition 1]
State 1
[Condition 2]
State 2
Exit
1.5
1.6 End

Change the Timing Details

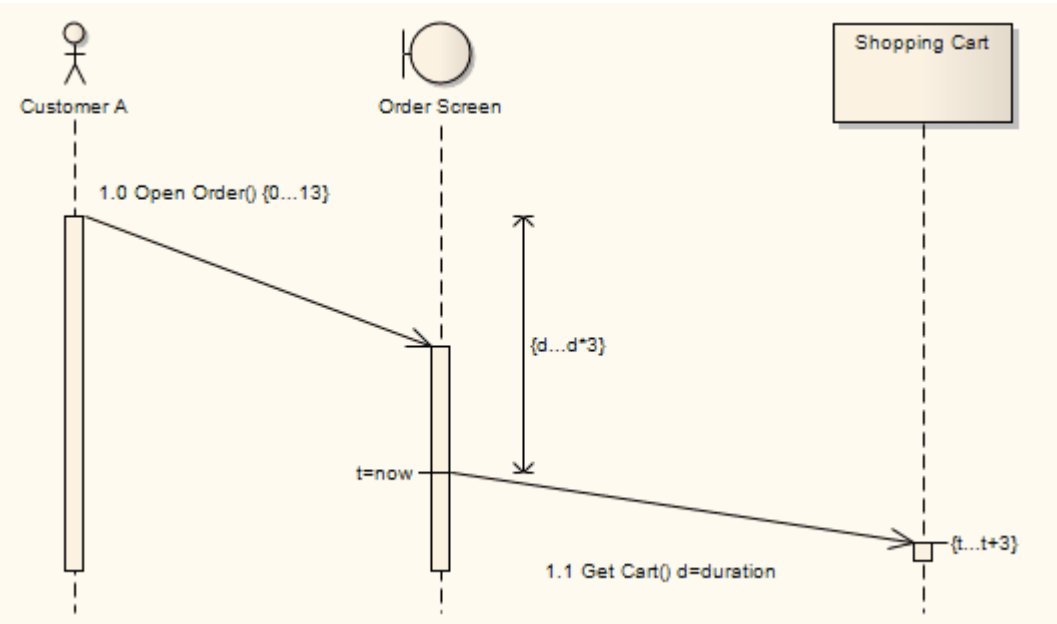
It is possible to change the timing details of a Message in a Sequence diagram.

Access

Context Menu	Right-click on the Message | Timing Details

Change Timing

See the OMG Unified Modeling Language specification, (v2.5.1, *p. 511*).



In this diagram, on the Open Order Message:

- 'Duration Constraint' has been set to 0...13

On the Get Cart Message:

- 'Duration Constraint Between Messages' has been set to d...d*3
- 'Duration Observation' has been set to d=duration
- 'Timing Constraint' has been set to t...t+3
- 'Timing Observation' has been set to t=now

By typing a value in the 'Duration Constraint' field, you enable the Message angle to be adjusted. After clicking on the OK button on the 'Timing Details' dialog, click on the head of the Message connector and drag the connector up or down to change the angle. You cannot extent the angle beyond the life line of the connecting sequence object or create an angle of less than 5 degrees.

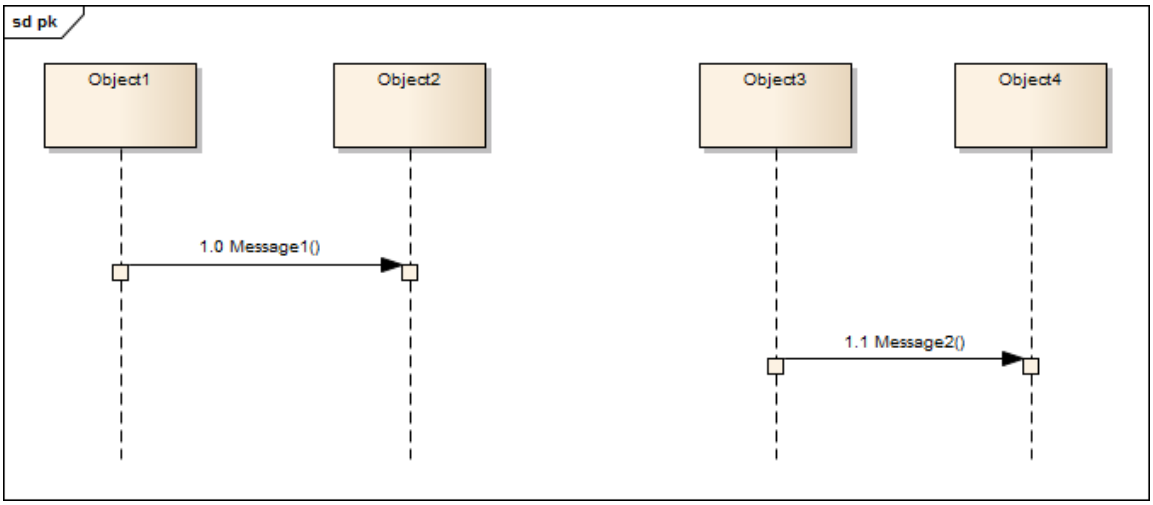
You can also create the 'Duration Constraint Between Messages' line by dragging the 'General Ordering' arrow up to the point at which the previous message joins the source Lifeline for the current message. A dialog displays on which you enter the value for the constraint. Having created the line, you can move it to any point within half way along the current message and half way along the previous message, to avoid overlap with other message timing details. You can edit or delete the value either through the 'Timing Details' dialog or by right-clicking on the line itself and selecting the appropriate context menu option.

Field	Action
Duration	Indicate the minimum and maximum

Constraint	limits on how long a message can last.
Duration Constraint Between Messages	Indicate the minimum and maximum interval between sending or receipt of the previous message at the current message's source Lifeline, and sending the current message.
Duration Observation	Capture the duration of a message.
Timing Constraint	Indicate the minimum and maximum time at which the message should arrive at the target.
Timing Observation	Capture the point at which the message was sent.

General Ordering

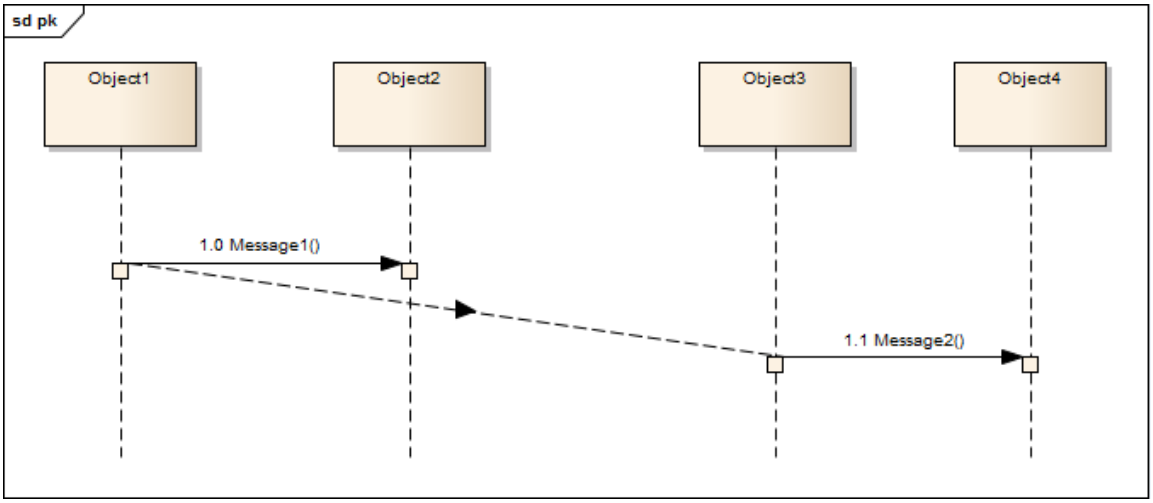
In a Sequence diagram, the workflow is represented by the sequence of Messages down the diagram. Messages near the top of the diagram are passed before Messages lower down the diagram.



In the diagram, Message 1 is earlier than Message 2. However, in a complex diagram, or when representing finely timed operations or parallel processing, this might not be apparent. You can reinforce the sequence using a 'General Ordering' arrow.

Click on the Message arrow. A small arrow displays at the source anchor point.

Click on this arrow and drag it to the start of the next Message in sequence (Message 2 in the example). The General Ordering arrow displays, indicating that the second Message follows the first.



You can have more than one General Ordering arrow issuing from or targeting a Message, if necessary.

Asynchronous Signal Message

You define a Message as an asynchronous signal message by displaying the Properties window for the Message and setting the 'Synch' field to 'Asynchronous', and the 'Kind' field to 'Signal'. A synchronous message cannot be used to convey signals, so setting the 'Synch' field to 'Synchronous' disables the 'Kind' field.

'Return Value', 'Assign To' and the Operations button, which are not applicable to asynchronous signals, are disabled.

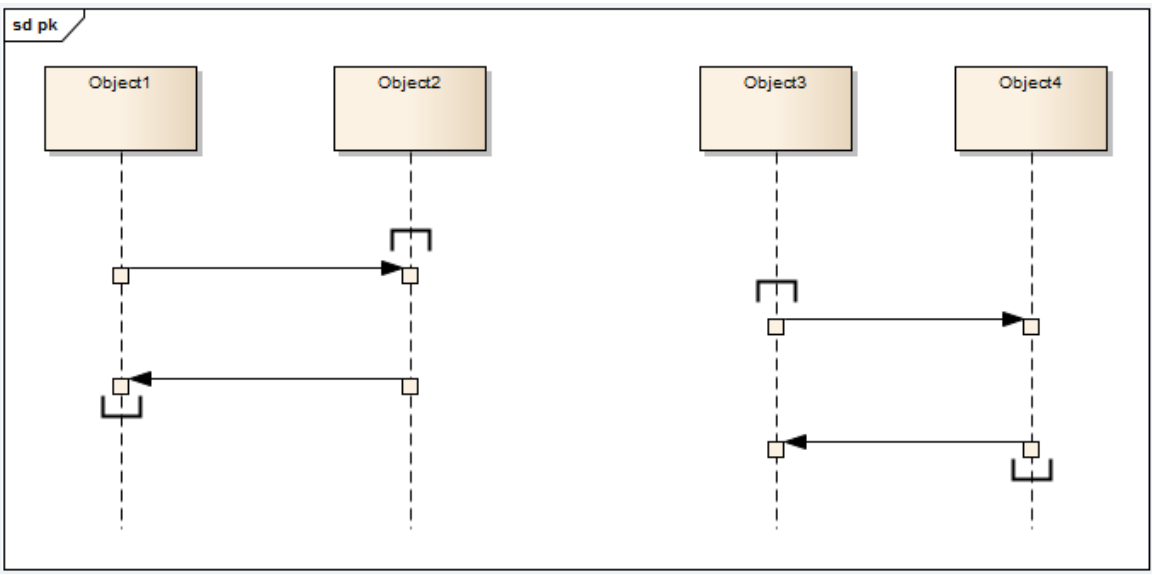
The Operations button changes to a Signal button, which you click on to associate the asynchronous signal message with a Signal element in the model. You can type the arguments corresponding to the Signal attributes into the 'Argument(s)' field.

When you click on the Signal button, the 'Select Signal' dialog displays, through which you locate and select the required Signal element.

Co-Region Notation

Co-Region notation can be used as a short hand for parallel combined fragments. You can add this notation to a Sequence diagram using the 'Co-Region' submenu, which you display by right-clicking on a connector in a Sequence diagram and selecting the 'Co-Region' option. There are four sub-options available:

- Start at head
- End at head
- Start at tail
- End at tail



Sequence Diagrams and Version Control

You might create Sequence diagrams that use elements from other Packages as the Lifelines within the diagram. In such cases, the diagrams could be corrupted when the element Packages are checked in and out under Version Control. This is because during checkout the elements are first deleted from the model and then re-imported, and although they are reinstated in the diagrams, any Messages connecting them are not.

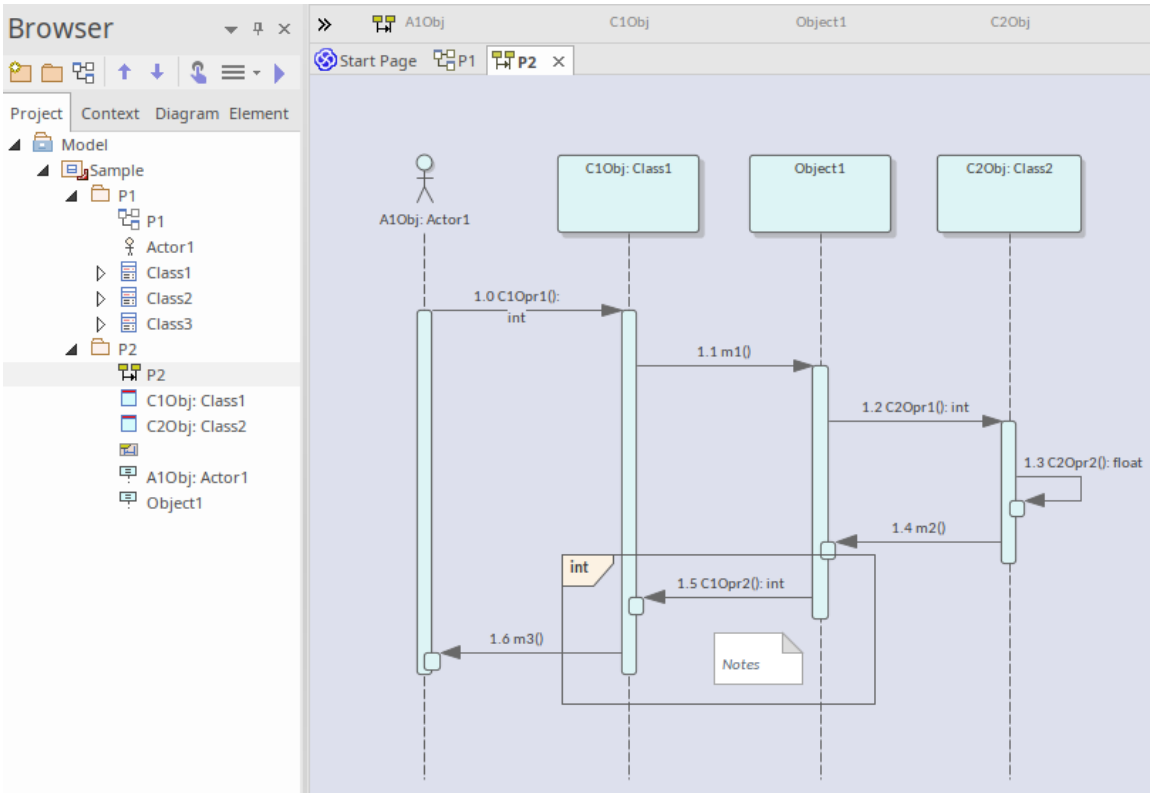
So, if the diagram and its elements reside in different Packages, a round-trip of the element Package through Version Control might damage the Sequence diagram.

The solution is to drag-and-drop each Class onto the Sequence diagram as an object - when you drop the Class onto the Sequence diagram, in the 'Paste Element' dialog select the 'as Instance of Element (Object)' option. This creates a new object in the diagram's parent Package, based on the selected Class element. You then create the Messages between the objects.

Therefore, to ensure that a Sequence diagram is not damaged by round-trips of other Packages through Version Control, remember that:

- The Lifelines must be objects (even though you can drop elements as Lifelines onto a Sequence diagram, it is not a strictly UML compliant construct)
- The Lifelines must be in the same Package as the diagram

This illustration shows the Browser window with two Packages: P1, containing the elements, and P2, containing a Sequence diagram that uses those elements. The diagram itself is also shown.



This diagram is not damaged when round-tripped through Version Control, because all the Lifelines are objects and these objects reside in the same Package as the Sequence diagram.

Notes

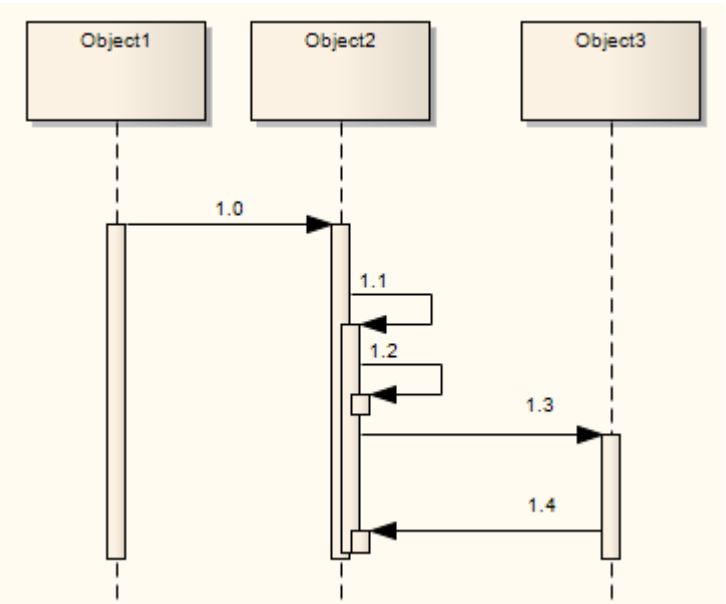
- These recommendations also apply to Communication diagrams

Sequence Element Activations

Sequence elements in a Sequence diagram have Activation rectangles drawn along their lifelines. These rectangles describe the time the element is active during the overall period of processing. This visual representation can be suppressed by right-clicking the Sequence diagram, and selecting 'Suppress Activations'.

In general, Enterprise Architect calculates the period of activation for you, but in some cases you might want to fine tune the rectangle length. There are several context menu options on a Sequence Message that you can use to accomplish this. To access the context menu, right-click on the message and select 'Activations'.

A more convenient way to change activation levels is directly on the diagram. Whenever appropriate, left arrows and/or right arrows display on specific connectors. In this diagram, see connector 1.3. Click on the arrow to raise or lower the activation level.

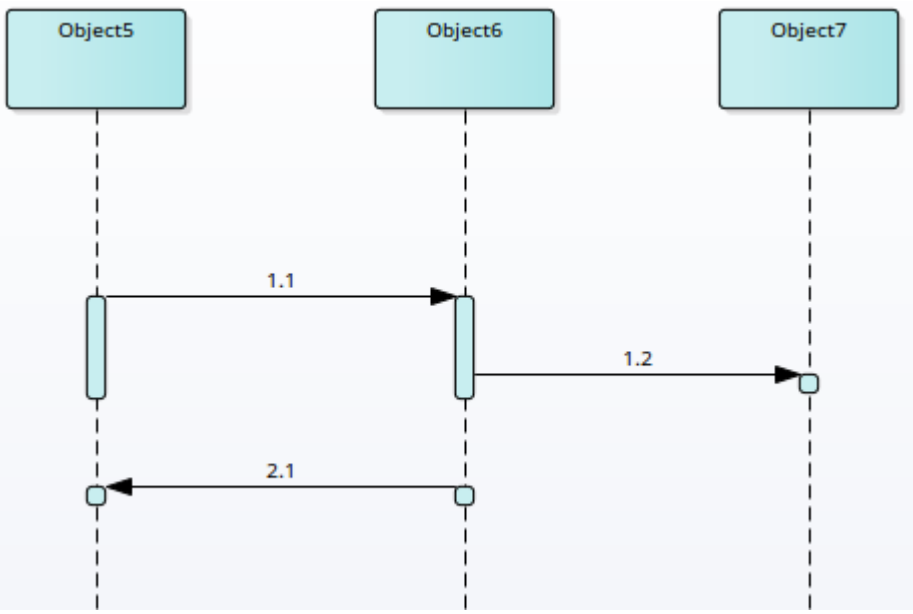


Branch With Previous Message

[This section describes a method of representing concurrent messages as defined in UML **prior to** UML 2.0, and is included to support models that might still apply it.

From UML 2.0 onwards, the notation has been replaced by Fragments. It is recommended that you consider upgrading your models to make use of Fragments and other more recent improvements in notation.]

Having set out the Lifelines and Sequence Messages with the appropriate message grouping and activation levels, you might want to indicate that two messages in different Message Groups and at different Activation levels issuing from a Lifeline are branches, or executed concurrently. Consider this example:



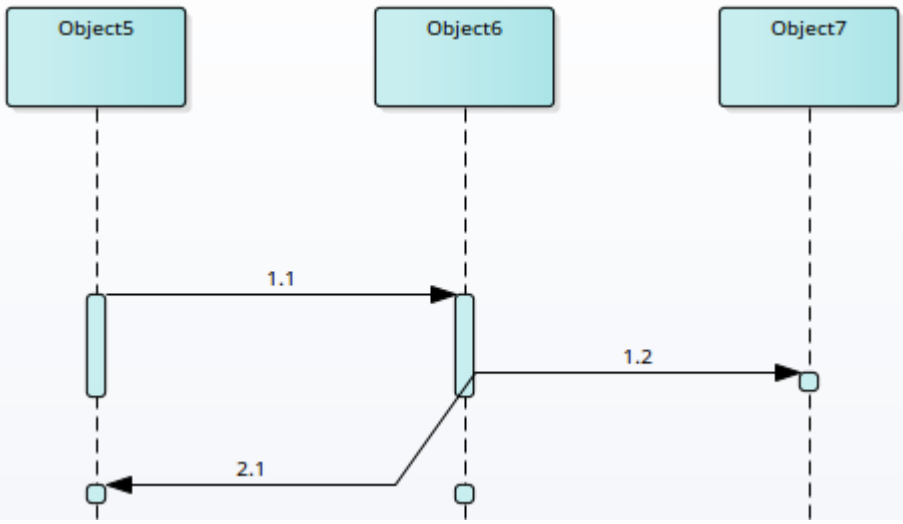
Message 1.1 passes from Object 5 to Object 6, and then Message 1.2 passes to Object 7 and Message 2.1 passes back to Object 5. It appears that the Messages go in the

sequence 1.1, 1.2 and then 2.1. However, you want to indicate that Message 2.1, whilst separate, is concurrent with Message 1.2.

In this case:

- Right-click on the later Message (2.1) and select the option 'Branch with Previous Message'

The source anchor for Message 2.1 then becomes the same as the source for Message 1.2, the immediately previous message. They are separate but concurrent Messages from the same Lifeline.



If it later becomes unnecessary to show that the Messages are branches, right-click on the later message (2.1) and deselect the 'Branch with Previous Message' option.

Context menu options

Option	Description
Start New	Starts off a new round of processing in

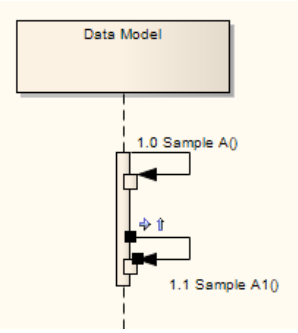
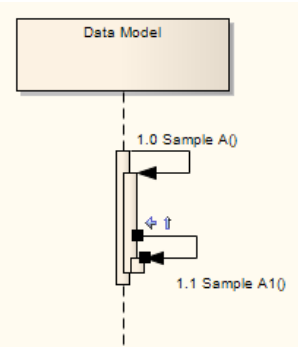
Message Group	the current diagram. This enables you to describe more than one processing scenario in a single diagram.
Extend Source Activation Down	Forces an element to stay active beyond the normal processing period. This could be used to express an element that continues its own processing concurrently with other processes.
Extend Source Activation Up	Forces an element's activation upwards.
End Source Activation	Truncates the activation of the source element after the current message. This is useful for expressing an asynchronous message after which the source element becomes idle.
End Target Activation	Ends a Forced Activation started by the 'Extend Source Activation' options.
Raise Activation Level	Displays on the context menu only where its use is appropriate. For example, after a self-message the next message starts by default at a lower

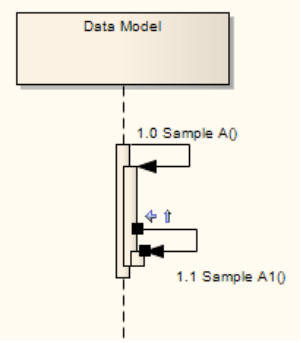
	activation level but the 'Raise Activation Level' command displays on the context menu to enable you to raise its level.
Lower Activation Level	Displays on the context menu only where its use is appropriate.

Lifeline Activation Levels

Complicated processing systems can be easily negotiated and reflected in Sequence diagrams, by adding activation layers on a single lifeline.

Examples

	A Class invokes the method Sample A, which in turn calls Sample A1. To produce the arrangement in the diagram: 1. In the Diagram Toolbox click on to display the 'Find Toolbox Item' dialog and specify 'Interaction'. 2. Click on the 'Self-message' icon in the 'Interaction Relationships' panel. 3. Click on the lifeline.
	In order to raise the Activation level of Sample A1, click on the raise arrow of the selected connector. The lifeline now visually depicts that method Sample A1 is called during the processing of Sample A.
	In this example, a few more



self-messages have been added.

The message Sample A2a is called from Sample A2, which in turn is called from Sample A (not Sample A1).

Sample A1 is called from Sample A.

Sequence Message Label Visibility

Hide and show labels used in Sequence messages

Step	Action
1	Right-click on the message within the Sequence diagram and select 'Set Label Visibility'. The 'Label Visibility' dialog displays.
2	Select or clear the checkbox against each message label to display or hide, respectively.
3	Click on the OK button to save the settings.

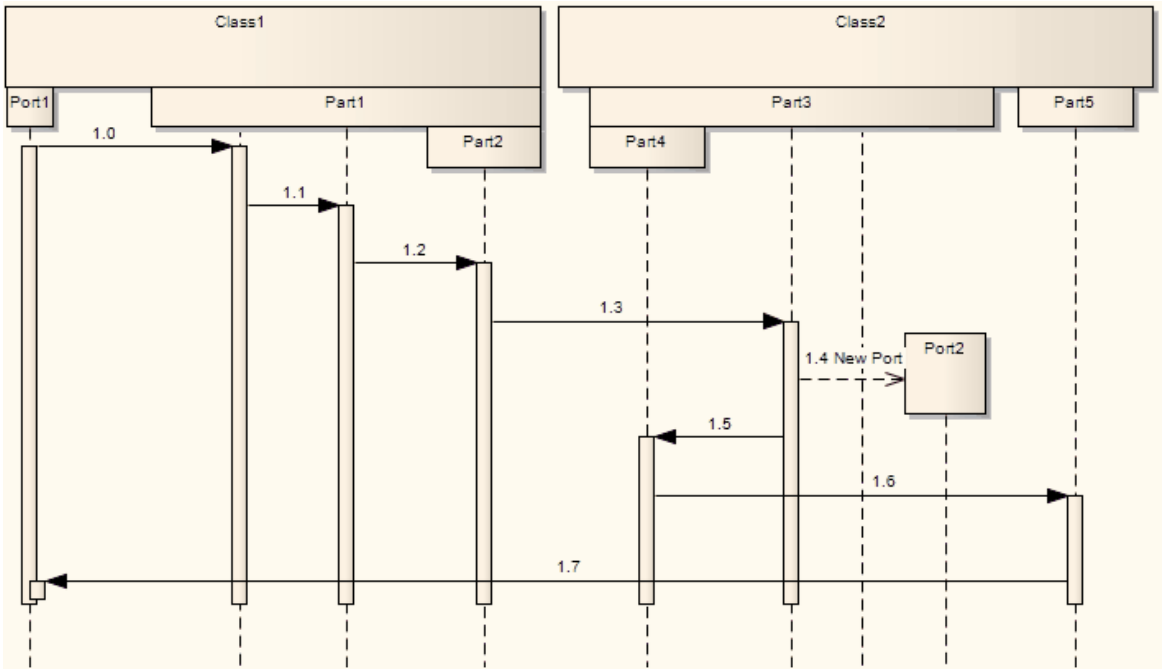
Change the Top Margin

In order to change the top margin of a Sequence diagram from the default 50 units, right-click on the diagram and select the 'Set Top Margin' option. You can set the top margin to any value between 30 and 250 units. You can then use this space to, for example, add Note or Text elements to provide documentation on the diagram.

Inline Sequence Elements

On a Sequence diagram it is possible to represent existing child Part and Port elements, which render as inline sequence elements under their parent Class sequence element.

Example Sequence Diagram with Parts and Ports



Represent Part and Port elements on a Sequence diagram

Step	Action

1	Right-click on the Sequence elements containing the child Ports or Parts, and select 'Features | Interaction Points'. The Features window displays at the 'Interaction Points' tab.
2	Select the checkbox against each Part or Port to show, and click on the Close button.

Communication Diagram

A Communication diagram is a diagram that shows the interactions between elements at run-time in much the same manner as a Sequence diagram. However, Communication diagrams are used to visualize inter-object relationships, while Sequence diagrams are more effective at visualizing processing over time.

Communication diagrams employ ordered, labeled associations to illustrate processing. Numbering is important to indicate the order and nesting of processing. A numbering scheme could be:

1

1.1

1.1.1

1.1.2

1.2, and so on.

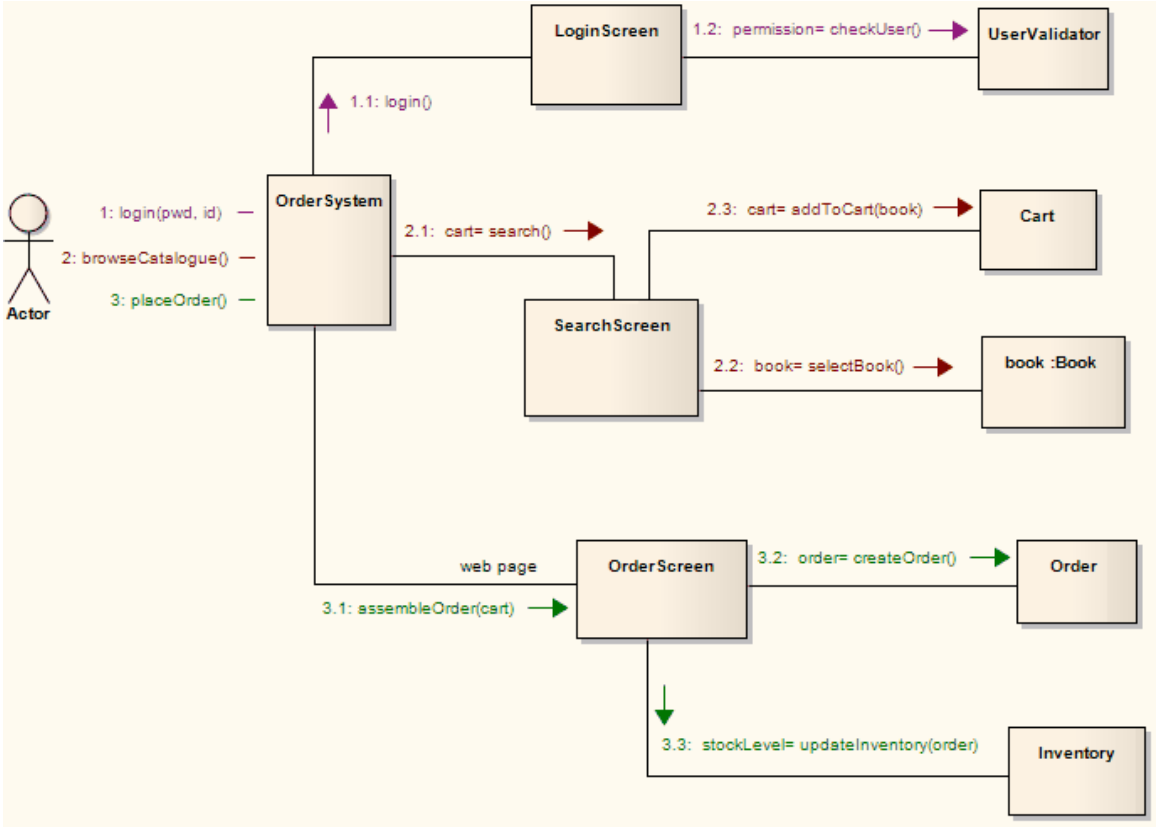
A new number segment begins for a new layer of processing, and would be equivalent to a method invocation.

You generate Communication diagram elements and connectors from the 'Communication' pages of the Diagram Toolbox.

Example Diagram

This example illustrates a Communication diagram among cooperating object instances. Note the use of message levels to capture related flows, and the different colors of the

messages.

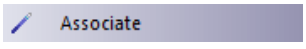


Communication Diagram Element Toolbox Icons

Icon	Description
Actor	An Actor is a user of the system; user can mean a human user, a machine, or even another system or subsystem in the model.
Object	An Object is a particular instance of a Class at run time.

	A Boundary is a stereotyped Object that models some system boundary, typically a user interface screen.
	A Control element represents a controlling entity or manager that organizes and schedules other activities and elements.
	An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system.
	Packages are used to organize your project contents, but when added onto a diagram they can be used to depict the structure and relationships of your model.

Communication Diagram Connector Toolbox Icons

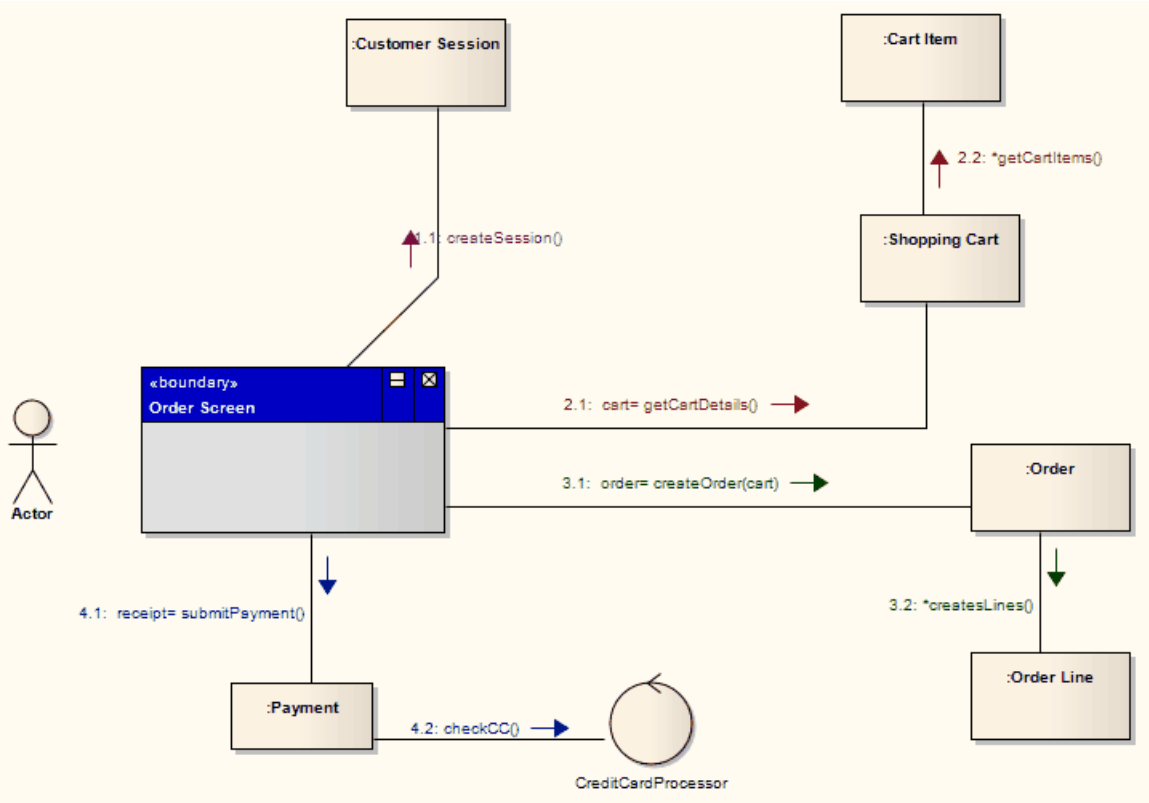
Icon	Description
	An Association implies that two model elements have a relationship, usually

	implemented as an instance variable in one or both Classes.
	The Nesting Connector is an alternative graphical notation for expressing containment or nesting of elements within other elements.
	A Realizes connector represents that the source object implements or Realizes its destination object.

Communication Diagrams in Color

It is possible to highlight particular message flows in a Communication diagram using different colors for each message set.

Highlight the colors in a Communication diagram



Step	Action
1	Select 'Start > Application > Preferences > Preferences > Communication Colors'. The 'Communication Message Coloring' page of the

	'Preferences' dialog displays.
2	Select the 'Use Communication Message Coloring' checkbox.
3	Click on the drop-down arrow of each 'Message n' field, and select the required color for each message group.
4	Click on the Close button. On your Communication diagram, each sequence group of messages displays in a different color, as shown.

Messages (Communication Diagrams)

A Message in a Communication diagram is equivalent in meaning to a Message in a Sequence diagram. It implies that one object uses the services of another object, or sends a message to that object. Communication Messages in Enterprise Architect are always associated with an Association connector between object instances. Always create the Association first, then add a Message to the connector.

Messages can be dragged into a suitable position by clicking and dragging on the message text.

Communication Messages are ordered to reflect the sequencing of the diagram. The numbering scheme should reflect the nesting of each event. A sequencing scheme could be:

1

2, 2.1, 2.2, 2.3

3

This would indicate the single sequence of events 2.1, 2.2 and 2.3 occurs within an operation initiated by event 2. This is the default pattern applied by Enterprise Architect.

Alternatively, the sequence could be:

1

2

2.1, 2.1.1, 2.1.1.1

2.2, 2.2.1, 2.2.1.1

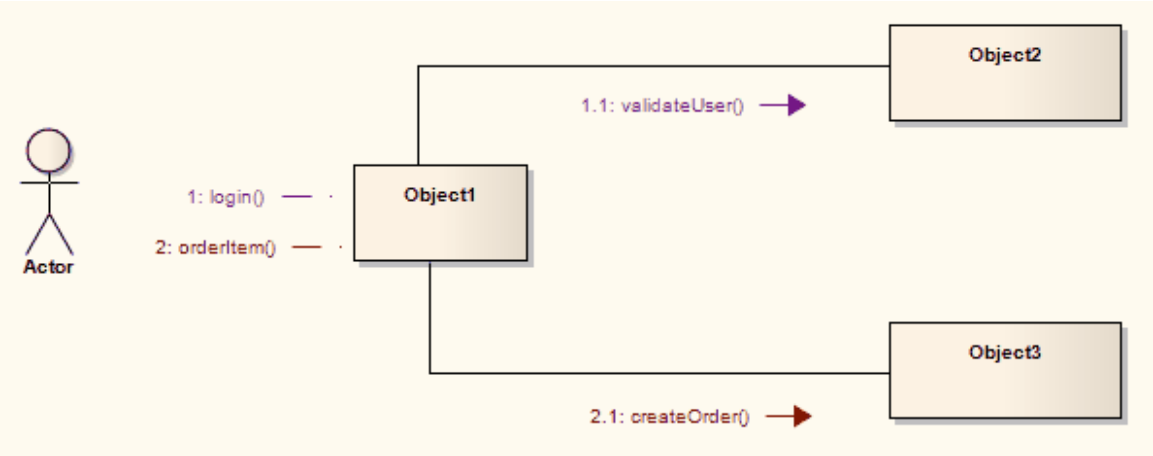
3

This would indicate that two sequences of events can be initiated by event 2, and 2.1 and 2.2 are separate sequences, not consecutive events in one sequence. You can set the sequence pattern and order using the Properties window for the Message and the 'Sequence Communications' dialog.

If the target object is a Class or has its instance classifier set, the drop-down list of possible message names includes the exposed operations for the base type.

Create a Communication Message

Create a Communication Message



Step	Action
1	Open a diagram (one of: Communication, Analysis, Interaction Overview, Object, Activity or StateMachine).
2	Add the required objects.
3	Add an Association relationship between each pair of objects that communicate.
4	Right-click on an Association to display the context menu.
5	Select the appropriate option to add a Message from one object to the other.

6	When the Properties window for the Message displays, type in a name and any other required details.
7	Click on the OK button. The Message is added, connected to the Association and Object instances.
8	Move the Message to the required position.

Re-Order Messages

When constructing your Communication diagram, it is frequently necessary to create or delete Message 'groups' and to re-order the sequence of Messages. There are two displays that help you perform these tasks: the Properties window for the Message and the 'Sequence Communications' dialog.

Organize Message Groups

If you have several Messages in the form 1.1, 1.2, 1.3, 1.4, for example, but want to start a new numbering group on, say, the third Message (that is, 1.1, 1.2, 2.1, 2.2, 2.3), you can change that Message in the series to a Start Group message.

Step	Action
1	Double-click on a Message name. The Properties window for the Message displays.
2	To make the selected Message the start of a new group, select the 'Start New Group' checkbox.
3	If required, in the Notes window for the Message, type an explanatory note. You can format the text using the Notes toolbar at

	the top of the field.
4	Click on the Save icon to save changes.

Sequence Messages

In larger and more complex diagrams, you might have to use deeper levels of Messages in a group; for example, 1, 1.2, 1.2.1, 1.2.1.1. You might also have to change the sequence of Messages, making Message 1.3, for example, into Message 1.1.

Step	Action
1	Select the 'Sequence Communication Messages' option after you: • Select the 'Design > Diagram > Options' ribbon option, or • Right-click on the diagram background, or • Right-click on a Message The 'Communication Messages' dialog displays.
2	Click on the Message to adjust and, at the bottom of the dialog, click on the: • Move Up or Move Down (Hand) buttons to move the Message up or down the sequence (for

	example, *Message 1.2* to *Message 1.1* or *1.3*) • Move Left or Move Right (Hand) buttons to move the Message up or down a level (for example, *Message 1.2.1* to *Message 1.2* or *Message 1.2.1.1*)
3	Repeat step 2 until the Message sequence and levels match your requirements. You might have to adjust other Message numbers (in group, sequence or level) to accommodate the changes you have made.
4	Click on the OK button to save changes.

Interaction Overview Diagram

Interaction Overview diagrams visualize the cooperation between other Interaction diagrams to illustrate a control flow serving an encompassing purpose. As Interaction Overview diagrams are a variant of Activity diagrams, most of the diagram notation is the same, as is the process of constructing the diagram.

Decision points, Forks, Joins, Start points and End points are the same. Instead of Activity elements, however, rectangular elements of two types are used:

- Interaction elements display an inline Interaction diagram, which can be any one of the four types (Sequence, Timing, Communication or Interaction Overview)
- Interaction Occurrence elements are references to an existing Interaction diagram: they are visually represented by a frame, with ref in the frame's title space; the diagram name is indicated in the frame contents

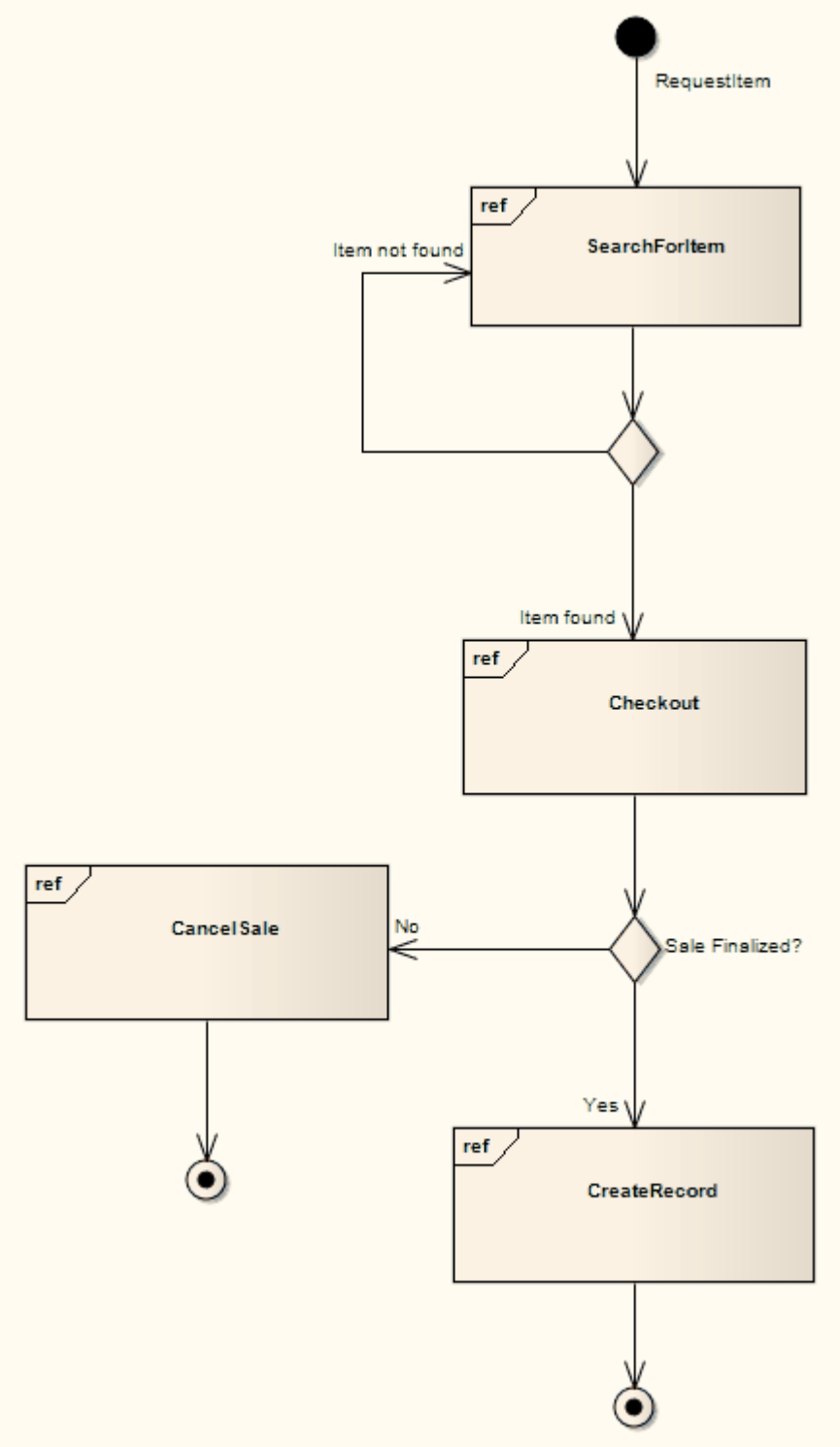
To create an Interaction Occurrence, simply drag an Interaction diagram from the Browser window onto your Interaction Overview diagram. The ref frame displays, encapsulating an instance of the Interaction diagram.

You generate Interaction Overview diagram elements and connectors from the 'Interaction Overview' pages of the Diagram Toolbox.

Example Diagram

This diagram depicts a sample sale process, shown in an Interaction Overview diagram, with sub-processes abstracted within Interaction Occurrences.

The diagram appears very similar to an Activity diagram, and is conceptualized the same way; as the flow moves into an interaction, the respective interaction's process must be followed before the Interaction Overview's flow can advance.

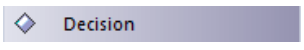


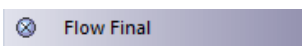
Interaction Overview Diagram Toolbox Icons

Icon	Description

	The Diagram Frame is a reference to an Interaction diagram.
	The Interaction Occurrence is a reference to an existing Interaction diagram. It is displayed as a frame, with 'ref' in the frame's title space and the diagram name in the frame contents. It is used inline as an AcitivityInvocation.
	The Control Flow is a connector connecting two nodes, modeling an active transition.

Interaction Overview Diagram Control Nodes Toolbox Icons

Icon	Description
	The Initial element defines the start of a flow when an Activity is invoked.
	A Decision is an element that indicates a point of conditional progression: if a condition is true, then processing

	continues one way; if not, then another.
	A Merge Node brings together a number of alternative flow paths in Activity, Analysis and Interaction Overview diagrams.
	A Synch state is useful for indicating that concurrent paths are synchronized. They are used to split and rejoin periods of parallel processing.
	A Fork/Join element can be used to: 1) split a single flow into a number of concurrent flows, 2) join a number of concurrent flows or 3) both join and fork a number of incoming flows to a number of outgoing flows.
	A Fork/Join element can be used to: 1) split a single flow into a number of concurrent flows, 2) join a number of concurrent flows or 3) both join and fork a number of incoming flows to a number of outgoing flows.
	The Flow Final element depicts an exit from the system, as opposed to the Activity Final, which represents the

	completion of the Activity.
	The Final element, indicates the completion of an Activity; upon reaching the Final, all execution is aborted.

UML Elements

UML elements are the building blocks of a model. They are contained in a repository and are depicted in diagrams connected by relationships to create narratives that describe the enterprise, business or software system. Each element has a type that dictates its presentation and the rules that govern how it is connected to other elements. Elements are displayed in a hierarchy in the Browser window and each element plays a role in defining the system being modeled. They are grouped into structural or behavioral element types, and each type can be used at any stage of the representation of a system. For example, Activities can be used to define the way an organization carries out a business function, or to define the steps in a computer algorithm.

Behavioral Diagram Elements

Behavioral diagrams depict the behavioral features of a system or business process. Elements that can appear on Behavioral diagrams include Activity, Interaction, Lifeline, StateMachine and Use Case.

Structural Diagram Elements

Structural diagrams depict the structural elements composing a system or function. Elements that can appear on Structural diagrams include Class, Component, Interface, Node and Package.

Behavioral Diagram Elements

This section provides detailed descriptions of the elements commonly used in modeling with Behavioral diagrams in Enterprise Architect.

Action

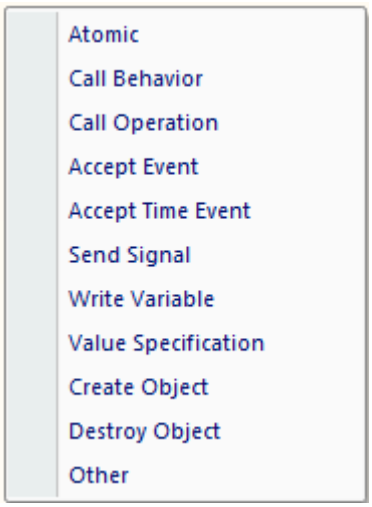
Description

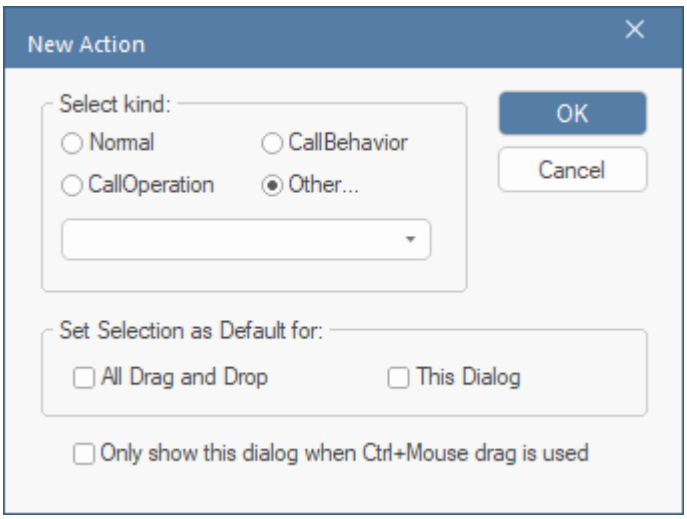
An Action element describes a basic process or transformation that occurs within a system, and is the basic functional unit within an Activity diagram. Actions can be thought of as children of Activities; both represent processes, but Activities can contain multiple steps or decomposable processes, each of which can be embodied by an Action. An Action cannot be further broken down or decomposed.

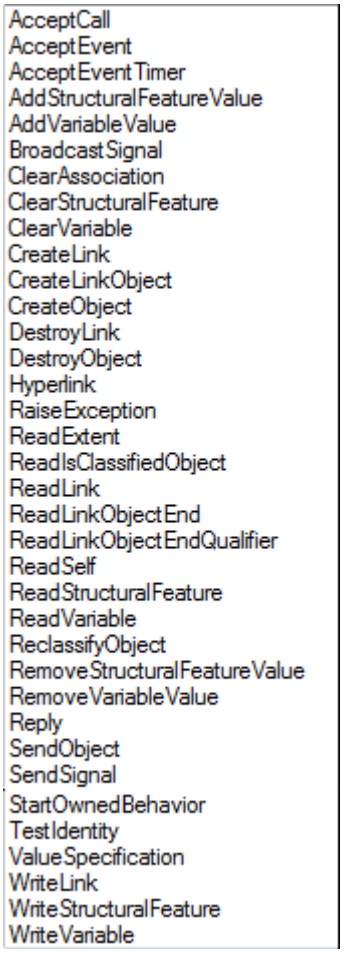
For the purposes of simulation, you can define the effect of a basic (Atomic) Action on the 'Action' tab of the Properties window for the element, using a JavaScript expression in the 'Effect' field to define the duration of the effect and selecting to display the effect on the diagram. An Action can be further defined with pre-condition and post-condition notes.

Certain properties can be graphically depicted on the Action. When you first drag the 'Action' icon from the Toolbox onto a diagram, the system prompts you to select from a list of the more common types of Action to create. If you select the 'Other' option on this list, the 'New Action'

dialog displays; the 'Other' drop-down list on this dialog enables you to select a more specialized type of Action from a complete list of Action types.







If you later decide that the Action type is not appropriate, you can change it on the 'Action' tab of the Properties window - select the required new type from the 'Kind' drop-down list. For a Value Specification Action, you can also set the value on this tab.

The data values passed out of and into an Action can be represented by Action Pins. For an Action type other than a basic Action, you can also assign Action Pins to represent specific properties.

An Action can also be depicted as an Expansion Node to indicate that the Action consists of an Expansion Region.

If you have defined a Decision Table for the Action element, you can select options on the element's context menu to render the element on a diagram as the Decision

Table, showing the rules as either rows or columns. You can also return the element to its normal element shape.

Note: Selecting the option 'Only show this dialog when Ctrl+Mouse drag is used' on the dialog 'New Action', suppresses display of the list of common Action types, as well as the dialog 'New Action'. In this case, dragging from the toolbox while holding the 'Ctrl' key causes the list to display.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p. 443) states:

An Action is a fundamental unit of executable functionality contained, directly or indirectly, within a Behavior. The execution of an Action represents some transformation or processing in the modeled system, be it a computer system or otherwise.

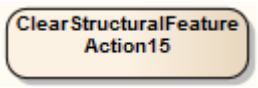
The OMG Unified Modeling Language specification, (v2.5.1, p. 443) also states:

An Action may accept inputs and produce outputs, as specified by InputPins and OutputPins of the Action,

respectively. Each Pin on an Action specifies the type and multiplicity for a specific input or output of that Action.

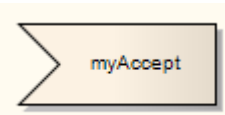
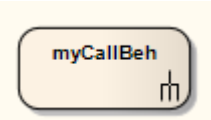
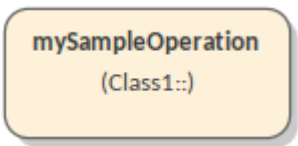
Action Types

Action elements are extremely versatile. Enterprise Architect supports a wide range of specific Action types that you can use to represent or enact a discrete object, operation or behavior. Actions of most types are depicted as a round-cornered rectangle containing the Action type and Action name, as shown.



Action Element Notation

Certain types of Action element have their own specific notation; for example:

Action Kind	Notation
AcceptEvent	
AcceptEvent Timer	
CallBehavior	
CallOperation	

SendSignal	

AcceptEvent Actions

An AcceptEvent Action element has a selectable output result Action Pin assigned to it, and one or more Triggers to denote the type of events accepted by the Action. You define the Triggers on the 'Triggers' tab of the Properties window. In a simulation, an AcceptEvent Action without a Trigger will block the simulation at the Action element.

Field	Action
Name	Type the name of the trigger.
Type	Click on the drop-down arrow and select the type of trigger: Call, Change, Signal or Time: • Call - specifies that the event is a CallEvent, which sends a message to the associated object by invoking an operation • Change - specifies that the event is a ChangeEvent, which indicates that the transition is the result of a change in value of an attribute

	• Signal - specifies that the event is a SignalEvent, which corresponds to the receipt of an asynchronous signal instance • Time - corresponds to a TimeEvent; which specifies a moment in time Code generation for StateMachines currently supports Change and Time trigger events only, and expects a specification value. In simulation, each Trigger should have a Signal. The result will be the Accept signal.
Specification	Specify the event instigating the Transition. For an AcceptEventTimer Action, you can type a JavaScript expression in this field evaluating to the number of ticks to wait for.

SendSignal Action & BroadcastSignal Action

A SendSignal Action has an assigned target ActionPin and a Signal. The Signal can have input ActionPins that bind its attribute parameters as arguments. For example:

::Sender: sig.binding_To_s1: Integer

In a model simulation, a SendSignal Action will transfer the values of the arguments into the attributes of the created Signal instance. The target ActionPin can have an empty object, to send the Signal into the root of the simulation space. If there is no target ActionPin, simulation will stop at the Action. If the target has an Object, the Signal will be sent to the Object. You must specify the Pin type of the target ActionPin in the classifier of the Object.

A BroadcastSignal Action is similar to a SendSignal Action, except that it does not have a target ActionPin. In a simulation, it always sends its Signal to the root of the simulation data.

You can model the Signal to be sent and the associated arguments to be conveyed, using the 'Signal' tab of the Properties window for the element.

Field/Button	Action
Signal	Click on ... and select the required signal from the 'Select Signal' dialog.
Attribute	Click on the drop-down arrow and select the attribute (as previously created in the Signal element) with which the arguments are to be associated.
Value	Type the appropriate value for the attribute.
	Click on this button and select the

Add	appropriate ActionPins from the 'Select Pin' dialog, to identify the arguments for the Signal. To assign more than one ActionPin, press the Ctrl key while you select each one.
Save	Click on this button to save your changes.

CallBehavior

A CallBehavior Action has a behavior such as an Activity, and a selectable ActionPin result that will put the return value. The CallBehavior Action can also transfer the values of its argument ActionPins into its behavior, if they are bound together. In a simulation, if the Action has no behavior, the simulation is blocked.

SendObject Action

A SendObject Action sends a copy of an Object from the requesting ActionPin to the target ActionPin. In a simulation, the SendObject Action must have both ActionPins, otherwise the simulation is blocked at the Action.

Structural Feature Actions

A StructuralFeature Action acts upon a modeling structural feature, such as a Port, Part or attribute of an Activity or of the classifier of an Object, which you identify within the Action element. Enterprise Architect supports these types of Structural Feature Action:

Action	Description
AddStructuralFeatureValue	Requires an object input ActionPin where the target object will be entered, and a result output ActionPin to hold the read result. If the object Port has no value at run time, the process will pause at the Action. In your model design, the Port should be connected to the Port of an Object or to an Object Node of an Activity. The properties of the Port or Object Node must be correctly set, and the value Port must be set up to capture the input value when the Action takes effect. The result ActionPin can be connected to an input consume Port or ActionPin. For example, it can be used at the next WriteStructuralFeature Action as the input value.
ClearStructuralFeature	Clears the single value of a structural attribute or a structural Port of an Object or an Activity, and sets the value of the

	structural feature to null.
ReadStructuralFeature	Resembles AddStructuralFeatureValue, except that the value Port is not necessary. In a simulation, if the Object's Port has no value at run time, the simulation will pause at the Action.
RemoveStructuralFeatureValue	Similar to ClearStructuralFeature except that it invokes a value ActionPin to input a value and, if that value matches the value of the specified structural feature, it sets the value to null. If the values do not match, the Action does not clear the structural feature value.
WriteStructuralFeature	Identical to AddStructuralFeatureValue. In a simulation, the value Port must be set up to capture the input value when the simulation runs the Action.

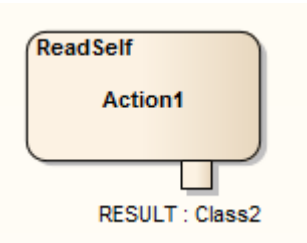
Set a StructuralFeature

Step	Action

1	Right-click on the Action element and select 'Advanced | Set Structural Feature: Add'.
2	On the 'Select Property' dialog (a variant of the 'Select <Item>' dialog), browse or search for the appropriate structural feature, and double-click on it. The feature name and location displays in the 'structuralFeature' field of the 'Set Structural Feature' dialog.
3	Click on the OK button to save the setting.

ReadSelf

A ReadSelf Action reads its own host object name into its result Port. You must set an output ActionPin for the result.



The Action must be within a Class, which is instanced during run time. When a simulation passes the Action, it puts the name of the instance of the Class into the result Port.

ReadSelf is one of a group of Object Actions, with CreateObject and DestroyObject.

Variable Actions

Variable Actions are closely concerned with the simulation of the behavior of and actions on Objects in a process. They have an association variable in the form of the Tagged Value variable with, as its value, the name of an Object in run-time. That is:

sim.ObjectName

Variable Actions provide the variable not only as an Object but also as a property (such as an attribute or Port) of an Object. For example:

sim.a.a1

The parameter a.a1 can have an integer value.

Variable Actions include:

- ReadVariable
- WriteVariable
- ClearVariable
- AddVariableValue
- RemoveVariable

ReadVariable

A ReadVariable Action has a Result Action Pin as an output Port. The value of the Port will be the result to be read, this being a copy of the variable read. Therefore, it is not affected by changes to the value of the variable. For example, if the variable is sim.Object.a that has the value 3,

and its value has been changed into 5 after it is read, the value read is still 3.

Before reading:

sim.Object.a = 3;

sim.Action1.result = null;

After reading:

sim.Object.a = 3;

sim.Action1.result = 3;

After a change in the value of the variable:

sim.Object.a = 5;

sim.Action2.value = 3;

In that example, the value is a Port of Action2 that is connected to the result Port of Action1 by an Object Flow connector.

WriteVariable

This Action has a Value Action Pin as an input Port. The value of the Port will be written into its variable. The result value is a copy of the variable from the Value Port.

ClearVariable

This Action clears all values of a variable, the variable being either an Object or a value.

AddVariableValue

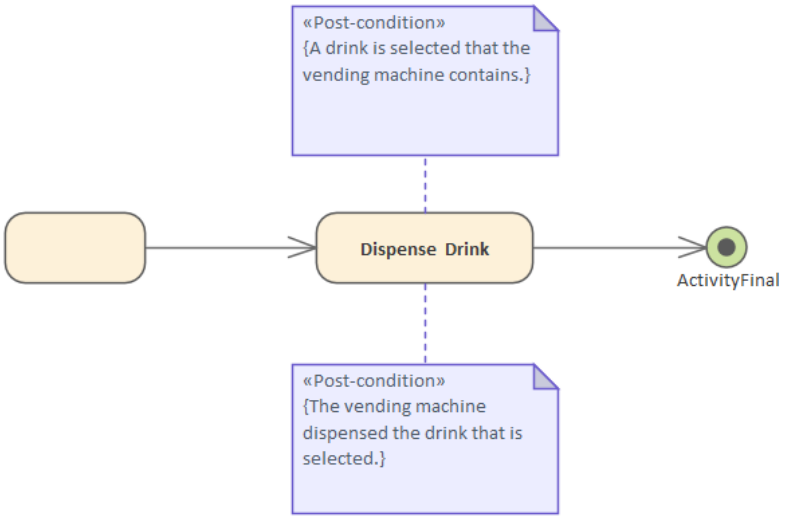
This Action is effectively the same as a WriteVariable Action, because the current variables of the simulation do not support multiple values.

RemoveVariableValue

This Action is effectively the same as a ClearVariable Action because the current variables of the simulation do not support multiple values.

Local Pre/Post Conditions

Actions can be further defined with pre-condition and post-condition notes, which constrain an Action's entry and exit.



Attach a constraint to an Action

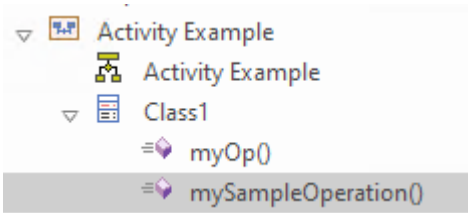
Step	Action
1	Right-click on the Action and select the 'New Child Element| Attach Constraint' option. A Note is created on the diagram, connected to the Action.
2	Right-click on the Note and select the 'View Properties' option. The 'Constraint' dialog displays.

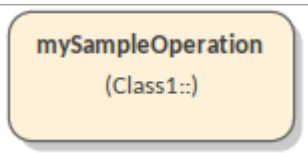
3	In the 'Constraint Type' field, click on the drop-down arrow and select the required constraint type.
4	In the 'Constraint' field, type the text for the constraint.
5	Click on the OK button to save the constraint.

Class Operations in Diagrams

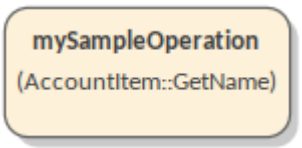
Operations from Classes can be represented by CallOperation Action elements on any diagram (such as an Activity, Custom or Analysis diagram). When an operation is shown as an Action, the notation of the element displays the name of the operation prefixed by the name of the Class from which it comes.

Add an Operation to a Diagram

Step	Action
1	Open the target diagram.
2	From the Browser window open a Class and locate the operation to be added to the diagram.
3	Drag the operation on to the diagram. 
4	When the operation has been added to the diagram, the CallOperation Action resembles this:

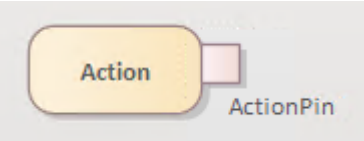
Change the Operation That an Action Refers to

Step	Action
1	Right-click on the Action and select the 'Advanced | Set Operation' option. The 'Set Operation' dialog displays.
2	If necessary, in the 'Go To Namespace' field, select the model that contains the operation. Browse for the operation.
3	When you have located the operation, double-click on it. The Action updates to show the new classifier and operation names. 

Notes

- If you want to locate, in the **Browser window**, the operation that an Action was created from, right-click on the Action in the diagram and select the 'Find | Locate Operation in Project Browser' option
- If you want to display the previously-generated code for the Class containing the operation, click on the Action in the diagram and press either **Ctrl+E** or **F12**; the 'Code Editor' view displays, with the code generated for the Class (if no code has been generated for the Class, the 'Code Editor' does not display)
- In a simulation, the CallOperation Action must have a calling operation and a target object ActionPin, the operation belonging to the object that comes from the target ActionPin; if you don't set these properties, simulation will be blocked at the Action
- If the 'Name' property of the CallOperation is empty, then the name of the Class operation name will be displayed in its place. If the operation's name is modified, the displayed name will be updated to reflect that change.

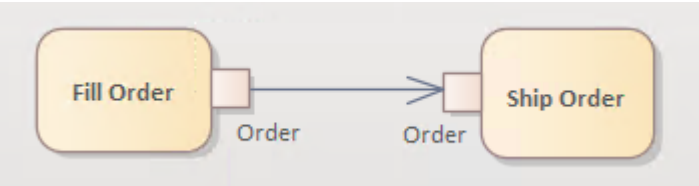
Action Pin



Description

An Action Pin is used to define the data values passed out of and into an Action. An Input Pin provides values to the Action, whereas an Output Pin contains the results from that Action.

Action Pins are used here to connect two Actions:



Action Pins can be further characterized as defining exception parameters, streams, or states. Associating a state with a Pin defines the state of input or output values. For instance, the Pin could be called 'Orders', but the state could be 'Validated' or 'Canceled'.

To add an Action Pin to an Action, right-click on the Action to display the context menu and select the 'New Child Element | Action Pin' option. (You can also assign Action Pins, to define specific properties of the Action.)

The Properties window for an Action Pin has a 'Pin' tab on which you define the specific actions of the Pin.

A Pin serves as an argument for Call Behavior Actions and Call Operation Actions - the Pin name and parameters are

shown on the 'Arguments' tab of the Properties window for the Action element. When an Action is associated with a valid behavior in the model, the associated behavior's parameters are listed in the 'Parameter' field drop-down list to facilitate one-to-one mapping between the argument and the parameter. The fields in the 'Argument' panel of the 'Pin' tab are enabled only for Pins belonging to Call Actions, and only when the Action is associated with a valid behavior with valid parameters. To observe this:

1. Create an Activity element and give it an Activity Parameter (right-click on it and select 'New Child Element | Activity Parameter').
2. Create an Action and set the 'Kind' property to 'CallBehavior' (on the 'Action' tab of the Properties window for the Action element).
3. Make the Activity element the classifier for the Action (on the Properties window for the Action, click on the 'Element' tab and, in the 'Advanced' section click on the 'Classifier' browse button and locate and select the Activity on the 'Select <Item>' dialog).
4. The Features window immediately displays at the 'Interaction Points' tab. Select the 'Show Owned/Inherited' checkbox; when this is selected, the Activity Parameter is listed in the 'Defined Elements' panel. Select the checkbox against the Activity Parameter.
5. The Action element now has an Action Pin representing an argument, with the Activity Parameter as the parameter of the argument.

You can also change the objectState property of an Action

Pin on the 'Pin' tab of the element's Properties window.

Assign Action Pins

Apart from adding Action Pins to any Action, you can assign specialized input or output Action Pins to Actions that have a specific type (that is, those that are not Basic or Atomic Actions). These input/output Pins signify various properties of the Action - they are not visible as structures on the diagram unless they have previously been added, but are listed in the Browser window as properties of the Action.

You can only assign Pins that have already been added or assigned to the Action, or that are being created specifically to be assigned to the Action.

Assign Action Pins to an Action

Step	Action
1	Click on the Action in the diagram and, in the docked Properties window, click on the 'Action' tab. Click on the drop-down arrow in the 'Kind' field and ensure that you have the correct Action type, then click on the Save icon.
2	Click on the 'Element' tab, and select the stereotype properties group. The group contains different fields depending on the Action type. The fields are

	populated by typing in or browsing for the appropriate object name or selecting a checkbox. If you use a Browser screen, you can either browse for and assign existing objects - in this case, ActionPins - or click on the Add New button and create and assign a new Action Pin.
3	Note that the Action Pins do not display on the diagram, but are shown in the Browser window under the Action element. Click on the OK button to return to the Properties window.

Notes

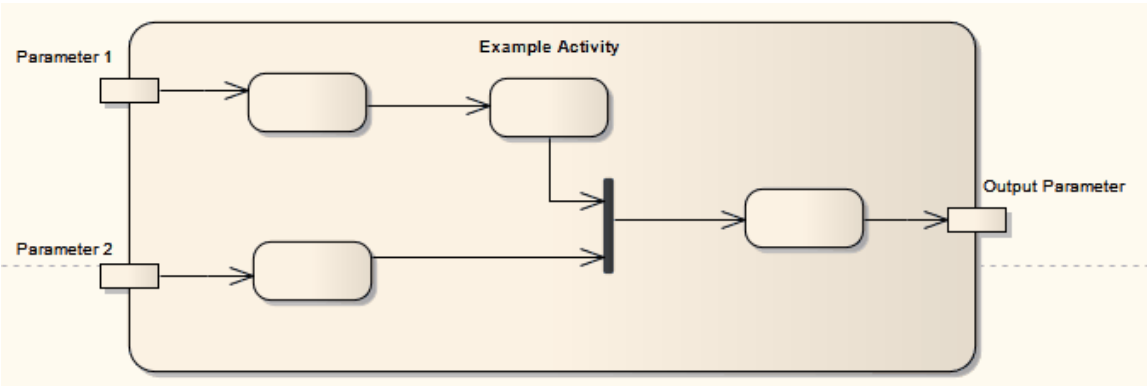
- To check the exact location of an assigned Action Pin, you can right-click on the Pin name in the Properties window and select the 'Find in Project Browser' option

Activity

Description

An Activity organizes and specifies the participation of subordinate behaviors, such as sub-Activities or Actions, to reflect the control and data flow of a process. Activities are used in Activity diagrams for various modeling purposes, from procedural-type application development for system design, to business process modeling of organizational structures or workflow.

This simple diagram of an Activity contains Action elements and includes input parameters and output parameters.



You can define an Activity as a composite element, either during creation or during later edits. When creating a composite Activity element, it is simpler to apply the mechanism for creating Structured Activity elements, which

reduces the number of steps to work through. If converting an existing Activity element, right-click on the element and select the 'New Child Diagram | Composite Structure Diagram' option.

Certain properties can be graphically depicted on an Activity. The Actions in an Activity can be further organized by Activity Partitions.

An Activity can also be depicted as an Expansion Node to indicate that the Activity consists of an Expansion Region.

If you have defined a Decision Table for the Activity element, you can select options on the element's context menu to render the element on a diagram as the Decision Table, showing the rules as either rows or columns. You can also return the element to its normal element shape.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.373-374) states:

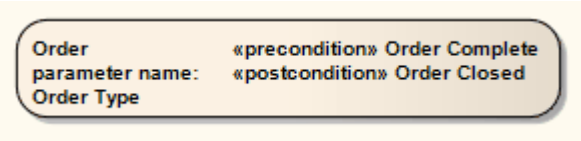
An Activity is a Behavior specified as sequencing of subordinate units, using a control and data flow model. Subordinate behaviors coordinated by these models may be initiated because other behaviors in the model finish executing, because objects and data become available or

because events occur externally to the flow. The flow of execution is modeled as ActivityNodes connected by ActivityEdges. An ExecutableNode can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents (...). ActivityNodes also include flow-of-control constructs, such as synchronization, decision, and concurrency control.

Tokens are not explicitly modeled in an Activity, but are used for describing the execution of an Activity. An object token is a container for a value that flows over ObjectFlow edges (some object tokens can flow over ControlFlow edges, as specified by the modeler, see isControlType for ObjectNodes in sub clause 15.4). An object token with no value in it is called a null token. A control token affects execution of ActivityNodes, but does not carry any data, and flows only over ControlFlow edges. Each token is distinct from any other, even if it contains the same value as another.

Activity Notation

Certain properties can be graphically depicted on an Activity element, as shown:



To define these properties, right-click on the Activity and select the 'Properties' option, then select the 'Advanced' tab of the 'Properties' dialog.

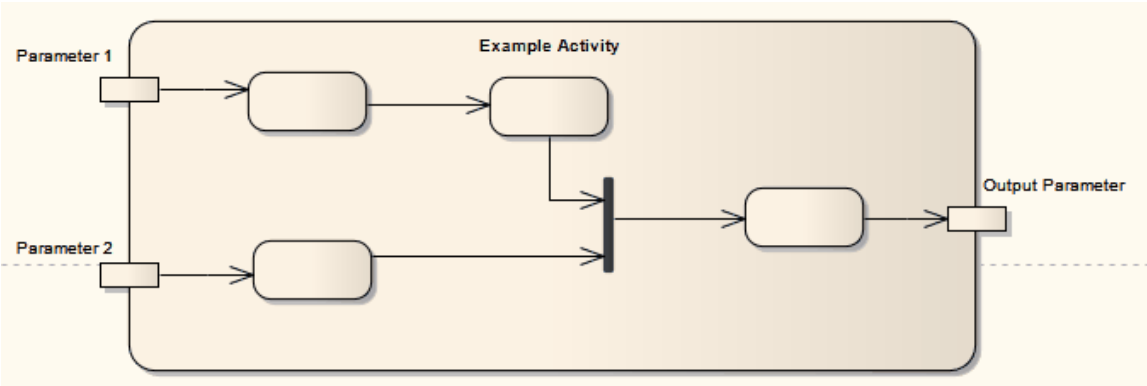
You can also define the duration (the number of ticks to wait for) of the Activity, using a JavaScript expression. Open the Properties window, click on the 'Behavior' tab and type the JavaScript expression in the 'Specification' field.

Activity Parameter Nodes

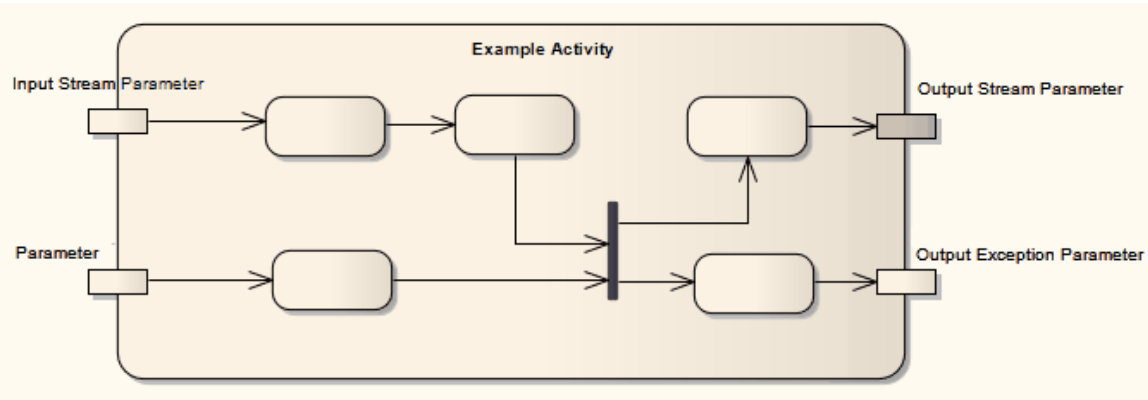
Description

An Activity Parameter Node accepts input to an Activity or provides output from an Activity.

This example depicts two entry parameters and one output parameter defined for the Activity.



Define an Activity Parameter Node for an Activity



Step	Action

1	Right-click on the element and select the 'New Element | Activity Parameter' option.
2	The 'Properties' dialog displays, which prompts for the name and other properties of the embedded element.
3	To further define the new Activity Parameter, select the 'Parameter' tab of the 'Properties' dialog and define: • Type • Default Value • Direction • Whether this is a fixed value • Multiplicity upper and lower bounds • Whether to allow duplicates and • Whether multiplicity is ordered Activity Parameter Nodes also have the 'Exception' and 'Stream' options: • Exception indicates that a parameter can emit a value at the exclusion of other outputs, usually because of some error • Stream indicates whether or not a parameter can accept or post values during the execution of the Activity

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.398) states:

As a kind of Behavior, an Activity may have Parameters (..). When the Activity is invoked, values may be passed into the Activity execution on input Parameters (i.e., those with direction in or inout) and values may be passed out of the Activity execution on output Parameters (i.e., those with direction inout, out or return).

Within an Activity, inputs to and outputs from an Activity are handled using ActivityParameterNodes. Each ActivityParameterNode is associated with one Parameter of the Activity that owns the node. The type of an ActivityParameterNode shall be the same as the type of its associated Parameter.

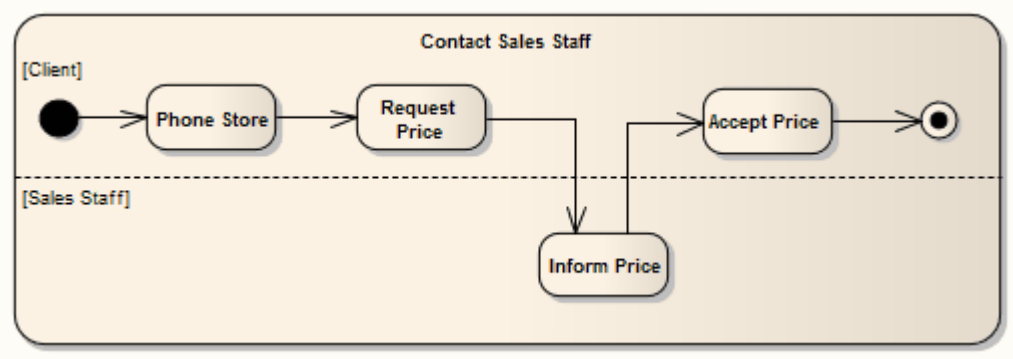
Activity Partition

Enterprise Architect supports two types of Activity Partition:

- The Activity Partition feature, described in this topic, which is used to logically organize an Activity element
- The Activity Partition element, which is used to logically organize an Activity diagram

In effect, these are the same. They partition the Actions of the Activity without affecting the token flow, helping to structure the view or parts of the Activity.

An example of a feature-partitioned Activity is shown here:



Define Partitions

Step
In a diagram, right-click on the Activity element and select the 'Advanced | Partition Activity' option. The 'Activity Partitions' dialog displays.

In the 'Name' field, type the name of a partition. Click on the Save button.
Repeat step 2 for each partition to be created.
Click on the Close button. If the partitions do not show on the element, click on the icon to the right of the element, to toggle display of the partitions.
Click on the partition borders and drag them into position to enclose the appropriate elements.

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.408) states:

An ActivityPartition is notated with two, usually parallel lines, either horizontal or vertical, and a name labeling the partition in a box at one end. Any ActivityNodes and ActivityEdges placed between these lines are considered to be contained within the partition. This notation for an ActivityPartition is colloquially known as a swimlane, (...).

Actor

Note that on a Construction diagram, an Actor element defaults to Rectangular Notation and looks like a Class element.

Description

An Actor is a user of the system; user can mean a human user, a machine, or even another system or subsystem in the model. Anything that interacts with the system from the outside or system boundary is termed an Actor. Actors are typically associated with Use Cases.

Actors can use the system through a graphical user interface, through a batch interface or through some other media. An Actor's interaction with a Use Case is documented in a Use Case scenario, which details the functions a system must provide to satisfy the user requirements.

Actors also represent the role of a user in Sequence diagrams, where you can display them using rectangle notation. Enterprise Architect supports a stereotyped Actor element for business modeling. The business modeling elements also represent Actors as stereotyped Objects.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.640/647) states:

An *Actor* specifies a role played by a user or any other system that interacts with the subject

An Actor models a type of role played by an entity that interacts with the subjects of its associated UseCases (e.g., by exchanging signals and data). Actors may represent roles played by human users, external hardware, or other systems.

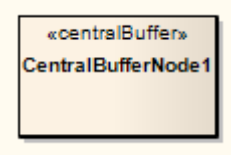
NOTE. An Actor does not necessarily represent a specific physical entity but instead a particular role of some entity that is relevant to the specification of its associated UseCases. Thus, a single physical instance may play the role of several different Actors and, conversely, a given Actor may be played by multiple different instances.

NOTE. The term “role” is used informally here and does not imply any technical definition of that term found elsewhere in this specification.

When an Actor has an association to a UseCase with a multiplicity that is greater than one at the UseCase end, it means that a given Actor can be involved in multiple UseCases of that type. The specific nature of this multiple involvement depends on the case on hand and is not defined in this specification. Thus, an Actor may initiate multiple UseCases in parallel (concurrently) or at different points in

time.

Central Buffer Node



Description

A Central Buffer Node is an object node for managing flows from multiple sources and destinations, represented in an Activity diagram. It acts as a buffer for multiple in-flows and out-flows from other object nodes, but does not connect directly to Actions.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.398) states:

A *CentralBufferNode* acts as a buffer between incoming ObjectFlows and outgoing ObjectFlows. It accepts all object tokens offered to it on all incoming flows, which are then held by the node. Held object tokens are offered to outgoing flows according to the general ordering rules for ObjectNodes. When an offer for a token is accepted by a downstream object node, that token is removed from the

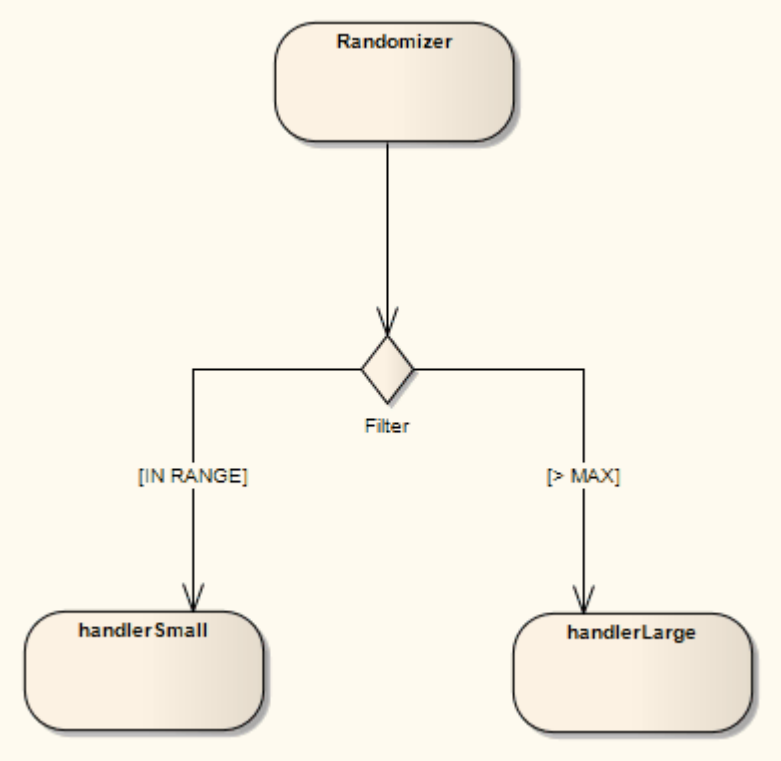
CentralBufferNode and moved to the accepting object node, as for any object node.

Choice

Description

The Choice pseudostate is used to compose complex transitional paths in, for example, a StateMachine diagram, where the outgoing transition path is decided by dynamic, run-time conditions. The run-time conditions are determined by the actions performed by the StateMachine on the path leading to the choice.

This example depicts the Choice element. Upon reaching the Filter pseudostate, a transition fires to the appropriate State based on the run-time value passed to the Filter. Very similar in form to a Junction pseudostate, the Choice pseudostate's distinction is in deciding transition paths at run-time.



Toolbox icon

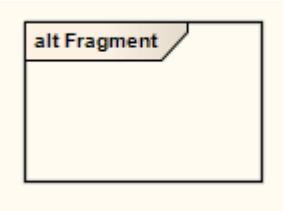
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

This type of Pseudostate is similar to a junction Pseudostate (...) and serves similar purposes, with the difference that the guard Constraints on all outgoing Transitions are evaluated dynamically, when the compound transition traversal reaches this Pseudostate. Consequently, choice is used to realize a dynamic conditional branch. It allows splitting of compound transitions into multiple alternative paths such that the decision on which path to take may depend on the

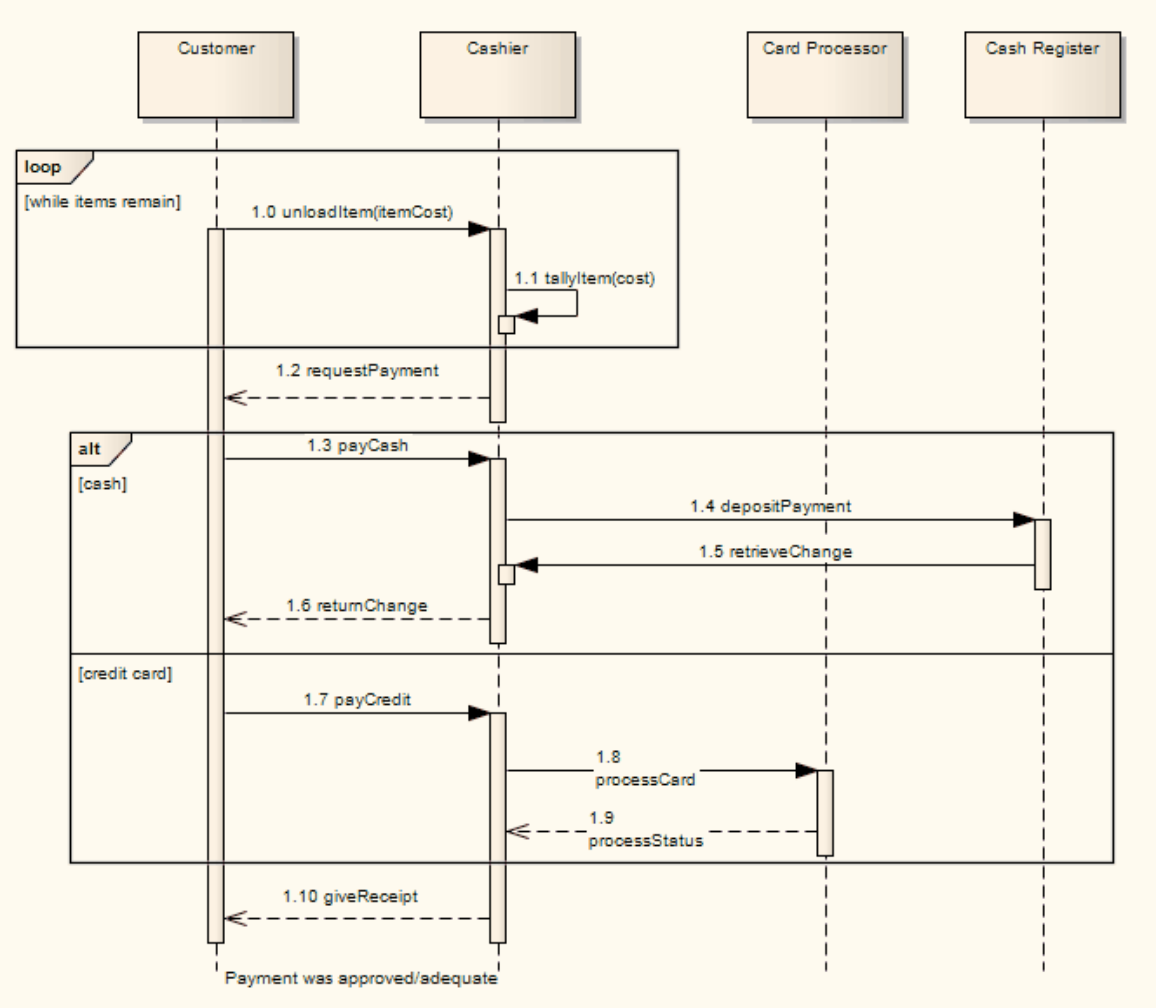
results of Behavior executions performed in the same compound transition prior to reaching the choice point. If more than one guard evaluates to true, one of the corresponding Transitions is selected. The algorithm for making this selection is not defined. If none of the guards evaluates to true, then the model is considered ill formed. To avoid this, it is recommended to define one outgoing Transition with the predefined “else” guard for every choice Pseudostate.

Combined Fragment



A Combined Fragment reflects one or more aspects of interaction (called interaction operands) controlled by an interaction operator, with corresponding Boolean conditions known as interaction constraints. The Fragment displays as a transparent window, divided by horizontal lines for each operand.

This Sequence diagram illustrates the use of Combined Fragments in modeling a simplified purchasing process. A loop fragment represents iteration through an unknown number of items for purchase, after which the cashier requests payment. An alternative fragment represents the payment options, the fragment being divided to show the two operands cash and credit card. After the fragment completes its trace, the cashier gives a receipt to the customer, under the fulfilled condition that payment requirements were met.



The order of interaction fragment conditions can be changed directly on the diagram:

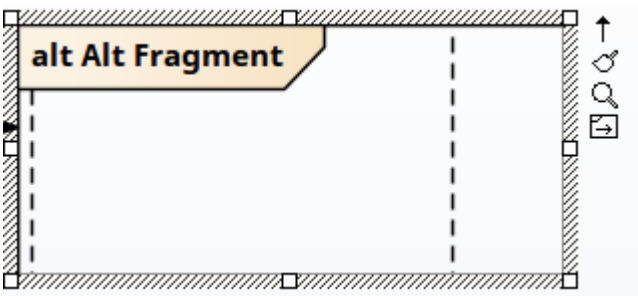
1. Select an interaction fragment with more than one condition defined; up and down arrows appear on the right hand side of each condition.
2. Click on the appropriate arrow to change the order.

Selecting and Moving a Combined Fragment

In order to select a Combined Fragment, you must click near the inside edge or drag a selection rectangle around the Fragment; this is designed to prevent accidental selection when moving Messages inside the Fragment.

Once contained within a Fragment or a Fragment Operand, Messages continue to be contained by it as they are moved up and down the diagram. To move a Message out of a Fragment, or to a different position in the sequence within the Fragment, press and hold the Alt key as you drag the Message into position. A Fragment on a Sequence diagram will resize when a Message within it is moved up or down, to continue to contain that Message.

When you select an Interaction Fragment on a diagram, it shows one of two element icons (off the top right corner) that control how freely you can move the fragment and any Messages within and below the fragment.



To move a Combined Fragment *independently* of its

contents, make sure the 'move freely' element icon is visible; if it is not shown, click on the 'move contents' icon and drag the element border.

Interaction Fragments inside a Combined Fragment operand cannot be moved outside the operand unless the fragment is in 'move freely' mode. Moving an operand line moves any objects and Messages below that line down or up by the amount the operand line is moved.

Fragments containing other fragments resize when the contained fragment is resized (unless the fragment is in 'move freely' mode).

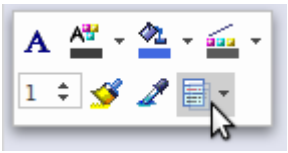
Fill Opacity

Whilst an Interaction Fragment usually encloses a number of other elements, there might be reasons for hiding those elements as well as times to fully show them, or perhaps just indicate that they are there, depending on the immediate purpose of the diagram. You can apply these nuances in the display of elements behind and covered or overlapped by an Interaction Fragment, by changing the opacity of the element.

Before setting the opacity, check that the element has a fill color.

You set the opacity using an icon from either of these two pop-up element toolbars:

- Click on the Interaction Fragment element and on the icon:

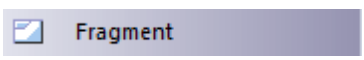


- Right-click on the Interaction Fragment element and look above the context menu:

Click on the icon and select:

- 100% for total opacity, where the elements behind and overlapping or covered by the Interaction Fragment are hidden (you could right-click on individual elements and select the 'Z-Order | Bring to Top' option to expose those elements only)
- 0% for no opacity, where the fill color is not applied and anything behind the Interaction Fragment is fully visible
- 75%, 50% or 25% to set the appropriate degree of opacity and make the covered elements visible but over-shaded

Toolbox icon



OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, p.607) states:

A CombinedFragment defines an expression of InteractionFragments. A CombinedFragment is defined by an interaction operator and corresponding

InteractionOperands. Through the use of CombinedFragments the user will be able to describe a number of traces in a compact and concise manner.

Create a Combined Fragment

Create a Combined Fragment

Step	Action
1	Drag the 'Fragment' icon onto the diagram from the 'Interaction Elements' page of the Diagram Toolbox.
2	In the 'Type' field, click on the drop-down arrow and select one of the various types of interaction operator.
3	In the 'Condition' field, specify a condition or interaction constraint for each operand.
4	A rectangular frame displays, partitioned by lines into segments for each operand.
5	Adjust the frame to encompass the required event occurrences for each operand.

Notes

- A message will always be contained within a fragment or

a fragment operand when it is moved within it

- Fragments on Sequence diagrams will resize when a message is moved down to ensure that messages, once within a fragment, always remain within the fragment

Interaction Operators

When creating Combined Fragments, you must apply an appropriate interaction operator to characterize the fragment. This table provides guidance on the various operators, and their corresponding descriptions.

Interaction Operator

Operator	Action
alt	Divide up interaction fragments based on Boolean conditions.
opt	Enclose an optional fragment of interaction.
par	Indicate that operands operate in parallel.
loop	Indicate that the operand repeats a number of times, as specified by interaction constraints.
critical	Indicate a sequence that cannot be interrupted by other processing.
neg	Assert that a fragment is invalid, and implies that all other interaction is valid.

assert	Specify the only valid fragment to occur. This operator is often enclosed within a consider or ignore operand.
strict	Indicate that the behaviors of the operands must be processed in strict sequence.
seq	Indicate that the Combined Fragment is weakly sequenced. This means that the ordering within operands is maintained, but the ordering between operands is undefined, so long as an event occurrence of the first operand precedes that of the second operand, if the event occurrences are on the same lifeline.
ignore	Indicate which messages should be ignored during execution, or can appear anywhere in the execution trace.
consider	Specify which messages should be considered in the trace. This is often used to specify the resulting event occurrences with the use of an assert operator.
ref	Provide a reference to another diagram. The ref fragment is not created using the

	method described in the *Create a Combined Fragment* topic. To create a ref fragment, simply drag an existing diagram from the Browser window onto the current diagram.

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.583-585) states:

The value of the *interactionOperator* is significant for the semantics of CombinedFragment, as specified below for each interactionOperator enumeration value.

Alternatives

The interactionOperator alt designates that the CombinedFragment represents a choice of behavior. At most one of the operands will be chosen. The chosen operand must have an explicit or implicit guard expression that evaluates to true at this point in the interaction. An implicit true guard is implied if the operand has no guard. The set of traces that defines a choice is the union of the (guarded) traces of the operands. An operand guarded by else designates a guard that is the negation of the disjunction of all other guards in the enclosing CombinedFragment. If none of the operands has a guard that evaluates to true, none of the operands are executed and the remainder of the enclosing InteractionFragment is executed. If an inner CombinedFragment Gate is used in any InteractionOperand

of an alt CombinedFragment, a Gate with that same name must be used by every InteractionOperand of that alt CombinedFragment.

Option

The interactionOperator opt designates that the CombinedFragment represents a choice of behavior where either the (sole) operand happens or nothing happens. An option is semantically equivalent to an alternative CombinedFragment where there is one operand with non-empty content and the second operand is empty.

Break

The interactionOperator break designates that the CombinedFragment represents a breaking scenario in the sense that the operand is a scenario that is performed instead of the remainder of the enclosing InteractionFragment. A break operator with a guard is chosen when the guard is true and the rest of the enclosing Interaction Fragment is ignored. When the guard of the break operand is false, the break operand is ignored and the rest of the enclosing InteractionFragment is chosen. The choice between a break operand without a guard and the rest of the enclosing InteractionFragment is done non-deterministically. A CombinedFragment with interactionOperator break should cover all Lifelines of the enclosing InteractionFragment.

Parallel

The interactionOperator par designates that the CombinedFragment represents a parallel merge between the behaviors of the operands. The OccurrenceSpecifications of the different operands can be interleaved in any way as long

as the ordering imposed by each operand as such is preserved. A parallel merge defines a set of traces that describes all the ways that OccurrenceSpecifications of the operands may be interleaved without obstructing the order of the OccurrenceSpecifications within the operand.

Weak Sequencing

The interactionOperator seq designates that the CombinedFragment represents a weak sequencing between the behaviors of the operands. Weak sequencing is defined by the set of traces with these properties:

1 The ordering of OccurrenceSpecifications within each of the operands are maintained in the result.

2 OccurrenceSpecifications on different lifelines from different operands may come in any order.

3 OccurrenceSpecifications on the same lifeline from different operands are ordered such that an OccurrenceSpecification of the first operand comes before that of the second operand.

Thus weak sequencing reduces to a parallel merge when the operands are on disjunct sets of participants. Weak sequencing reduces to strict sequencing when the operands work on only one participant.

Strict Sequencing

The interactionOperator strict designates that the CombinedFragment represents a strict sequencing between the behaviors of the operands. The semantics of strict

sequencing defines a strict ordering of the operands on the first level within the CombinedFragment with interactionOperator strict. Therefore OccurrenceSpecifications within contained CombinedFragment will not directly be compared with other OccurrenceSpecifications of the enclosing CombinedFragment.

Negative

The interactionOperator neg designates that the CombinedFragment represents traces that are defined to be invalid. The set of traces that defined a CombinedFragment with interactionOperator negative is equal to the set of traces given by its (sole) operand, only that this set is a set of invalid rather than valid traces. All InteractionFragments that are different from Negative are considered positive meaning that they describe traces that are valid and should be possible.

Critical Region

The interactionOperator critical designates that the CombinedFragment represents a critical region. A critical region means that the traces of the region cannot be interleaved by other OccurrenceSpecifications (on those Lifelines covered by the region). This means that the region is treated atomically by the enclosing fragment when determining the set of valid traces. Even though enclosing CombinedFragments may imply that some OccurrenceSpecifications may interleave into the region, such as with par-operator, this is prevented by defining a region. Thus the set of traces of enclosing constructs are

restricted by critical regions.

Ignore / Consider

The interactionOperator ignore designates that there are some message types that are not shown within this combined fragment. These message types can be considered insignificant and are implicitly ignored if they appear in a corresponding execution. Alternatively, one can understand ignore to mean that the message types that are ignored can appear anywhere in the traces. Conversely, the interactionOperator consider designates which messages should be considered within this combined fragment. This is equivalent to defining every other message to be ignored.

Assertion

The interactionOperator assert designates that the CombinedFragment represents an assertion. The sequences of the operand of the assertion are the only valid continuations. All other continuations result in an invalid trace. Assertions are often combined with Ignore or Consider as shown in Figure 17.17.

Loop

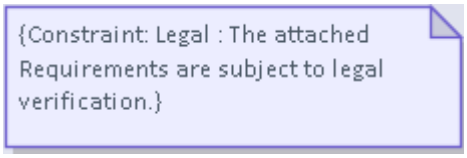
The interactionOperator loop designates that the CombinedFragment represents a loop. The loop operand will be repeated a number of times.

The Guard may include a lower and an upper number of iterations of the loop as well as a Boolean expression. The semantics is such that a loop will iterate minimum the ‘minint’ number of times (given by the iteration expression in the guard) and at most the ‘maxint’ number of times.

After the minimum number of iterations have executed and the Boolean expression is false the loop will terminate. The loop construct represents a recursive application of the seq operator where the loop operand is sequenced after the result of earlier iterations.

If the loop contains a separate InteractionConstraint with a specification, the loop will only continue if that specification evaluates to true during execution regardless of the minimum number of iterations specified in the loop.

Constraint



Description

A Constraint element identifies a constraint on other elements; it can be connected to other elements of any type. The Constraint element icon is available in any Enterprise Architect diagram, through the 'Common' pages of the Toolbox.

This element is a means of documenting the fact that there are constraints on related elements; it has no impact on the other elements. However, a Constraint is a named element, listed in the Browser window, and able to be copied and re-used where appropriate. You define the types of constraint in the project reference data, apply them to the element in the element 'Properties' dialog, and manage them through the Responsibility window.

Toolbox icon

Datastore

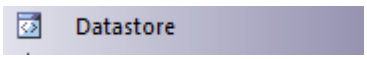
Description

A Datastore is an element used to define permanently stored data. A token of data that enters into a Datastore is stored permanently, updating tokens for data that already exists. A token of data that comes out of a Datastore is a copy of the original data.

Use Object Flow connectors to connect elements (such as Activities) to Datastores, as values and information are being passed between nodes. Selection and transformation behavior, together composing a sort of query, can be specified as to the nature of data access. For instance, selection behavior determines which objects are affected by the connection to the Datastore. Transformation behavior might then further specify the value of an attribute pertaining to a selected object.

To define the behavior of access to a Datastore, attach a note to the Object Flow connector. To do this, right-click on the Object Flow and select the 'Attach Note or Constraint' option. A dialog indicates other flows in the Activity diagram, to which you can attach the note (if the behavior applies to multiple flows). To comply with UML 2.x, preface the behavior with the notation «selection» or «transformation».

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.399) states:

A *DataStoreNode* is a CentralBufferNode that holds its object tokens persistently while its activity is executing. When an offer for an object token held by a DataStoreNode is accepted by a downstream object node, the offered token is removed from the DataStoreNode, per the usual CentralBufferNode semantics. However, a copy is made of the removed object token, with the same value, and this is immediately placed back onto the DataStoreNode. Thus, the values held by a DataStoreNode appear to persist for the duration of each execution of its containing activity, even as tokens move downstream from the node. When a DataStoreNode accepts an object token, if that token contains an object with the same identity as an object contained in a token already held by the node, then the duplicate object token shall not be placed on the DataStoreNode. Unlike a regular CentralBufferNode, a DataStoreNode contains objects uniquely.

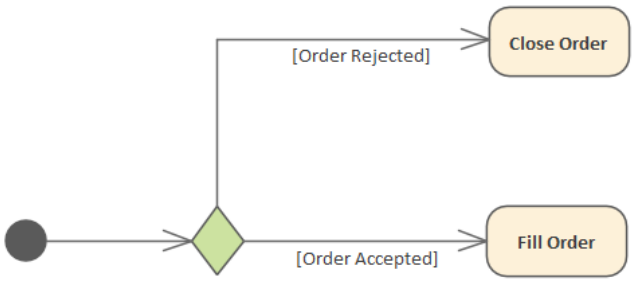
Decision

Description

A Decision is an element of an Activity diagram or Interaction Overview diagram that indicates a point of conditional progression: if a condition is True, then processing continues one way; if not, then another.

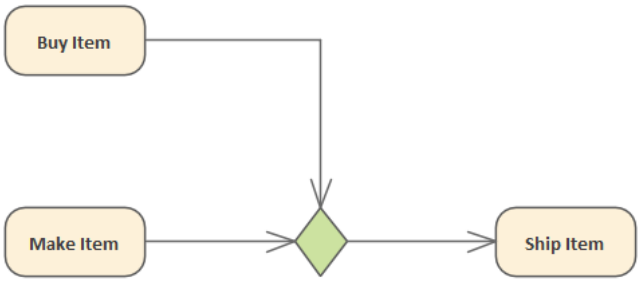
The element can also be used as a Merge node in that multiple alternative flows can be merged (but not synchronized) to form one flow. These examples show both of these methods of using the Decision element.

Used as a decision:



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.77, p. 363.)

Used as a merge:



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.106, p. 388.)

You can choose a Behavior element as the Decision Input property of the Decision (UML: decisionInput) in the Properties window. To show the chosen Decision Input property on a diagram, attach a Note to the Decision, then right-click on the Note Link and choose the 'Link this Note to an Element feature' option. Then select 'Decision Input' as the linked feature.

You can also choose an Object Flow as the Decision Input Flow of the Decision (UML: decisionInputFlow). Select the incoming Object Flow and select the 'Decision Input Flow' option in the Properties window.

Notes

- Moving a diagram generally does not affect the location of elements in Packages; if you move a diagram out of one Package into another, all the elements in the diagram remain in the original Package

  However, Decision elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram; therefore, if you move

a diagram containing these elements, they are moved to the new parent Package with the diagram

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.390 (Decision Node)) states:

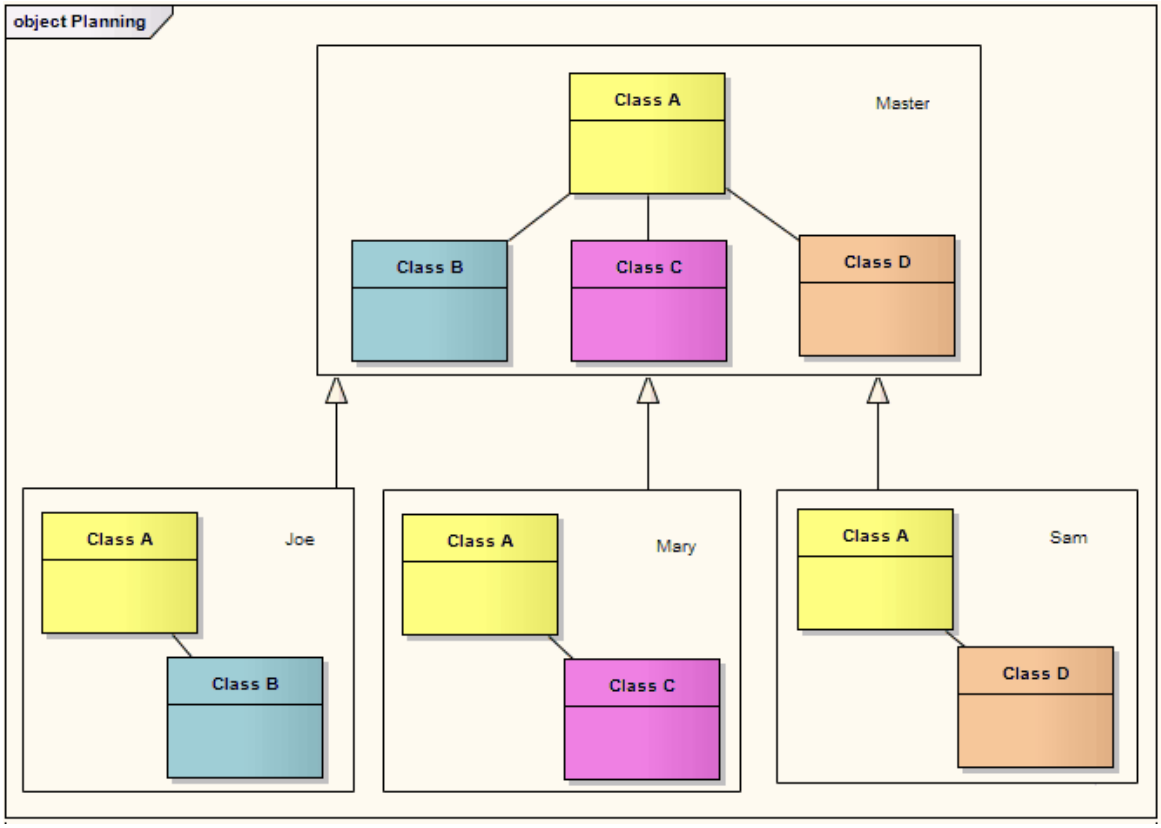
A DecisionNode is a ControlNode that chooses between outgoing flows. A DecisionNode shall have at least one and at most two incoming ActivityEdges, and at least one outgoing ActivityEdge. If it has two incoming edges, then one shall be identified as the decisionInputFlow, the other being called the primary incoming edge. If the DecisionNode has only one incoming edge, then it is the primary incoming edge. If the primary incoming edge of a DecisionNode is a ControlFlow, then all outgoing edges shall be ControlFlows and, if the primary incoming edge is an ObjectFlow, then all outgoing edges shall be ObjectFlows.

A DecisionNode accepts tokens on its primary incoming edge and offers them to all its outgoing edges. However, each token offered on the primary incoming edge shall traverse at most one outgoing edge. Tokens are not duplicated.

If any of the outgoing edges of a DecisionNode have guards,

then these are evaluated for each incoming token. The order in which guards are evaluated is not defined and may be evaluated concurrently. If the primary incoming edge of a DecisionNode is an ObjectFlow, and the DecisionNode does not have a decisionInput or decisionInputFlow, then the value contained in an incoming object token may be used in the evaluation of the guards on outgoing ObjectFlows.

Diagram Frame



A Diagram Frame element is a rendition of a diagram dropped from the Browser window into another diagram. It is a type of Combined Fragment with an 'Interaction Operator' ref. However, it can be created on any type of diagram, and is not created in the same way as other Combined Fragments.

When you drop the diagram from the Browser window onto the open diagram, a dialog shows providing (amongst others) these options:

- 'Diagram Frame' - a Diagram Frame is inserted into the diagram, containing an image of the dropped diagram
- 'Diagram Reference' - an empty frame is inserted with the name of the dropped diagram in the frame label

In both cases, the object acts as a hyperlink to the real referenced diagram. You can also define properties for the objects, as for other elements, by right-clicking on the object and selecting the element 'Properties' option.

All options on the 'Select Type' dialog are discussed in the *Add Diagram Links to Diagrams* Help topic.

Diagram Frame Appearance

You can change the appearance of a Diagram Frame, as for other elements, but the available options are tailored for this element type. If you right-click on the frame and select the 'Appearance | Diagram Frame Appearance' option, a sub-menu displays with these options:

- 'Normal' - the default appearance of a visible rectangular frame with a visible frame label; you can use this option to reset the appearance after using one of the other options
- 'Boundary' - hides the frame label of the Diagram Frame
- 'Boundary With Name' - hides the border of the frame label
- 'Name Only' - hides the border of the Diagram Frame and frame label, leaving the text only
- 'Hidden' - hides the border and text of the Diagram Frame

In a SysML, State or StateMachine diagram:

- If the frame is set to non-selectable it will auto-resize to fit the bounds of the diagram, expanding from its default size but not shrinking smaller
- Diagrams showing Diagram Frames applied using

Enterprise Architect release 14.0 or later will, if opened using a release of Enterprise Architect *earlier* than release 14.0, draw the parent object on the diagram

Moving Elements via Diagram Frames

A useful feature of Diagram Frames *as a diagram reference* is that they provide the facility of moving elements currently displayed on the Host diagram through the frame to the referenced diagram, the parent Package of the referenced diagram, or both. You can also add the element to the referenced diagram as a link to its current location.

To move or link the element, simply drag it on the current diagram over the Diagram Frame. A dialog displays, listing options to:

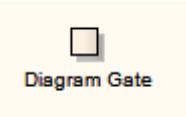
- Move the element to the referenced diagram
- Create a link to the element on the referenced diagram
- Move the element to the parent Package
- Move the element to both the referenced diagram and its parent Package

If you select one of the options to move the element to the referenced diagram, it and any connectors it has are removed from the current diagram and placed in a clear area of the referred diagram. If the element already has relationships with other elements on the diagram, those relationships are included.

Notes

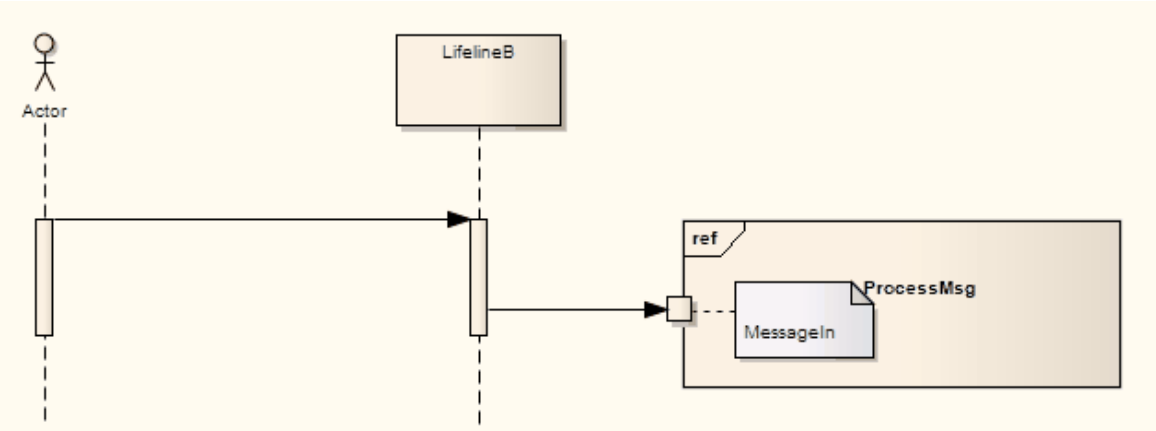
- You can change the size of both objects, but you cannot reduce a Diagram Frame to less than the size of the enclosed diagram
- You cannot change the diagram within a Diagram Frame; to edit the diagram, double-click within the frame and edit the original diagram
- The Diagram Frame element is not the same as the diagram frame border that you can set (using the 'Diagram Frames' panel on the 'Diagram' page of the 'Preferences' dialog) on images of diagrams that you print out, copy to file or paste into other tools; it is possible, but not usual, to paste the diagram image from the clipboard into another Enterprise Architect diagram, in which case the image initially looks the same as the Diagram Frame element, but element options do not function on this image

Gate



Description

A Diagram Gate is a simple graphical way to indicate the point at which messages can be transmitted into and out of Interaction Fragments. A fragment might be required to receive or deliver a message; internally; an ordered message reflects this requirement, with a Gate indicated on the boundary of the fragment's frame. Any external messages 'synching' with this internal message must correspond appropriately. Gates can appear on Interaction diagrams (Sequence, Timing, Communication or Interaction Overview), Interaction Occurrences and Combined Fragments (to specify the expression).



Toolbox icon

OMG UML Specification:

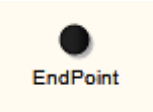
The OMG Unified Modeling Language specification, (v2.5.1, pp.575-576) states:

A Gate is a MessageEnd that is used on the boundary of an Interaction, or an InteractionUse, or a CombinedFragment to establish the concrete sender and receiver for every Message.

Gate instances, since they occur in a paired manner linking two Message instances, are also not themselves explicitly ordered. Gates are MessageEnds that provide a connection point between, either:

- a Message instance outside of an InteractionUse and a Message instance inside the used Interaction, or
- a Message instance outside a CombinedFragment and a Message instance inside an InteractionOperand within the CombinedFragment.

Endpoint

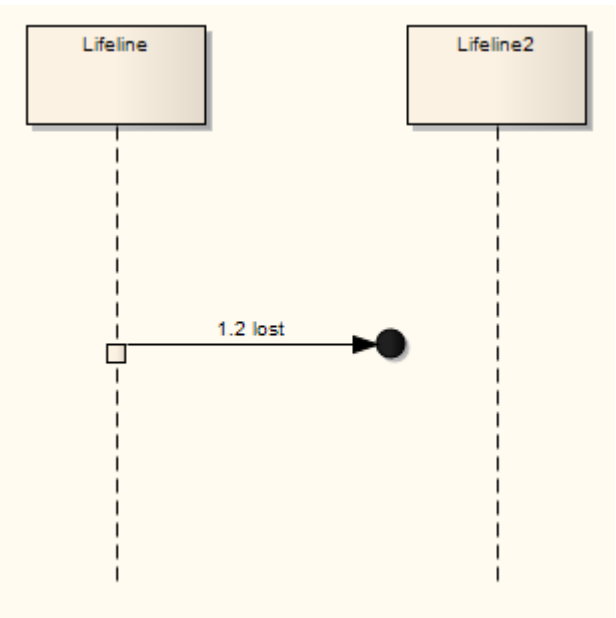


Description

An Endpoint is used in Interaction diagrams (Sequence, Timing, Communication or Interaction Overview) to reflect a lost or found Message in sequence. To model this, drag an Endpoint element onto the workspace.

With Sequence diagrams, drag a Message from the appropriate Lifeline to the Endpoint. With Timing diagrams, the Message connecting the Lifeline to the Endpoint requires some timing specifications to draw the connection.

This example depicts a lost Message in a Sequence diagram.



Toolbox icon

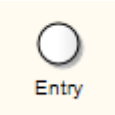
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.574) states:

A *lost Message* is a Message where the sending event occurrence is known, but there is no receiving event occurrence. We interpret this to be because the destination of the [lost]Message is outside the scope of the description.

A *found Message* is a Message where the receiving event occurrence is known, but there is no (known) sending event occurrence. We interpret this to be because the origin of the Message is outside the scope of the description. This may for example be noise or other activity that we do not want to describe in detail.

Entry Point



Description

Entry Point pseudostates are used to define the beginning of a StateMachine. An Entry Point exists for each region, directing the initial concurrent state configuration.

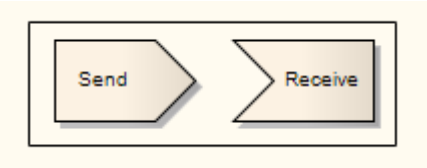
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

An *entryPoint* Pseudostate represents an entry point for a StateMachine or a composite State that provides encapsulation of the insides of the State or StateMachine. In each Region of the StateMachine or composite State owning the entryPoint, there is at most a single Transition from the entry point to a Vertex within that Region.

Event



Description

Two elements are used to model Events; the:

- Send Event which models the generation of a stimulus in the system and the passing of that stimulus to other elements, either within the system or external to the system
- Receive Event, depicted as a rectangle with a recessed 'V' on the left side, which indicates that an event occurs in the system due to some external or internal stimulus; typically this invokes further activities and processing

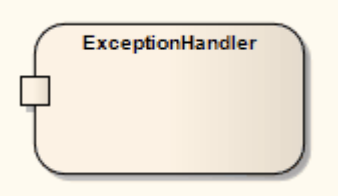
Send and Receive Events can be added from the Analysis, State and Activity Element pages of the Toolbox.

If you should select the wrong type of event, or otherwise want to change the type, right-click on the Event and select the 'Advanced | Make Sender' or 'Advanced | Make Receiver' option, as appropriate.

Toolbox icon

Receive

Exception



Description

The Exception Handler element defines the group of operations to carry out when an exception occurs. In an Activity diagram, the protected element can contain a set of operations and is connected to the exception handler via an Interrupt Flow connector. Any defined error contained within an element's parts can trigger the flow to move to an exception.

Toolbox icon

OMG UML Specification:

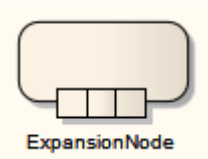
The OMG Unified Modeling Language specification, (v2.5.1, p.404) states:

An *exception* is a value used to identify a non-normal completion mode of an execution. If an exception is raised (e.g., using a RaiseExceptionAction; ...) within the execution of an ExecutableNode and not handled within that

execution, then the execution is terminated and the exception is propagated out of the ExecutableNode.

An ExecutableNode may have one or more ExceptionHandlers that are used to deal with exceptions that may be propagated out of the ExecutableNode, which is the protectedNode of those handlers. If an exception is propagated out of the protectedNode, then the set of handlers is examined for a handler that matches the exception. A handler matches if the type of the exception is the same as, or a (direct or indirect) subtype of, one of the exceptionTypes of the handler. If there is a match, the handler catches

Expansion Node

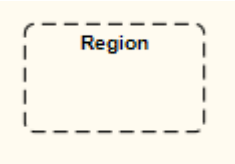


Description

Representing an Action or an Activity as an Expansion Node is a shorthand notation to indicate that the Action/Activity consists of an Expansion Region.

To specify an Action or Activity as an Expansion Node, right-click on the Action and select the 'New Child Element | Expansion Node' option.

Expansion Region



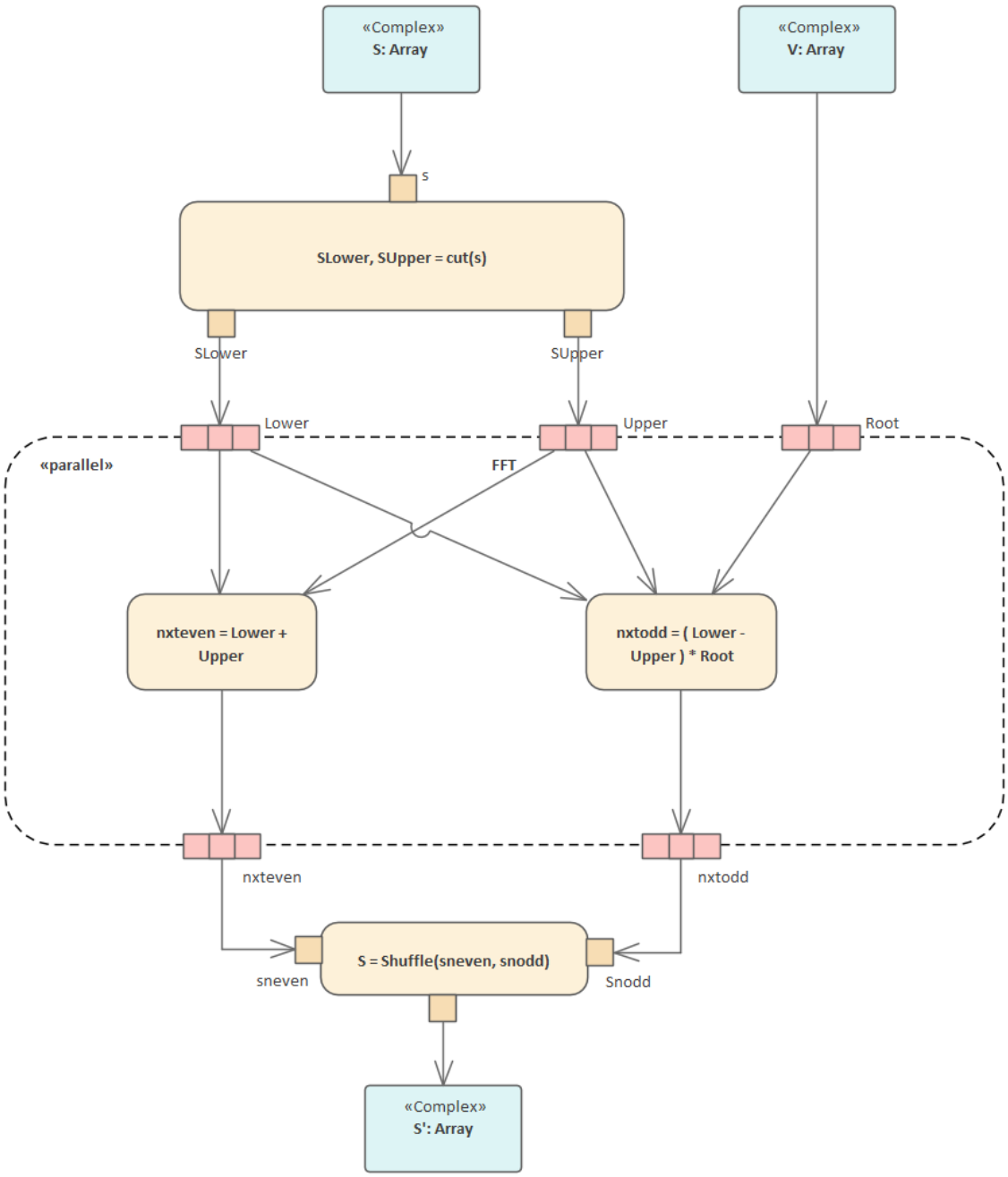
Description

On an Activity diagram, an Expansion Region encloses a group of ActivityNodes and ActivityEdges that are to be executed several times on the incoming data, once for every element in the input collection. If there are multiple inputs, the collection sizes should match; if they do not, the smallest collection determines the number of executions. The collections must also be of the same type (such as set, or bag). Any outputs must be in the form of a collection of at least the same size as the input collection; the output collection can be larger if each execution can produce more than one output.

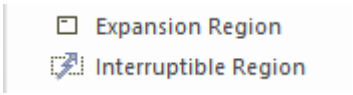
You create an Expansion Region as one variant of a Region (the other is an Interruptible Activity Region) using the Activity pages of the Diagram Toolbox. You are prompted to specify the concurrency of the Expansion Region's multiple executions (parallel, iterative or stream). Parallel reflects that the elements in the incoming collections can be processed at the same time or overlapping, whereas an iterative concurrency mode specifies that execution must occur sequentially. A stream mode Expansion Region indicates that the input and output come in and exit as streams, and that the Expansion Region's process must have

some method to support streams.

To modify the mode of an Expansion Region, right-click on it and select the 'Properties | Special Action' option, then select the 'Advanced' tab and click on the drop-down arrow in the 'mode' field..



Toolbox icon



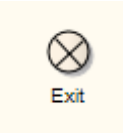
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.480-481) states:

An *ExpansionRegion* is a StructuredActivityNode that executes its contained elements multiple times corresponding to elements of an input collection.

An *ExpansionRegion* is a StructuredActivityNode that takes as input one or more collections of values and executes its contained ActivityNodes and ActivityEdges on each value in those collections. If the computation produces results, these may be collected into output collections. The number of output collections can differ from the number of input collections.

Exit Point



Description

Exit Points are used in StateMachine elements and StateMachine diagrams to denote the point where the machine is exited and the transition sourcing this exit point, for StateMachine elements, is triggered. Exit points are a type of pseudostate used in the StateMachine diagram.

Toolbox icon

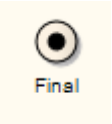
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

An *exitPoint* Pseudostate is an exit point of a StateMachine or composite State that provides encapsulation of the insides of the State or StateMachine. Transitions terminating on an exit point within any Region of the composite State or a StateMachine referenced by a submachine State implies exiting of this composite State or submachine State (with execution of its associated exit Behavior). If multiple Transitions from orthogonal Regions within the State

terminate on this Pseudostate, then it acts like a join Pseudostate.

Final

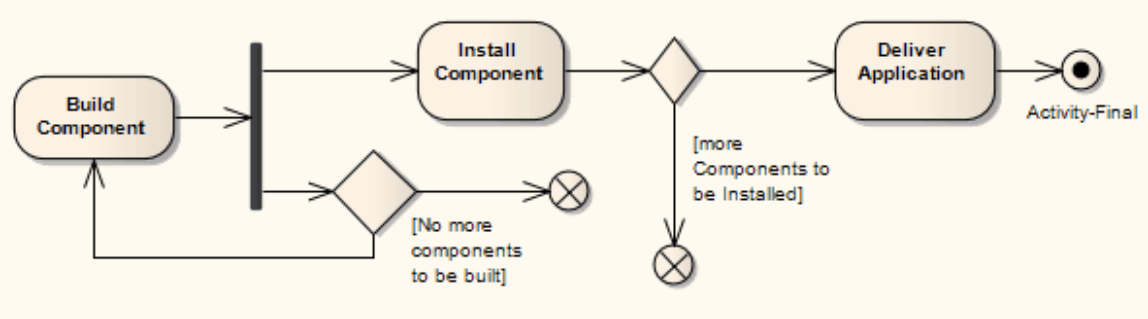


Description

Two nodes can be used to define a Final state in an Activity, both defined in UML 2.1 as of type Final Node. The Activity Final element indicates the completion of an Activity; upon reaching the Final, all execution in the Activity diagram is aborted. The other type of final node, Flow Final, depicts an exit from the system that has no effect on other executing flows in the Activity.

The next example illustrates the development of an application. The process comes to a Flow Final node when there are no more components to be built; note that the Fork element indicates a concurrent process with the building of new components and installation of completed components. The Flow Final terminates only the sub-process building components. Similarly, only those tokens entering the decision branch for the installation of further components terminate with the connecting Flow Final (that is, stop installing this component, but keep on installing other components). It is only after the Deliver Application activity is completed, after the control flow reaches the Final node, that all flows stop.

The node that initiates a flow is the Initial node.



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.91, p.374.)

Notes

- Moving a diagram generally does not affect the location of elements in Packages; if you move a diagram out of one Package into another, all the elements in the diagram remain in the original Package

  However, Final elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram; therefore, if you move a diagram containing these elements, they are moved to the new parent Package with the diagram

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification,

(v2.5.1, p.388) states:

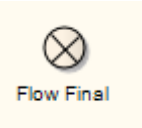
A *FinalNode* is a ControlNode at which a flow in an Activity stops. A FinalNode shall not have outgoing ActivityEdges. A FinalNode accepts all tokens offered to it on its incoming ActivityEdges.

There are two kinds of FinalNode:

1 A FlowFinalNode is a FinalNode that terminates a flow. All tokens accepted by a FlowFinalNode are destroyed. This has no effect on other flows in the Activity.

2 An ActivityFinalNode is a FinalNode that stops all flows in an Activity (...). A token reaching an ActivityFinalNode owned by an Activity terminates the execution of that Activity. If an Activity owns more than one ActivityFinalNode, then the first one to accept a token (if any) terminates the execution of the Activity, including the execution of any other ActivityFinalNodes.

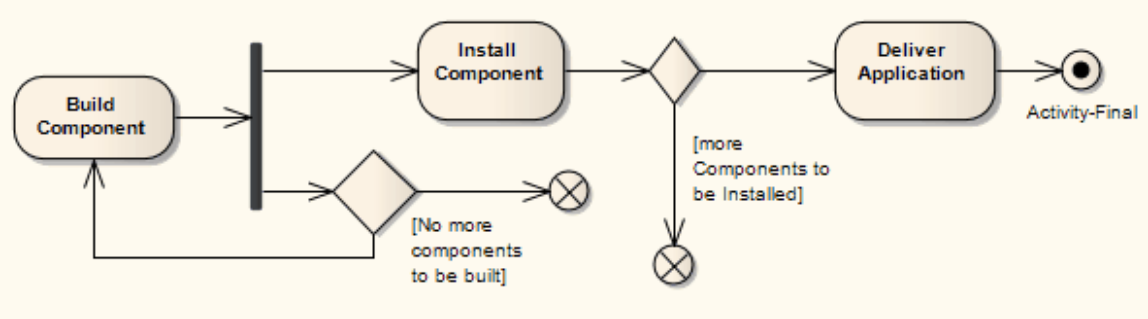
Flow Final



Description

There are two nodes used to define a final state in an Activity, both defined in UML 2.1 as of type Final Node. The Flow Final element depicts an exit from the system, as opposed to the Activity Final, which represents the completion of the Activity. Only the flow entering the Flow Final node exits the Activity; other flows continue undisturbed.

This example Activity diagram illustrates the development of an application. The process comes to a Flow Final node when there are no more components to be built; note that the Fork element indicates a concurrent process with the building of new components and installation of completed components. The Flow Final terminates only the sub-process building components. Similarly, only those tokens entering the decision branch for the installation of further components terminate with the connecting Flow Final (that is, stop installing this component, but keep on installing other components). It is only after the Deliver Application activity is completed, after the control flow reaches the Final node, that all flows stop.



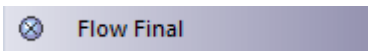
See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.91, p.374.)

Notes

- Moving a diagram generally does not affect the location of elements in Packages: if you move a diagram out of one Package into another, all the elements in the diagram remain in the original Package

  However, Flow Final elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram; therefore, if you move a diagram containing these elements, they are moved to the new parent Package with the diagram

Toolbox icon



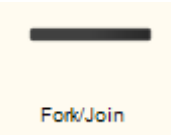
OMG UML Specification:

The OMG Unified Modeling Language specification,

(v2.5.1, p.388) states:

A *FlowFinalNode* is a FinalNode that terminates a flow. All tokens accepted by a FlowFinalNode are destroyed. This has no effect on other flows in the Activity.

Fork/Join



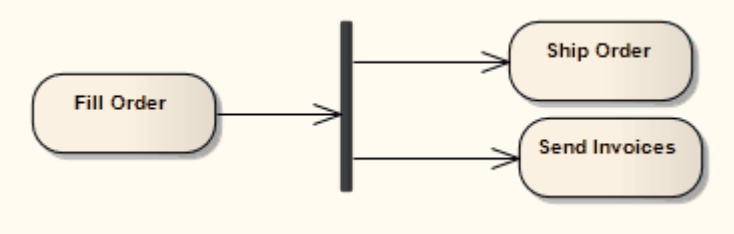
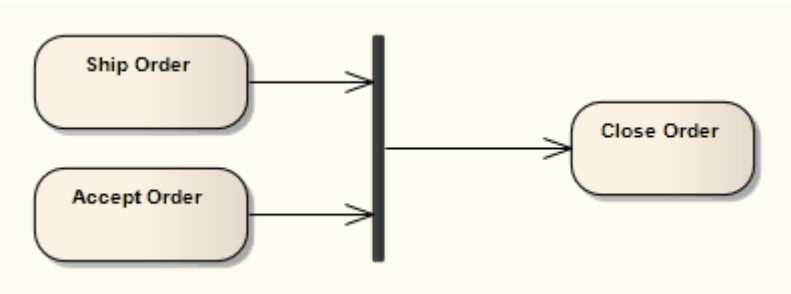
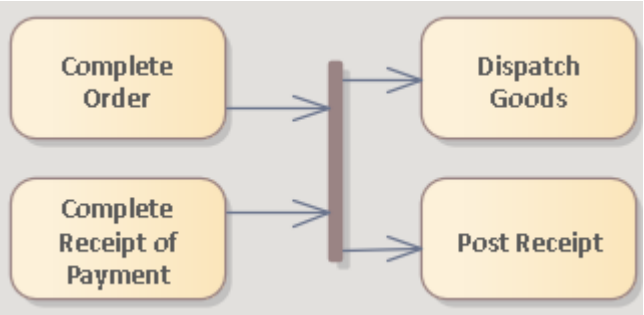
The Fork/Join elements can be used to:

- Fork or split the flow into a number of concurrent flows
- Join the flow of a number of concurrent flows
- Both join and split a number of incoming flows to a number of outgoing flows

These elements are used in both Activity and StateMachine diagrams, in either vertical or horizontal orientation. With respect to StateMachine diagrams, Forks and Joins are used as pseudostates. Other pseudostates include History states, Entry Points and Exit Points. Forks are used to split an incoming transition into concurrent multiple transitions leading to different target states. Joins are used to merge concurrent multiple transitions into a single transition leading to a single target. They are semantic inverses. To learn more about these individual elements see their specific topics.

Example Diagrams

Description	Diagram
Fork or split the flow into	

a number of concurrent flows	
Join the flow of a number of concurrent flows	
Join and Fork a number of incoming flows to a number of outgoing flows	

Toolbox icon

or

OMG UML Specification:

Forks in Activity Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.388) states:

Fork vertices serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices (i.e. vertices in different regions of a composite state). The segments outgoing from a fork vertex must not have guards or triggers.

Forks in State Machine Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

(A) *fork* Pseudostates serve to split an incoming Transition into two or more Transitions terminating on Vertices in orthogonal Regions of a composite State. The Transitions outgoing from a fork Pseudostate cannot have a guard or a trigger.

Joins in Activity Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.389) states:

A *JoinNode* is a ControlNode that synchronizes multiple flows. A JoinNode shall have exactly one outgoing ActivityEdge but may have multiple incoming ActivityEdges. If any of the incoming edges of a JoinNode are ObjectFlows, the outgoing edge shall be an ObjectFlow. Otherwise the outgoing edge shall be a ControlFlow.

Joins in State Machine Diagrams

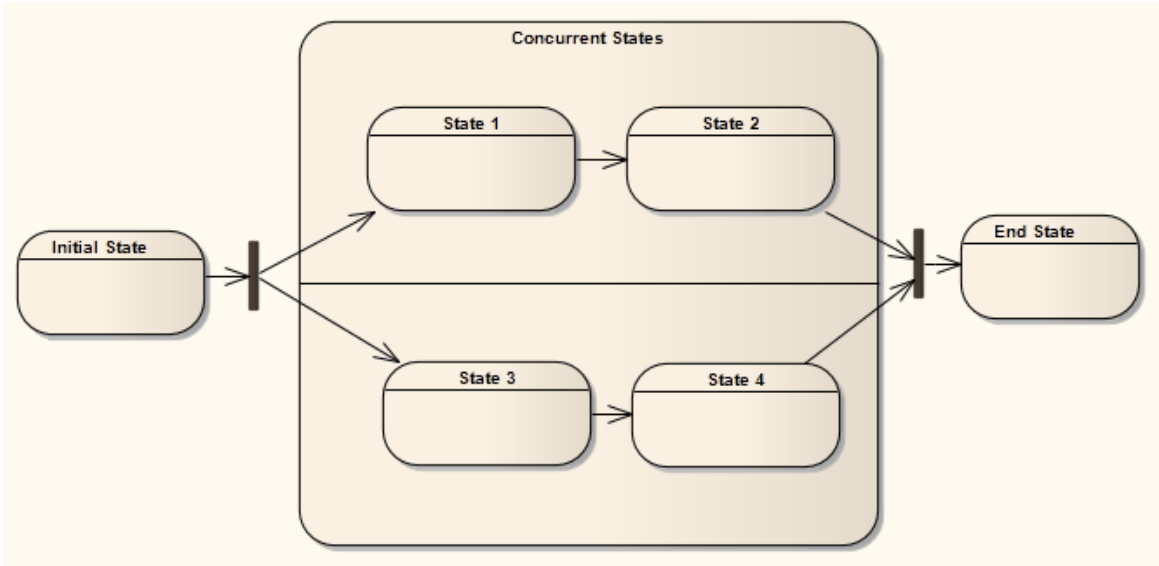
The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

(A *Join*) Pseudostate serves as a common target Vertex for two or more Transitions originating from Vertices in different orthogonal Regions. Transitions terminating on a join Pseudostate cannot have a guard or a trigger. Similar to junction points in Petri nets, join Pseudostates perform a synchronization function, whereby all incoming Transitions have to complete before execution can continue through an outgoing Transition.

Fork

Description

The Fork element is used in both Activity and StateMachine diagrams. With respect to StateMachine diagrams, a Fork pseudostate signifies that its incoming transition comes from a single state, and it has multiple outgoing transitions. These transitions must occur concurrently, requiring the use of concurrent regions, as depicted here in the Composite State. Unlike Choice or Junction pseudostates, Forks must not have triggers or guards. This diagram demonstrates a Fork pseudostate dividing into two concurrent regions, which then return to the End State via the Join pseudostate.



OMG UML Specification:

Forks in Activity Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.388) states:

Fork vertices serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices (i.e. vertices in different regions of a composite state). The segments outgoing from a fork vertex must not have guards or triggers.

Forks in State Machine Diagrams

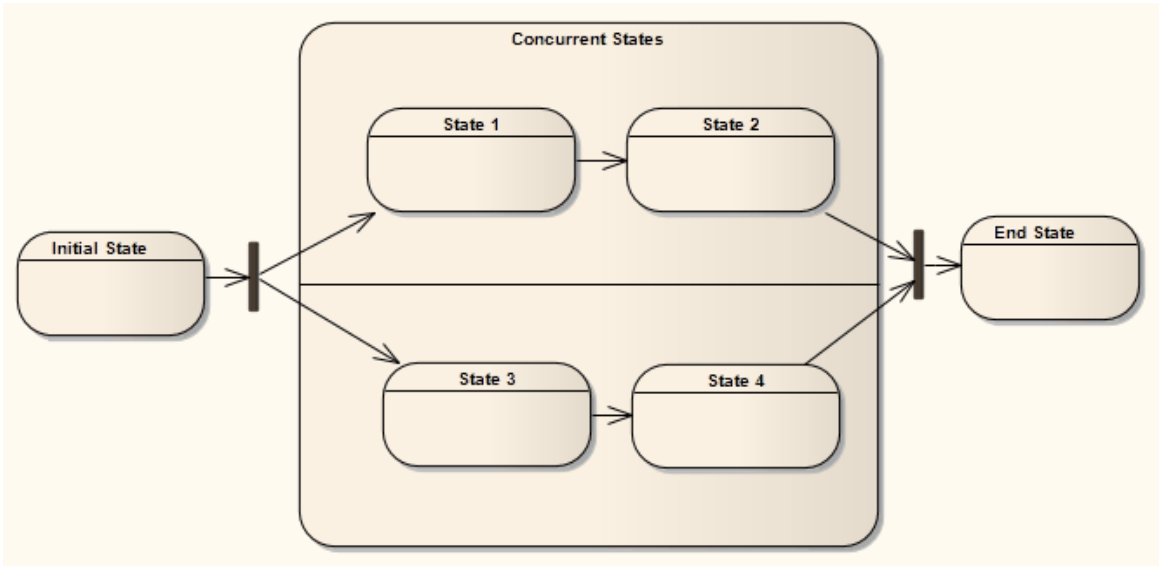
The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

(A) *fork* Pseudostates serve to split an incoming Transition into two or more Transitions terminating on Vertices in orthogonal Regions of a composite State. The Transitions outgoing from a fork Pseudostate cannot have a guard or a trigger.

Join

Description

The Join element is used by Activity and StateMachine diagrams. The example illustrates a Join transition between Activities. With respect to StateMachine diagrams, a Join pseudosate indicates multiple States concurrently transitioning into the Join and onto a single State. Unlike Choice or Junction pseudostates, Joins must not have triggers or guards. This diagram demonstrates a Fork pseudostate dividing into two concurrent Regions, which then return to the End State via the Join.



OMG UML Specification:

Joins in Activity Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.389) states:

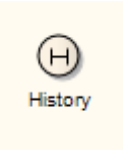
A *JoinNode* is a ControlNode that synchronizes multiple flows. A JoinNode shall have exactly one outgoing ActivityEdge but may have multiple incoming ActivityEdges. If any of the incoming edges of a JoinNode are ObjectFlows, the outgoing edge shall be an ObjectFlow. Otherwise the outgoing edge shall be a ControlFlow.

Joins in State Machine Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

(A *Join*) Pseudostate serves as a common target Vertex for two or more Transitions originating from Vertices in different orthogonal Regions. Transitions terminating on a join Pseudostate cannot have a guard or a trigger. Similar to junction points in Petri nets, join Pseudostates perform a synchronization function, whereby all incoming Transitions have to complete before execution can continue through an outgoing Transition.

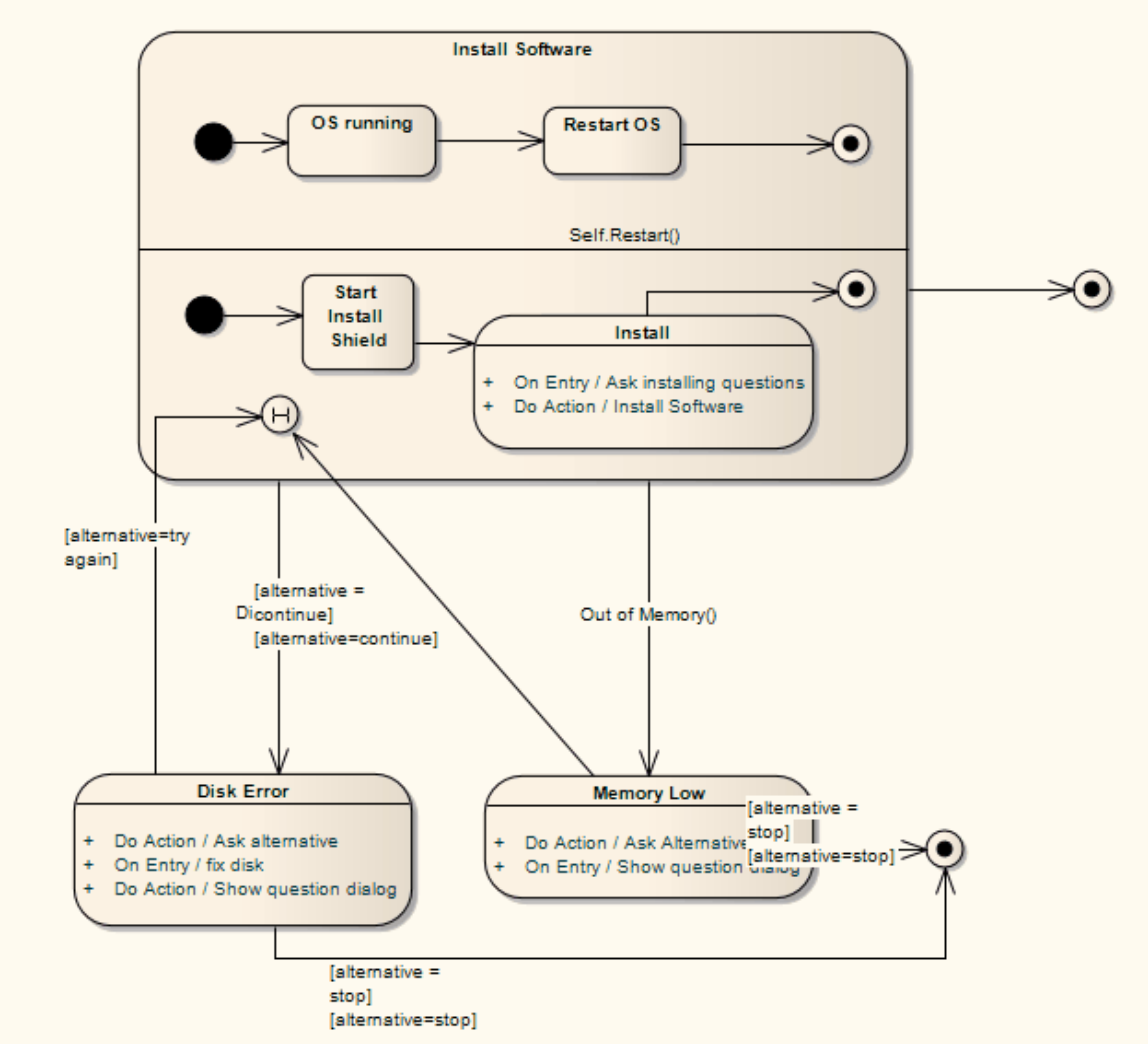
History



Description

There are two types of History pseudostate defined in UML: shallow and deep history. A shallow History sub-state is used to represent the most recently active sub-state of a Composite State; this pseudostate does not recurse into this sub-state's active configuration, should one exist. A single connector can be used to depict the default shallow History state, in case the Composite State has never been entered.

A deep History sub-state, in contrast, reflects the most recent active configuration of the Composite State. This includes active sub-states of all regions, and recurses into those sub-states' active sub-states, should they exist. Only one deep history and one shallow history can exist within a composite state. You can reassign a shallow History sub-state as a deep History sub-state using the 'Advanced' element context menu.



Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.309) states:

The concept of State *history* was introduced by David Harel in the original statechart formalism. It is a convenience concept associated with Regions of composite States

whereby a Region keeps track of the state configuration it was in when it was last exited. This allows easy return to that same state configuration, if desired, the next time the Region becomes active (e.g., after returning from handling an interrupt), or if there is a local Transition that returns to its history.

Two types of history Pseudostates are provided.

Deep history (deepHistory) represents the full state configuration of the most recent visit to the containing Region. The effect is the same as if the Transition terminating on the deepHistory Pseudostate had, instead, terminated on the innermost State of the preserved state configuration, including execution of all entry Behaviors encountered along the way.

Shallow history (shallowHistory) represents a return to only the topmost substate of the most recent state configuration, which is entered using the default entry rule.

Initial

Description

The Initial element is used by Activity and StateMachine diagrams. In Activity diagrams, it defines the start of a flow when an Activity is invoked. With StateMachines, the Initial element is a pseudostate used to denote the default state of a Composite State; there can be one Initial vertex in each Region of the Composite State.

This simple example shows the start of a flow to receive an order.

See the OMG Unified Modeling Language specification, (v2.5.1, Figure 12.97, p.378.)

The activity flow is completed by a Final or Flow Final node.

Notes

- Moving a diagram generally does not affect the location of elements in Packages; if you move a diagram out of one Package into another, all the elements in the diagram remain in the original Package

However, Initial elements are used only within one diagram, have no meaning outside that diagram, and are never re-used in any other diagram; therefore, if you move a diagram containing these elements, they are moved to the new parent Package with the diagram

Toolbox icon

OMG UML Specification:

Initial in Activity Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.387) also states:

An InitialNode is a ControlNode that acts as a starting point for executing an Activity. An Activity may have more than one InitialNode. If an Activity has more than one InitialNode, then invoking the Activity starts multiple concurrent control flows, one for each InitialNode.

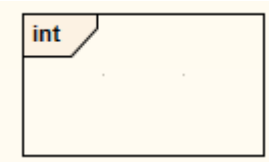
An InitialNode shall not have any incoming ActivityEdges, which means the InitialNodes owned by an Activity will always be enabled when the Activity begins execution and a single control token is placed on each such InitialNode when Activity execution starts. The outgoing ActivityEdges of an InitialNode must all be ControlFlows. The control token placed on an InitialNode is offered concurrently on all outgoing ControlFlows.

Initial in State Machine Diagrams

The OMG Unified Modeling Language specification, (v2.5.1, p.312) states:

An initial Pseudostate represents a starting point for a Region; that is, it is the point from which execution of its contained behavior commences when the Region is entered via default activation. It is the source for at most one Transition, which may have an associated effect Behavior, but not an associated trigger or guard. There can be at most one initial Vertex in a Region.

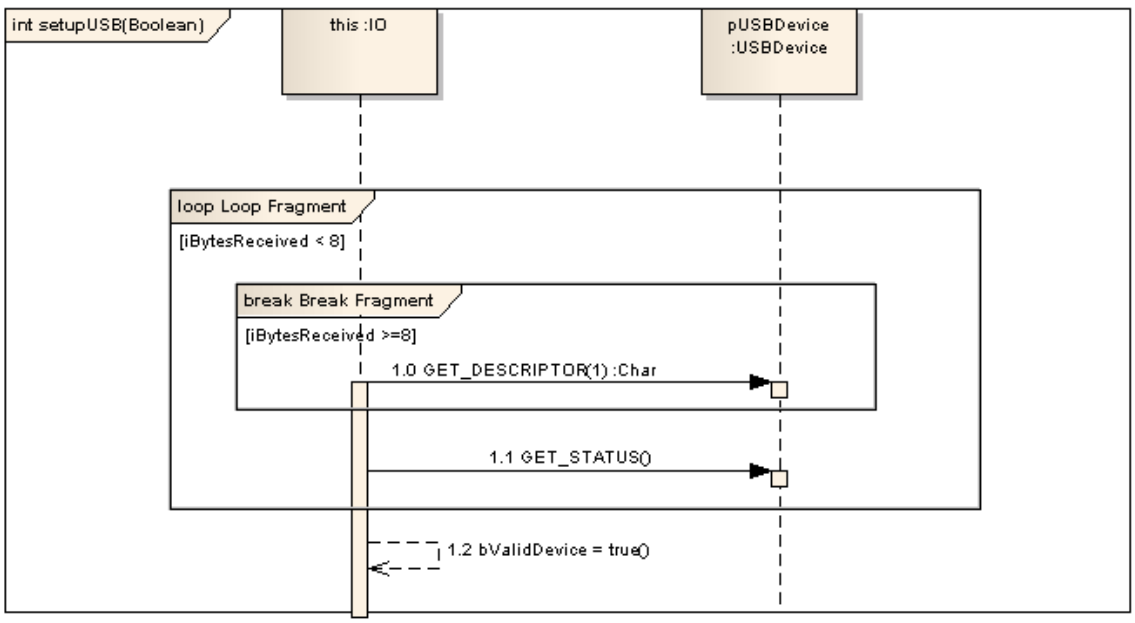
Interaction



Description

You can use an Interaction element to insert an Interaction diagram as a child of a Class element. The Interaction element can contain a diagram of any of these types:

- Sequence
- Communication
- Timing



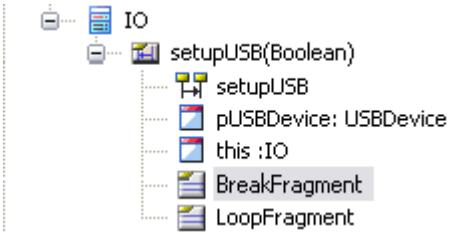
An Interaction element in Enterprise Architect is treated as a behavior of the classifier it is encapsulated within. It can have parameters and return types, which are modeled using the 'Behavior' tab of the Interaction element's 'Properties'

dialog. The element is interpreted as a method of the containing Class in the generated code (see the *Generate Code From Behavioral Model* topic).

An Interaction element can also be set as the classifier for an Interaction Occurrence in a Sequence diagram, or for a Call Behavior Action in an Activity diagram. Establishing such an association (between a behavior and a behavior call) facilitates adding arguments that can be individually mapped to the associated behavior's parameters.

Notes

- The behavioral code generation engine expects the Sequence diagram and all its associated Messages and Combined Fragments to be encapsulated within an Interaction element (such as setupUSB in this example)



  (The IO Class is available in the EAExample model, under 'Systems Engineering Model | Implementation Model | Software')

- The 'Interaction' icon is listed on the 'Additional' page of the 'Interaction' Toolbox, but should only be added to elements through the element context menu on the

diagram or in the Browser window

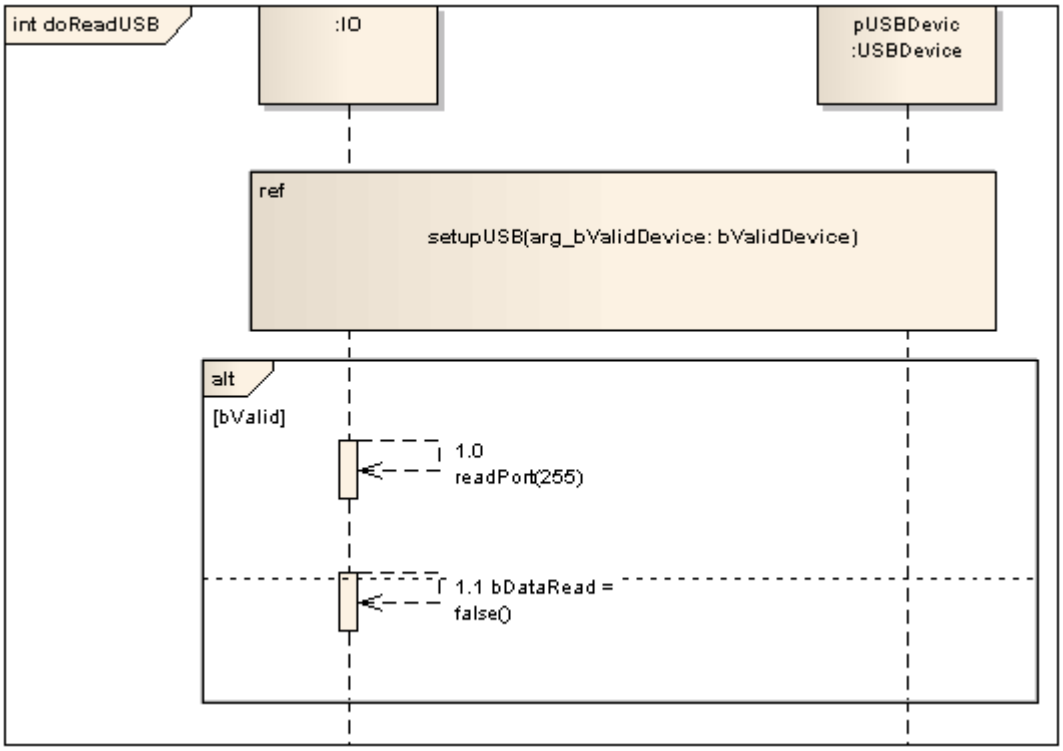
Toolbox icon

Interaction Occurrence

An Interaction Occurrence (or InteractionUse) is a reference to an existing Interaction (Sequence) diagram. Interaction Occurrences are visually represented by a frame, with 'ref' in the frame's title space. The diagram name is indicated in the frame contents.

To create an Interaction Occurrence, simply open a Sequence diagram (preferably contained within an Interaction element) and drag another Sequence diagram (also preferably contained within an Interaction element) into its workspace. A dialog displays, providing configuration options. The resulting Interaction Occurrence acts as an invocation of the original Interaction. You use the 'Call' tab of the Properties window for the element to set up the actual arguments of the Interaction and also to change to a different associated Interaction element.

This figure illustrates the use of an Interaction Occurrence in another Interaction (Sequence) diagram. You can display the sequence represented by the Interaction Occurrence by double-clicking on the element.



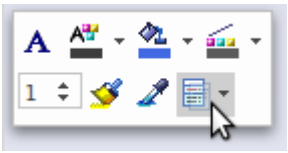
Fill Opacity

Whilst an Interaction Occurrence usually encloses a number of other elements, there might be reasons for hiding those elements as well as times to fully show them, or perhaps just indicate that they are there, depending on the immediate purpose of the diagram. You can apply these nuances in the display of elements behind and covered or overlapped by an Interaction Occurrence, by changing the opacity of the element.

Before setting the opacity, check that the element has a fill color.

You set the opacity using an icon from either of these two pop-up element toolbars:

- Click on the Interaction Occurrence element and on the icon:



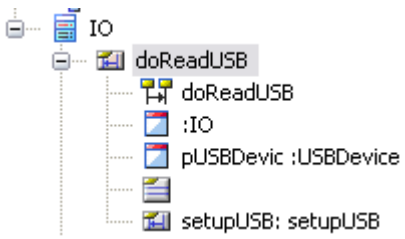
- Right-click on the Interaction Occurrence element and look above the context menu:

Click on the icon and select:

- 100% for total opacity, where the elements behind and overlapping or covered by the Interaction Occurrence are hidden (you could right-click on individual elements and select the 'Z-Order | Bring to Top' option to expose those elements only)
- 0% for no opacity, where the fill color is not applied and anything behind the Interaction Occurrence is fully visible
- 75%, 50% or 25% to set the appropriate degree of opacity and make the covered elements visible but over-shaded

Notes

- The behavioral code generation engine expects the Sequence diagram and all its associated messages and interaction fragments to be encapsulated within an Interaction element (such as doReadUSB in this example)

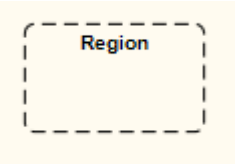


OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.620) refers to an Interaction Occurrence as an InteractionUse, and states:

An InteractionUse refers to an Interaction. The InteractionUse is a shorthand for copying the contents of the referenced Interaction where the InteractionUse is. To be accurate the copying must take into account substituting parameters with arguments and connect the formal Gates with the actual ones.

Interruptible Activity Region

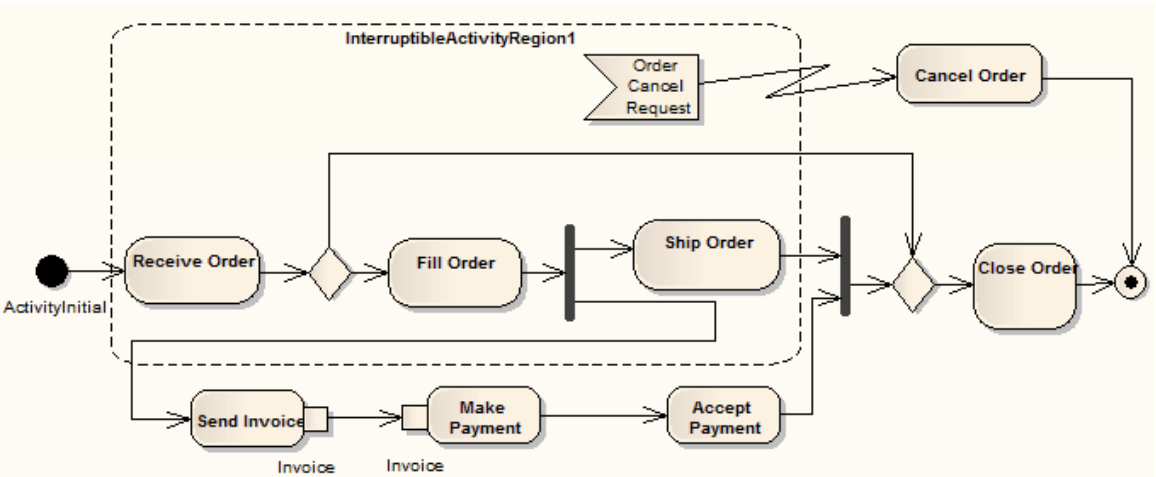


Description

In an Activity diagram, an Interruptible Region surrounds a group of Activity elements, all affected by certain interrupts in such a way that all tokens passing within the region are terminated should the interruption(s) be raised. Any processing occurring within the bounds of an Interruptible Activity Region is terminated when a flow is instigated across an interrupt flow to an external element.

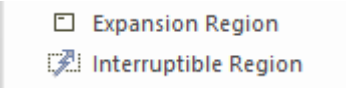
You create an Interruptible Activity Region as one variant of a Region (the other is an Expansion Region), using the Activity pages of the Diagram Toolbox.

This example illustrates that an order cancellation kills any processing of the order at the receipt, filling or shipping stage.



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.100, p.381.)

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.407) states:

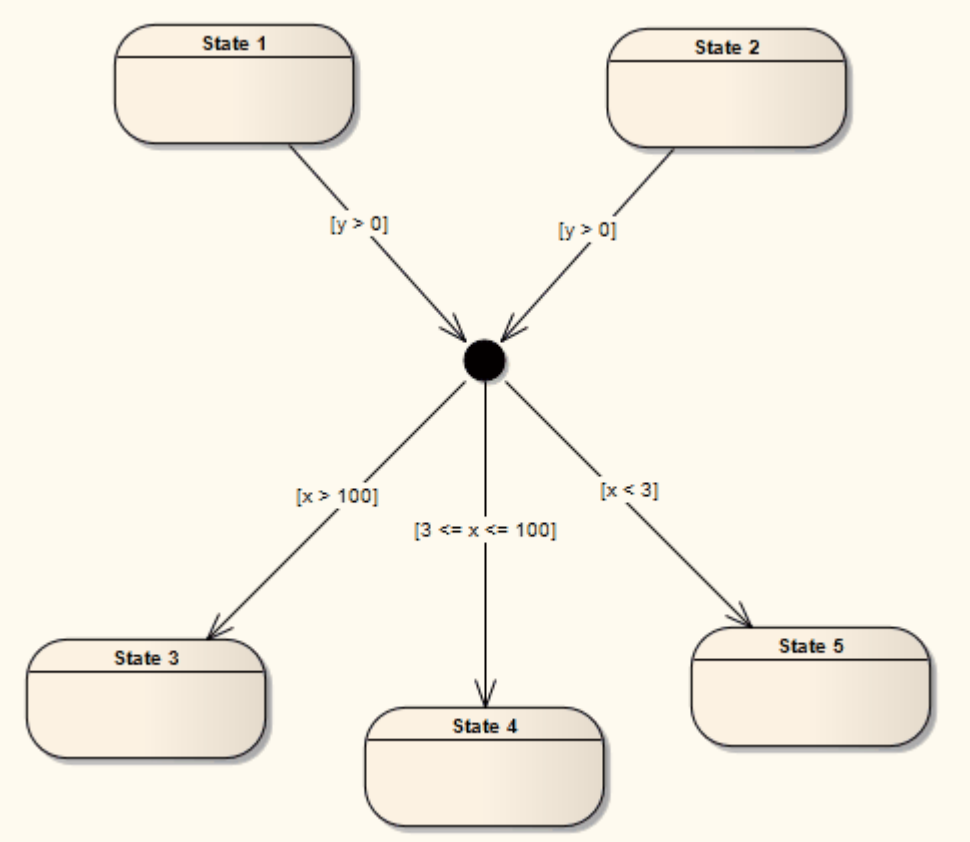
An InterruptibleActivityRegion is an ActivityGroup that supports termination of a portion of an Activity. An InterruptibleActivityRegion contains only ActivityNodes. It also identifies as interruptingEdges certain ActivityEdges that have their source within the region and their target outside the region. When a token offered along an interruptingEdge is accepted and traverses that edge, then the execution of all containedNodes of the region is terminated and all tokens are removed from them.

Junction

Description

Junction pseudostates are used to design complex transitional paths in StateMachine diagrams. A Junction can be used to combine or merge multiple paths into a shared transition path. Alternatively, a Junction can split an incoming path into multiple paths, similar to a Fork pseudostate. Unlike Forks or Joins, Junctions can apply guards to each incoming or outgoing transition, such that if the guard expression is False, the transition is disabled.

This example illustrates how guards can be applied to transitions coming into or out of a Junction pseudostate.



Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.313) states:

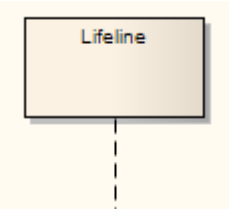
This type of Pseudostate is used to connect multiple Transitions into compound paths between States. For example, a junction Pseudostate can be used to merge multiple incoming Transitions into a single outgoing Transition representing a shared continuation path. Or, it can be used to split an incoming Transition into multiple

outgoing Transition segments with different guard Constraints.

NOTE. Such guard Constraints are evaluated before any compound transition containing this Pseudostate is executed, which is why this is referred to as a static conditional branch.

It may happen that, for a particular compound transition, the configuration of Transition paths and guard values is such that the compound transition is prevented from reaching a valid state configuration. In those cases, the entire compound transition is disabled even though its Triggers are enabled. (As a way of avoiding this situation in some cases, it is possible to associate a predefined guard denoted as “else” with at most one outgoing Transition. This Transition is enabled if all the guards attached to the other Transitions evaluate to false). If more than one guard evaluates to true, one of these is chosen. The algorithm for making this selection is not defined.

Lifeline



Description

A Lifeline is an individual participant in an interaction (that is, Lifelines cannot have multiplicity). A Lifeline represents a distinct connectable element. To specify that representation within Enterprise Architect, right-click on the Lifeline and select the 'Advanced | Instance Classifier' option. The 'Select <Item>' dialog displays, which you use to locate the required project classifiers.

Lifelines are available in Sequence diagrams. There are different Lifeline elements for Timing diagrams (State Lifeline and Value Lifeline); however, although the representation differs between the two diagram types, the meaning of the Lifeline is the same.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification,

(v2.5.1, p.572) states:

In an interaction diagram a Lifeline describes the time-line for a process, where time increases down the page. The distance between two events on a time-line does not represent any literal measurement of time, only that non-zero time has passed.

Events on the same time-line are ordered linearly down the page, except where they occur within a parallel combined fragment, or along a lifeline within a “coregion”. (...) Within a parallel combined fragment or a coregion, events are not locally ordered unless that is directly imposed by a general ordering construct. (...).

The order of OccurrenceSpecifications along a Lifeline is significant denoting the order in which these OccurrenceSpecifications will occur. The absolute distances between the OccurrenceSpecifications on the Lifeline are, however, irrelevant for the semantics.

Merge

Description

A Merge Node brings together a number of alternative flow paths in Activity, Analysis and Interaction Overview diagrams. For example, if a Decision is used after a Fork, the two flows coming out of the Decision must be merged into one before going to a Join; otherwise, the Join waits for both flows, only one of which arrives.

A Merge Node has multiple incoming edges and a single outgoing edge. The edges coming into and out of a Merge Node must be either all object flows or all control flows.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.427) states:

A merge node is a control node that brings together multiple alternate flows. It is not used to synchronize concurrent flows but to accept one among several alternate flows.

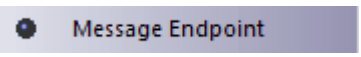
Message Endpoint

Description

A Message Endpoint element defines the termination of a State or Value Lifeline in a Timing diagram. It indicates that the Message:

- Terminates at an undefined point outside the State or Value Lifeline, having started at an identified point within the Lifeline, or
- Originates from an undefined point outside a State or Value Lifeline, terminating at an identified point within the Lifeline

Toolbox icon

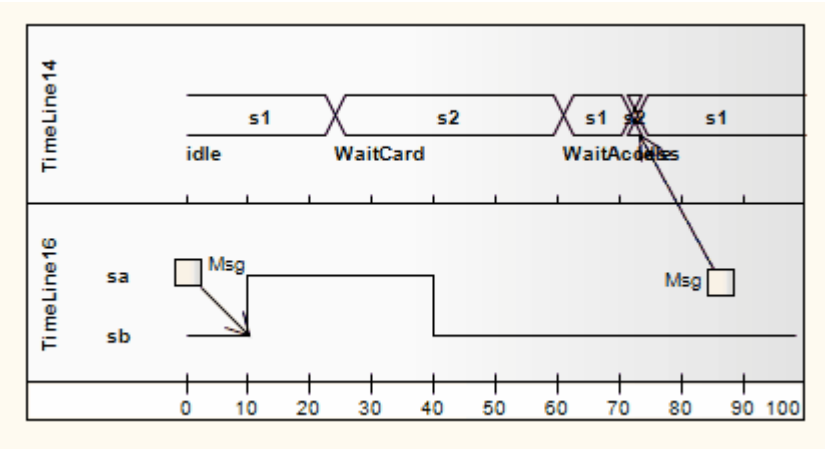


Message Label

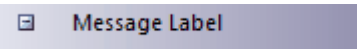
Description

A Message Label is an alternative way of denoting Messages between Lifelines, which is useful for 'uncluttering' Timing diagrams strewn with messages. To indicate a Message between Lifelines, draw a connector from the source Lifeline into a Message Label. Next, draw a connector from another Message Label to the target Lifeline. Note that the label names must match to reflect that the message occurs between the two Message Labels.

This diagram illustrates how Message Labels are used to construct a message between Lifelines.



Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.577) states:

The message-name appearing in a request-message-label is the name property of the Message. If the Message has a signature, this will be the name of the Operation or Signal referenced by the signature. Otherwise the name is unconstrained. If a request-message-label includes an input-argument-list, then either all input-arguments must have an in-parametername given or none may have one.

Note

Description

A Note element is a textual annotation that can be attached to a set of elements of any other type. The attachment is created separately, using a Notelink connector. Both Note and Notelink are available in any Enterprise Architect diagram, through the 'Common' pages of the Toolbox.

A Note is also called a Comment.

Note elements do not have a Properties dialog. You can give them a name in the Properties window, but this does not display on the diagram.

You can configure Enterprise Architect to display the text in all Notes elements in italics. Select the 'Start > Appearance > Preferences > Preferences' ribbon option and on the 'Diagram > Appearance' page select the 'Italic Note Element text' checkbox. This has an immediate effect, as does clearing the checkbox to show the text in normal font.

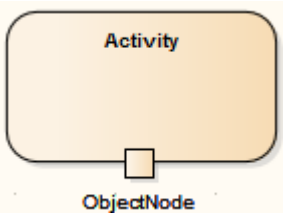
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.40) states:

A Comment is a textual annotation that can be attached to a set of Elements.

Object Node

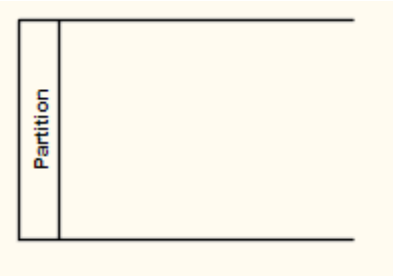


Description

An Object Node holds data that is input to or output from an Activity element. To set the type of an Object Node, click on it and press Ctrl+L (to select an Instance Classifier). Object Nodes can be connected by Object Flow connectors; if the Object Nodes on each end of an Object Flow are typed, their types should be compatible.

An Action Pin is a specialized form of Object Node.

Partition



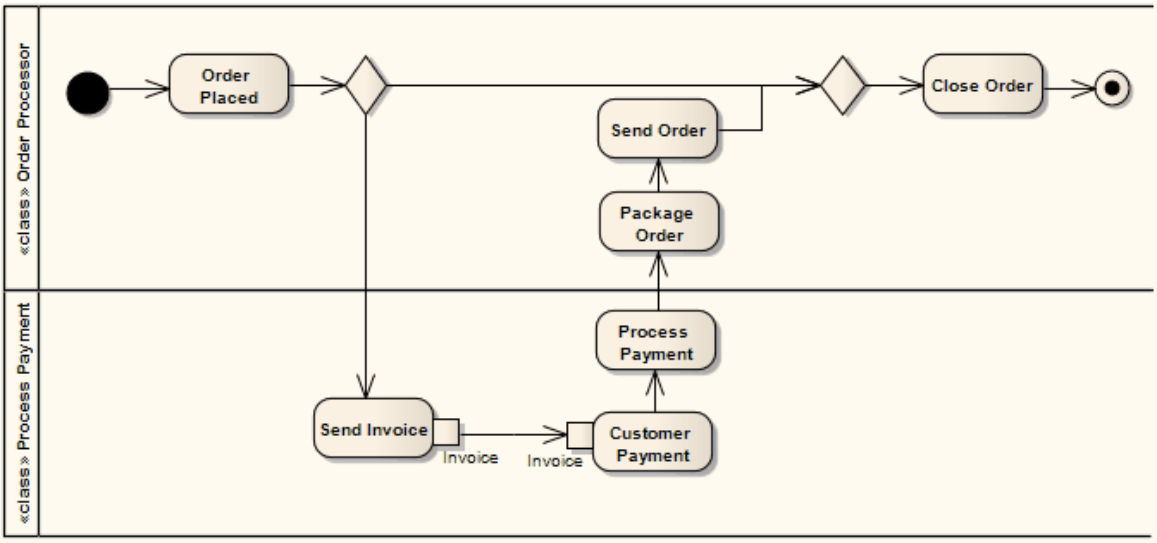
Description

Enterprise Architect supports two types of Activity Partition:

- The Activity Partition feature, which is used to logically organize an Activity element
- The Activity Partition element, described in this topic, which is used to logically organize an Activity diagram

These have similar effects - they partition the Actions of the Activity without affecting the token flow, helping to structure the view or parts of the Activity.

This example depicts the partitioning between the Classes *Process Payment* and *Order Processor*.



The Partition orientation defaults to horizontal. To turn it into a vertical Partition, right-click on it and select the 'Advanced | Vertical Partition' option.

You can neatly align and join the Activity Partitions on a diagram using the element context menu 'Dockable' option. For Partitions, the option defaults to selected.

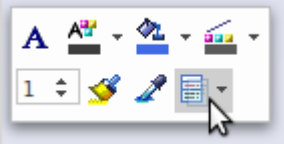
Setting Opaque Fill

Whilst an Activity Partition usually contains a number of other elements, there might be reasons for hiding those elements as well as times to fully show them, or perhaps just indicate that they are there, depending on the immediate purpose of the diagram. You can apply these nuances in the display of the Partition contents by changing the opacity of the element's fill color.

Before setting the opacity, give the element a fill color.

You set the opacity using an icon from either of these two pop-up element toolbars:

- Click on the Activity Partition element and on the icon:

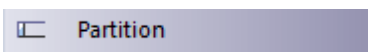


- Right-click on the Activity Partition element and look above the context menu:

Click on the icon and select:

- 100% for total opacity, where the contents of the Activity Partition are hidden (you can first right-click on individual elements and select the 'Z-Order | Bring to Top' option to expose those elements only)
- 0% for no opacity, where the fill color is not applied and the Activity Partition contents are fully visible
- 75%, 50% or 25% to set the appropriate degree of opacity and make the content elements visible but over-shaded

Toolbox icon



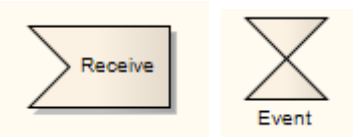
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.406) states:

An ActivityPartition is a kind of ActivityGroup for identifying ActivityNodes that have some characteristics in common. ActivityPartitions can share contents. They often

correspond to organizational units in a business model. They may be used to allocate characteristics or resources among the nodes of an Activity..

Receive

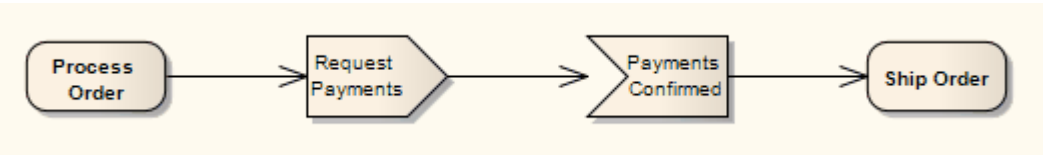


Description

A Receive element is used to define the acceptance or receipt of a request, in an Activity diagram. Movement from a Receive element occurs only once receipt is fulfilled according to its specification. The Receive element comes in two forms:

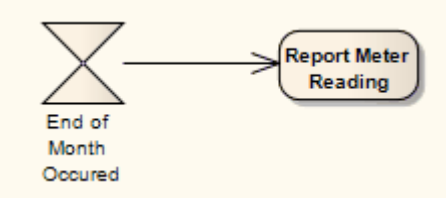
- Accept Event Action element (pennant shape)
- Accept Time Event Action element (hourglass shape)

This example reflects a payment process on an order. Upon receiving the payment (from Request Payments, a Send element), the payment is confirmed and the flow continues to ship the order.



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.26, p.312.)

To depict an Accept Time Event, use the standard Receive element from the Toolbox. Right-click on this element, and select the 'Advanced | Accept Time Event' option. This example shows the hourglass-shaped Accept Time Event Action:



See The OMG Unified Modeling Language specification, (v2.5.1, figure 12.27, p.312.)

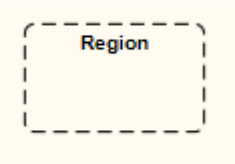
Toolbox icon

OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, p.489) states:

An AcceptEventAction is an Action that waits for the occurrence of one or more specific Events.

Region

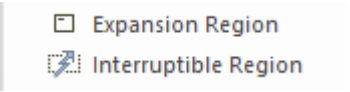


Description

Enterprise Architect supports two types of Region element:

- Expansion Region
- Interruptible Activity Region

Toolbox icon

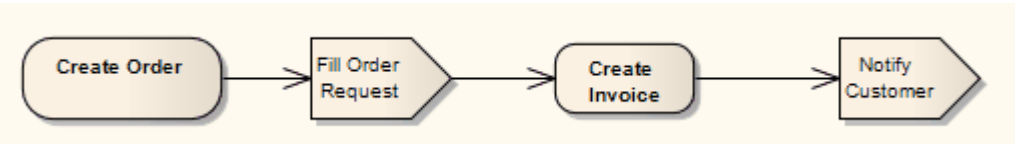


Send

Description

The Send element is used to depict the action of sending a signal, in an Activity diagram. It is the opposite of a Receive element. You can also create Send events using the 'Event' icon on the 'State' page of the Diagram Toolbox.

This example shows an order being processed, where a signal is sent to fill the processed order and, upon creation of the resulting invoice, a notification is sent to the customer.



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.132, p.408.)

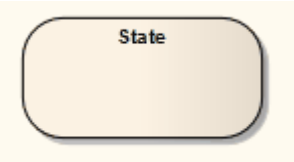
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.452) states:

A SendObjectAction is an Invocation action that transmits any kind of object to the object given on its target InputPin. The object to be transmitted is given on the single request InputPin of the SendObjectAction. If the object is a Signal instance, then it may be handled by the target object in the same way as an instance sent from a SendSignalAction or BroadcastSignalAction. Otherwise, the reception of the object can only be handled using an AnyReceiveEvent (...).

State



Description

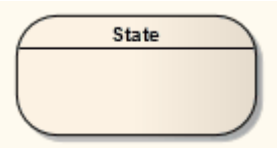
A State represents a situation where some invariant condition holds; this condition can be static (waiting for an event) or dynamic (performing a set of activities). State modeling is usually related to Classes, and describes the enable-able states a Class or element can be in and the transitions that enable the element to move there. There are two types of State: Simple States and Composite States, both created from the 'State' icon from the Toolbox.

Furthermore, there are pseudostates, resembling some aspect of a State but with a pre-defined implication. Pseudostates model complex transitional paths, and classify common StateMachine behavior.

You can define entry, internal and exit actions for a State using operations. State elements can have three operations (*entry*, *do* and *exit*) that are created and defined through the 'Behavior' tab of the Features window (Start > All Windows > Properties > Element Features > Features). The tab displays only when the selected element is a State. It automatically lists the three operations, and you can either type a text value in the 'Name/Comment' field, or assign a behavior element of code using the 'Behavior' page of the

Properties window (see the *Operation Behavior* Help topic).

If a State element has features such as operations, internal triggers or inherited operations and attributes, the depiction of the element in a diagram has a line under the element name. This line persists if the features are hidden. The line also displays if the 'Show State Compartment' checkbox is selected on the 'Objects' page of the 'Preferences' dialog (select the 'Start > Appearance > Preferences > Preferences' ribbon option and the 'Objects' page).



Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.308) states:

A State models a situation in the execution of a StateMachine Behavior during which some invariant condition holds. In most cases this condition is not explicitly defined, but is implied, usually through the name associated with the State.

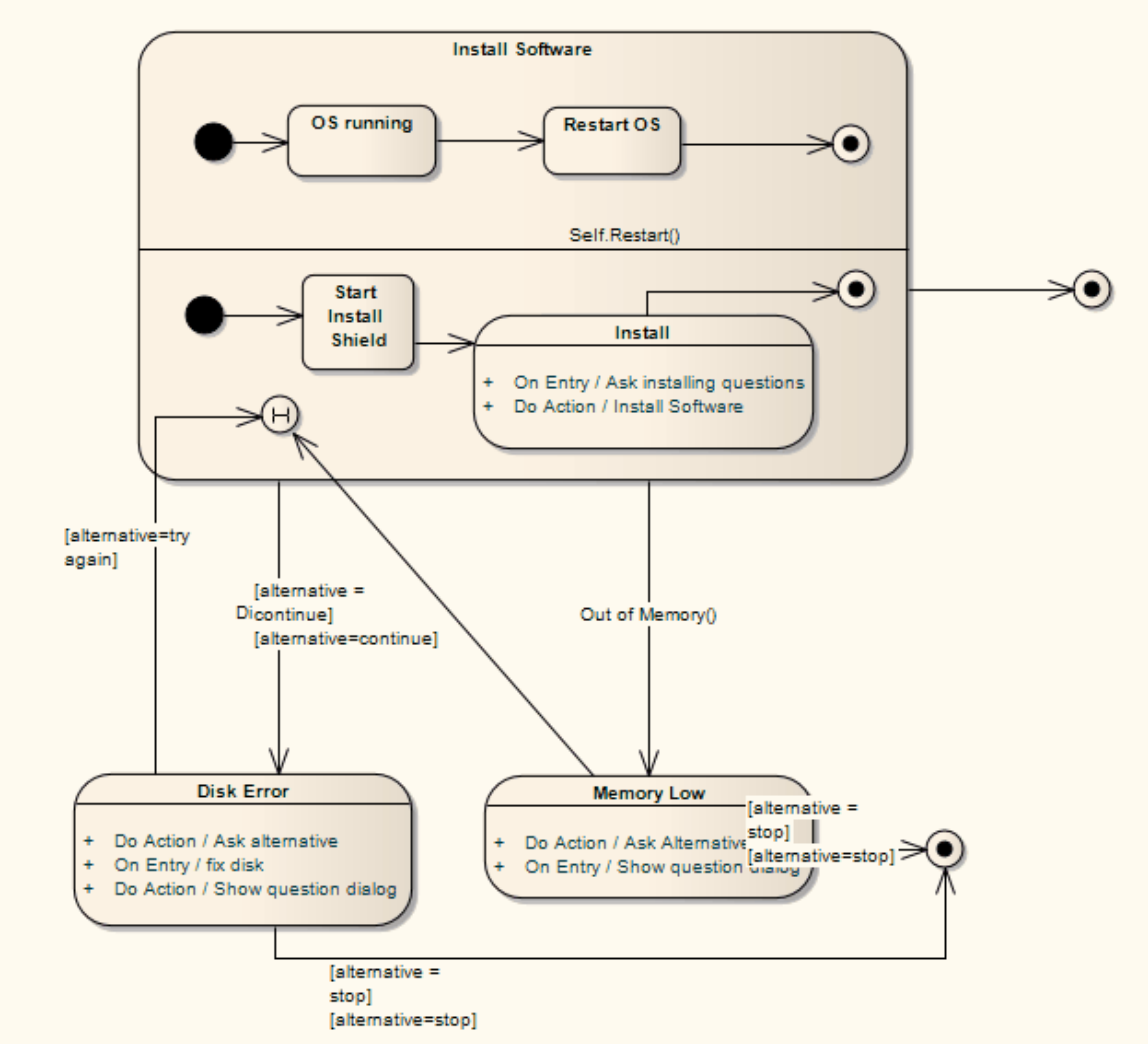
Composite State

Description

Composite States are composed within the StateMachine diagram by expanding a State element, adding Regions if applicable, and dragging further State elements, related elements and connectors within its boundaries. The internal State elements are then referred to as Substates.

(You can also define a State element, as with many other types of element, as a composite element; this then has a hyperlink to a child diagram that can be another StateMachine diagram or other type of diagram elsewhere in the model.)

Composite States can be orthogonal, if Regions are created. If a Composite State is orthogonal, its entry denotes that a single Substate is concurrently active in each Region. The hierarchical nesting of Composite States, coupled with Region use, generates a situation of multiple States concurrently active; this situation is referred to as the active State configuration.



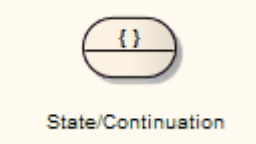
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.308) states:

A composite State contains at least one Region, whereas a submachine State refers to an entire StateMachine, which is, conceptually, deemed to be “nested” within the State. A composite State can be either a simple composite State with exactly one Region or an orthogonal State with multiple Regions (isOrthogonal = true). (...) Any State enclosed within a Region of a composite State is called a substate of

that composite State. It is called a direct substate when it is not contained in any other State; otherwise, it is referred to as an indirect substate.

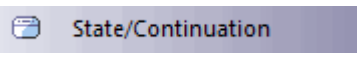
State/Continuation



Description

The State/Continuation element serves two different purposes for Interaction (Sequence) diagrams, as State Invariants and Continuations. The system prompts you to identify the purpose when you create the element.

Toolbox icon



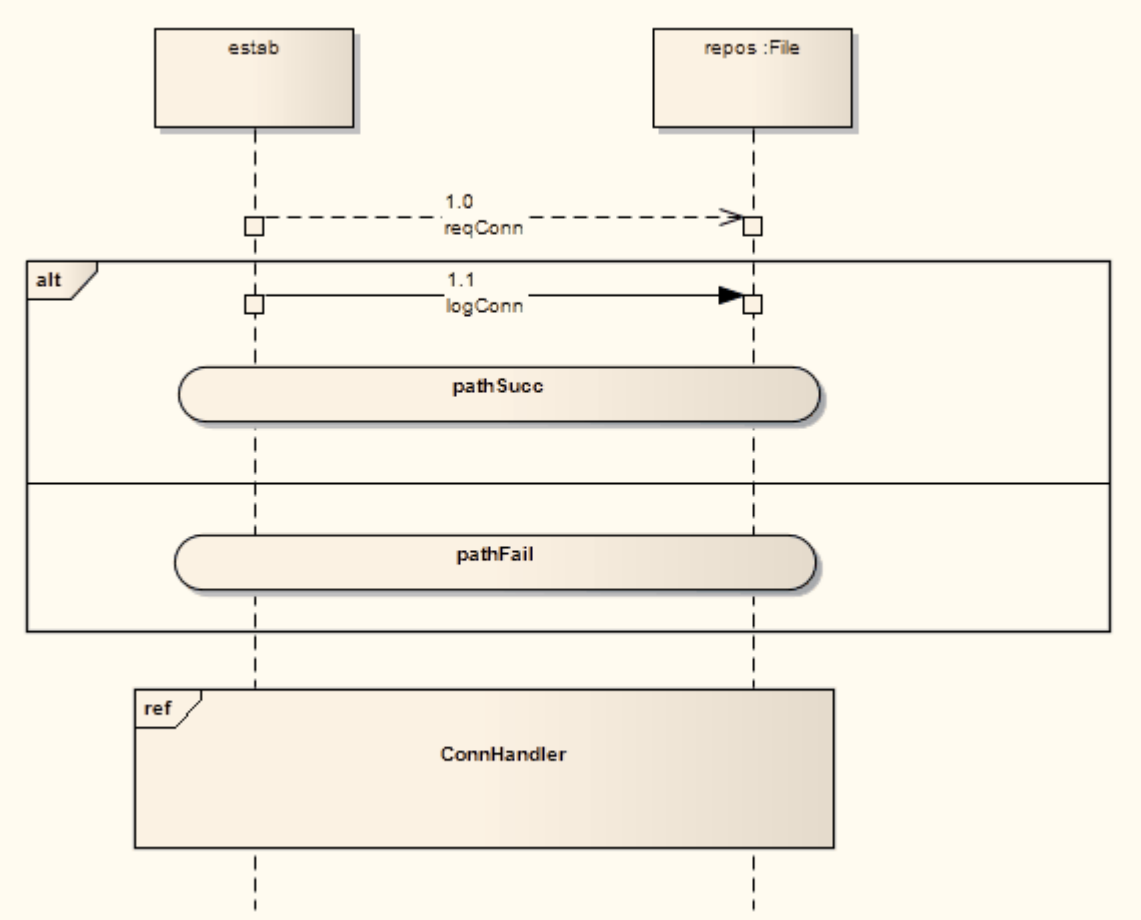
Continuation

Description

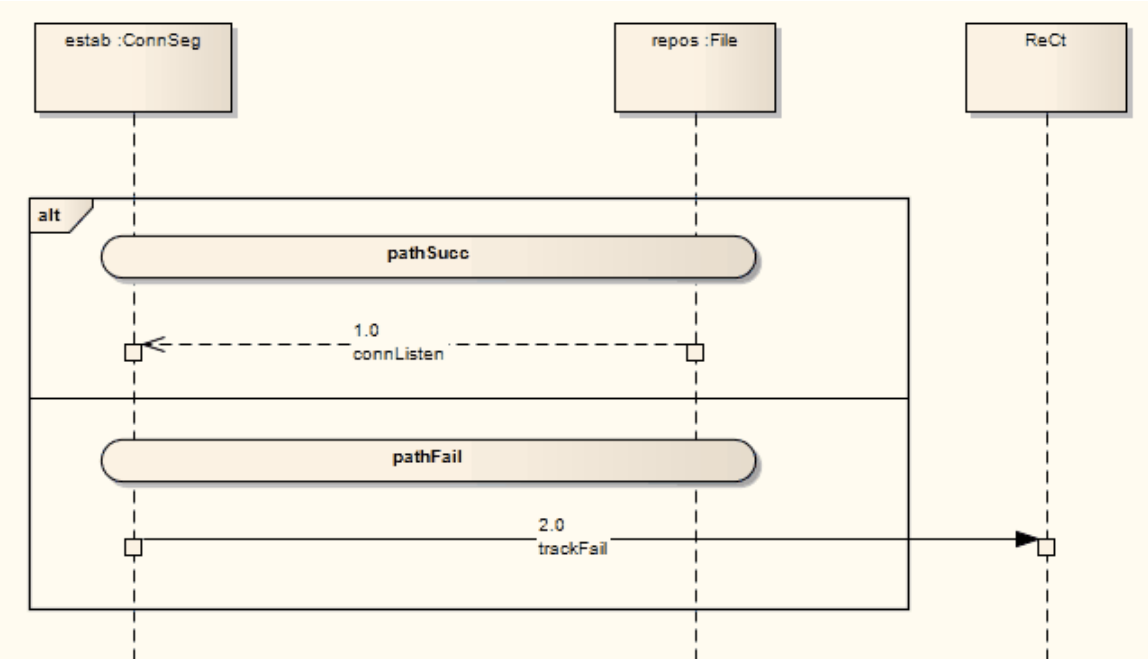
A Continuation is used in seq and alt Combined Fragments, to indicate the branches of continuation that an operand follows. To indicate a continuation, end an operand with a Continuation, and indicate the continuation branch with a matching Continuation (same name) preceding the Interaction Fragment.

You create a Continuation by dragging the State/Continuation element onto the diagram from the 'Interaction Elements' page of the Toolbox.

For this Continuation example, an alt Combined Fragment has Continuations pathSucc and pathFail. These Continuations are located within the Interaction Occurrence ConnHandler, which has subsequent events based on the continuation.



This diagram shows the interaction referenced by the Interaction Occurrence.



OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, p. 609) states:

A Continuation is a syntactic way to define continuations of different branches of an alternative CombinedFragment. Continuations are intuitively similar to labels representing intermediate points in a flow of control.

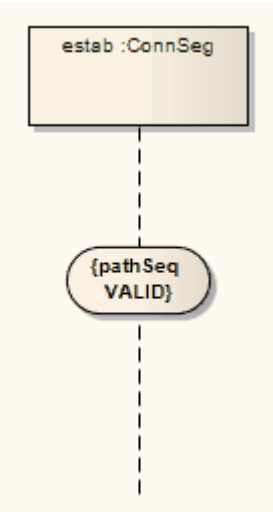
The OMG Unified Modeling Language specification, (v2.5.1, pp. 582-583) also states:

Continuations have semantics only in connection with Alternative CombinedFragments and (weak) sequencing. If an InteractionOperand of an Alternative CombinedFragment ends in a Continuation with name (say) X, only InteractionFragments starting with the Continuation X (or no continuation at all) can be appended.

State Invariant

A State Invariant is a condition applied to a Lifeline, which must be fulfilled for the Lifeline to exist. You create a State Invariant by dragging the State/Continuation element onto the diagram from the Interaction Elements page of the Toolbox.

This diagram illustrates a State Invariant.



When a State Invariant is moved near to a Lifeline, it snaps to the center. If the sequence object is dragged left or right, the State Invariant moves with it.

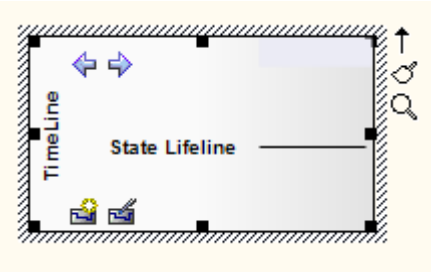
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p. 630) states:

A StateInvariant is a runtime constraint on the participants of the Interaction. It may be used to specify a variety of different kinds of Constraints, such as values of Attributes or Variables, internal or external States, and so on. A

StateInvariant is an InteractionFragment and it is placed on a Lifeline.

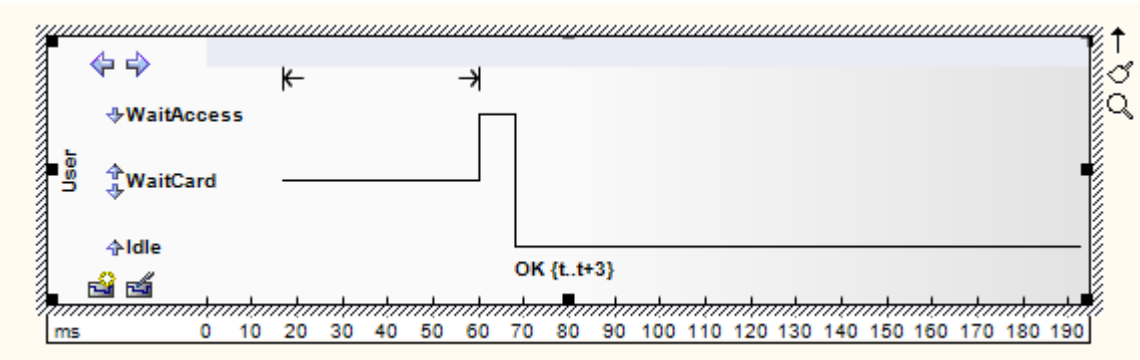
State Lifeline



Description

A Lifeline is the path an object takes across a measure of time, as indicated by the x-axis. There are two sorts: State Lifelines (defined here) and Value Lifelines, both used in Timing diagrams.

A State Lifeline follows discrete transitions between States, which are defined along the y-axis of the timeline. Any transition has optional attributes of timing constraints, duration constraints and observations. An example of a State Lifeline is shown here:



Transition point properties

A State Lifeline consists of a set of transition points. Each

transition point can be defined with these properties:

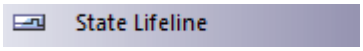
Property	Description
At time	Specifies the starting time for a change of state.
Transition to	Indicates the state to which the lifeline changes.
Event	Describes the occurring event.
Timing constraints	Refers to the time taken for a state to change within a lifeline, or the time taken to transmit a message (for example, t..t+3).
Timing observations	Provides information on the time of a state change or sent message.
Duration constraints	Pertains to a lifeline's period at a particular state. The constraint could be instigated by a change of state within a lifeline, or that lifeline's receipt of a message.
Duration observations	Indicates the interval of a lifeline at a particular state, begun from a change in state or message receipt.

Example properties

In the example diagram, the OK transition point has these properties:

Property	Value
At Time	68 ms
Transition to	Idle
Event	OK
Timing constraints	t..t+3
Timing observations	–
Duration constraints	–
Duration observations	–

Toolbox icon



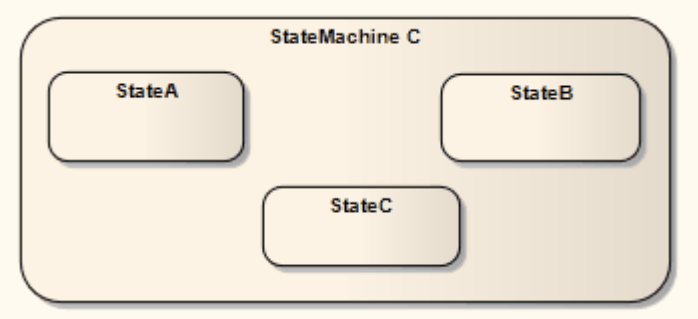
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.604) states:

This is the state of the classifier or attribute, or some testable condition, such as a discrete enumerable value.

It is also permissible to let the state-dimension be continuous as well as discrete. This is illustrative for scenarios where certain entities undergo continuous state changes, such as temperature or density

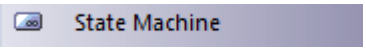
StateMachine



Description

A StateMachine element is a container for groups of related State elements. You can create sections of a StateMachine diagram, showing the organization of the inter-related State elements, and enclose each section in a StateMachine element. You can also create Regions on a StateMachine element.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.306) states:

A behavior StateMachine comprises one or more Regions, each Region containing a graph (possibly hierarchical) comprising a set of Vertices interconnected by arcs

representing Transitions. State machine execution is triggered by appropriate Event occurrences. A particular execution of a StateMachine is represented by a set of valid path traversals through one or more Region graphs, triggered by the dispatching of an Event occurrence that match active Triggers in these graphs. ... In the course of such a traversal, a StateMachine instance may execute a potentially complex sequence of Behaviors associated with the particular elements of the graphs that are being traversed (transition effects, state entry and state exit Behaviors, etc.)

Structured Activity

Structured Activity elements are used in Activity diagrams. A Structured Activity is an activity node that can have subordinate nodes as an independent Activity Group. You can set an option to ensure that no other Activities or their side effects interfere with this Activity's processing (the 'Must Isolate' checkbox in the Structured Activity element 'Properties' dialog).

Enterprise Architect provides a number of forms of Structured Activity, both basic and specialized.

Access

Ribbon	Design > Diagram > Toolbox : Specify 'Activity' in the 'Find Toolbox Item' dialog
Keyboard Shortcuts	Ctrl+Shift+3 : > Specify 'Activity' in the 'Find Toolbox Item' dialog
Other	You can display or hide the Diagram Toolbox by clicking on the or icons at the left-hand end of the Caption Bar at the top of the Diagram View.

Create Structured Activities

When you drag a Structured Activity icon from the Toolbox onto a diagram, a short menu displays from which you select one of these options:

- Loop Node
- Conditional Node
- Other

The first two options specifically create a Loop Node or Conditional Node.

The 'Other' option displays the 'New Structured Activity' dialog, on which you can select to create one of five types of Structured Activity element.

Structured Activity Types

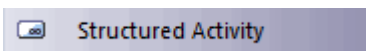
Type	Description
Simple Composite Activity	Generates a Composite Activity element with a child Activity diagram.
Loop Node	Represents a sequence of Actions and Activities that are to be repeated within the object.
Conditional	Represents an arrangement of Actions

Node	and Activities where choice determines which Activities are performed.
Structured Activity Node	Represents an ordered arrangement of executable Activity nodes (Actions, Decisions, Merges and so on) that can include branched and nested nodes; this is the base element from which the other types of Structured Activity are derived.
Sequential Node	Represents a sequential arrangement of executable Activity nodes.

Notes

- To protect the processing of a Loop or Conditional Node Structured Activity from interference from other Activities or their side effects, open the Properties window and select the 'Must Isolate' checkbox on the 'Loop' or 'Condition' tab

Toolbox icon



Structured Node

On a diagram, Structured Activity Nodes have broken borders, as shown.

You can nest other elements underneath the Structured Node, including other Structured Activity elements such as Conditional, Loop and other Structured Node elements.

OMG UML Specification:

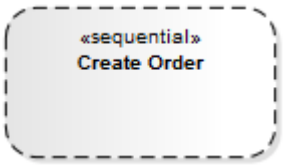
The OMG Unified Modeling Language specification, (v2.5.1, p.476) states:

A StructuredActivityNode is an Action that is also an ActivityGroup (...) and whose behavior is specified by the ActivityNodes and ActivityEdges it so contains. Unlike other kinds of ActivityGroup, a StructuredActivityNode owns the ActivityNodes and ActivityEdges it contains, and so a node or edge can only be directly contained in one StructuredActivityNode. StructuredActivityNodes may be nested (as a StructuredActivityNode, as an Action, is also an ActivityNode), however, so an edge or node may be indirectly contained in a number of nested StructuredActivityNodes.

Sequential Node

On a diagram, Sequential Activity Nodes have broken borders, and can contain nested elements that define a sequence of actions.

Sequential Nodes are flagged as composite elements in the context menu ('New Child Diagram | Composite'); however, when you add the child diagram the element converts to a simple composite Activity.



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.480) states:

A SequenceNode defines a complete, sequential ordering of all the ActivityNodes it contains, which must all be ExecutableNodes. When the SequenceNode executes, each of the nodes within it are executed in sequential order. The SequenceNode may also contain ActivityEdges between its nodes, and ActivityEdges may cross into and out of the SequenceNode. The semantics are equivalent to a general StructuredActivityNode containing the same nodes and edges, but with ControlFlows added to sequentially order the nodes as specified for the SequenceNode.

Loop Node

A Loop Structured Activity Node is used for defining a loop, and is commonly associated with ‘While’, ‘Repeat’ or ‘For’ loop statements.

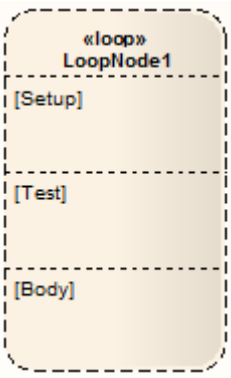
Each Loop Node has three partitions:

- Setup commonly initiates variables to be used in the loop's exit-condition; it is executed once on entry to the loop
- Test defines the loop exit-condition
- Body can contain Actions to be executed repeatedly until the Test produces a false value

The results of the final execution of the Test or Body are available after execution of the Loop is complete.

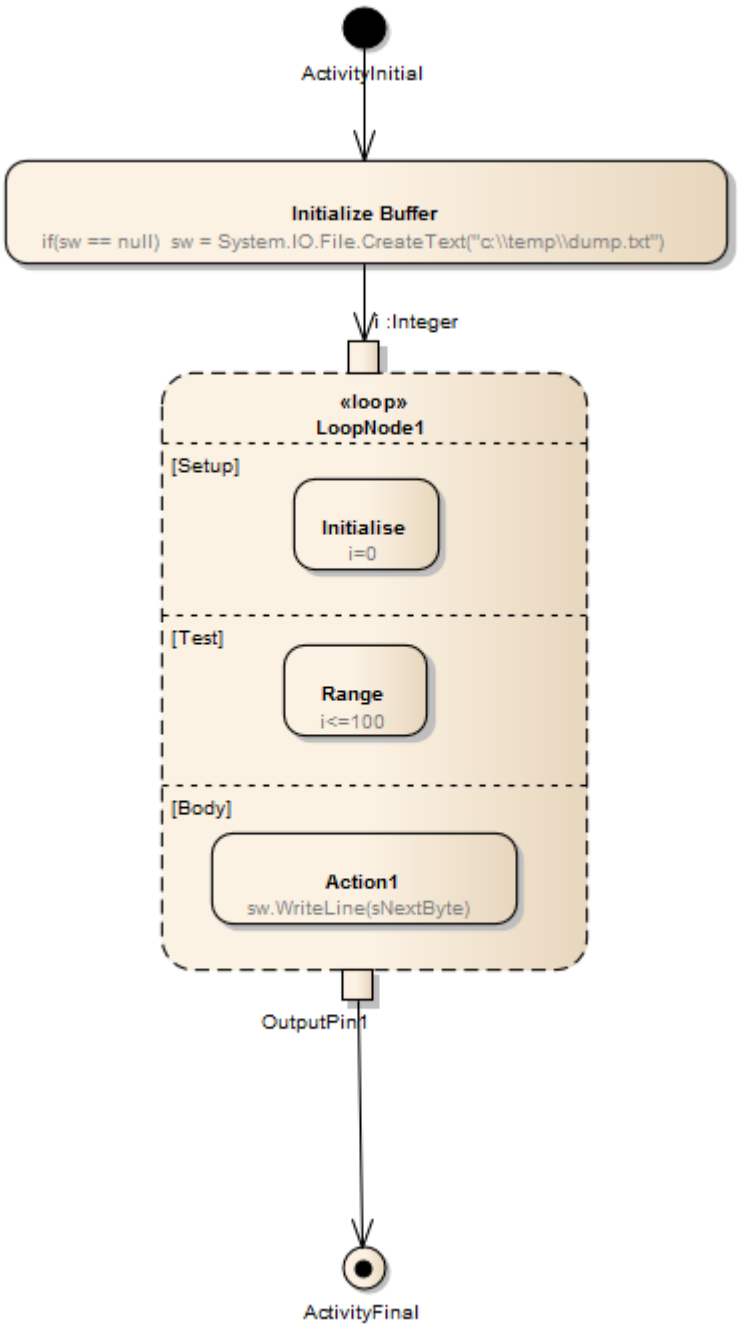
Create a Loop Node

A Loop Node is depicted on an Activity diagram in this way:



You define the Loop nodes by dragging Action elements from the Diagram Toolbox page into the 'Setup', 'Test' and 'Body' partitions. The 'Body' partition can contain several

Actions, which can be linked and organized into the required structure. The elements are aligned on the top left of the partition, so that resizing the node maintains the organization of the structure within and between the partitions. If you try to shrink the node below the structure size, the node automatically defaults to the 'best fit' size.



Ste	Action

p	
1	From the Activity page of the Diagram Toolbox, drag a Structured Activity icon onto the Activity diagram. A short menu displays.
2	Select the 'Loop Node' option. The Loop Node displays on the diagram, with the element 'Properties' dialog (if the dialog does not display, double-click on the element).
3	Complete as many of the common element Properties fields as required, then close the 'Properties' dialog.
4	Display the Properties window ('Start > Application > Design > Properties') for the Loop Node, and click on the 'Loop' tab. Set these checkboxes as required: • 'Must Isolate' - defines concurrency: if selected, no object within the node can be used outside it; the objects are isolated from parallel use • 'Tested First' - defines the loop type; select for a For / While loop, deselect for a Repeat Until loop
5	For each of these fields, click on the ... or Add button as appropriate, to display the 'Select Pins' dialog and select an Action Pin:

	• Decider (an Output Pin within the 'Test' partition, the value of which is examined after execution of the Test to determine whether to execute the loop Body) • Loop Variable Input • Loop Variable • Body Output and • Result The 'Select Pins' dialog lists only Input Pins for the 'Loop Variable Input' field and only Output Pins for the other fields. If the required Action Pin does not already exist, you can click on the Add New button on the dialog to automatically create the Input pin or an Output pin for the node.
6	In the 'Nodes' panel, click on one of the 'Setup', 'Test' or 'Body' radio buttons to list the Actions and Activities contained in the corresponding partition of the Loop Node. An element must be completely below the top edge of a partition to be listed for that partition - if it overlaps with the partition above in any way, it is treated as being part of that partition.
7	Click on the OK button to save the properties of the Loop Node.

8	Right-click on the Node in the diagram and select the 'Features | Interaction Points' option. The Features window displays, showing the 'Interaction Points' tab. Select the checkbox against each Interaction Point. The Action pins should now be visible in the diagram, attached to the Node.

Notes

- You can check on the exact location of an existing Action Pin by right-clicking on the pin name in the Loop Node's Properties window and selecting the 'Find in Project Browser' option; the location of the Action Pin in the Browser window is expanded and highlighted

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.479) states:

A LoopNode is a StructuredActivityNode that represents an iterative loop. A LoopNode consists of a setupPart, a test and a bodyPart, which identify subsets of the ExecutableNodes contained in the LoopNode. Any ExecutableNode in the LoopNode must be included in the setupPart, test or bodyPart for the LoopNode. When a LoopNode begins execution, any InitialNodes within it are

immediately enabled. An ExecutableNode contained in the LoopNode, however, can only become enabled when the setupPart, test or bodyPart section that contains it is executed.

When a section is executed, any ExecutableNode in the section that has no mandatory input data and no incoming ControlFlow with a source in the same section is enabled and receives a single control token. Execution then proceeds according to the usual semantics of Activities, except that any offers made to an ExecutableNode in a section that is not executing are not immediately delivered but remain pending. The target ExecutableNode may accept any pending offers if it eventually executes as part of a later execution of the section that contains it.

Conditional Node

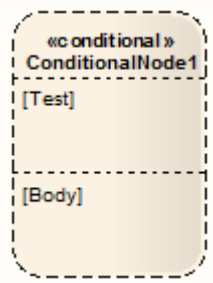
A Conditional Structured Activity Node is the modeling equivalent of an 'If-Then-Else' programming construct. At its simplest, it consists of a Clause containing:

- A Test partition that evaluates a condition, and
- A Body partition that performs one or more actions if the Test condition is satisfied

You can have more than one Clause, so that if the Test condition is not satisfied its Body is ignored and processing moves to the next Clause and evaluates another Test condition.

Each Clause has a 'Decider' ActionPin to hold the result of the Test, and a 'Body Output' ActionPin to hold the result of the Body's actions (if executed). The Conditional Node itself has a result ActionPin that makes available the overall result of the Node (the output of the first Body to be executed).

The representation of a Conditional Node on an Activity diagram resembles this:



You define Conditional Nodes by dragging other Activity diagram elements from the Toolbox page into the appropriate partition of the element, and linking and organizing the structure as required. The elements are

aligned on the top left of the partition, so that resizing the node maintains the organization of the structure within and between the partitions. If you try to shrink the node below the structure size, the node automatically defaults to the 'best fit' size.

When you create a Conditional Node, the 'Properties' dialog displays. Much of this you can complete as for any other element. However, for the Conditional Node you also display the Properties window, which has an additional 'Condition' tab.

On this tab, in the 'Result' panel, add an Action Pin to hold the result for the node, clicking on the Add button to display the 'Select Pins' dialog.

A Conditional Node automatically contains one Clause containing a Test partition and a Body partition, and a Decider Pin and Body Output Pin. You can add further Clauses as required. For each Clause you add an Action Pin for the Decider and for the Body Output. Click on the Save button to save the Clause definition.

The 'Select Pin' dialog reveals only Output pins as appropriate to the context. If the required Action Pin does not already exist, you can click on the Add New button on the dialog to automatically create an Output pin under the appropriate parent node.

For the 'Result' and 'Body Output' entries, you can check on the exact location of each Action Pin by right-clicking on the entry and selecting the 'Find in Project Browser' option.

The 'Nodes' panel, by default, lists the Actions and Activities contained in the Test partition. Click on the 'Body'

radio button to list the elements contained in the Body partition. An element must be completely contained in the Body partition to be listed there - if it overlaps with the Test partition in any way, it is treated as being part of the Test partition.

Add or Remove Clauses

To add another Clause, click on the Add button underneath the 'Clause(s)' list. This inserts a new Clause in the list, and identifies which is the preceding (Predecessor) Clause and (if appropriate) which is the following (Successor) Clause. The remaining fields in the 'Clause(s)' panel are cleared so that you can add Decider and Body Output Action Pins. New 'Test' and 'Body' partitions are immediately added to the element on the diagram, and you can populate these partitions with Activity elements, which are then identified in the 'Nodes' panel.

To remove a Clause, highlight it in the list and click on the Delete button. This immediately removes the Clause's corresponding partitions from the diagram, along with all their contained Activity elements. Removing a Clause from between two other Clauses adjusts the numerical order; for example, if Clause 2 is removed from between Clause 1 and Clause 3, Clause 3 is renamed as Clause 2, and any further Clauses are also moved up one place.

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.478) states:

A ConditionalNode is a StructuredActivityNode that chooses one among some number of alternative collections of ExecutableNodes to execute. A ConditionalNode consists of one or more Clauses, each of which represents a single branch of the conditional. A Clause consists of a test section and a body section, which identify disjoint subsets of the ExecutableNodes contained in the ConditionalNode. Any ExecutableNode in the ConditionalNode must be included in the test section or body section of exactly one Clause.

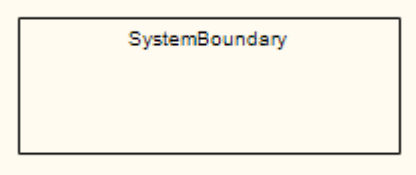
Synch

Description

A Synch state is useful for indicating that concurrent paths of a StateMachine are synchronized. They are used to split and rejoin periods of parallel processing. After bringing the paths to a synch state, the emerging transition indicates unison.

Toolbox icon

System Boundary



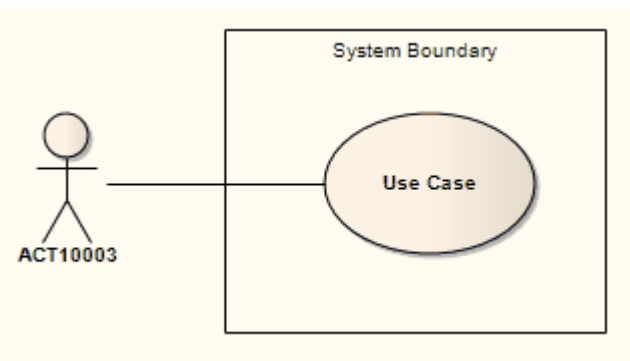
Description

A System Boundary element is a non-UML element used to define conceptual boundaries. You can use System Boundaries to help group logically related elements (from a visual perspective, not as part of the UML model).

In the OMG Unified Modeling Language specification, (v2.5.1), System Boundaries are described in the sections on Use Cases, because the System Boundary is often used to indicate the application of a Use Case to another entity. In this context, the System Boundary:

- Encloses the Use Case, and
- Is associated with a classifier such as a Class, Component or Subsystem (Actor) through the 'Select <Item>' dialog

By associating the System Boundary - and not the Use Case - with the classifier, the classifier is linked to the Use Case as a user, but not as an owner.



You can also define a Use Case as the classifier of a System Boundary element, to link the elements enclosed in the System Boundary (such as parts of an Activity diagram) to their representation in a logical Use Case.

The element properties for a System Boundary element comprise the name, the border style, and the number of horizontal or vertical swim lanes. You can also change the overall shape of the System Boundary, which includes an option to add dividing lines to the element other than by using the swimlanes, and you can make the element fully opaque, fully transparent or various degrees of opacity in between.

A System Boundary element can be marked as 'Selectable', using the element's context menu. When the element is not selectable, you can click on the other elements within the System Boundary space without activating or selecting the System Boundary itself.

Toolbox icon

Notes

- A System Boundary is the basis for the Image element, which enables you to add icons or backgrounds to a diagram, automatically displaying the Image Manager window from which to select the appropriate image
- A System Boundary is not the same as the Boundary

element used to capture user interactions in, for example, Analysis diagrams

OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, p.641) states:

A subject for a set of UseCases (sometimes called a system boundary) may be shown as a rectangle with its name in the top-left corner, with the UseCase ellipses visually located inside this rectangle. The same modeled UseCase may be visually depicted as separate ellipses within multiple subject rectangles.

System Boundary Properties

The System Boundary element has a small set of properties that are mainly concerned with the appearance of the element. You can also apply other element control options such as default appearance, locking the element and applying an image to the element.

The element must be set to 'Selectable' in order for you to be able to change its properties.

Access

Context Menu	Right-click on Boundary element | Properties | Properties
Keyboard Shortcuts	Alt+Enter
Other	In the Properties window (Ctrl+2), click on the 'Boundary' tab

Set System Boundary Properties

Option	Action

Name	(Optional) Type a name for the element. (This field does not appear on the 'Boundary' tab of the Properties window).
Shape	Click on the drop-down arrow and select from these options: • 'Rectangle' - if you have previously switched from the default rectangular border with sharp corners, return to that default • 'Rounded Rectangle' - set the shape to a rectangle with rounded corners • 'Ellipse' - set the shape to a circle or oval to accommodate the enclosed elements • 'User Defined - Orthogonal' - enable setting drag-points on the border to create a custom orthogonal (block) shape (see the *Customize System Boundary - Orthagonal/Freeform* section) • 'User Defined - Freeform' - enable setting drag-points on the border to create a custom freeform shape (see the *Customize System Boundary - Orthagonal/Freeform* section) • 'User Defined - Custom Grid' - enable adding vertical and horizontal lines to the Boundary or to a cell of the

	Boundary (see the *Customize System Boundary - Custom Grid* section); this style disables the use of Swimlanes in the Boundary element
Style	Click on the drop-down arrow and select from these options: • Solid - a solid line border with the system default element fill color • Dotted - a dotted line border with no element fill color • Dashed - a broken line border with no element fill color • Solid - No Fill - a solid line border with no element fill color (this setting is blocks the fill opacity of the element; see the *Fill Opacity* section)
Horizontal Swim Lanes	Type in the number of horizontal segments you want to divide the element into, to group the elements in the System Boundary in horizontal contexts (for example, Client, Application and Database tiers could be represented in swim lanes). The field defaults to 1. Leave it on this setting if you intend to use the custom grid. The swim lanes are equal divisions of the

	System Boundary - you cannot change their relative heights.
Vertical Swim Lanes	Type in the number of vertical segments you want to divide the element into, to group the elements in the System Boundary in vertical contexts (for example, Start, Progress and Terminate segments). The field defaults to '1'. Leave it on this setting if you intend to use the custom grid. The swim lanes are equal divisions of the System Boundary - you cannot change their relative widths.
Instance Classifier	(On the Properties window.) Shows the classifier for the Boundary. If one has not been specified, or you want to change it, click on the ... icon and locate and select the required Classifier using the 'Select Element' dialog. The Features window also displays.
Multiplicity	(On the Properties window.) Shows the number of instances of the element that can exist in a set; if there is no figure set or you want to change it, click on the drop-down arrow and select an

	appropriate value.

Customize System Boundary - Orthagonal/Freeform

When you have selected one of the 'User Defined' options in the 'Shape' field, you can add way-points to the sides of the System Boundary, to drag in a direction to create a new shape. This helps you to create irregular shapes that enclose dispersed elements that cannot be captured in a simple rectangle or ellipse.

The 'Orthogonal' variant helps you to create shapes with vertical and horizontal lines, whilst the 'Freeform' variant helps you to create diagonal lines.

To set a way-point on an edge:

- Press Shift+click, Ctrl+click or Ctrl+Q on the appropriate point on the edge

To clear a way-point:

- Press Shift+click, Ctrl+click or Ctrl+Q on it

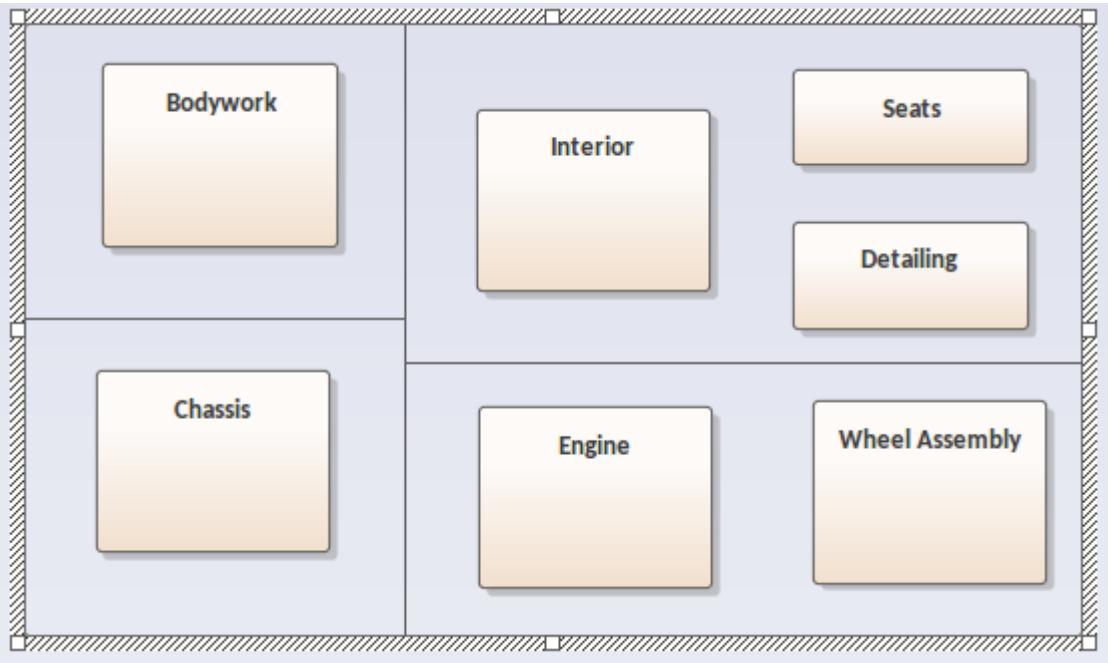
To move a way-point:

- Click on the boundary to display all way-points, position the cursor on the required way-point so that it changes to green, and then click and drag the way-point; when you move the cursor onto the border, the way-points on either side of the cursor turn green and you can click and drag that segment of the border

If you create a 'Freeform' shape and then change the 'Shape' setting to 'User Defined - Orthogonal', the system converts all diagonal lines to vertical or horizontal lines. You might then have to adjust the shape so that it has fewer lines. If you try to drag an orthogonal way-point in a diagonal direction, the horizontal and vertical lines adjust to maintain a right-angle at the cursor position.

Customize System Boundary - Custom Grid

When you select this option, you are able to draw horizontal and vertical lines within the Boundary, using the same style as you have set for the Boundary borders. Note that this option is an alternative to using swimlanes - you don't use both options together, and the swimlane options are disabled.



The custom lines link existing lines, so if you have a Boundary with no divisions you can draw a line between the

borders of the element (as for the vertical line in the illustration), but if you have already added cells you can draw a line between the vertical or horizontal borders of a cell (as for the two horizontal lines in the illustration). This helps you to add further non-regular groupings of elements within the Boundary, to create an effect such as appears in a Business Model Canvas. Just as you can drag elements between swimlanes, you can also drag elements between custom cells.

To add custom cells to the Boundary element:

1. Select the 'User Defined - Custom Grid' option.
2. Return to the diagram.
3. Press the Shift key and hold the left mouse button down as you drag the cursor across or up the Boundary element. A dotted guideline displays to show where the line will be created.
4. When the line is roughly where you want it, release the mouse button. The line becomes solid.

You can move the line by hovering the cursor over it so that a small double-headed arrow displays, then holding the left mouse button down and moving the mouse as required. Abutting lines will extend or retract to maintain the join.

To delete a line, right-click on it and select the 'Delete Region' context menu option. This is very similar to merging cells in a table. The system highlights the region bounded by the line and prompts you to confirm the deletion. Click on the Yes button. Any abutting lines will extend to the next perpendicular line.

If you resize the Boundary element, all the cells are resized proportionately. However, if you simply want to create more room for adding more cells and do not necessarily want to change the size of existing cells internal to the Boundary, press and hold the Shift key while you resize the element. Only those lines that attach to the border are extended; the internal cells maintain their size.

Fill Opacity

Whilst a Boundary usually encloses a number of other elements, there might be reasons for hiding those elements as well as times to fully show them, or perhaps just indicate that they are there, depending on the immediate purpose of the diagram. You can apply these nuances in the display of elements behind and covered or overlapped by a Boundary by changing the opacity of the element.

Before setting the opacity, check that the element has a fill color and that the 'Style' option in the Boundary 'Properties' dialog or Properties window 'Boundary' tab is set to a value other than 'Solid - No Fill'.

You set the opacity using an icon from either of these two pop-up element toolbars:

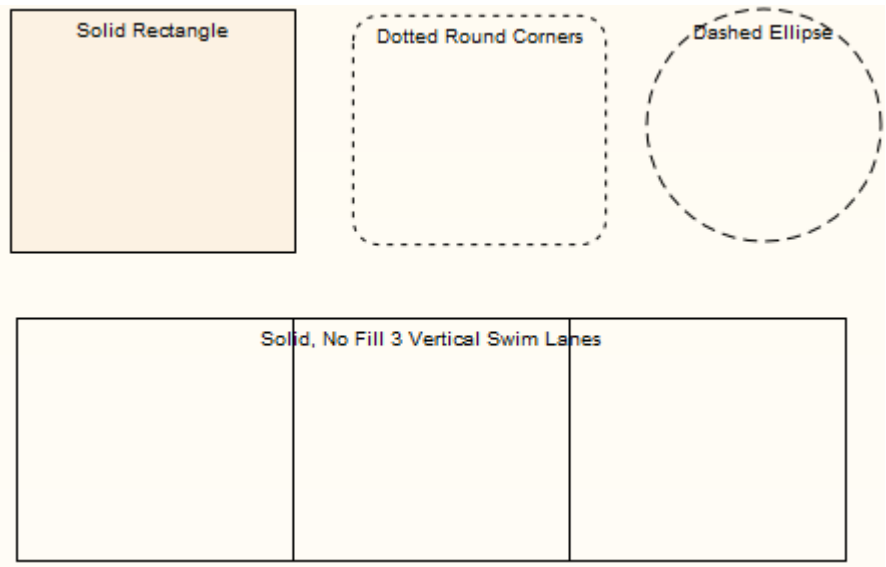
- Click on the Boundary element and on the icon:

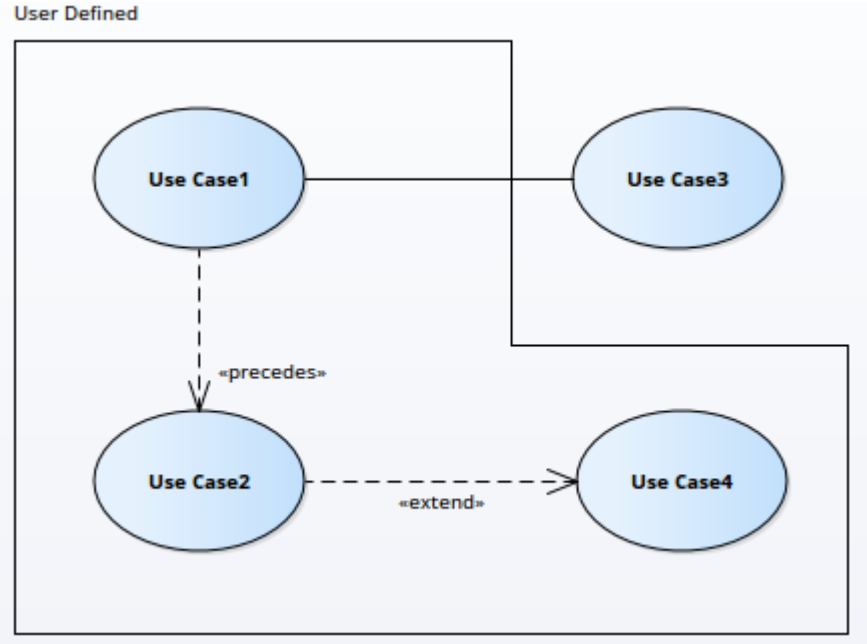
- Right-click on the Boundary element and look above the context menu:

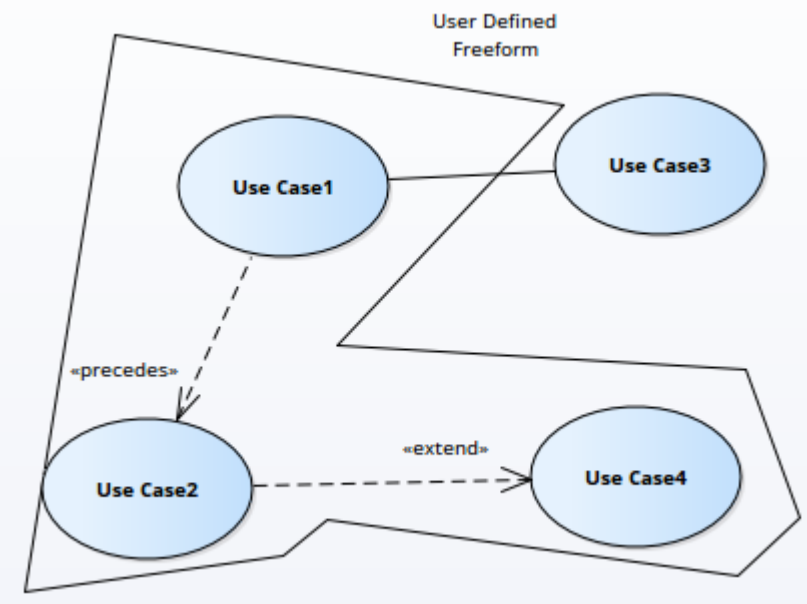
Click on the icon and select:

- 100% for total opacity, where the elements behind and overlapping or covered by the Boundary are hidden (you could right-click on individual elements and select the 'Z-Order | Bring to Top' option to expose those elements only)
- 0% for no opacity, where the fill color is not applied and anything behind the Boundary is fully visible
- 75%, 50% or 25% to set the appropriate degree of opacity and make the covered elements visible but over-shaded

Example Shapes







Notes

- Diagram-specific options for Boundaries (shape, border style, swimlane count) are locked if the diagram is locked or if the user does not have access permissions to update diagrams; the 'Name' field can be updated
- The Boundary element 'Name' field is locked if the element is locked or the user does not have access

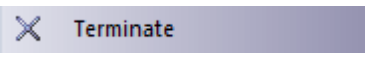
permissions to update elements; the other fields can be changed

Terminate

Description

The Terminate pseudostate indicates that upon entry of its pseudostate, the StateMachine's execution ends.

Toolbox icon



Trigger

Description

A Trigger indicates an event that initiates an action (and might arise from completion of a previous action). You initially define a Trigger in one of four ways:

- As a property of a Transition relationship
- As a property of an Accept Event Action (on the 'Triggers' tab of the element 'Properties' dialog)
- As an event in a StateMachine Table
- Directly, as a Trigger element, through the 'New Element' dialog or Diagram Toolbox ('State Additional' page)

When you save the Trigger, it is added to the list of elements for the parent Package in the Browser window. You can then click on it and press Ctrl+2 to display the Properties window for the element, to view and, if required, edit its properties as an element rather than as a property itself. Triggers created as events remain as Event elements, whilst Triggers created in other ways are Trigger elements, with a 'Trigger' tab in the Properties window.

Field	Action
Type	If necessary, edit the type of trigger: • Call - specifies that the event is a

	CallEvent, which sends a message to the associated object by invoking an operation • Change - specifies that the event is a ChangeEvent, which indicates that the transition is the result of a change in value of an attribute • Signal - specifies that the event is a SignalEvent, which corresponds to the receipt of an asynchronous signal instance • Time - corresponds to a TimeEvent; which specifies a moment in time
Specification	Either type in the event instigating the Trigger, or click on the ... button and select the event (depending on the Type value).
Ports	Click on the Add button and select the appropriate Port from the 'Select Port' dialog. • To create new Ports using the 'Select Port' dialog, the Trigger should be created as child of a Class or Component element • To add several Ports at once, press Ctrl as you select each Port • To check the exact location of a Port,

	right-click on the Port name and select the 'Find in Project Browser' option

Notes

- You can also drag an existing Trigger element onto another diagram, although there are limited uses for the element in that context
- This element is not the same as a Trigger Operation, which is an operation automatically executed as a result of the modification of data in a database

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.300) states:

Events may cause execution of behavior (e.g. the execution of the effect activity of a transition in a state machine). A trigger specifies the event that may trigger a behavior execution as well as any constraints on the event to filter out events not of interest.

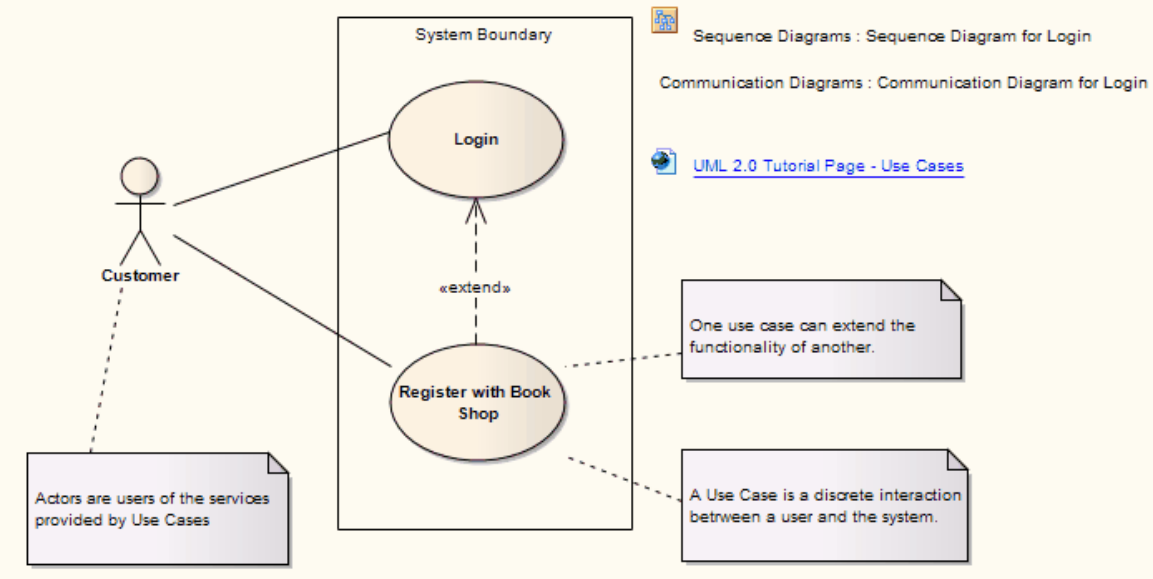
Use Case

Description

A Use Case is a UML modeling element that describes how a user of the proposed system interacts with the system to perform a discrete unit of work. It describes and signifies a single interaction over time that has meaning for the end user (person, machine or other system), and is required to leave the system in a complete state: the interaction either completed or rolled back to the initial state. A Use Case:

- Typically has requirements and constraints that describe the essential features and rules under which it operates
- Can have an associated Sequence diagram illustrating behavior over time; who does what to whom, and when
- Typically has scenarios associated with it that describe the workflow over time that produces the end result; alternative workflows (for example, to capture exceptions) are also enabled

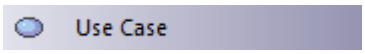
Example Use Case diagram



If extending a Use Case, you can specify the points of extension with Use Case Extension Points. To display the attributes, operations or constraints of a Use Case on a diagram, use Rectangle Notation.

Enterprise Architect also provides two stereotyped Use Cases: the Test Case and the Business Use Case.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.649) states:

A UseCase specifies a set of actions performed by its subjects, which yields an observable result that is of value for one or more Actors or other stakeholders of each subject.

Use Case Extension Points

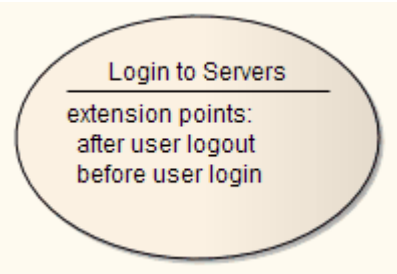
The behavior defined for a Use Case can add to the behavior of another Use Case; that is, the first Use case extends the second one. This is represented on the model by an Extend connector from the first Use Case to the second. If the extended behavior takes effect at a specific point, you can define that point as an extension point on the extended Use Case. The name (description) text of the extension point can be as informal or precise as is appropriate to define the point in behavior at which the extension applies. A Use Case can have more than one extension point, to allow for different source Use Cases to extend this target Use Case, or for changes in where the extending behavior applies depending on the constraints defined for the Extend connector. The connector also identifies which extension point is in effect.

Access

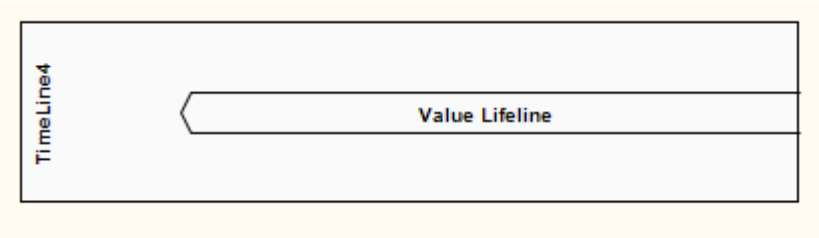
Context Menu	On diagram | Right-click on extended Use Case element | Advanced | Edit Extension Points

Add extension points to a Use Case

Field/Button	Action
Defined Extension Points	Lists the extension points currently defined for the selected Use Case.
Add	Click on this button to display a prompt for the name of a new extension point. Type the name and click on the OK button. The name is added to the Defined Extension Points list.
Edit	Click on an existing extension point and click on this button to display a prompt for changes to the name of the selected extension point. Overtype the name and click on the OK button. The name is updated in the Defined Extension Points list.
Remove	Click on an existing extension point and click on this button to immediately remove the name from the Defined Extension Points list.
OK	Click on this button to save all changes to the extension points, and to close the dialog. The extension points you have defined

	are represented on the Use Case element in the diagram as shown. 

Value Lifeline

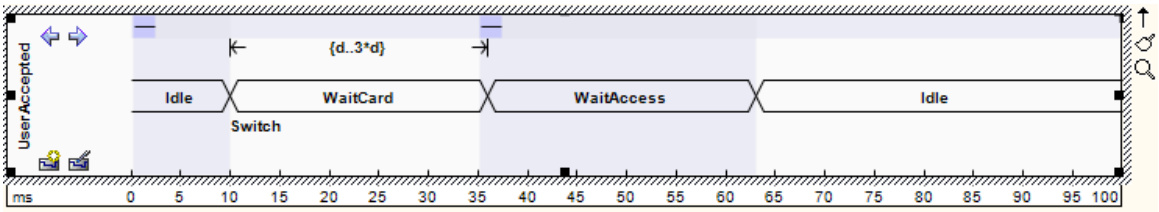


Description

A Lifeline is the path an object takes across a measure of time, indicated by the x-axis. There are two sorts: Value Lifelines (defined here) and State Lifelines, both used in Timing diagrams.

A Value Lifeline shows the Lifeline's state across the diagram, with parallel lines indicating a steady state. A cross between the lines indicates a transition or change in state.

This is an example of a Value Lifeline:



See the OMG Unified Modeling Language specification, (v2.5.1, Figure 14.30, p.520.)

Transition point properties

A Value Lifeline consists of a set of transition points. Each transition point can be defined with these properties:

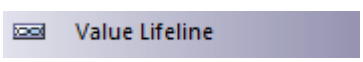
Property	Description
At time	Specifies the starting time for a change of state.
Transition to	Indicates the state to which the Lifeline is to change.
Event	Describes the occurring event.
Timing constraints	Refers to the time taken for a state to change within a Lifeline, or the time taken to transmit a message.
Timing observations	Provides information on the time of a state change or sent message.
Duration constraints	Pertains to a Lifeline's period at a particular state. The constraint could be instigated by a change of state within a Lifeline, or that Lifeline's receipt of a message.
Duration observations	Indicates the interval of a Lifeline at a particular state, begun from a change in state or message receipt.

Example properties

In the example diagram, the 10ms transition point has these properties:

Property	Text
At Time	10ms
Transition to	Waitcard
Event	Switch
Timing constraints	–
Timing observations	–
Duration constraints	d..3*d
Duration observations	–

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.604) states:

Shows the value of the connectable element as a function of time. Value is explicitly denoted as text. Crossing reflects the event where the value changed.

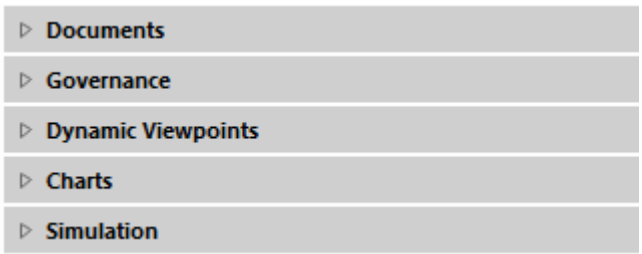
Structural Diagram Elements

This section provides detailed descriptions of the elements commonly used when modeling with UML Structural Diagrams in Enterprise Architect.

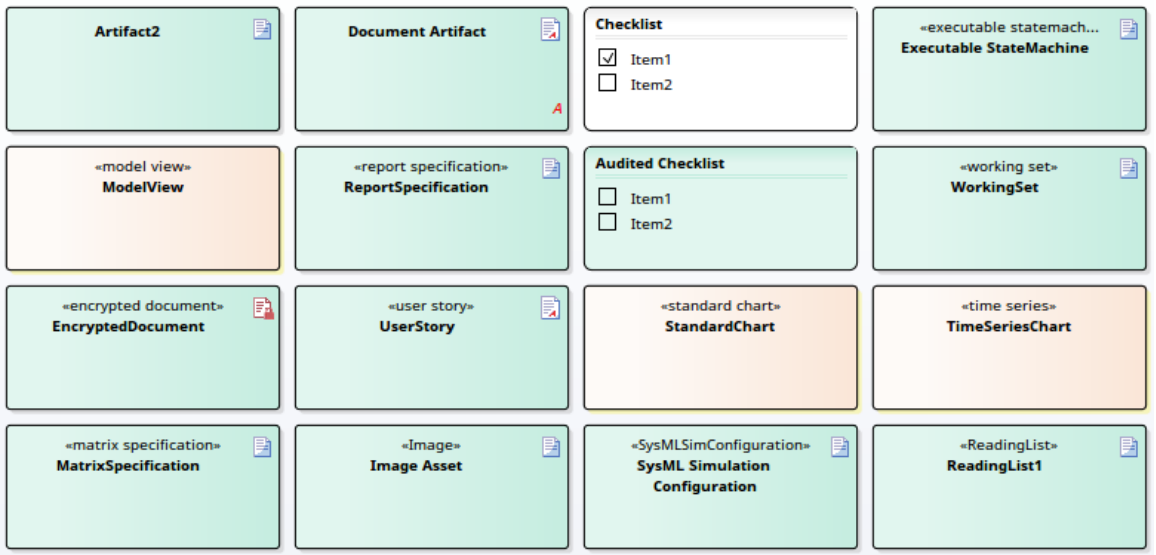
Artifact

An Artifact is any physical piece of information used or produced by a system. In Enterprise Architect an Artifact is represented by an Artifact element, which can have one of a number of stereotypes to tailor it to a specific purpose, including internal operations and structures within the model, as indicated in the examples. Artifacts can have associated properties or operations, and can be instantiated or associated with other Artifacts according to the object they represent.

You can create an Artifact element by dragging one of the 'Artifact' icons from one of several Diagram Toolbox pages according to type. The 'Common Elements' page of the Toolbox has a generic 'Artifact' icon that - when you drag it onto a diagram - offers a choice of types of Artifact to create.

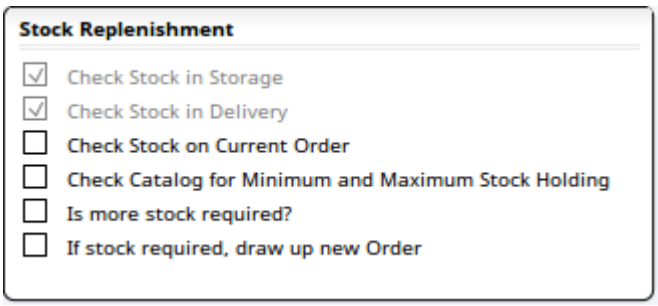


Types of Artifact



Type	Description
(Base) Artifact	A Base Artifact defines the external artifacts used in a process and the internal artifacts generated in the process, such as model files, source files, database tables, development deliverables or support documents. The files represented by the Artifact are listed on the 'Files' tab of the Properties window for the element. To open the files represented by the Artifact, click on the element on the diagram and press Ctrl+E. Each file is opened either on a separate tab in the Diagram View workspace (if the file can be opened within Enterprise Architect) or in the default Windows viewer/editor for the file type (if the file cannot be opened within Enterprise Architect). Files can also be launched individually

	from the 'Files' tab (opening in the Windows default editor), as for elements of any other type that have associated files.
Document Artifact	A Document Artifact is an Artifact having a stereotype of «document». You create the Document Artifact using the Artifacts, Component, Documentation or Deployment pages of the Diagram Toolbox, and associate it with an RTF document or CSV file. Double-click on the element to display the Linked Document Editor. When you have created the Linked Document, the Document Artifact element on the diagram shows an 'A' symbol in the bottom right corner.
Checklist Artifact	A Checklist Artifact provides the facility for generating a list of items, each with a checkbox, to be used as a checklist. You can set each checkbox to default to being selected or unselected, and set all selected options to be grayed out or struck through, so that the unselected options are more prominent. The end users can

	then work with the element to tick off items that have been obtained or activities that have been completed. A version of this Artifact - the Requirements Checklist - is available in the 'Extended Requirements' Toolbox page. This automatically contains ten characteristics of a good Requirement, for a Requirement author to tick off as they check that their Requirement has been set up to show those characteristics. 
Audited Checklist Artifact	This Artifact is identical to the Checklist Artifact, except that it has an associated 'Audit Log' page that identifies any changes made to the checklist, when those changes were made, who made them and what the changes were. This is a very useful Project Management tool, adding tracking and accountability to the use of checklists.
Encrypted Document	An Encrypted Document Artifact is used to create and hold a Linked Document

Artifact	that is automatically encrypted, and that cannot be opened and automatically decrypted within Enterprise Architect without entering a password. You can therefore use the generated Artifact element to record sensitive information, which you protect from general access by assigning a password. When you drag the 'Encrypted Document' icon onto the diagram from the 'Documents' page of the Toolbox, a prompt displays to type in a password. When you enter this password, you can create the Linked Document. Thereafter, when any user attempts to open the document, the same password prompt displays. If the user does not provide the password that you originally specified, the document will not open, whilst if the correct password is provided the document is decrypted and opened, and the text can be viewed and edited. You cannot change the password, nor can you delete the Linked Document from the Artifact (although you can delete the Artifact itself). Other facilities that display 'normal' Linked Documents - such as the Document window or report generator -

	will ignore an encrypted document. The Encrypted Document is indicated in the Toolbox page, diagram and Browser window by a red 'document' icon - .
User Story	A User Story Artifact provides a means of documenting a business Use Case in the context of Agile methodologies such as Extreme Programming (XP). In the Linked Document, you define the functions a business system must provide; it captures the 'who', 'what' and 'why' of a requirement in a simple, concise format. The User Story Artifact behaves as a Document Artifact, prompting you to select a Linked Document template to base the document on.
Working Set Artifact	A Working Set Artifact defines a Working Set that opens various windows, diagrams and views, recreating a work environment that you frequently use. • To create or modify the Working Set, right-click on the element and select the 'Edit Working Set' option • To execute the Working Set to open the defined windows and views and execute any commands, double-click

	on the element
Custom Table Artifact	A Custom Table Artifact generates a diagram object that displays custom data in a grid format similar to a spreadsheet, providing extra 'non-modeled' information on elements, diagrams or project management exactly where it is applicable.
Standard Chart	A Standard Chart Artifact provides the facilities for generating a Pie Chart or Bar Chart on an aspect of the data in your model. It adds three 'Chart Details' tabs to the standard tabs of the element 'Properties' dialog. After you have added the element to your diagram, double-click on it. The element 'Properties' dialog automatically opens at the 'Chart Details - Source' tab. Define the chart type and data source, then go on to define any filters you want to apply, and how the chart should display. Once you have defined the chart, it automatically displays with the latest information whenever you open the parent diagram.
Time Series	A Time Series Chart Artifact provides the

Chart	facilities for generating a linear graph of a model property over time. After you have added the element to your diagram, double-click on it. The element 'Properties' dialog automatically opens at the 'Chart Details - Source' tab. Define the Package from which the data is to be extracted, and the time interval over which the data is to be sampled. Then go on to define the appearance of the chart. Once you have defined the chart, it automatically displays with the latest information whenever you open the parent diagram.
Model View	A Model View Artifact provides the facilities for generating a tabular Model View Chart on a segment of the data in your model, extracted using a custom SQL search. After you have added the element to your diagram, double-click on it. The element 'Properties' dialog automatically opens at the 'Chart Details - Source' tab. Define the SQL Search to extract and tabulate the information. Once you have defined the chart, it automatically displays with the latest information whenever you open the

	parent diagram.
Report Specification	A Report Specification Artifact encapsulates a report definition. When you have created the element on the diagram, you double-click on it to display the 'Generate Documentation' dialog, on which you enter the report parameters and generate the report. After you create the Report Specification, each time you double-click on the Artifact element the 'Generate Documentation' dialog again displays with the same report parameters. You can continue to generate the same report, or alter the parameters if necessary. If you change the parameters, they are re-presented until such time as you change them again,
Matrix Specification	A Matrix Specification Artifact encapsulates a Relationship Matrix Profile definition. When you have created the element on the diagram, you double-click on it to display the 'Matrix Specification' dialog, in which you create the Profile definition. The Profile takes the name of the element. The profile defined in the Artifact is independent of

	the Package that contains the Artifact element, and therefore could specify source and target Packages other than parent Package. After you create the Profile definition, each time you double-click on the Artifact element the Relationship Matrix displays with the Profile applied. To edit the Profile, right-click on the Artifact and select the 'Documentation | Edit Matrix Profile' option.
Executable StateMachine	An Executable StateMachine Artifact is the vehicle through which you can generate, build (compile) and execute - via simulation - code for a StateMachine or complex of StateMachines. Each StateMachine is the child of a Class element; when you drag the Class from the Browser window onto the Artifact element, it is pasted inside the Artifact as a Part. You can paste several Classes - and, therefore, Parts - into a single Artifact. Having set up the Executable StateMachine Artifact, you use simple context menu options on the Artifact to perform the code generation, build and execution operations on all

	StateMachines bound within the Artifact.
Business Process Simulation	The Business Process Simulation Artifact appears in the 'Simulation' page of the Toolbox when the BPSim Simulation Engine has been installed and registered on your system. You use the Artifact as a container for - and an access point to - a Business Process Simulation Configuration, which defines what business process model to simulate and what parameters to apply during the simulation. Right-click on the element and select the 'Configure BPSim' context menu option. The Configure BPSim window displays.
BPSim Result Chart	The BPSim Result Chart and BPSim Custom Result Chart Artifacts appear in the 'Charts' page of the Toolbox when the BPSim Simulation Engine has been installed and registered on your system. These Artifacts generate Charts that reflect selected results from BPSim Simulations: • BPSim Result Chart - generates a Chart that reflects selected results from a series of standard BPSim Simulations • BPSim Custom Result Chart -

	generates a Chart that reflects results from a series of customized BPSim Simulations As for other Chart Artifacts, both BPSim Chart types can be quickly configured to display the Simulation results in variations of a Line Chart, two-dimensional Bar Chart or 3-dimensional Bar Chart. The results that both Artifacts operate on are captured in Results Artifacts that are automatically generated during a Business Process Simulation.
SysMLSim Configuration	This Artifact provides access to the Configure SysML Simulation Window, and contains a specific SysML simulation configuration. To access the Configure SysML Simulation Window, double-click on the Artifact or right-click and select the 'Show SysML Configuration' option.
Image Asset	Image Assets are model elements that are used to store images in the model. You can create them by dragging the 'Image Asset' icon onto a diagram and choosing an image file, or by dragging an image file from your file system straight onto a diagram. Enterprise Architect creates an

	Image Asset Artifact, then stores the image from the file in the Artifact as the 'owned image'. The element can display on the diagram either as the image or as a rectangular element; to toggle between them, right-click on the object and click on the 'Show Owned Image' option. When the Artifact is displaying as an image, you can double-click on it to display the element 'Properties' dialog. If the Artifact is displaying in element format, double-clicking on it will open the image in the default external viewer for images. An Image Asset image can be used in model documentation in the same way as for an image from the Image Manager, by inserting a hyperlink to the Image Asset element. As you create the hyperlink, you are prompted to select the type of object to link to; click on 'Element Image' and select the appropriate Image Asset element from the list. If you Ctrl+click on the hyperlink, the image is displayed in the default external viewer for images. When a report is generated, the hyperlinked Image Asset element is rendered using its 'owned image'.

Reading List	The Reading List Artifact provides a list of elements that contain information of particular significance to a task or process. The intention is for the information from each element to be displayed in the Document window, in the sequence in which the elements are organized in the Reading List. The Document window provides 'Next' and 'Previous' options to move through the elements.

Create File Artifacts

A **File Artifact** is an Artifact element that represents and is linked to an external file. You can create the Artifact element on a diagram, from the file itself.

Create the Artifact

Step	Action
1	Locate the file in a file list (such as Windows Explorer) or on your Desktop. Drag the file onto your diagram. A context menu displays.
2	Click on the menu option you need: • 'Hyperlink' - to create a Hyperlink element on the diagram; you can select a sub-option to define whether users, when they double-click on the Hyperlink, will either just display the file content or open it within the appropriate file editor • 'Artifact External' - to create an Artifact element on the diagram; the element 'Properties' dialog displays, in which you enter any element properties you need Save the data you have entered and close the 'Properties' dialog (note that the file name becomes

	the element name) If you double-click on the Artifact the 'Properties' dialog redisplays; click on the 'Files' tab to see the file pathname listed in the 'Files' panel, from which you can launch it in its registered application • 'Artifact Internal' - to immediately create an Artifact element on the diagram with the file name as the element name The file is stored in your model, but is managed by the registered external application for the file type; if you double-click on the Artifact, the file is opened within its external application If the file is changed, you are prompted to update the element within the model - click on the Save button to update the element, or on the Discard button to ignore the changes • 'Insert' - (graphics files) to insert the file into the diagram as a filled Boundary element; double-click on the image to display the Boundary 'Properties' dialog

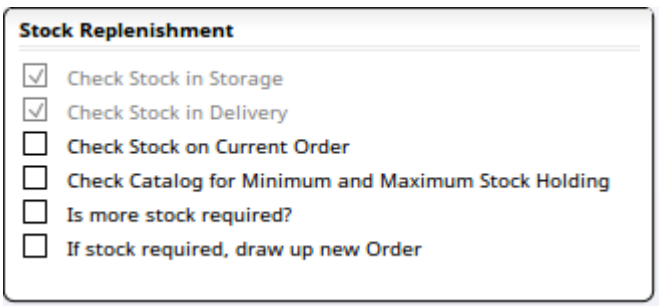
Notes

- This feature is available in all editions of Enterprise Architect
- Hyperlinks to images stored as Artifacts can be created using a hyperlink of type 'Element Image'

- For operating systems other than Windows; as the file source for dragging a file onto a diagram, use: Ribbon > Settings > User Tools > Windows Explorer.

Using the Checklist and Audited Checklist Artifacts

Using the Checklist Artifact, you can create any number of Checklist elements that you and other users can work through to ensure that the required aspects of a task or object have been addressed in completing the task or developing the object. You can select a checkbox against each item to indicate that the point has been addressed, and you can configure the checklist to show selected items in gray or struck out so that unselected items are more obvious.

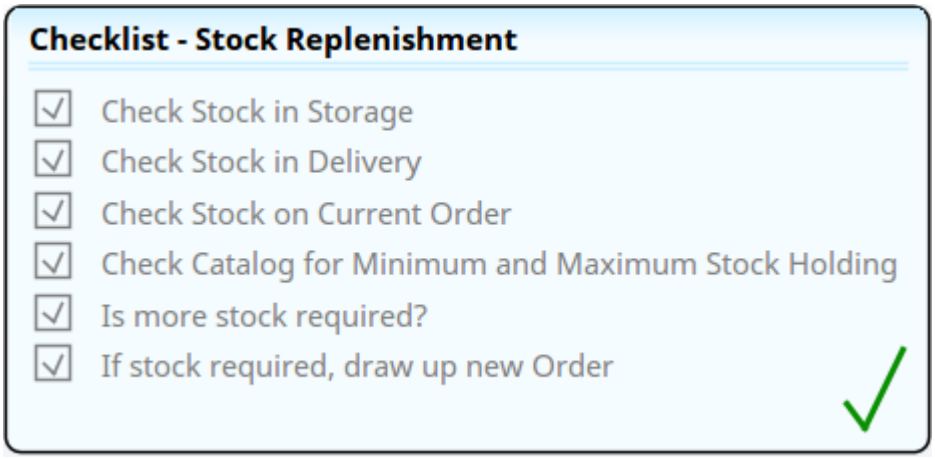


You can also display items in a flat list or a numbered or bulleted list, which helps if you do not necessarily want to check-off the items as you work through the list. If it does become necessary to select such items, you can do so through the 'Checklist Items' dialog.

You can re-use the checkboxes as well - the context menus for individual elements in a diagram, multiple elements in a diagram, and the diagram itself all have options for clearing the checkboxes so that the Checklists are ready to use again to restart a process.

When all the items on a Checklist have been selected, a green arrow displays in the bottom right corner of the

element to indicate that the Checklist has been completed.



The Audited Checklist is an extension of the Checklist Artifact, having an associated 'Audit Records' page. It is a useful tool for Project Management, providing the facility to monitor accountability and track completion of tasks.

You can have a sequence of Checklists that must be completed in order - the first Checklist in the sequence must be complete before any checkboxes on the next Checklist are enabled for selection. This order of completion is established by creating Dependency connectors between the Checklist elements.

- The target Checklist element of the Dependency becomes a prerequisite, and the source Checklist becomes a dependent
- All items on the prerequisite Checklist must be selected before any items on the dependent Checklist can be selected
- A prerequisite Checklist cannot have items deselected whilst the dependent Checklist has items selected
- Clearing a prerequisite Checklist will also clear any dependent Checklists (as they cannot have selected items

whilst the prerequisite Checklist is clear)

- New items can be added to a prerequisite Checklist, but only if they are set to selected before saving (for the first new item a warning message displays; thereafter the new items are automatically set to selected)
- Clearing a Checklist and its dependents will affect all Checklists in the hierarchy even if they are external to the current Package or diagram

Create a Checklist element

You create an empty Checklist element by dragging the 'Checklist' icon onto a diagram from the 'Governance' page of the Diagram Toolbox. The 'Governance' page is always present, at the bottom of every Toolbox.

You can also create a ready-to-use Requirements Checklist with appropriate items already in place, by selecting the 'Requirements Checklist' icon from the 'Extended Requirements' page of the Diagram Toolbox (select one of the search facilities in the Toolbox header and specify 'Requirements').

Set up a Checklist

To populate a Checklist element with items, double-click on the Checklist element in the diagram. The 'Checklist items' dialog displays, with the cursor in the field at the top of the dialog ready to receive the first item name.

Simply type in the item name, then press the Enter key to display and move to the next item entry field.

To the left of each item is a checkbox. If you want the item to default to selected when the element is used, click on the checkbox here.

At the bottom of the dialog are three fields:

- Display - either accept the default of '<None>' for a simple list, or click on the drop down arrow and select for the checklist items to follow on from checkboxes, bullets or numbers
- Check Style - click on the drop-down arrow and select whether to apply no style to items with a selected **checkbox** ('<None>') or to display the 'checked off' items with a line through them ('Strikeout') or in pale gray ('Grayed')

- Spacing - either accept the default of single spacing between items, or click on the drop-down arrow and select a larger spacing between the items; the line spacing within an item remains at single spacing
  You can also change the spacing using the 'Checklist Spacing' option on the element context list

When you have finished setting up the Checklist, click on the Close button.

Edit a Checklist

To change the text, selection status or sequence of items in a Checklist element, double-click on the element to display the 'Checklist Items' dialog and click on the item to change. To:

- Change the text of the item, simply click again and type over the current text (or right-click and select the 'Edit item' option)
- Delete the item, right-click on it and select the 'Delete item' option
- Toggle the status of the checkbox, click on the checkbox (or right-click and select the 'Toggle item check' option)
- Move the item to a different position in the sequence of items, right-click and select either the 'Move item up' or 'Move item down' option
- Change the display format to simple list, checkboxes, bullets or numbers, click on the drop-down arrow in the 'Display' field and select the appropriate option

- Change the selected item style for all items in the checklist, click on the 'Check Style' drop-down arrow and select the appropriate option
- Change the item spacing, click on the drop-down arrow in the 'Spacing' field and select the appropriate line spacing

Click on the Close button to save your changes and close the dialog.

Make a Checklist available to users

To make a Checklist available to other users, create it on a diagram that the users can access directly. Alternatively, save the diagram as a Pattern that the users can draw on as the basis for creating their own Checklist diagrams.

Work with a Checklist

To record, on a Checklist element with a checkbox, the completion of an action or the presence of an object, simply click on the appropriate checkbox to select it.

If you need to clear a checkbox, or several checkboxes, you can either:

- Click on each checkbox again, or
- Right-click on the Checklist element and select the 'Clear Checklist' option (all checkboxes in that element), or
- Select a number of Checklist elements, right-click on one of them and select the 'Clear Checklist' option (all checkboxes in all selected elements), or

- Right-click on the diagram background and select the 'Clear all Checklists' option (all checkboxes in all checklists on the diagram)

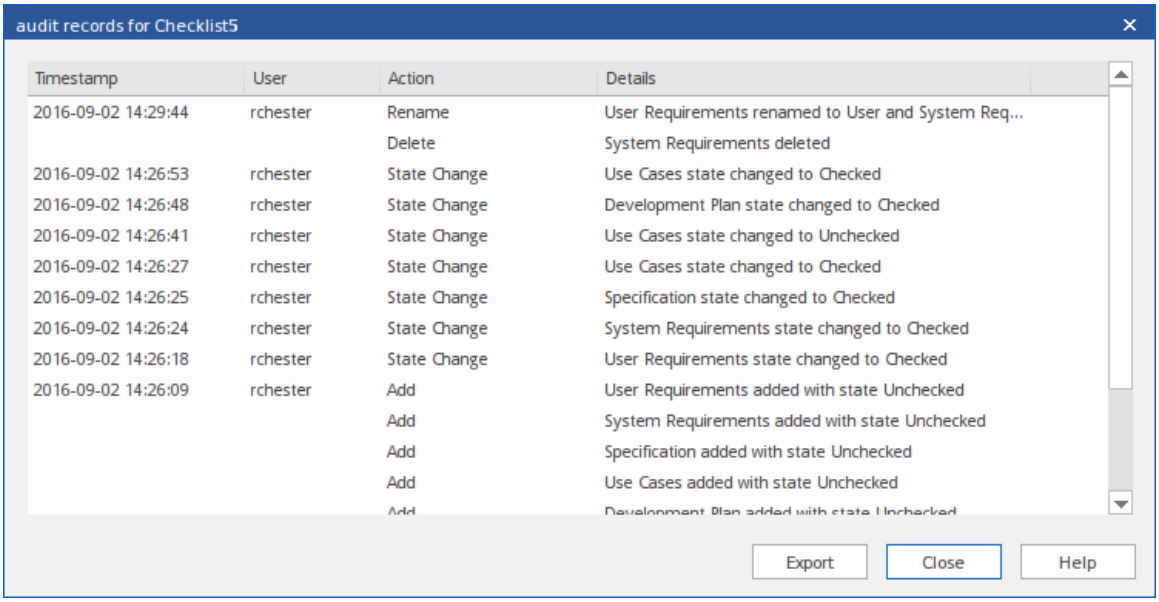
You can re-use the cleared Checklist for another step, stage or process.

If you have selected to display a simple, bulleted or numbered list, and you want to temporarily strike off items in the list, you can do so in the 'Checklist Items' dialog by selecting the checkbox against each item.

You can also use the element context menu to change the separation of items on the Checklist you are viewing; right-click on the element and select the line spacing you prefer.

The Audited Checklist

You can set up an Audited Checklist and other workers can make use of it, both in exactly the same way as for a Checklist. However, there is an additional context menu option - 'View audit log' - that displays an 'Audit records' page for the Audited Checklist. This 'Audit records' page is generated only for Audit Checklists, created using the 'Artifacts' toolbox page.



This page shows each change made to the checklist, the date and time of the change, who made the change, the type of change and what the change is. As shown, the changes include those made in creating or updating the checklist *and* the changes made by users selecting or clearing the checkboxes, including:

- Adding Checklist items (which also records the state they were set in)
- Deleting items
- Re-setting the state of each item
- Completing a Checklist
- De-selecting an item on a completed Checklist
- Making a Checklist a pre-requisite or dependent of another Checklist
- Renaming a pre-requisite or dependent Checklist
- Making a Checklist no longer the pre-requisite or dependent of another Checklist

Changes committed at the same time (such as when you click on an OK button) have one time stamp, as illustrated by the 'Add' changes in the audit log image.

If necessary, you can export the total contents of the 'Audit Records' page to a .csv file. To do this:

1. Click on the Export button; the 'Export List' dialog displays.
2. Browse for and select the appropriate directory in which to store the file.
3. Enter a filename for the file, and click on the Save button.

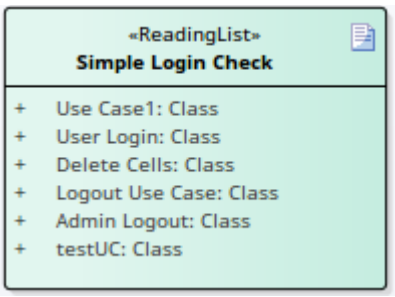
The file is saved and can then be opened in any external spreadsheet application, such as Excel.

Notes

- Unlike many other Artifact elements, Checklists can have a Priority of High, Medium or Low, set in the element 'Properties' dialog; this is useful for making sure a Checklist in a process is flagged as being of importance, such as in a Kanban diagram

Using the Reading List Artifact

The Reading List Artifact provides a list of elements that contain information of particular significance to a task or process. The intention is for the information from each element to be displayed in the 'Dynamic Document' view, in the sequence in which the elements are organized in the Reading List. The 'Dynamic Document' view provides 'Next' and 'Previous' options to move through the elements.



Set up a Reading List

Task	Action
Create the Artifact	Drag the 'Reading List' icon onto the diagram from the 'Documents' page of the Diagram Toolbox. Double-click on the element to display the 'Properties' dialog and give the element a name; perhaps identifying the subject of the reading list.
Create the Reading List	In the Browser window, identify the elements containing the required information and drag each one onto the

	Reading List Artifact element. The element names are displayed on the Artifact, in the sequence in which they are added to the Artifact. The element names are attributes of the Artifact. If you want to change the sequence in which they are listed: 1. Click on the Artifact and press Ctrl+5. The Features window displays, showing the 'Attributes' page. 2. Right-click on the attribute to move, and select either the 'Move Up' option or the 'Move Down' option. The attribute moves one space up or down the list for each click.

Read a Reading List

To read the information identified by a Reading List icon, you can either:

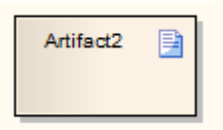
- Right-click on the Artifact and select the 'Show Reading List' option to display the 'Dynamic Document' view, or
- Display the 'Dynamic Document' view (Publish > Model Reports > Preview Mode) and click on the element name in the Browser window

The information from the first element in the Reading List is shown in the 'Dynamic Documents' view. When you have

finished reading this information, click on the [Reading] header and select the 'Next' option to display the information from the next element in the Reading List. As you work down the list, you can also select the 'Previous' option to return to information from an element further up the list.

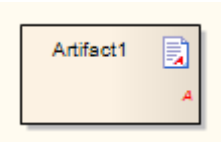
The 'Dynamic Document' view provides other options to, for example, use another style template to display the information, or change the display size. These options are available as you review the Reading List.

Document Artifact



Description

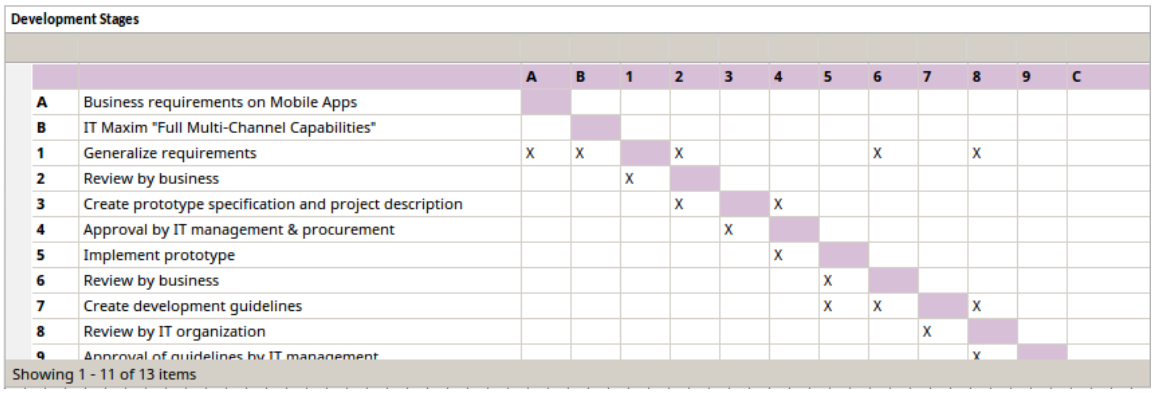
A Document Artifact is an artifact having a stereotype of «document». You create the Document Artifact using the Common, Documents, Component, Documentation or Deployment pages of the Diagram Toolbox, and associate it with a document or CSV file. Double-click on the element to display the Linked Document Editor. When you have created the Linked Document, the Document Artifact element on the diagram shows an 'A' symbol in the bottom right corner.



Toolbox icon

Custom Table Artifact

The Custom Table artifact is a diagram object that displays custom data in a grid format similar to a spreadsheet. For example:



Development Stages

		A	B	1	2	3	4	5	6	7	8	9	C
A	Business requirements on Mobile Apps												
B	IT Maxim "Full Multi-Channel Capabilities"												
1	Generalize requirements	X	X		X				X		X		
2	Review by business			X									
3	Create prototype specification and project description				X		X						
4	Approval by IT management & procurement					X							
5	Implement prototype						X						
6	Review by business							X					
7	Create development guidelines							X	X		X		
8	Review by IT organization									X			
9	Approval of guidelines by IT management										X		

Showing 1 - 11 of 13 items

The benefits of using this element include:

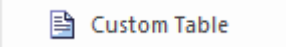
- Providing extra 'non-modeled' information on elements, diagrams or project management exactly where it is applicable, such as a SWAT Analysis or Capability Matrix
- Providing such information in a convenient human-readable and - if appropriate - human-editable format
- The ability to read and update data using scripts and Add-Ins

This feature is available in the Corporate, Unified and Ultimate Editions of Enterprise Architect, from Release 15.0.

Create a Custom Table

From the 'Dynamic Viewpoints' page of the Diagram

Toolbox, drag the Custom Table icon onto a diagram.



The Table element is created on the diagram and in the Browser window. On the diagram, click on the element to select it, then drag the borders of the element to expand it to a more comfortable size for editing.

Working with Custom Tables

To modify a Custom Table, it must first be placed into edit mode. To begin editing, right-click the Custom Table element on the diagram and select 'Edit Custom Table'. Alternatively, click on the element and press the Enter key or the F2 key.

While in edit mode you can modify and format the table contents.

To exit edit mode, right-click on the element and choose the option 'Exit Edit'. Alternatively, deselecting the table element (by clicking outside of the element, or by pressing the 'Esc' key) will also exit the edit mode.

Exiting edit mode will automatically save your changes.

1. The Custom Table data content will automatically be saved in XML format into the element's 'data' property.

2. The Custom Table row/cell formatting data will automatically be saved in XML format into the 'dataformat' property.

If you update a Custom Table element's properties by directly editing the XML, you can refresh the element in the diagram by right-clicking the element and selecting the option 'Refresh Custom Table'.

Operation	Description
Define Grid Size	The new Custom Table element does not yet have a defined grid size. 1. Right-click in the blank space in the body of the element and select the 'Set Grid Size...' option. 2. In the 'Set Grid Size' dialog, type in the required numbers of rows and columns. 3. Click on the OK button. Columns are autosized to occupy the visual area of the Table element. Rows, however, default to a single line height.
Add More Columns	You have two options for adding further columns to the table: • Add a new column to the right hand end of the table - right click on the table and select the 'Add Column' option • Insert a column at a specific point in

	the table - right-click on a column and select the 'Insert Column' option and either 'Before Selected Column' or 'After Selected Column' as required
Add More Rows	You also have two options for adding rows to the table: • Add a new row to the bottom of the table - right-click on the table and select the 'Add Row' option; the row might not be visible without scrolling to it, but note that the 'Showing x - n of y items' counter in the bottom right of the element will be incremented • Insert a row at a specific point in the table - right-click on a row and select the 'Insert Row' option and either 'Above Selected Row' or 'Below Selected Row' as required; the inserted row has the default height of one line
Delete Columns or Rows	Right-click on a cell in the column or row and select 'Delete Selected Row' or 'Delete Selected Column'. You cannot delete more than one row or column at a time, nor can you delete a row or column containing merged cells (even if they do not contain cells outside the row or column).

Copy Content	When you have data in the table, you can select to copy the content of either selected cells or the entire table to the clipboard, to paste into an external spreadsheet tool or text file. Select the 'Copy to Clipboard' option and then either 'Selected' or 'All'.
Change Column Width	Click on a column cell and either drag the border of the column header cell to the required width, or right-click on the Table element and select the 'Set Column(s) Width' option. (To select multiple columns, press the Ctrl key as you select each column.) If you select the 'Set Column(s) Width' option, the 'Set Column Width' dialog displays. Either type in the required width in pixels, or click on the arrows to increase or decrease the value by one pixel per click. Click on the OK button when you have entered the width. Note that: • Column widths are confined by the width of the element; you cannot increase the width of a column indefinitely, as the width increase will be blocked prior to the point at which

	one of the columns will lose its visible presence on the Table • Manually setting a column width by dragging the header border adjusts the width of any unset columns by equal amounts; for example, in a three-column Table, increasing one column width by eight pixels will decrease the width of the other two columns by 4 pixels each • Setting a column width through the 'Set Column Width' dialog changes the width of any unset and manually set columns by equal amounts • Adjusting the size of the Table element will adjust the column widths within the Table in proportion to each other • Text strings will wrap in the width of the column
Change Row Height	If you want to increase or decrease the height of one or more rows by one line height, right-click on a cell in that row and select the 'Increase row lines' or 'Decrease row lines' option, as appropriate. (To select multiple rows, press the Ctrl key as you select each row.) If you want to increase or decrease the

	height of one or more rows by several line heights, right-click on a cell in the selected row(s) and select the 'Set row(s) lines option'. The 'Set Row Lines' dialog displays. Either type in the required height in lines, or click on the arrows to increase or decrease the value by one line per click. Click on the OK button to save the settings. Note that: • Changing the height of one or more rows has no effect on the height of unselected rows in the Table • Increasing the height of a row can scroll other rows out of sight, beyond the borders of the Table element • If the text of a cell is more than can be displayed in the cell, the text scrolls out of sight beyond the top and bottom of the cell; in these situations, mouse-over the cell to display the full text in a pop-up field, or add more height to the row to accommodate the text
Format the Grid	You can perform operations to format the appearance of cells in the table, and of the table as a whole. Right-click on the table and select:

	• Show Grid Lines - to hide or display all lines separating cells; this does not hide those lines specifically displayed using the 'Set Cell(s) Border' option • Set Cell(s) Border - if 'Show Grid Lines' is toggled off and you have highlighted one or more cells, select the option for the required cell edge to display that edge on the cell(s): - Top - Right - Bottom - Left - Reset Default (to hide the visible borders of the currently-selected cells) Note that the border width is fixed at 1px. • Set Grid Color - the 'Select item color' dialog displays; click on the 'Set Color' drop-down arrow and on the appropriate color in the palette, then click on the OK button to apply the color to either all cell borders in the table ('Show Grid Lines' on) or those specifically selected for display ('Set Cell(s) Border') • Merge Selected - (available if you have selected two or more cells by sweeping

	the cursor across them) the separate cells become one cell, with the formatting of the uppermost and/or leftmost cell; the selected cells must form a regular block - you cannot merge two cells on one line with one cell on the next line unless the cells form a rectangle • UnMerge Selected - the previously-merged cells reappear with their original dimensions, color, text and formatting
Add Text to a Cell	Double-click on the cell and start typing. Alternatively, if you have copied text into the buffer, click twice on the cell, right-click and select the 'Paste' option. Note that formatting (even from another table cell) is not transferred in the copy.
Format Text	You can perform a number of operations to format the complete text of a cell. The options cannot operate on partial text strings in a cell. Note that these options take effect when you click off the cell. Right-click on the cell and select the required option: • Horizontal Align Text - select the appropriate sub-option to align the text

	with the left, center or right of the cell (new cells default to left-aligned text) • Vertical Align Text - select the appropriate sub-option to align the text with the top, center or bottom of the cell (new cells default to center-aligned text) • Set Text Color - the 'Select item color' dialog displays; click on the 'Set Color' drop-down arrow and on the appropriate color in the palette, then click on the OK button • Set Background Color - (to set the background for the cell, whether or not it contains text) the 'Select item color' dialog displays; click on the 'Set Color' drop-down arrow and on the appropriate color in the palette, then click on the OK button • Toggle Bold Text - the text is changed to bold or back to normal

Using the Tagged Values

The simplest method for initially defining and populating the Custom Table is to use the context menu options. However, if you want to set up a number of tables of similar

structure it becomes more efficient to copy the XML from the Tagged Values of one table to the Tagged Values of another, or to add a script to read and/or populate the tables. Each Tagged Value is a <memo> type that can contain a lengthy collection of XML definitions.

Tagged Value	Content
data	As you build up the grid, the basic structure is defined in this Tagged Value in XML, and as you add data values they are inserted into the appropriate lines of the structure definition. For example: <?xml version="1.0"?> <adhoctable> <table> <row> <column>Heading 1</column> <column>Heading 2</column> <column>Heading 3</column> <column>Heading 4</column> <column>Heading 5</column> </row> <row> <column>Rowname</column> <column></column> <column></column> <column></column>

	<column></column> </row> If you want to enter data directly into the XML in this Tagged Value, the 'Edit Custom Table' context menu must be off (you have selected the 'Exit Editing' option) and you should select the 'Refresh Custom Table' context menu option frequently to update the table image.
dataFormat	As you define the format and appearance of the grid, the definition is stored in XML in this Tagged Value. For example: <?xml version="1.0"?> <dataformat> <style> <grid rows="8" columns="5"> <gridcolor>16646398</gridcolor> </grid> <cells> <cell row="0" col="0"> <bold>true</bold> <txtcolor>255</txtcolor> <borders> <left>0</left> <top>0</top> <right>1</right>

	<bottom>1</bottom> </borders> </cell> <cell row="0" col="1"> <txtcolor>13434880</txtcolor> <borders> <left>0</left> <top>0</top> <right>1</right> <bottom>1</bottom> </borders> </cell> If you want to modify the definition directly in the XML in the Tagged Value, the 'Edit Custom Table' context menu must be off (you have selected the 'Exit Editing' option) and you should select the 'Refresh Custom Table' context menu option frequently to update the table image.

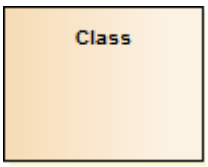
Custom Table Scripts

You can also associate a JavaScript script with your Custom Table element. Typically, a script might be used to either read data from the table or update the data in the table.

Scripts are saved in the operation named 'script'.

Action	Description
Define a Script	To define or edit the script: • Click on the table element to select it • Choose the ribbon option 'Develop > Source Code > Behavior' (Alt+7) • Select the operation named 'script' in the left hand panel of the editor • Enter the script code in the right-panel of the editor For detail on the methods available see the *ElementGrid Class* Help topic.
Run a Script	If an associated script has been defined for the table, it can be run by right-clicking the table while it is not in edit mode then choosing the option 'Run Custom Table Script'.

Class

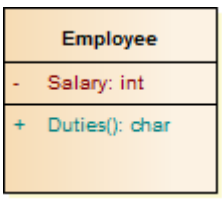


Description

A Class is a representation of a type of object that reflects the structure and behavior of such objects within the system. It is a template from which actual running instances are created, although a Class can be defined either to control its own execution or as a template or parameterized Class that specifies parameters that must be defined by any binding Class.

A Class can have attributes (data) and methods (operations or behavior). Classes can inherit characteristics from parent Classes and delegate behavior to other Classes. Class models usually describe the logical structure of the system and are the building blocks from which components are built.

The top section of a Class shows the attributes (or data elements) associated with the Class. These hold the 'state' of an object at run-time. If the information is saved to a data store and can be reloaded, it is termed 'persistent'. The lower section contains the Class operations (or methods at run-time). Operations describe the behavior a Class offers to other Classes, and the internal behavior it has (private methods).



Class elements are generally used in Class diagrams and Composite Structure diagrams.

Enterprise Architect also supports a number of stereotyped Class elements to represent various entities in web-page modeling. A Class can also be integrated with an Associate connector to form an Association Class, to allow the Associate connector to have operations and attributes that define certain types of UML relationship.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.194-195) states:

The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

Class is a kind of EncapsulatedClassifier whose Features are Properties, Operations, Receptions, Ports and Connectors. Attributes of a Class are Properties that are owned by the Class. Some of these attributes may represent the ends of binary Associations. Objects of a Class must contain values

for each attribute that is a member of that Class, in accordance with the characteristics of the attribute, for example its type and multiplicity.

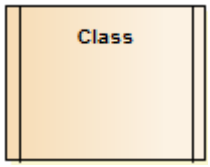
When an object is instantiated in a Class, for every attribute of the Class that has a specified default, if an initial value of the attribute is not specified explicitly for the instantiation, then the default ValueSpecification is evaluated to set the initial value of the attribute for the object.

Operations of a Class can be invoked on an object, given a particular set of values for the parameters of the Operation, (...).

A Class cannot access private Features of another Class, or protected Features on another Class that is not its ancestor.

A Class acts as the namespace for various kinds of Classifiers defined within its scope, including Classes. Nested Classifiers are members of the namespace of the containing Class. Classifier nesting is used for reasons of information hiding.

Active Classes



Description

An Active Class indicates that, when instantiated, the Class controls its own execution. Rather than being invoked or activated by other objects, it can operate standalone and define its own thread of behavior.

Define an Active Class in Enterprise Architect

Step	Action
1	Highlight a Class, and display its 'Properties' dialog (right-click on the Class and select the 'Properties | Properties option).
2	Select the 'Details' tab on the lower right of the dialog.
3	Select the 'Is Active' checkbox.
4	Click on the OK button to save the changes.

OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, p.438) states:

An active object is an object that, as a direct consequence of its creation, commences to execute its classifier behavior, and does not cease until either the complete behavior is executed or the object is terminated by some external object. (This is sometimes referred to as "the object having its own thread of control.") The points at which an active object responds to communications from other objects is determined solely by the behavior of the active object and not by the invoking object. If the classifier behavior of an active object completes, the object is terminated.

Parameterized Classes (Templates)

Enterprise Architect supports parameterized (Template) Classes, which specify parameters that must be defined by any binding Class.

Parameterized Classes are commonly implemented in C++; Enterprise Architect imports and generates templated Classes for C++.

The functionality of a template Class can be reused by any bound Class. If a default value is specified for a parameter, and a binding Class doesn't provide a value for that parameter, the default is used.

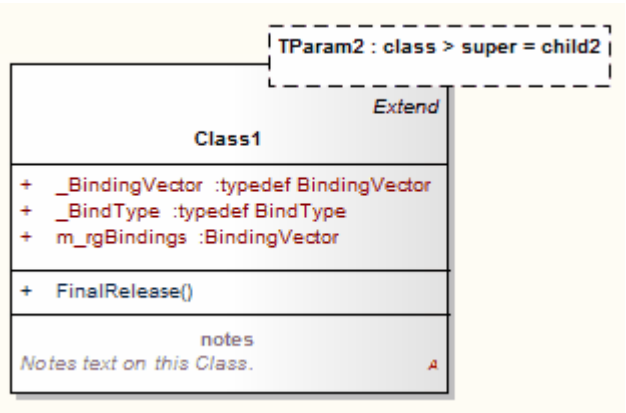
Create a parameterized Class

Step	Action
1	Click on the required Class.
2	Select the 'Design > Element > Manage > Template Parameters' ribbon option. The 'Templates' dialog displays.
3	In the 'Template Parameter(s)' panel, click on the Add button. The 'Template Parameter' dialog displays.

4	Type in the name and type of the parameter and, if required, click on the ... button after the 'Constraints' and 'Default' fields to select the required constraining and default Classes from the 'Select <Item>' dialog. The default Class can be either the constraining classifier or any Class that derives from the constraining classifier.

Notation Example

On a diagram, template Classes are shown with the parameters in a dashed outline box in the upper right corner of the Class.



OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, p. 622) states:

A template is a parameterized element that can be used to generate other model elements using TemplateBinding

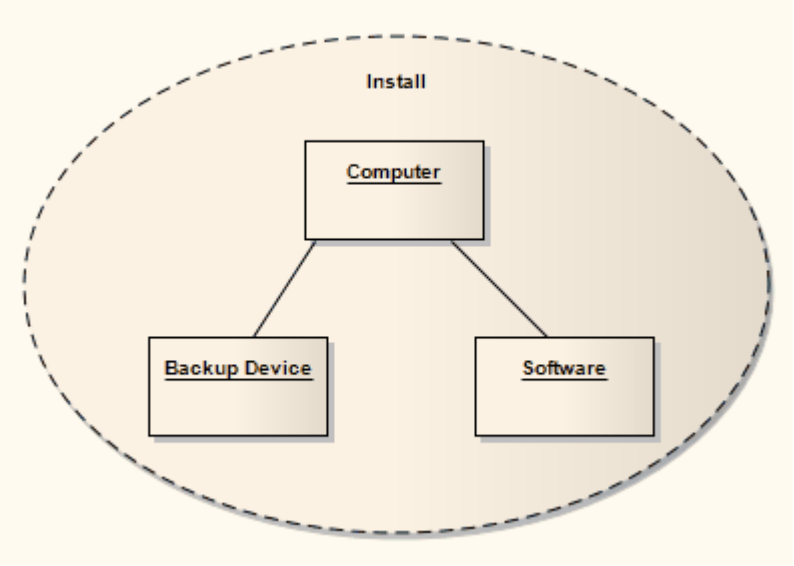
relationships. The template parameters for the template signature specify the formal parameters that will be substituted by actual parameters (or the default) in a binding.

Collaboration

Description

A Collaboration defines a set of cooperating roles and their connectors. These are used to collectively illustrate a specific functionality, in a Composite Structure diagram. A Collaboration should specify only the roles and attributes required to accomplish a specific task or function. Although in practice a behavior and its roles could involve many tangential attributes and properties, isolating the primary roles and their requisites simplifies and clarifies the behavior, as well as providing for reuse. A Collaboration often implements a Pattern to apply to various situations.

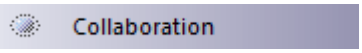
This example illustrates an Install Collaboration, with three roles (Objects) connected as shown. The process for this Collaboration can be demonstrated by attaching an Interaction diagram (Sequence, Timing, Communication or Interaction Overview).



To understand referencing a Collaboration in a specific situation, see the *Collaboration Use* Help topic.

Enterprise Architect supports a stereotyped Collaboration to represent a Business Use Case Realization in business modeling.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, v2.5.1, p.222) states:

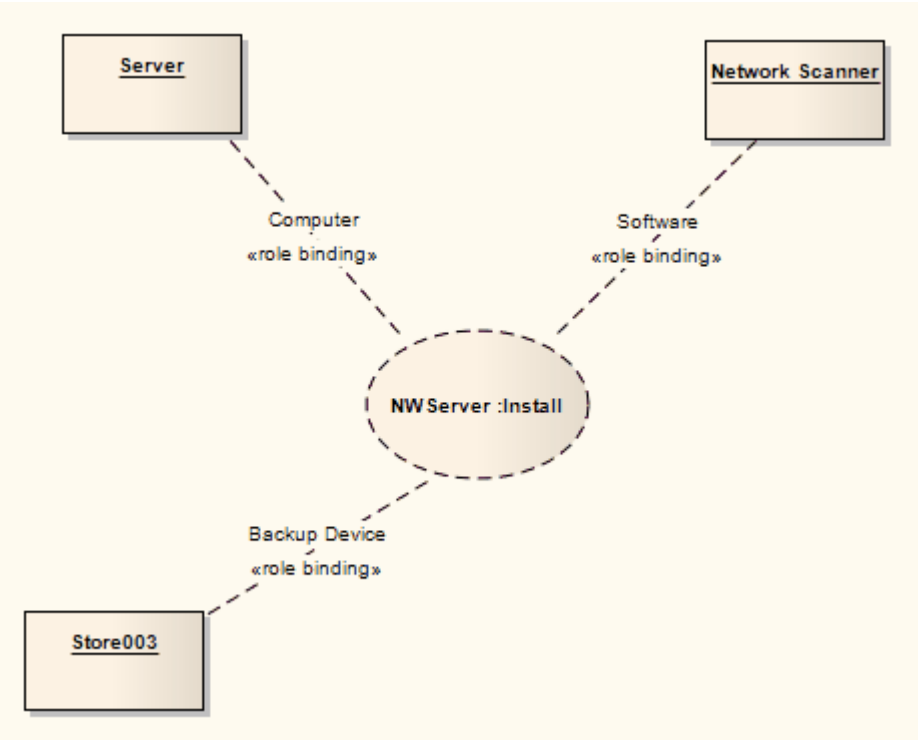
A Collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplish some desired functionality.

Collaboration Use

Description

Use a Collaboration Use to apply a Pattern defined by a Collaboration to a specific situation, in a Composite Structure diagram.

This example shows a Use, 'NWServer', of the Collaboration 'Install', to define the installation process of a network scanner. This process can be defined by an interaction attached to the Collaboration. (See the *Collaboration* Help topic for a representation of the Install Collaboration.)



To create a Collaboration Use, drag the icon from the 'Composite' page of the Diagram Toolbox onto the diagram.

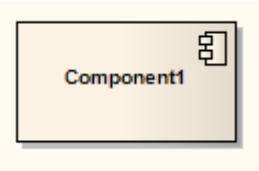
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.216) states:

A CollaborationUse represents a particular use of a Collaboration to explain the relationships between a set of elements. A CollaborationUse shows how the pattern described by a Collaboration is applied in a given context Classifier, by binding specific ConnectableElements from that context to the collaborationRoles of the Collaboration. There may be multiple CollaborationUses related to a given Collaboration within a Classifier, each bound differently. A given collaborationRole or Connector may be involved in multiple uses of the same or different Collaborations.

Component



Description

A Component is a modular part of a system, whose behavior is defined by its provided and required interfaces; the internal workings of the Component should be invisible and its usage environment-independent. Source code files, DLLs, Java beans and other artifacts defining the system can be manifested in Components.

A Component can be composed of multiple Classes, or Components pieced together. As smaller Components come together to create bigger Components, the eventual system can be modeled, building-block style, in Component diagrams. By building the system in discrete Components, localization of data and behavior enables decreased dependency between Classes and Objects, providing a more robust and maintainable design.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, v2.5.1, pp.208-209) states:

This sub clause specifies a set of constructs that can be used to define software systems of arbitrary size and complexity. In particular, it specifies a Component as a modular unit with well-defined Interfaces that is replaceable within its environment. The Component concept addresses the area of component-based development and component-based system structuring, where a Component is modeled throughout the development life cycle and successively refined into deployment and run-time.

An important aspect of component-based development is the reuse of previously constructed Components. A Component can always be considered an autonomous unit within a system or subsystem. It has one or more provided and/or required Interfaces (potentially exposed via Ports), and its internals are hidden and inaccessible other than as provided by its Interfaces. Although it may be dependent on other elements in terms of Interfaces that are required, a Component is encapsulated and its Dependencies are designed such that it can be treated as independently as possible. As a result, Components and subsystems can be flexibly reused and replaced by connecting ("wiring") them together.

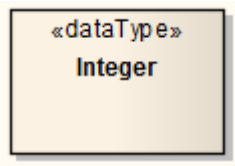
The aspects of autonomy and reuse also extend to Components at deployment time. The artifacts that implement Component are intended to be capable of being deployed and re-deployed independently, for instance to update an existing system.

The OMG Unified Modeling Language specification, v2.5.1, p.224) states:

A Component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment.

A component defines its behavior in terms of provided and required interfaces. As such, a component serves as a type whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics).

Data Type



Description

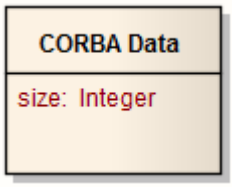
A Data Type is a specific kind of classifier, similar to a Class except that a Data Type cannot own sub Data Types, and instances of a Data Type are identified only by their value. For example, an instance of a Person Class is a Helen object, but an instance of an Integer Data Type is 12.

All copies of an instance of a Data Type, and any instances of that Data Type with the same value, are considered to be the same instance. That is, instances of Helen are not necessarily the same Helen, but all 12s are the same 12. For example, the 12 on a watch face is exactly the same integer as the number of months in a year.

Instances of a Data Type that have attributes (that is, are instances of a structured Data Type) are considered to be the same if the structure is the same and the values of the corresponding attributes are the same. If a Data Type has attributes, instances of that Data Type contain attribute values matching the attributes.

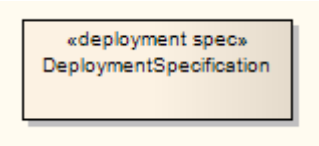
A typical use of Data Types would be to represent programming language primitive types or CORBA basic types. For example, integer and string types are often treated as Data Types.

A Data Type is denoted by a rectangle with the keyword ?dataType? or, when it is referenced by (for example) an attribute, by a string containing the name of the Data Type, as shown:



Toolbox icon

Deployment Specification

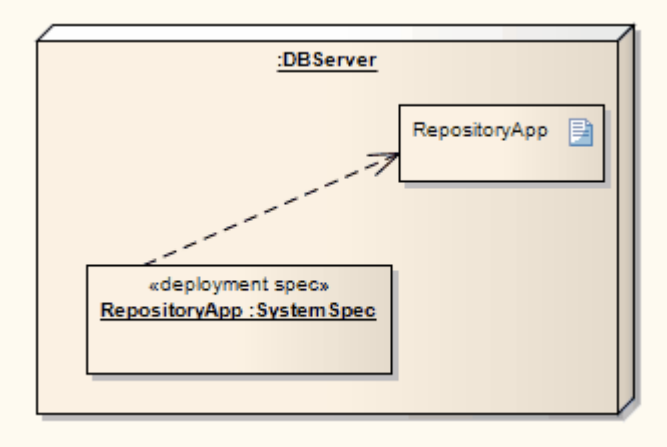


Description

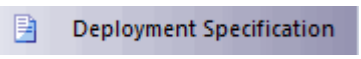
A Deployment Specification specifies parameters guiding deployment of an artifact, as is necessary with most hardware and software technologies. A specification lists those properties that must be defined for deployment to occur, as represented in a Deployment diagram. An instance of this specification specifies the values for the parameters; a single specification can be instantiated for multiple artifacts.

These specifications can be extended by certain component profiles. Examples of standard Tagged Values that a profile might add to a Deployment Specification are ?concurrencyMode? with Tagged Values {thread, process, none} or ?transactionMode? with Tagged Values {transaction, nestedTransaction, none}.

This example depicts the artifact RepositoryApp deployed on the server node, as per the specifications of RepositoryApp, instantiated from the Deployment Specification SystemSpec.



Toolbox icon

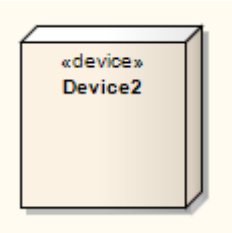


OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.662) states:

A deployment specification specifies a set of properties that determine execution parameters of a component artifact that is deployed on a node. A deployment specification can be aimed at a specific type of container. An artifact that reifies or implements deployment specification properties is a deployment descriptor.

Device



Description

A Device is a physical electronic resource with processing capability upon which Artifacts can be deployed for execution, as represented in a Deployment diagram. Complex Devices can consist of other devices; that is, a Device can be a nested element, where a physical machine is decomposed into its elements either through namespace ownership or through attributes that are typed by Devices.

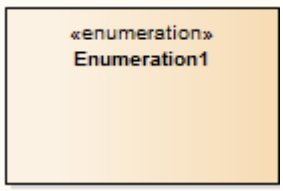
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, v2.5.1, p.663) states:

A device is a physical computational resource with processing capability upon which artifacts may be deployed for execution. Devices may be complex (i.e., they may consist of other devices).

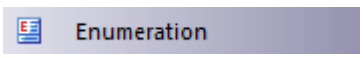
Enumeration



Description

An Enumeration is a data type, whose instances can be any of a number of user-defined enumeration literals. It is possible to extend the set of applicable enumeration literals in other Packages or profiles. You create Enumerations in Class or Package diagrams, and in diagrams developed using the Metamodel and Profile pages of the Diagram Toolbox.

Toolbox icon

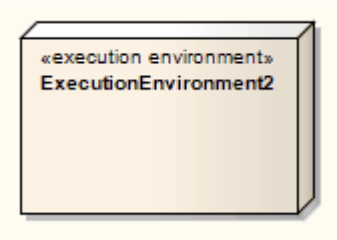


OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.175) states:

An Enumeration is a DataType whose values are enumerated in the model as EnumerationLiterals.

Execution Environment

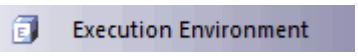


Description

An Execution Environment is a node that offers an execution environment for specific types of component that are deployed on it in the form of Executable Artifacts. This is depicted in a Deployment diagram.

Execution Environments can be nested; for example, a database Execution Environment can be nested in an operating system Execution Environment. Components of the appropriate type are then deployed to specific Execution Environment nodes.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.664) states:

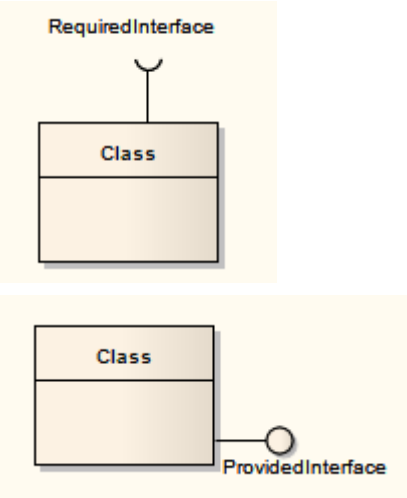
An execution environment is a node that offers an execution environment for specific types of components that are

deployed on it in the form of executable artifacts.

The OMG Unified Modeling Language specification, (v2.5.1, p.658) also states:

Typically, ExecutionEnvironments are assigned to some, often higher level, Device or general system Node via the composition relationship defined on Node. ExecutionEnvironments can be nested (for example, a database ExecutionEnvironment might be nested in an operating system ExecutionEnvironment). ExecutionEnvironment may have explicit interfaces for system level services that can be called by the deployed elements. In such cases, software ExecutionEnvironment services should be explicitly modeled. Application Components of the appropriate type are then deployed, with a Deployment relationship, to specific ExecutionEnvironment nodes or the Manifestations relationships of DeployedArtifacts. For each component Deployment, aspects of these services may be determined by properties in a DeploymentSpecification for a particular kind of ExecutionEnvironment.

Expose Interface



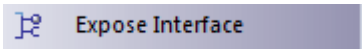
Description

The Expose Interface element is a graphical method of depicting the required or supplied interfaces of a Component, Class or Part, in a Component or Composite Structure diagram. It just identifies the fact that the element provides or requires an interface; to depict the fact that the provided interface is used, or the required interface provided, by another element, use the Assembly connector.

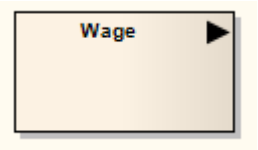
The Expose Interface element must be attached to the Class or Component element, and it becomes a child element of that Class or Component; it cannot exist independently. You can attach more than one Expose Element to another element.

When you create the Expose Interface element, a dialog displays in which you enter a name for the element and specify whether it represents a required interface or a provided interface.

Toolbox icon



Information Item

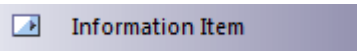


Description

An Information Item element represents an abstraction of data, which data can be conveyed between two objects. The term 'Information Item' is also more loosely applied to any classifier that represents a more specific identification of the type of data that can be conveyed between two objects

The conveyance and realization of Information Items (of either kind) between the two objects is represented by an Information Flow connector.

Toolbox icon



OMG UML Specification:

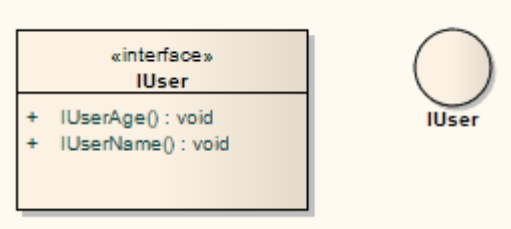
The OMG Unified Modeling Language specification, (v2.5.1, p.673) states:

InformationItems represent many kinds of information that can flow from sources to targets in very abstract ways. They represent the kinds of information that may move within a system, but do not elaborate details of the transferred

information. Details of transferred information are the province of other Classifiers that may ultimately define InformationItems. Consequently, InformationItems cannot be instantiated and do not themselves have features, generalizations, or associations.

An important use of InformationItems is to represent information during early design stages, possibly before the detailed modeling decisions that will ultimately define them have been made. Another purpose of InformationItems is to abstract portions of complex models in less precise, but perhaps more general and communicable, ways.

Interface



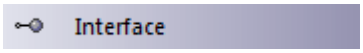
Description

An Interface is a specification of behavior (or contract) that implementers agree to meet. By implementing an Interface, Classes are guaranteed to support a required behavior, which enables the system to treat non-related elements in the same way; that is, through the common interface. You also use Interfaces in a Composite Structure diagram.

Interfaces are drawn in a similar way to a Class, with operations specified, as shown here. They can also be drawn as a circle with no explicit operations detailed - right-click on the element and select the 'Use Circle Notation' option to switch between styles. Realize connectors to an Interface drawn as a circle are drawn as a solid line without target arrows.

An Interface cannot be instantiated (that is, you cannot create an object from an Interface). You must create a Class that 'implements' the Interface specification, and in the Class body place operations for each of the Interface operations. You can then instantiate the Class.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.171) states:

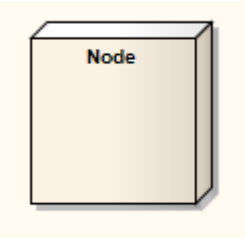
An Interface is a kind of Classifier that represents a declaration of a set of public Features and obligations that together constitute a coherent service. An Interface specifies a contract; any instance of a Classifier that realizes the Interface shall fulfill that contract. The obligations associated with an Interface are in the form of constraints (such as pre- and postconditions) or protocol specifications, which may impose ordering restrictions on interactions through the Interface. Interfaces may not be instantiated. Instead, an Interface specification is implemented or realized by a BehavioredClassifier, which means that the BehavioredClassifier presents a public facade that conforms to the Interface specification.

NOTE. A given BehavioredClassifier may implement more than one Interface and that an Interface may be implemented by a number of different BehavioredClassifiers. Interfaces provide a way to partition and characterize groups of public Features and obligations that realizing BehavioredClassifiers shall possess.

An Interface does not specify how it is to be implemented, but merely what needs to be supported by realizing BehavioredClassifiers. That is, such BehavioredClassifiers

shall provide a public façade consisting of attributes, Operations, and externally observable Behavior that conforms to the Interface.

Node



Description

A Node is a physical piece of equipment on which the system is deployed, such as a workgroup server or workstation. A Node usually hosts components and other executable pieces of code, which again can be connected to particular processes or execution spaces. Typical Nodes are client workstations, application servers, mainframes, routers and terminal servers.

Nodes are used in Deployment diagrams to model the deployment of a system, and to illustrate the physical allocation of implemented artifacts. They are also used in web modeling, from dedicated web modeling pages in the Toolbox.

Toolbox icon

OMG UML Specification:

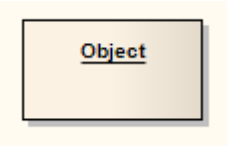
The OMG Unified Modeling Language specification,

(v2.5.1, p.658) states:

A Node is computational resource upon which Artifacts may be deployed, via Deployment relationships, for execution. For advanced modeling applications, Nodes may have complex internal structure defined by nesting and may be interconnected to represent specific situations. The internal structure of Nodes can only consist of other Nodes. Besides participating in Deployments, Nodes acquire a set of associated elements derived from the Manifestation relationships of the Artifacts deployed on them.

Nodes may be further sub-typed as Devices and ExecutionEnvironments. Devices represent physical machine components. ExecutionEnvironments represent standard software systems that application components may require at execution time. Specific profiles might, for example, define stereotypes for ExecutionEnvironments such as «OS», «workflow engine», «database system», and «J2EE container».

Object

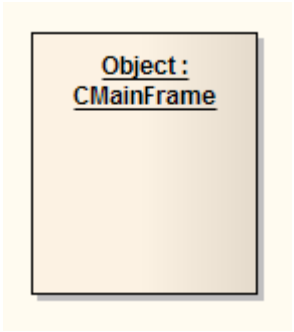


Description

An Object is a particular instance of a Class at run time. For example a car with the license plate AAA-001 is an instance of the general Class of cars with a license plate number attribute. Objects are often used in analysis to represent the numerous artifacts and items that exist in any business, such as pieces of paper, faxes and information. To model the varying behavior of Objects at run-time, use run-time states.

Early in analysis, Objects can be used to quickly capture all the things that are of relevance within the system domain, in an Object, Composite Structure or Communication diagram. As the model progresses these analysis Objects are refined into generic Classes from which instances can be derived to represent common business items. Once Classes are defined, Objects can be typed; that is they can have a classifier set that indicates their base type - see the *Classifiers and Instances* topic.

Enterprise Architect also supports a number of stereotyped Object elements to represent various entities in business modeling.



Toolbox icon

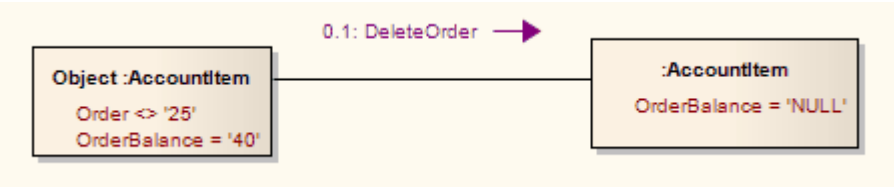
Run-time State

At run-time, an Object instance can have specific values for its attributes, or exist in a particular state. To model the varying behavior of Objects at run-time, use instance values selected from the 'Select <Item>' dialog and run-time states or run-states.

Typically there is interest in the run-time behavior of Objects that already have a classifier set. You can select from the classifier's attribute list and apply specific values for your Object instance. If the classifier has a child StateMachine, its States propagate to a list where the run-time state for the Object can be defined.

Example

This example defines run-time values for the listed variables, which are attributes of the AccountItem classifier for the instance.



Access

Ribbon	Start > Application > Design > Properties, click on an Object in the

	diagram or Browser window > Run States Design > Element > Editors > Properties, click on an Object in the diagram or Browser window > Run States
Context Menu	In a diagram or the Browser window, right-click on the 'Object | Features | Set Run State' option
Keyboard Shortcuts	Ctrl+Shift+R Ctrl+2 > click on an Object in the diagram or Browser window > Run States

Add run-time state instance variables to an Object

On the Properties window, or the '<object name> : Features' dialog the 'Run States' page lists any variables inherited from the classifier of the Object element. These inherited variables initially have no values and are inactive. You can activate and define a run state for them, or you can right-click on the '<object name> : Features' dialog and select the 'Hide Inherited Variables' option to hide them from view.

Step	Action

1	In the 'Variable' field, either: • Overtype the *New Variable* text with the name of the new variable, or • Click on the name of an inherited variable to activate
2	In the 'Operator' field, click on the drop-down arrow and select the operator that will qualify the run state value. The operators include: • blank (no operator) • != • < • <= • <> • = • => • >
3	In the 'Value' field, type the value for the run state of the variable.
4	If necessary, type in some explanatory notes.
5	Click on or add the next variable, or click on the Close button to save the changes.

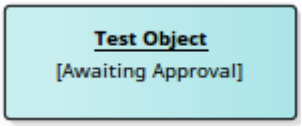
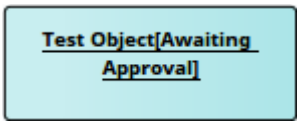
Delete a run-time state variable for an Object

Step	Action
1	In the 'Variable' field, right-click on the variable to delete and select the 'Delete' option. (Alternatively, click on the variable and press Ctrl+Del.)
2	Click on the Close button.

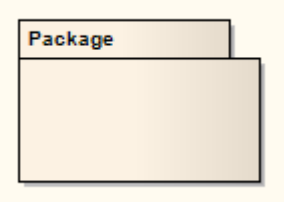
Object State

Set the Object state for a Class instance

Step	Action
1	Right-click on the required Object in a diagram and select the 'Advanced | Set Object State' option. The 'Set Instance State' dialog displays.
2	In the 'State' field, either type the required State (such as 'Awaiting Approval') or select a State from the drop-down list. The drop-down list for the 'State' field is populated with: • Any States owned by the object's classifier • Any States owned by any superclasses of the object's classifier • Any States owned by StateMachines owned by the object's classifier • Any States owned by StateMachines owned by any superclasses of the object's classifier
3	Below the 'State' field is the 'Merge State with Instance Name' checkbox, which defaults to unselected. Either:

	• Ignore the checkbox to display the object state on the element beneath the object name  • Or select the checkbox to display the object state on the element as a suffix to the object name 
4	Click on the OK button to apply the State. The object now shows the run-time state in the format you have selected.

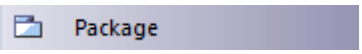
Package



Description

A Package is a namespace as well as an element that can be contained in other Package's namespaces. A Package can own or merge with other Packages, and its elements can be imported into a Package's namespace. In addition to using Packages in the Browser window to organize your project contents, you can drag the Packages onto a diagram workspace (most diagram types, both standard and extended) for structural or relational depictions, including Package imports or merges.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.241-242) states:

A Package is a namespace for its members, which comprise those elements associated via packagedElement (which are

said to be owned or contained), and those imported. A Package definition can extend the contents of other Packages through the merging of the contained elements.

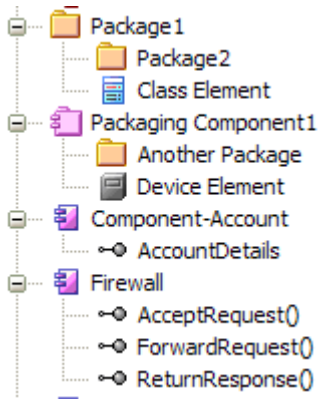
A Package may be defined as a template and bound to other templates: see sub clause 7.3, Templates, for further information. The URI can be specified to provide a unique identifier for a Package. Within UML there is no predetermined usage for this, with the exception of profiles (...). It may, for example, be used by model management facilities for model identification.

Packaging Component

Description

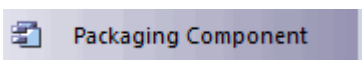
A Packaging Component is an element that appears very similar to a Component in a diagram but behaves as a Package in the Browser window (that is, it can be Version Controlled and can contain other Packages and elements). It is typically used in Component diagrams.

In the Browser window, the three elements display as shown:

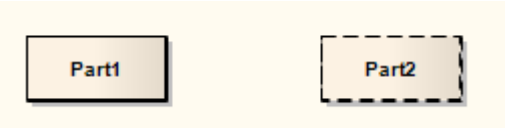


The Component element cannot contain child Packages or Packaging Components.

Toolbox icon



Part



Description

Parts are run-time instances of Classes or Interfaces. Multiplicity can be specified for a Part, using the notation:

(x{...y})

where x specifies the initial or set number of instances when the composite structure is created, and y indicates the maximum number of instances at any time.

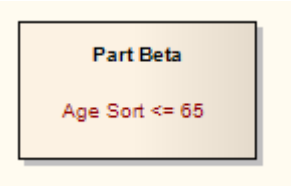
Parts are used to express composite structures, or modeling Patterns that can be invoked by various objects to accomplish a specific purpose. When illustrating the composition of structures, Parts can be embedded as properties of other Parts. When embedded as properties, Parts can be bordered by a solid outline, indicating the surrounding Part owns the Part by composition. Alternatively, a dashed outline indicates that the property is referenced and used by the surrounding Part, but is not composed within it.

Toolbox icon

Add Property Value

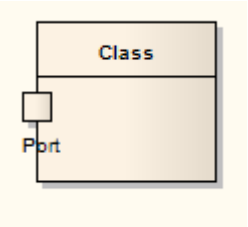
Add property value variables to a Part

A Part with a property value resembles this illustration:



Step	Action
1	Right-click on the Part and select the 'Features | Set Property Values' option (or press Ctrl+Shift+R). The 'Set Property Values' dialog displays.
2	In the 'Variable' field, click on the drop-down arrow and select the variable, or type in the new variable name.
3	Set the Operator and the Value, and optionally type in a Note.
4	Click on the OK button to save the variable.

Port



Description

Ports define the interaction between a classifier and its environment. Interfaces controlling this interaction can be depicted using the Interface element. Any connector to a Port must provide the required interface, if defined. Ports can appear on a contained Part, a Class, or the boundary of a Composite element.

A Port is a typed structural feature or property of its containing classifier. Ports are typically created in Class diagrams, Object diagrams and Composite Structure diagrams.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p. 230) states:

A Port is a property of an EncapsulatedClassifier that

specifies a distinct interaction point between that EncapsulatedClassifier and its environment or between the (behavior of the) EncapsulatedClassifier and its internal parts. Ports are connected to Properties of the EncapsulatedClassifier by Connectors through which requests can be made to invoke BehavioralFeatures. A Port may specify the services an EncapsulatedClassifier provides (offers) to its environment as well as the services that an EncapsulatedClassifier expects (requires) of its environment. A Port may have an associated ProtocolStateMachine.

Add a Port to an Element

Add a new Port to an element

Use one of these steps:

Step	Action
1	Click on the Port symbol in the 'Composite' page of the Toolbox, and drag the symbol to (or click on) the target host element. This creates an untyped, simple Port on the boundary, near the cursor position.
2	On the context menu of a suitable Class, Part or Composite element in the Browser window, select the 'Add | Port' option.
3	Drag a suitable classifier from the Browser window onto a Class or Part on a diagram. A prompt displays to add a typed Port or Part at the cursor position. The new Port is typed by the original dragged classifier.
4	Use the 'Ports' sub-menu (on a diagram, right-click on element | New Child Element) to add a new Port to the currently selected element.

Inherited and Redefined Ports

A Port is a redefinable and re-useable property of a composite classifier such as a Component. A Component can inherit Ports from its parent; if a Component's parent owns Ports, when you open the Features window at the 'Interaction Points' tab for the Component and select the 'Show Owned/Inherited' checkbox, the inherited Ports and their named owners are listed.

If you want to show an inherited Port on a Component, the Features window provides two options:

- Expose an inherited Port - tick the 'Show Owned/Inherited'checkbox to create a read-only copy of the Port; this is convenient for modeling Port interactions in child elements where the Ports are defined in the parent elements
- Redefine an inherited Port - right-click on the Port and select the 'Redefine' option, to create an editable copy of the Port; this is useful where a child element places additional restrictions or behavior on the Port

This also applies to Components that inherit Ports from realized Interfaces, and to Component instances that inherit Ports from their classifying Component.

Ports as Owners of Parts

If a Port is typed to a Class that has Parts, the Port can be shown on the diagram as the owner of these Parts. To do this either:

- Right-click on the Port | Advanced | Port Size Customizable, then enlarge the Port or
- Right-click on the Port | Features > Parts / Properties : Select the Parts you want to show : Select 'Show Owned/Inherited'

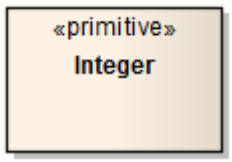
This feature is helpful when you want a connector to connect to the internal structure of the Port.

Properties Window - Property, Redefined/Subsetted

The Properties window for Ports and Parts has a 'Property' tab - which defines the type, initial value and multiplicity of the element - and a 'Redefined/Subsetted' tab that identifies the redefined and subsetted properties and Qualifiers of the element.

You set the Qualifiers by clicking on the Qualifiers button, to display the 'Qualifiers' dialog. You add Redefined and Subsetted Properties by clicking on the appropriate Add button, to display the 'Select Property' dialog.

Primitive



Description

A Primitive element identifies a predefined data type, without any relevant substructure (that is, it has no parts in the context of UML). It could be regarded as a conceptual Data Type. The Primitive element can be used to support the Meta-Object Facility (MOF) specification.

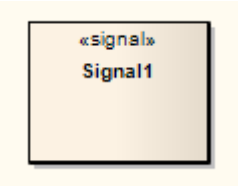
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.168) states:

A PrimitiveType defines a predefined DataType, without any substructure. A PrimitiveType may have algebra and operations defined outside of UML, for example, mathematically. The run-time instances of a PrimitiveType are values that correspond to mathematical elements defined outside of UML (for example, the Integers).

Signal



Description

A Signal is a specification of Send request instances communicated between objects, typically in a Class or Package diagram. The receiving object handles the Received request instances as specified by its Receptions. The data carried by a Send request is represented as attributes of the Signal. A Signal is defined independently of the classifiers handling the signal occurrence.

A Reception is defined as a feature of the receiving object, derived from the Signal element. The Reception takes the name of the Signal, and the Signal's attributes as its parameters.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.178) states:

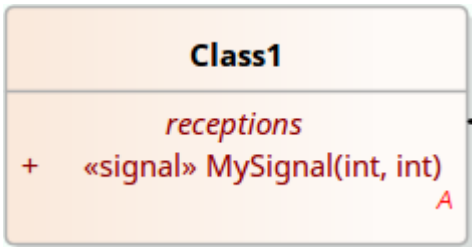
A Signal is a specification of a kind of communication between objects in which a reaction is asynchronously triggered in the receiver without a reply.

The OMG Unified Modeling Language specification, (v2.5.1, p.178) also states:

A Reception is a declaration stating that a Classifier is prepared to react to the receipt of a Signal.

Reception

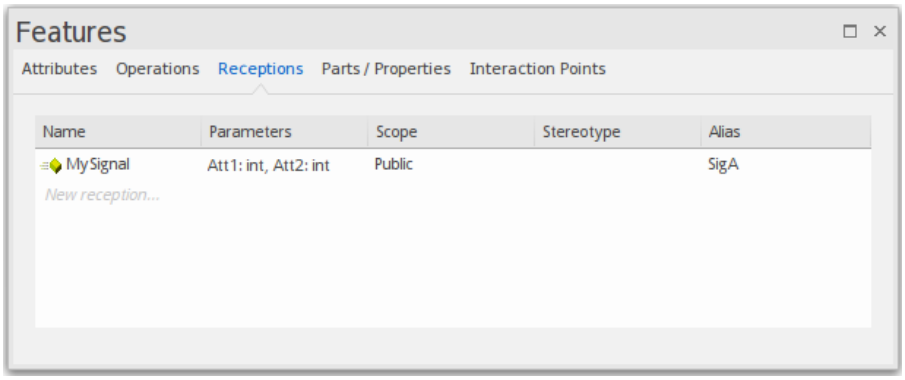
A Reception is a feature of a Classifier, and is derived from a Signal element; it models receipt of the Signal. Receptions are treated in the same way as Operations and, where element compartments are displayed, are shown within their own compartment.

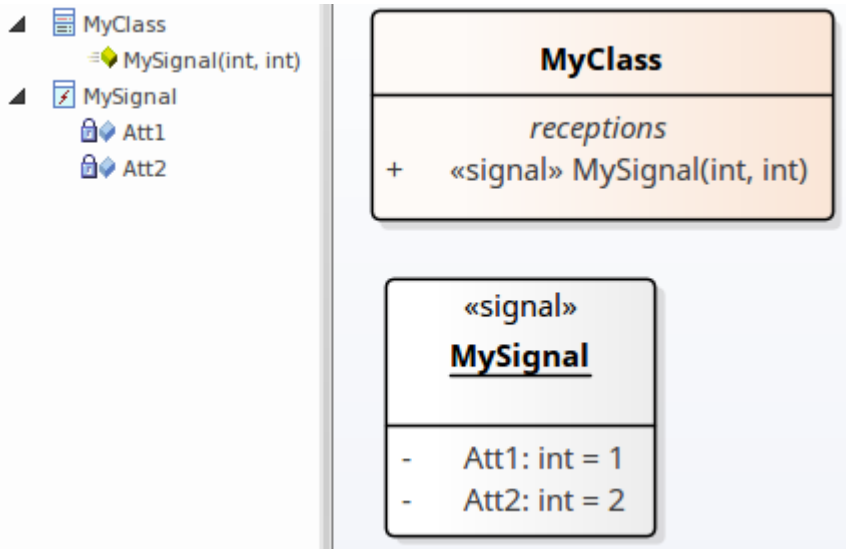


A Reception cannot be created without a Signal element, which either already exists or is created as a step in creating the Reception. The name of the Reception is drawn from the name of the Signal element, and the attributes of the Signal define the parameters of the Reception. Any changes to the Signal are reflected in the Reception, and vice versa.

Create a Reception

Step	Action
1	Create a Signal element either in the Browser window or on a diagram.
2	Create or drag a Class or Interface element on a diagram.
3	

	Either: • Drag the Signal element from the Browser window onto the Class or Interface element and go to step 8, or • Right-click on the Class element and select the 'Features | Receptions' option The Features window displays, at the 'Receptions' page. 
4	Click on the … button to the right of the *New Reception* text in the 'Name' column. The 'Select Signal' dialog displays (a variation of the 'Select <item>' dialog).
5	Browse for and click on the Signal element. (If you didn't create a Signal element at the start, you could use the Add New button to create a new Signal element now.)
6	Click on the OK button. The Signal element's name and attributes are used to generate the Reception's name and

	parameters.
7	Click on the Close button.
8	Note the appearance of the Class and Signal elements in the diagram and Browser window. 

OMG UML Specification

The OMG Unified Modeling Language specification, (v2.5.1, pp.447 - 448):

A reception is a declaration stating that a classifier is prepared to react to the receipt of a signal. A reception designates a signal and specifies the expected behavioral response. The details of handling a signal are specified by the behavior associated with the reception or the classifier itself. ... Receptions are shown using the same notation as for operations with the keyword <signal>.

Properties Window for Receptions

The docked Properties window provides a convenient and immediate way to view and edit common properties of modeling objects, including features such as Receptions. When you click on a Reception in the Browser window or a diagram, the Properties window immediately shows the Reception's properties, and you can swiftly display the properties of other types of object - elements, attributes, connectors and diagrams - on the same window without having to open and close separate dialogs.

The Properties window for Receptions has three tabs:

- Reception, on which you define general settings
- Behavior, on which you define the behavior to be taken on receipt of the signal
- Redefines, on which you set up any redefinition of target operations that takes place, and check for exceptions

For information on the 'Behavior' and 'Redefines' tabs, see the *Operation Behavior* and *Redefine Operation and Check for Exceptions* Help topics.

Access

Ribbon	Start > Application > Design > Properties; click on a Reception in a diagram or in the Browser window Design > Element > Editors > Properties,

	click on a Reception in a diagram or in the Browser window
Keyboard Shortcuts	Ctrl+2 or Ctrl+Enter, then click on a Reception in a diagram or in the Browser window

Reception - General Settings

Field	Description
Name	Displays the name of the Reception, which is the name of the Signal element it is derived from. You cannot change the name in this field. If the name of the Signal is changed, the Reception name is automatically updated to match.
Parameters	Displays the names and types of the attributes of the Signal element, from which the Reception parameters are derived. You can only change the parameters by updating the Signal attributes.
Scope	Displays the scope of the Reception, derived from the Signal element. If

	necessary, you can change this scope - click on the drop-down arrow at the end of the field and select the new value from the list.
Stereotype	(Optional) If you need to group or specialize the Reception, you can type the name of an appropriate stereotype in this field.
Alias	(Optional) if necessary, you can type in an alternative name or reference for the Reception.
Concurrency	Defaults to Sequential. If you want to change this to 'Guarded' or 'Synchronous', click on the drop-down arrow and select the appropriate value.
Abstract	Identifies whether or not the Reception is abstract.
Static	The flag indicating if the Reception is a static member; to change this flag, type 'False' or 'True' in the field, as appropriate.

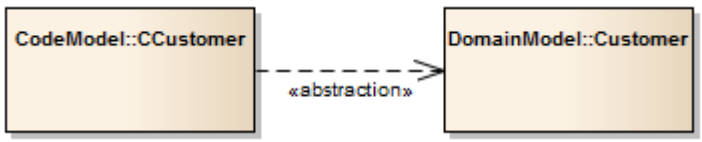
UML Connectors

Connectors link elements together and are typically represented as lines on diagrams showing how the elements relate to each other. Making a comparison to natural languages, if the elements are nouns the connectors are verbs that describe how the nouns relate to each other.

The UML has a wide variety of connector types that are used to express the nature of the relationship between the model elements involved. Some connectors such as the Association define structural relationships whereas others such as the Control Flow define the passage of time. Each connector type has a notation that helps modelers recognize the connector and understand its purpose.

Connectors can be viewed in a wide range of windows such as the Relationships Windows, the Hierarchy Window, the Relationship Matrix, the 'Details' tab of the Inspector window and an element's 'Properties' dialog.

Abstraction

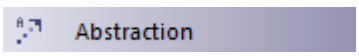


Description

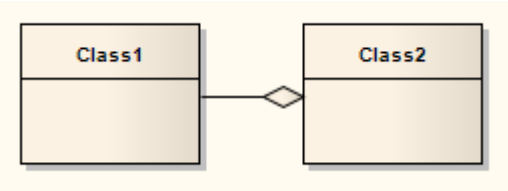
An 'Abstraction' is a relationship between two elements that represent the same concept, either at different levels of abstraction or from different viewpoints. This diagram shows that two different customer Classes from different models (the Domain model and the Code model) represent the same concept.

The 'Abstraction' relationship is a subtype of a 'Dependency' relationship.

Toolbox icon



Aggregation

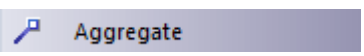


Description

An Aggregation connector is a type of association that shows that an element contains or is composed of other elements. It is used in Class models, Package models and Object models to show how more complex elements (aggregates) are built from a collection of simpler elements (component parts; for example, a car from wheels, tires, motor and so on).

A stronger form of aggregation, known as Composite Aggregation, is used to indicate ownership of the whole over its parts. The part can belong to only one Composite Aggregation at a time. If the composite is deleted, all of its parts are deleted with it.

Toolbox icon



Change Aggregation Connector Form

In your modeling, when you create an Aggregation relationship it defaults to the weak (shared) form of the relationship, represented by a hollow diamond head. You can change this to the strong form (Composition), represented by a solid black diamond head.

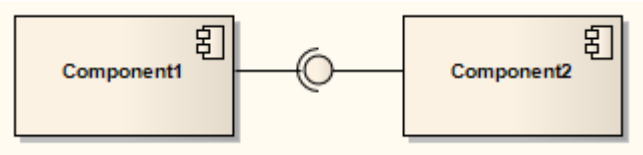
Change the form of an Aggregation connector from weak to strong

Step	Action
1	Right-click on an Aggregation connector to display the context menu.
2	Select Set Aggregation to Composite. The diamond is shown as filled.

Notes

- If the connector is already a Strong (Composition) connector, the context menu option changes to 'Set Aggregation to Shared'

Assembly



Description

An Assembly connector bridges a component's required interface (Component1) with the provided interface of another component (Component2), typically in a Component diagram.

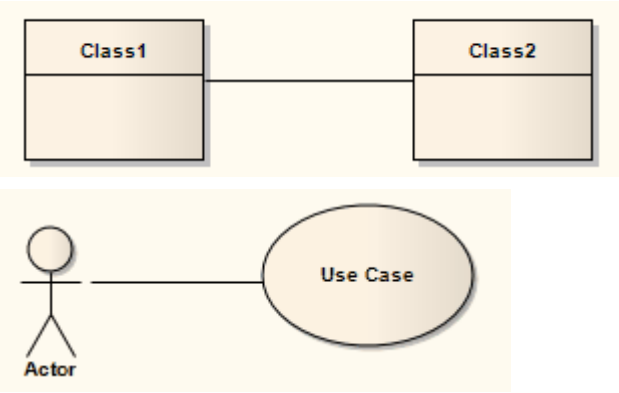
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.210) states:

The execution time semantics for an assembly Connector in a Component are that requests (signals and operation invocations) travel along an instance of a Connector.

Association



Description

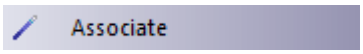
An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes. The connector can include named roles at each end, multiplicity, direction and constraints. You can also indicate the reading direction by adding a Name Direction Indicator arrow to the name-label on the connector (see the *Manage Object Labels* Help topic), and define template binding parameters for an Association connector between a binding Class and a parameterized Class.

Associations act as the keys in providing possible classifiers for a structure of instance elements, and for automatically generating Property (Part) elements on the source SysML Block element in the Association.

When code is generated for Class diagrams, Associations become member variables in the target Class. The relationship is also used in Package, Object, Communication, Data Modeling and Deployment diagrams.

'Association' is the general relationship type between two elements; to connect more than two elements in an Association, you can use the N-Ary Association element. An Association connector can also be integrated with a Class element to form an Association Class, to allow the connector to have operations and attributes that define certain types of UML relationship.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.199) states:

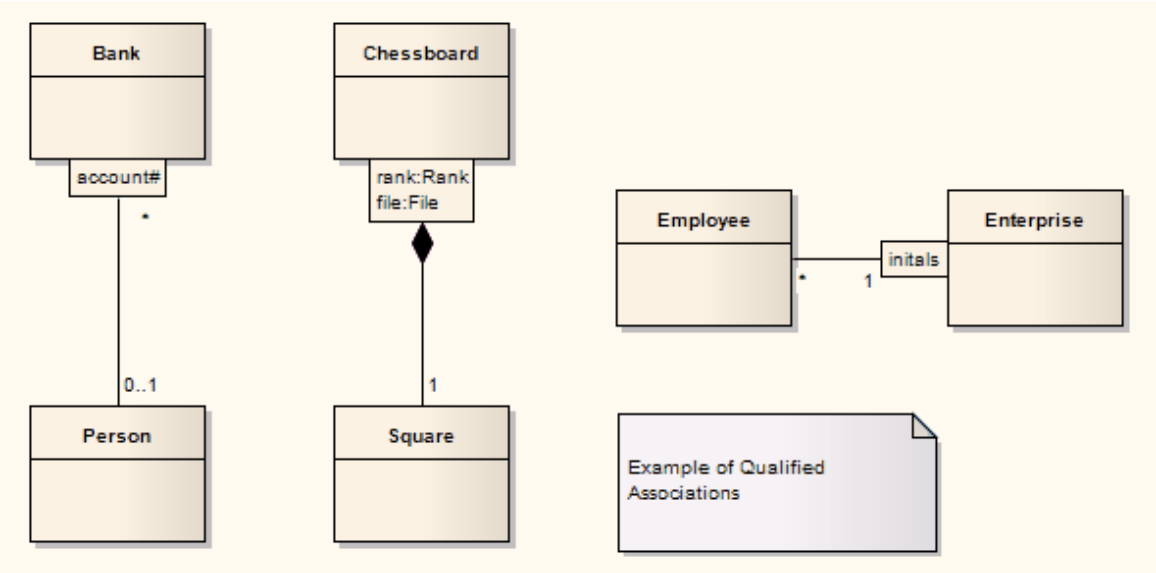
An Association specifies a semantic relationship that can occur between typed instances. It has at least two memberEnds represented by Properties, each of which has the type of the end. More than one end of the Association may have the same type.

An Association declares that there can be links between instances whose types conform to or implement the associated types. A link is a tuple with one value for each memberEnd of the Association, where each value is an instance whose type conforms to or implements the type at the end.

Qualifiers

Qualifiers are ordered sets of properties of an Association end point, a Part, a Port, or an attribute that limit the nature of the relationship between two classifiers or objects. You define a qualifier on the 'Qualifiers' dialog, which you display by clicking on the ... button at the end of the 'Qualifiers' field on the Association, Part, Port or attribute 'Properties' dialog.

Examples



Notes

- When typing multiple Qualifiers into the 'Qualifier(s)' field on a 'Properties' dialog, separate them with a semi-colon; each Qualifier then displays on a separate line; for example, in the diagram the Qualifier 'rank:Rank;file:File' has been rendered in two lines, with a

line break at the ; character

- You can enable or disable Qualifier rectangles in the 'Diagram' page of the 'Preferences' dialog (select the 'Start > Appearance > Preferences > Preferences > Diagram' ribbon option) - if disabled, the old style text Qualifiers are used; it is not recommended that you disable Qualifiers as they are an integral part of the UML
- You can enable or disable a mild shading on the Qualifier rectangles in the 'Links' page of the 'Preferences' dialog

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.129) states:

A qualifier declares a partition of the set of associated instances with respect to an instance at the qualified end (the qualified instance is at the end to which the qualifier is attached). A qualifier instance comprises one value for each qualifier attribute. Given a qualified object and a qualifier instance, the number of objects at the other end of the association is constrained by the declared multiplicity. In the common case in which the multiplicity is 0..1, the qualifier value is unique with respect to the qualified object, and designates at most one associated object. In the general case of multiplicity 0..*, the set of associated instances is partitioned into subsets, each selected by a given qualifier instance. In the case of multiplicity 1 or 0..1, the qualifier has both semantic and implementation consequences. In the case of multiplicity 0..*, it has no real semantic

consequences but suggests an implementation that facilitates easy access of sets of associated instances linked by a given qualifier value.

Qualifiers Dialog

The 'Qualifiers' dialog is used to define the Qualifiers of an Association connector end, Port, Part or Attribute.

General Tab

Review, edit or complete the fields as indicated in the table.

To change the position of a Qualifier in the list in the 'Qualifiers' panel, click on the Scroll Up button or Scroll Down button (the 'hand' buttons).

Field	Action
Name	Display the name of the Qualifier. For a new Qualifier, type the name (with no spaces).
Alias	Display an optional alias for the Qualifier. If necessary, type in a new alias.
Type	Display the Qualifier type. The type can be defined by the code language (data type) or by a classifier element. When you click on the drop-down arrow, the set of values in the list provides the appropriate data types. To select or define possible classifiers,

	either click on the 'Select Type' option in the list, or click on the ... button to display the 'Select <Item>' dialog. To add new code language data types that can be displayed in this list, see the *Data Types* topic.
Scope	Define the Qualifier as Public, Protected, Private or Package. If necessary, click on the drop-down arrow and select a different scope.
Stereotype	Define the optional stereotype of the Qualifier. If necessary, either type a different stereotype name or click on the drop-down arrow and select a stereotype.
Derived	Indicate that the Qualifier is a calculated value. If you select this checkbox, the Qualifier name on the element has the derived symbol (/) as a prefix.
Static	Indicate that the Qualifier is a static member.
Const	Indicate that the Qualifier is a constant.

Initial	Display an optional initial value. If necessary, type in a new initial value.
Notes	Enter any free text notes associated with the Qualifier. You can format the notes text using the Notes toolbar at the top of the field.

Detail Tab

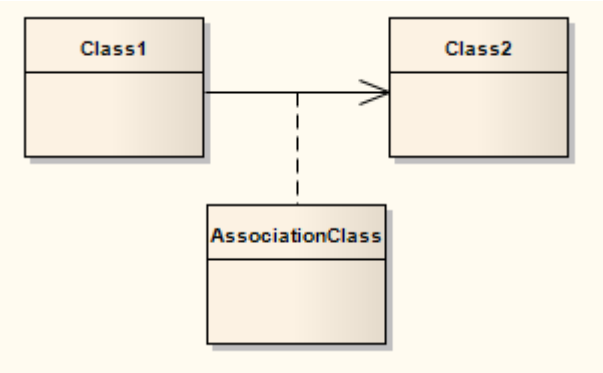
Use the 'Detail' tab to model additional properties of a selected Qualifier, such as its multiplicity, redefined properties and subsetted properties.

Select a Qualifier on the 'General' tab, then review, edit or complete the 'Detail' tab fields as indicated in this table.

Field	Action
Lower bound	Define a lower limit to the number of elements allowed in the collection.
Upper bound	Define an upper limit to the number of elements allowed in the collection.
Allow Duplicates	Indicate that duplicates are allowed. Maps to the UML property isUnique, value FALSE.

Multiplicity is Ordered	Indicate that the collection is ordered.
Redefined Property	Review the redefined properties for the Qualifier. Add redefined properties by clicking on the Add button to display the 'Select Property' dialog.
Subsetted Property	Review the subsetted properties for the qualifier. Add subsetted properties by clicking on the Add button to display the 'Select Property' dialog.

Association Class



Description

An Association Class is a UML construct that enables an Association to have attributes and operations (features). This results in a hybrid relation with the characteristics of an Association and a Class.

When you add an Association Class connection, Enterprise Architect also creates a Class that is automatically connected to the Association. When you hide or delete the Association, the Class is also hidden or deleted.

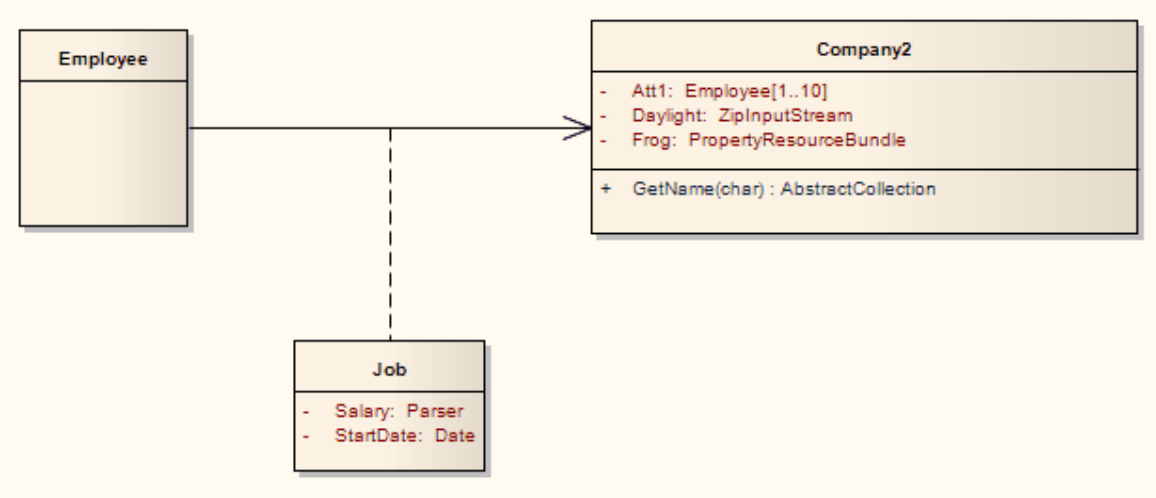
To add an Association Class to a Class or Deployment diagram, click on the 'Association Class' icon in the Toolbox. Click and hold on the source object in the diagram while you drag the line to the target element, then release the mouse button. Enterprise Architect draws the connector and adds the Class, then prompts you to add the Class name. Note that the names of the Class and the connector are the same. You can also connect a new Class to an existing Association.

You can highlight the Class part of an Association Class in

the Browser window, by selecting the 'Find Association Class' context menu option on the Association connector.

Example

This diagram illustrates an Association Class between model elements. Note the dotted line from the Class to the Association. You cannot move or delete this line.



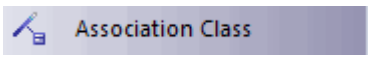
Notes

- If you are applying a stereotype with a Shape Script to an Association Class, be aware that the Shape Script is applied to both the Class part and the Association part; therefore, you might have to include logic in the shape main that tests the type of the element so that you can give separate drawing instructions for Class and for Association
- Such logic is not necessary in the:
    - Shape source or shape target, which are ignored by

Classes, or the
- Decoration shapes, which are ignored by Association connectors

- If you dissociate the Class from the Association connector, both parts keep their Shape Scripts until the stereotypes are removed

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.200) states:

An AssociationClass is a declaration of an Association that has a set of Features of its own. An AssociationClass is both an Association and a Class, and preserves the static and dynamic semantics of both. An AssociationClass describes a set of objects that each share the same specifications of Features, Constraints, and semantics entailed by the AssociationClass as a kind of Class, and correspond to a unique link instantiating the AssociationClass as a kind of Association.

Connect New Class to Existing Association

Connect Class to Association

Step	Action
1	Create a Class in the diagram containing the Association to connect.
2	Right-click on the new Class and select the 'Advanced | Association Class' menu option. The 'Create Association Class' dialog displays.
3	Select the connector to connect to.
4	Click on the OK button.

Communication Path

Description

A Communication Path defines the path through which two DeploymentTargets are able to exchange signals and messages. Communication Path is a specialization of Association. A DeploymentTarget is the target for a deployed Artifact and can be a Node, Property or InstanceSpecification in a Deployment diagram.

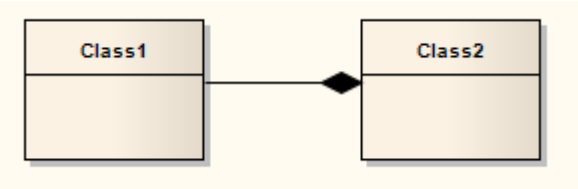
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.661) states:

A communication path is an association between two deployment targets, through which they are able to exchange signals and messages.

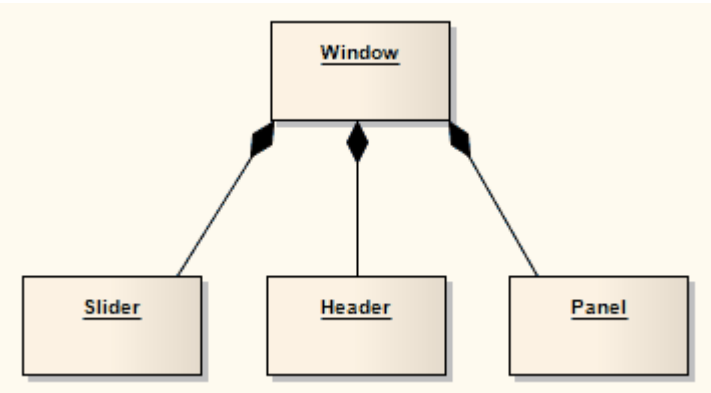
Composition



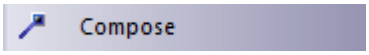
Direction:

A Composition is used to depict an element that is made up of smaller components, typically in a Class or Package diagram. A component - or part instance - can be included in a maximum of one composition at a time. If a composition is deleted, usually all of its parts are deleted with it; however, a part can be individually removed from a composition without having to delete the entire composition. Compositions are transitive, asymmetric relationships and can be recursive.

Example



Toolbox icon



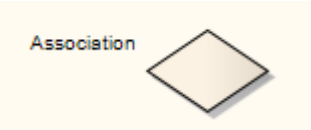
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.112) states:

Composite aggregation is a strong form of aggregation that requires a part object be included in at most one composite object at a time. If a composite object is deleted, all of its part instances that are objects are deleted with it.

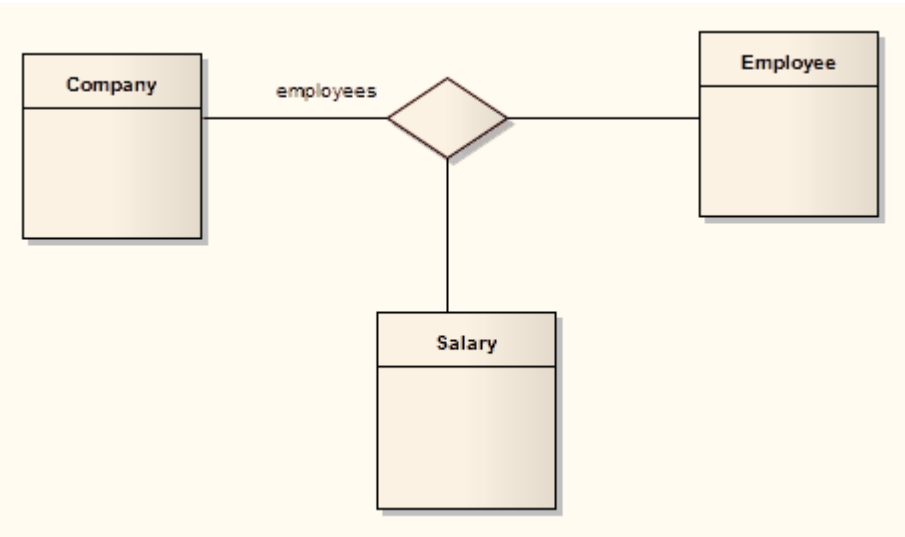
Compositions may be linked in a directed acyclic graph with transitive deletion characteristics; that is, deleting an object in one part of the graph will also result in the deletion of all objects of the subgraph below that object. The precise lifecycle semantics of composite aggregation is intentionally not specified. The order and way in which composed objects are created is intentionally not defined. The semantics of composite aggregation when the container or part is typed by a DataType are intentionally not specified.

N-Ary Association



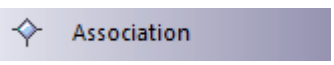
Description

An n-Ary Association element is used to model complex relationships between three or more elements, typically in a Class diagram. It is not a commonly-employed device, but can be used to good effect where there is a dependant relationship between several elements. It is generally used with the Association connector, but the relationships can include other types of connector.



In this example there is a relationship between a Company, an Employee and a Salary.

Toolbox icon

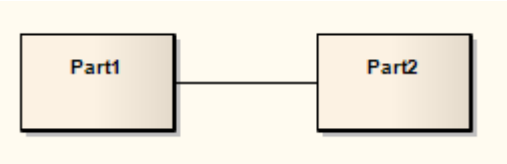


OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.201) states:

Any Association may be drawn as a diamond (larger than a terminator on a line) with a solid line for each Association memberEnd connecting the diamond to the Classifier that is the end's type. An Association with more than two ends can only be drawn this way.

Connector



Description

Connectors illustrate communication links between Parts to fulfill the structure's purpose, typically in a Composite Structure diagram. Each Connector end is distinct, controlling the communication pertaining to its connecting element. These elements can define constraints specifying this behavior. Connectors can have multiplicity.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.227) states:

A Connector specifies links that enables communication between two or more instances. In contrast to Associations, which specify links between any instance of the associated Classifiers, Connectors specify links between instances playing the connected parts only.

Control Flow

Description

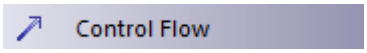
The Control Flow is a connector connecting two nodes in an Activity diagram, modeling an active transition. Control Flow connectors bridge the flow between Activity nodes, by directing the flow to the target node once the source node's activity is completed.

Control Flows and Object Flows can define a Guard and a Weight condition.

A Guard defines a condition that must be True before control passes along that activity edge. A practical example of this is where two or more activity edges (Control Flows) exit from a Decision element. Each flow should have a Guard condition that is exclusive of the other and defines which edge is taken under what conditions. The Control Flow 'Properties' dialog enables you to set up Guard conditions on Control Flows and on Object Flows.

A Weight defines the number of tokens that can flow along a Control or Object Flow connection when that edge is traversed. Weight can also be defined on the Control Flow and Object Flow 'Properties' dialogs.

Toolbox icon

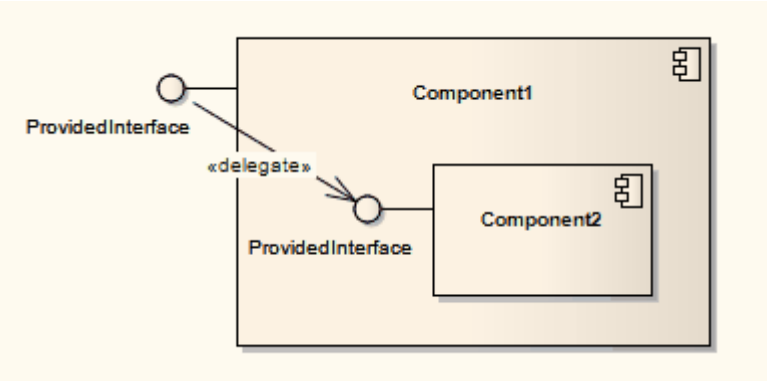


OMG UML specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.376) states:

A ControlFlow is an ActivityEdge that only passes control tokens (and some object tokens as specified by modelers, ...). ControlFlows are used to explicitly sequence execution of ActivityNodes, as the target ActivityNode cannot receive a control token and start execution until the source ActivityNode completes execution and produces the token.

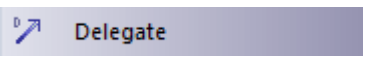
Delegate



Description

A Delegate connector defines the internal assembly of a component's external Ports and Interfaces, on a Component diagram. Using a Delegate connector wires the internal workings of the system to the outside world, by a delegation of the external interfaces' connections.

Toolbox icon



OMG UML Specification:

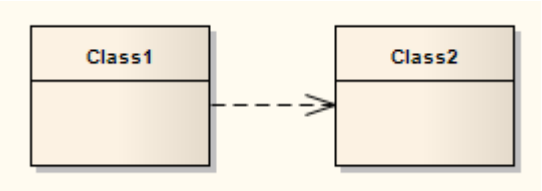
The OMG Unified Modeling Language specification, (v2.5.1, p.191) states:

A delegation Connector is a Connector that links a Port to a role within the owning EncapsulatedClassifier. It represents the forwarding of requests (Operation invocations and

Signals). A request that arrives at a Port that has a delegation Connector to one or more Properties or Ports on Properties will be passed on to those targets for handling.

Delegation Connectors can be used to model the hierarchical decomposition of behavior, where services provided by an EncapsulatedClassifier may ultimately be realized by one that is nested multiple levels deep within it.

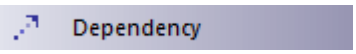
Dependency



Description

Dependency relationships are used to model a wide range of dependent relationships between model elements in Use Case, Activity and Structural diagrams, and even between models themselves. You can create the Dependency from the Common page of the Toolbox. The Dependencies Package as defined in UML 2.1 has many derivatives, such as Realize, Deployment and Use. Once you create a Dependency you can further refine its meaning by applying a specialized stereotype.

Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.42) states:

A Dependency is a Relationship that signifies that a single model Element or a set of model Elements requires other model Elements for their specification or implementation.

This means that the complete semantics of the client Element(s) are either semantically or structurally dependent on the definition of the supplier Element(s).

Apply a Stereotype

This topic defines how to apply a stereotype to a Dependency relationship.

Apply Stereotype

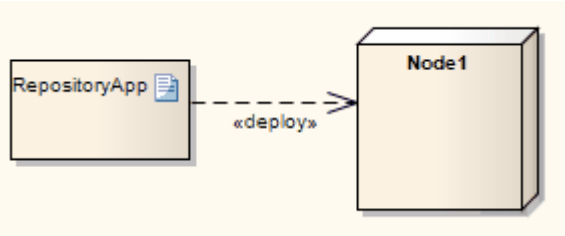
Step	Action
1	Select the Dependency relationship to change.
2	Right-click on the connector and select the 'Dependency Properties' option. The 'Dependency Properties' dialog displays.
3	In the 'Stereotype' field, either type in the required stereotype name or click on the drop-down arrow and select the stereotype from the list.
4	Click on the OK button.

Alternatively

Right-click on the Dependency relationship and select the 'Advanced | Dependency Stereotypes' option, then select

from a shorter list of standard stereotypes.

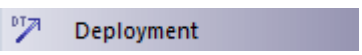
Deployment



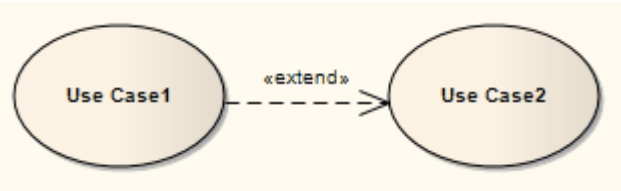
Description

A Deployment is a type of Dependency relationship that indicates the deployment of an artifact onto a node or executable target, typically in a Deployment diagram. A Deployment can be made at type and instance levels. At the type level, a Deployment would be made for every instance of the node. Deployment can also be specified for an instance of a node, so that a node's instances can have varied deployed artifacts. With composite structures modeled with nodes defined as Parts, Parts can also serve as targets of a Deployment relationship.

Toolbox icon



Extend



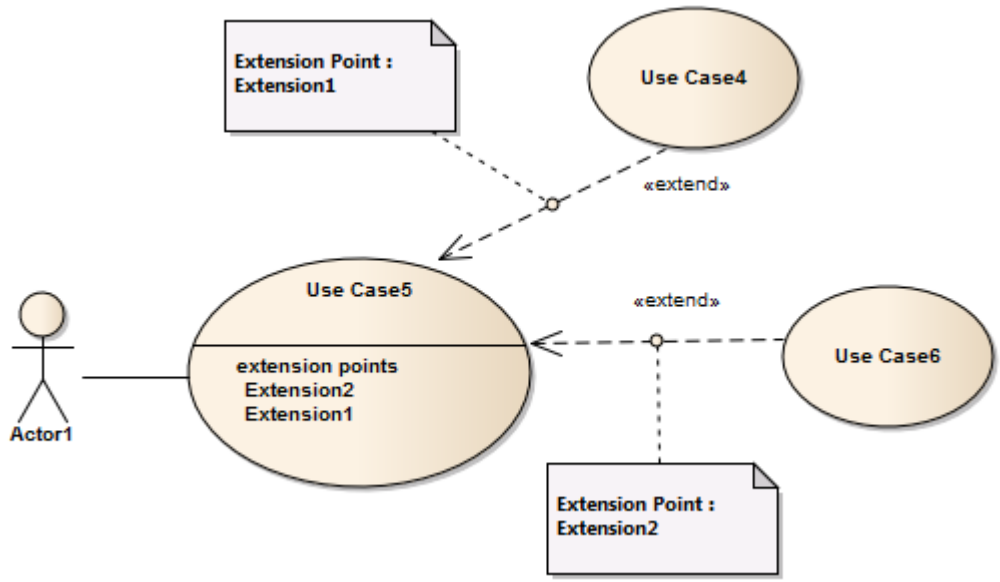
Description

An Extend connector is used to indicate that an element extends the behavior of another, mainly in Use Case models where one Use Case (optionally) extends the behavior of another Use Case. An extending Use Case often expresses alternative flows that are integrated with the behavior of the extended Use Case, at a specific point in the behavior flow identified within the element by an extension point. The extension point is represented by a text string such as 'on startup' or 'before connection is established'.

A Use Case can have more than one extension point, and can extend or be extended by more than one other Use Case. The precise relationship between the extending Use case, extended Use Case and the point at which the extension applies can be identified on the Extend relationship, as shown.

Identify Extension Point

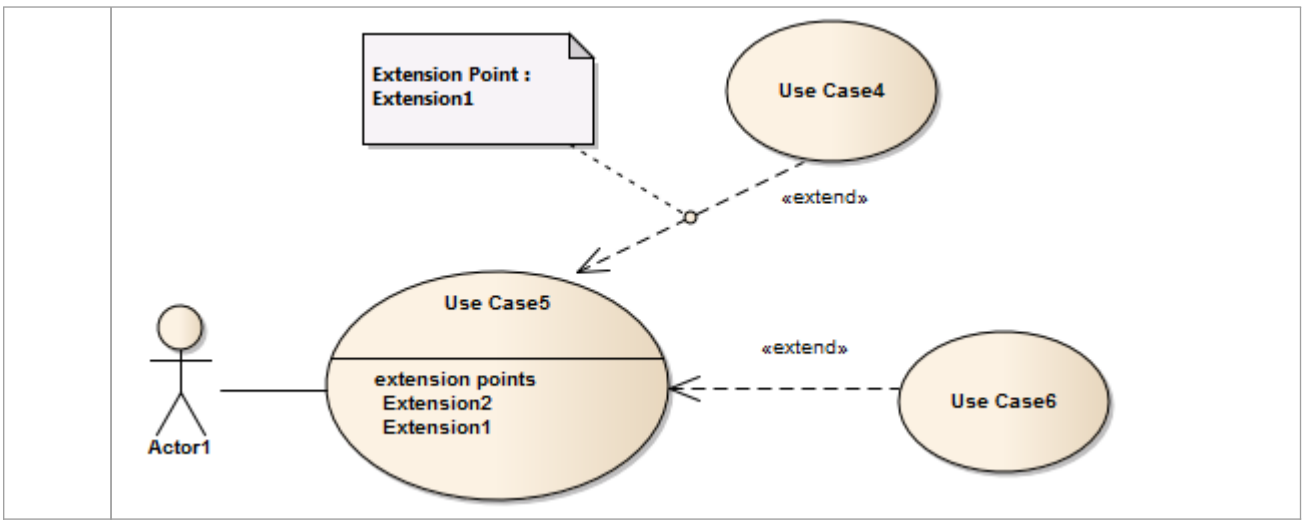
Step	Action

1	Right-click on the Extend connector and select the 'Advanced | Extension Point | Set Extension Point' option. The 'Element Usage' dialog displays, listing the Extension Points currently defined in the target Use Case element.
2	Click on the Extension Point on which the source Use Case acts, and click on the Open button. The dialog closes and the Extend connector shows a small circle at the mid-point, with a Notelink to a Note element that identifies the selected Extension Point.  (The Note might not initially display close to the Extend connector - check the upper left corner of the diagram and drag the Note to the position you want it to occupy.)

	Use these same steps to change the extension point identified in the Note.

Show/Hide Extension Point Note

Step	Action
1	Right-click on the Extend connector and select the 'Advanced | Extension Point | Show Extension Point' option. If there are any Extension Points identified on the selected Extend connector, they are displayed as shown.
2	Right-click on the Extend connector and deselect the 'Advanced | Extension Point | Show Extension Point' option. Any Extension Points identified on the selected Extend connector are hidden, as shown:



Toolbox icon

Notes

- The Extend connector is not the same as the Extension connector, which is used in Profile diagrams to indicate that a Stereotype element extends a Metaclass or another Stereotype element; the two types of connector have different appearances

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.640-641) states:

An Extend is a relationship from an extending UseCase (the extension) to an extended UseCase (the extendedCase) that

specifies how and when the behavior defined in the extending UseCase can be inserted into the behavior defined in the extended UseCase. The extension takes place at one or more specific extension points defined in the extended UseCase. Extend is intended to be used when there is some additional behavior that should be added, possibly conditionally, to the behavior defined in one or more UseCases. The extended UseCase is defined independently of the extending UseCase and is meaningful independently of the extending UseCase. On the other hand, the extending UseCase typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending UseCase defines a set of modular behavior increments that augment an execution of the extended UseCase under specific conditions.

NOTE. The same extending UseCase can extend more than one UseCase. Furthermore, an extending UseCase may itself be extended.

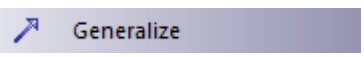
Generalization

Description

A Generalization is used to indicate inheritance. Drawn from the specific classifier to a general classifier, the generalization's implication is that the source inherits the target's characteristics. It is used typically in Class, Component, Object, Package, Use Case and Requirements diagrams.

You can also define template binding parameters for a Generalize connector between a binding Class and a parameterized Class.

Toolbox icon

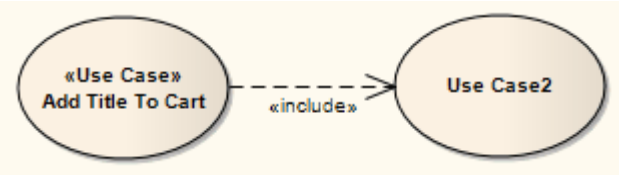


OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.138) states:

A Generalization is a taxonomic relationship between a more general Classifier and a more specific Classifier. Each instance of the specific Classifier is also an instance of the general Classifier. The specific Classifier inherits the features of the more general Classifier. A Generalization is owned by the specific Classifier.

Include



Description

An Include connection indicates that the source element includes the functionality of the target element. Include connections are used in Use Case models to reflect that one Use Case includes the behavior of another. Use an Include relationship to avoid having the same subset of behavior in many Use Cases; this is similar to delegation used in Class models.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.641) states:

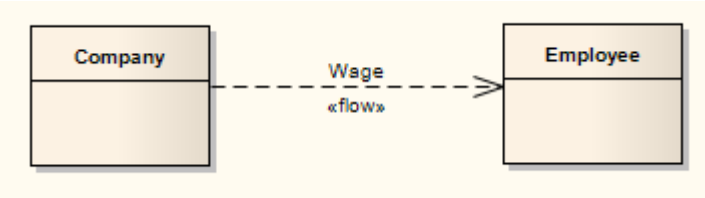
Include is a DirectedRelationship between two UseCases, indicating that the behavior of the included UseCase (the addition) is inserted into the behavior of the including UseCase (the includingCase). It is also a kind of NamedElement so that it can have a name in the context of

its owning UseCase (the includingCase). The including UseCase may depend on the changes produced by executing the included UseCase. The included UseCase must be available for the behavior of the including UseCase to be completely described.

The Include relationship is intended to be used when there are common parts of the behavior of two or more UseCases. This common part is then extracted to a separate UseCase, to be included by all the base UseCases having this part in common. As the primary use of the Include relationship is for reuse of common parts, what is left in a base UseCase is usually not complete in itself but dependent on the included parts to be meaningful. This is reflected in the direction of the relationship, indicating that the base UseCase depends on the addition but not vice versa.

All of the behavior of the included UseCase is executed at a single location in the included UseCase before execution of the including UseCase is resumed.

Information Flow



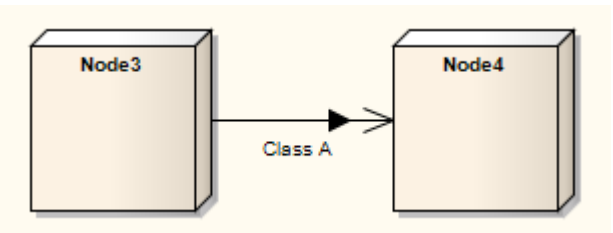
Description

An Information Flow represents the flow of Information Items (either Information Item elements or classifiers) between two elements in any diagram. The connector is available from:

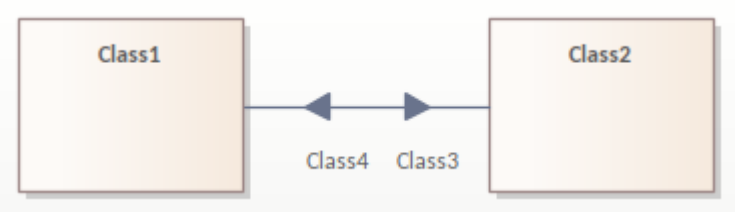
- The 'Common' page of the Toolbox
- Every Quick Link menu, and
- Automatically whilst directly defining Information Item realization

When you create the Information Flow connector, Enterprise Architect automatically prompts you to identify which information items are conveyed.

You can have more than one Information Flow connector between the same two elements, identifying which items flow between the elements under differing conditions. The connector can flow in one direction:

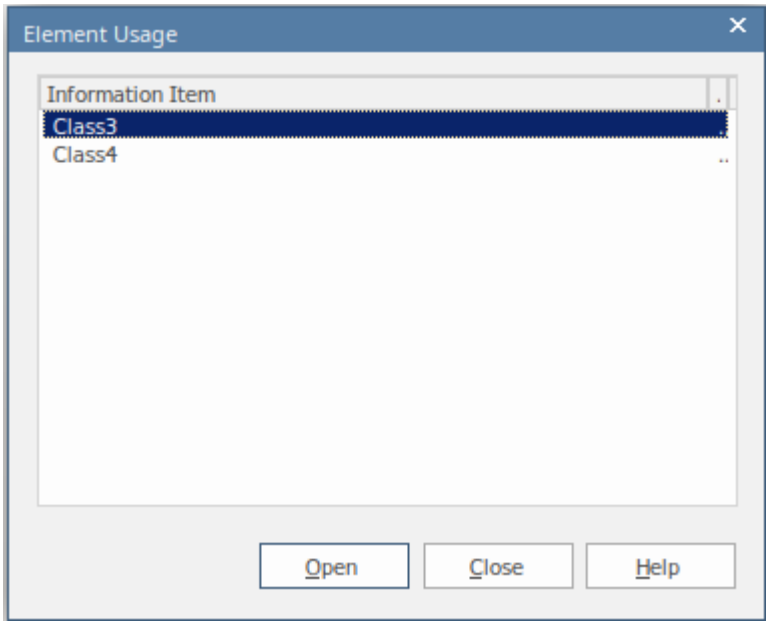


They can also be defined to flow in opposite directions:

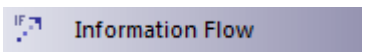


For more details see the *Using Information Flows* Help topic.

You can locate the items conveyed in any Information Flow, by right-clicking on the connector and selecting the 'Find Items Conveyed' option.



Toolbox icon



OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.669) states:

InformationFlows describe circulation of information

through a system in a general manner. They do not specify the nature of the information, mechanisms by which it is conveyed, sequences of exchange, or any control conditions. During more detailed modeling, representation and realization links may be added to specify which model elements implement an InformationFlow and to show how information is conveyed.

The OMG Unified Modeling Language specification, (v2.5.1, p.670) also states:

InformationItems represent many kinds of information that can flow from sources to targets in very abstract ways. They represent the kinds of information that may move within a system, but do not elaborate details of the transferred information. Details of transferred information are the province of other Classifiers that may ultimately define InformationItems.

Using Information Flows

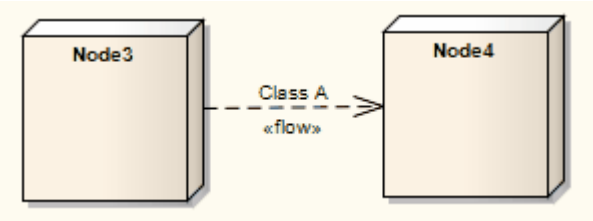
When you drag an Information Flow connector between two elements on a diagram, Enterprise Architect automatically prompts you to identify the Information items conveyed.

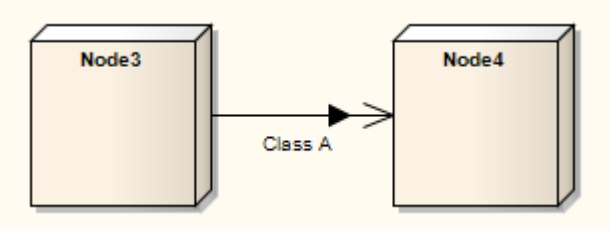
You can also create an Information Flow automatically whilst directly defining Information Flow realization, as you might do on a Message on a Sequence diagram.

If you tend to create, populate and realize Information Flows immediately in one work session, between Classifiers and between Ports and/or Parts, you will appreciate the fact that you can now set up the Information Flow quickly and completely just on the 'Information Items Conveyed' dialog, using the 'Add Realizing Relationship' checkbox.

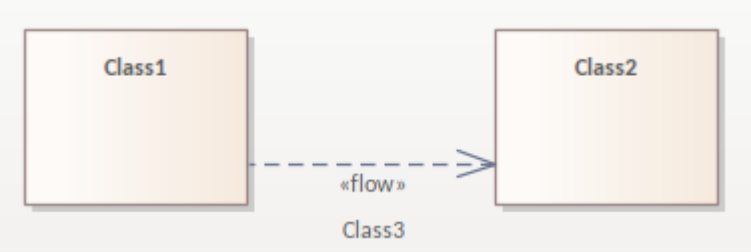
Create and realize Information Flows

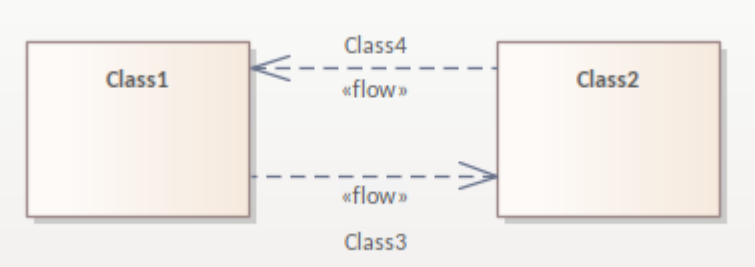
Step	Action
1	Open a diagram and add two elements (for example, Nodes on a Deployment diagram).
2	Click on the Information Flow connector in the 'Common' page of the Toolbox and drag the cursor between the two elements. The 'Information Items Conveyed' dialog displays.

3	Add the classifier or Information Item element(s) to the Information Flow. If you cannot complete this now, you can return to the Information Flow and add the remaining Information items later.
4	If you are creating an Information Flow between two Classifier elements (such as Class, Component or Use Case), or between Ports and/or Parts, the 'Add Realizing Relationship' checkbox is enabled (although not for other combinations of elements). If you have finished assigning information items to the flow, select this checkbox and click on the OK button: • For a relationship between two Classifiers, a new Association connector is automatically created between the two elements, realizing the Information Flow • For a relationship between Ports and/or Parts, a new Connector-type connector is automatically created between the two elements, realizing the Information Flow In these two cases, refer to the illustration in step 7. Otherwise, click on the OK button to close the dialog. The diagram now resembles this example: 

5	If you have finished assigning Information Items to the flow, add another connector between the same two elements (for example, a Communication Path connector). If you have not finished assigning information items you can return to the realization at a later time, either using the 'Information Flows Realized' dialog or - for flows between classifiers and between Ports and/or Parts - returning to the 'Information Items Conveyed' dialog and selecting the 'Add Realizing Relationship' checkbox.
6	Right-click on the connector and select the 'Advanced | Information Flows Realized' option. The 'Information Flows Realized' dialog displays.
7	Tick the checkbox against each required information item in the realized flow and click on the OK button. The connector now resembles this example, where the black triangle indicates the presence and direction of the Information Flow connector: 

Create bi-directional Information Flows

Step	Action
1	Open a diagram and add four elements (for example four Classes).
2	Using the Quicklinker, drag an Information Flow from Class1 to Class2. The *Information Items Conveyed* dialog appears. • Click the *Add* button and select Class3. Ensure the option *'Add Realizing Relationship'* is NOT ticked. • Click OK. Move the connector away from the center to ensure the visibility of the next connector. 
3	Using the Quicklinker, drag an Information Flow from Class2 to Class1. The *Information Items Conveyed* dialog appears. • Click the *Add* button and select Class4. Ensure the option *'Add Realizing Relationship'* is NOT ticked.

	• Click OK. Move the connector away from the center to ensure the visibility of the next connector. 
4	Using the Quicklinker, drag an Association between Class1 and Class2. • Right-click on the Association and select: Advanced | Information Flows Realized • In the *Information Flows Realized* dialog, select Class3 and Class4 and click on the OK button.

Notes

- Once the Information Flow is realized, you cannot access the 'Information Items Conveyed' dialog directly; to add or delete information items on the connector, you 'unrealize' the connector on the 'Information Items Realized' dialog
- If you have more than one Information Flow connector

between the elements, they form part of the same combined connector; you can again work on them separately through the 'Information Items Realized' dialog

- If you have information flows in a diagram that you use as the source for a Pattern, the 'Information Items Conveyed' and 'Information Flows Realized' data is not copied into the Pattern
- You can locate, in the Browser window, the classifier or information item element(s) conveyed on the Information Flow connector, using the 'Find Items Conveyed' context menu option on the connector

Convey Information on a Flow

When you create an Information Flow connector between two elements, Enterprise Architect automatically prompts you to specify which Information Items or classifiers are conveyed on this flow. If you do not realize the Information Flow with its existing information items immediately, you can change and/or add to the information items conveyed at a later time. The menu path helps you to return to an incomplete Information Flow, but the process steps apply to both new and unfinished flows.

Access

Context Menu	Right-click on Information Flow connector | Advanced | Information Items Conveyed
Diagram	Drag a Classifier from the Browser, the Diagram Toolbox or from the diagram itself and drop it onto an Information Flow connector, to add that Classifier as a conveyed item

Specify the Information Items conveyed on an Information Flow

Step	Action
1	On the 'Information Items Conveyed' dialog, click on the Add button. The 'Select Classifier' dialog displays.
2	Browse or search for the required Information Item or classifier element or elements, and select them as required. If you do not want to retain a selected item on the 'Select Classifier' dialog, press Ctrl and click on the item.
3	Click on the OK button to return to the 'Information Items Conveyed' dialog. Each information item you have selected is listed on a separate line in the dialog.
4	If you do not want to retain a selected item on this dialog, click on it and click on the Remove button. For a link between Classifiers or between Ports and/or Parts, and if the Information Flow is complete, you can create the realizing connector by selecting the 'Add Realizing Relationship' option. For Information Flows between other types of element, you can perform the realization separately.

	Click on the OK button to close the dialog and to show the selected information item element names on the Information Flow connector label.

Realize an Information Flow

After you create a UML information Flow connector you might want to:

- Realize one or more existing flows on the Information Flow connector
- Edit an existing flow on the Information Flow connector

You might also want to create and realize information flows on a non Information Flow connector, such as a Message on a Sequence diagram. You can perform these actions using the 'Information Flows Realized' dialog, which displays all existing flows that can be realized on the selected connector.

For a relationship between Classifiers or between Ports and/or Paths, having finished assigning flow items to the Information Flow you can also automatically realize it by selecting the 'Add Realizing Relationship' option on the 'Information Items Conveyed' dialog.

Access

Context Menu	Right-click on connector | Advanced | Information Flows Realized

Review Item Flows on an Information Flow Connector

Operation	Action
Realize Information Flows on the selected connector	Select the checkbox for each required flow and click on the OK button.
Cancel realization of a flow	Deselect the checkbox against the appropriate flow, and click on the OK button.
Change the classifier or Information Item elements conveyed on an Information Flow	Click on the appropriate Information Item row, and on the ... button at the right-hand end of it. The 'Select Classifier' dialog displays: • Click on a single item to select it • Ctrl+click on each of several items to select them all, or • Ctrl+click on a selected item to deselect it Click on the OK button to return to the 'Information Flows Realized' dialog and, if required, realize the changed flow.
Create a realized	1. Right-click on the connector and select the 'Information Flows Realized'

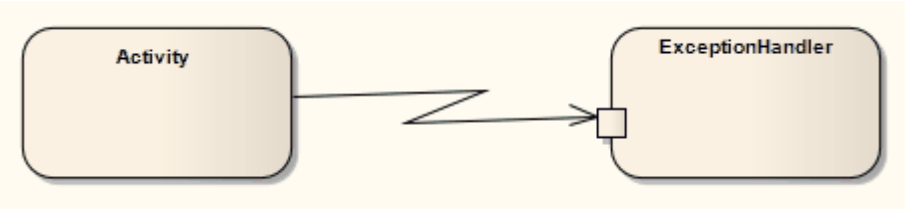
information flow directly on a new connector	option. 2. Click on the *Click to create new information flow...* text. The 'Select Classifier' dialog displays. 3. Select the required classifier or Information Item elements, and click on the OK button to return to the 'Information Flows Realized' dialog; the selected elements are listed first on the dialog, with the activation checkbox ticked. 4. Click on the OK button to return to the diagram; the connector now displays as a realized Information Flow, with the selected classifier or Information Item elements named in the connector label.

Notes

- If there are several Information flows and you do not realize all of them, those that are not realized are represented by a separate Information Items Conveyed iteration of the Information Flow connector; you can only realize those flows on the original connector, at which point the flow is represented on that original connector
- If you realize all of the flows, they are combined on the one connector line

- If you realize an information flow on a connector, you can use the 'Find Items Conveyed' context menu option to locate the corresponding Information Flow item in the Browser window
- You realize Information Flows on UML connectors only; you cannot realize Information Flows on, say, ArchiMate connectors, so the menu option is not provided for them

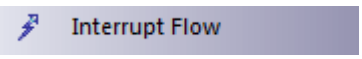
Interrupt Flow



Description

The Interrupt Flow is a connection used to define the two UML concepts of connectors for Exception Handler and Interruptible Activity Region. An Interrupt Flow is a type of activity edge. It is typically used in an Activity diagram, modeling an active transition.

Toolbox icon

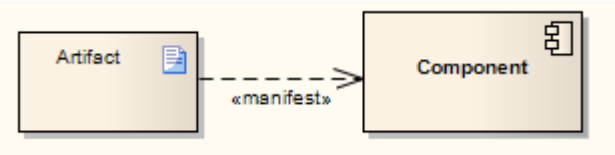


OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.375) states:

An ActivityEdge is a directed connection between two ActivityNodes along which tokens may flow, from the source ActivityNode to the target ActivityNode..

Manifest



Description

A Manifest relationship indicates that the Artifact source embodies the target model element, typically in Component and Deployment diagrams. Stereotypes can be added to Enterprise Architect to classify the type of manifestation of the model element.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.657) states:

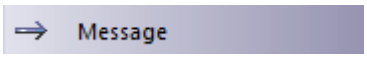
An Artifact may embody, or manifest, a number of model elements. The Artifact owns the Manifestations, each representing the utilization of some PackageableElement. Profiles may extend the Manifestation relationship to indicate particular forms of embodiment. For example, «tool generated» and «custom code» might be two Manifestations for different Classes embodied in an Artifact.

Message

Messages indicate a flow of information or transition of control between elements. Messages can be used in Timing diagrams, Sequence diagrams and Communication diagrams (but not Interaction Overview diagrams) to reflect system behavior. If between Classes or classifier instances, the associated list of operations is available to specify the event.

Moving a Message can disrupt the organization of other features on the diagram. To avoid this, and move only the Message, press Alt while you move the Message.

Toolbox icon



OMG UML Specification:

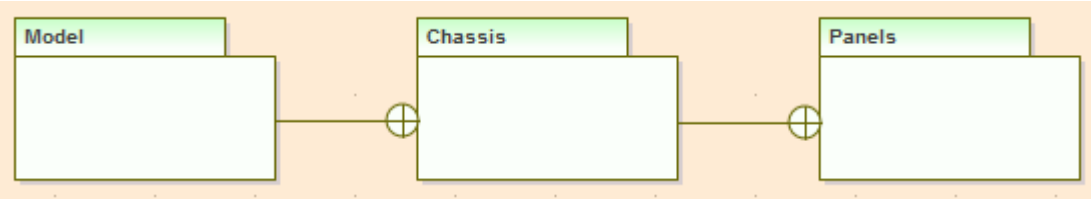
The OMG Unified Modeling Language specification, (v2.5.1, p.623) states:

A Message defines a particular communication between Lifelines of an Interaction.

The OMG Unified Modeling Language specification, (v2.5.1, p.574) also states:

The signature of a Message refers to either an Operation or a Signal. The name of the Message must be the same as the name of the referenced Operation or Signal.

Nesting

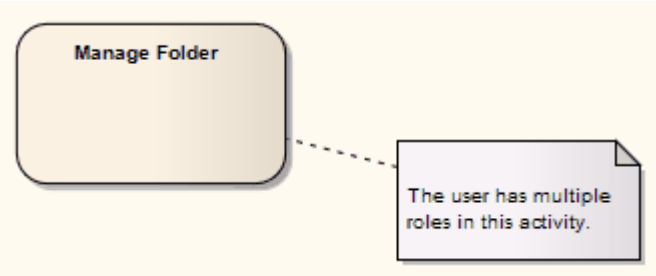


Description

The Nesting Connector is an alternative graphical notation for expressing containment or nesting of elements within other elements. It is most appropriately used for displaying Package nesting in a Package diagram.

Toolbox icon

Notelink



Description

A Notelink connector connects a Note to one or more other elements of any other type.

Both Note and Notelink are available in any category of the Toolbox, in the Common page. You can also select them from the UML Elements toolbar.

Toolbox icon

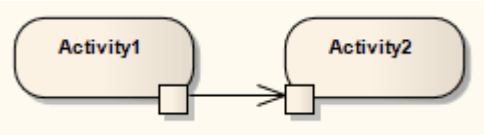
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.22) states:

A Comment is shown as a rectangle with the upper right corner bent (this is also known as a “note symbol”). The rectangle contains the body of the Comment. The connection to each annotatedElement is shown by a separate dashed line. The dashed line connecting the note symbol to

the annotatedElement(s) may be suppressed if it is clear from the context, or not important in this diagram.

Object Flow



Description

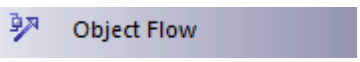
Object Flows are used in Activity diagrams and StateMachine diagrams. When used in an Activity diagram, an Object Flow connects two elements, with specific data passing through it, modeling an active transition. To view sample Activity diagrams using Object Flows, see the *Object Flows in Activity Diagrams* topic.

In StateMachine diagrams, an Object Flow is a specification of a state flow or transition. It implies the passing of an Object instance between elements at run-time.

You can insert an Object Flow from the 'State' or 'Activity' pages of the Toolbox, or from the drop-down list of all relationships located in the header toolbar. You can also modify a transition connection to an Object Flow by selecting the 'ObjectFlow' checkbox on the connection 'Properties' dialog.

See the *Control Flow* topic for information on setting up Guards and Weights on Object Flows.

Toolbox icon



OMG UML Specification:

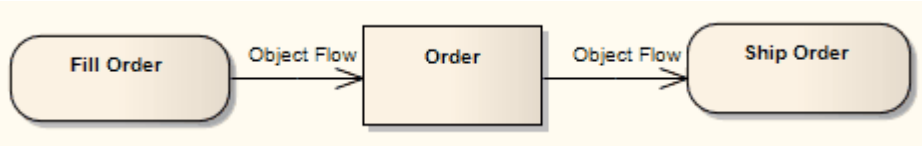
The OMG Unified Modeling Language specification, (v2.5.1, p.376) states:

An ObjectFlow is an ActivityEdge that can have object tokens passing along it. ObjectFlows model the flow of values between ObjectNodes. Tokens are offered to the target ActivityNode in the same order as they are offered from the source. If multiple tokens are offered at the same time, then the tokens are offered in the same order as if they had been offered one at a time from the source. If the source is an ObjectNode with an ordering specified, then tokens from the source are offered to the ObjectFlow in that order and, consequently, are offered from the ObjectFlow to the target in the same order

Object Flows in Activity Diagrams

In Activity diagrams, there are several ways to define the flow of data between objects.

This diagram depicts a simple Object Flow between two actions, Fill Order and Ship Order, both accessing order information.

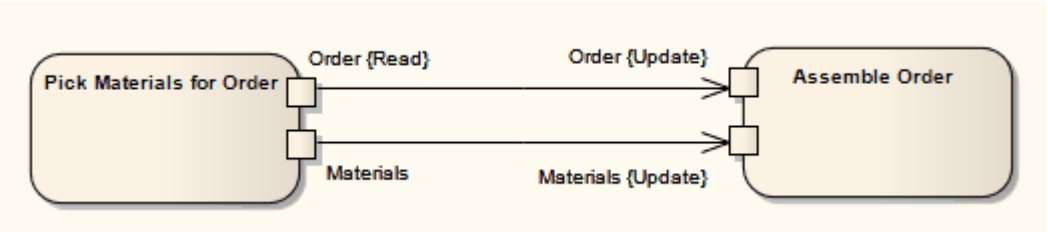


See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.110, p.391.)

This explicit portrayal of the data object Order, connected to the Activities by two Object Flows, can be refined by using this format. Here, Action Pins are used to reflect the order.

See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.110, p.391.)

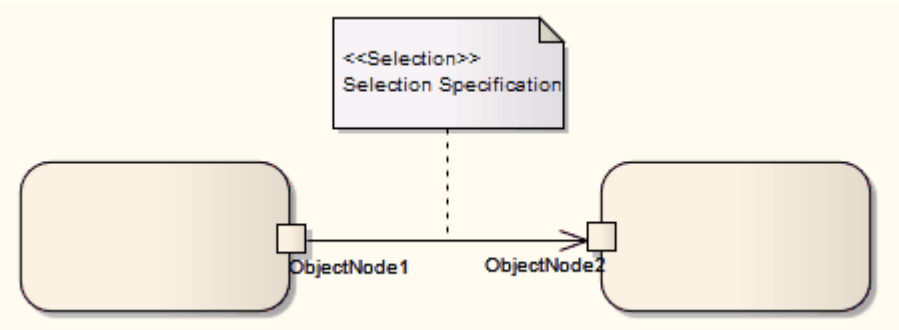
This diagram is an example of multiple Object Flows exchanging data between two actions.



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.111, p.391.)

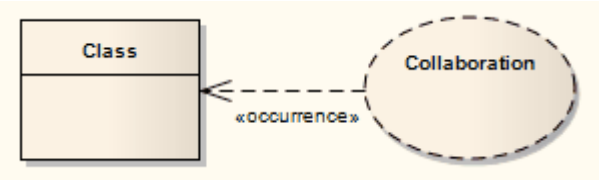
Selection and transformation behavior, together composing a sort of query, can specify the nature of the Object Flow's data access. Selection behavior determines which objects are affected by the connection. Transformation behavior might then further specify the value of an attribute pertaining to a selected object.

Selection and transformation behaviors can be defined by attaching a note to the Object Flow. To do this, right-click on the Object Flow and select the 'Attach Note or Constraint' option. A dialog lists other flows in the diagram to which you can select to attach the note, if the behavior applies to multiple flows. To comply with UML 2, preface the behavior with the notation «selection» or «transformation».



See the OMG Unified Modeling Language specification, (v2.5.1, figure 12.112, p.392.)

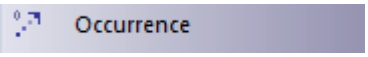
Occurrence



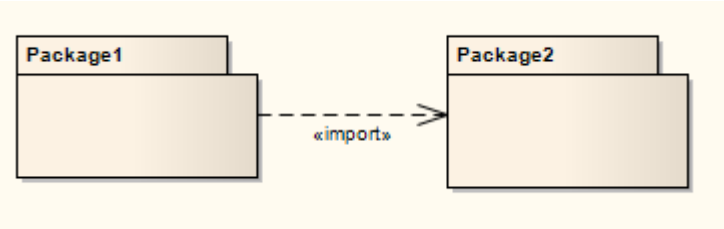
Description

An Occurrence relationship indicates that a Collaboration represents a classifier, in a Composite Structure diagram. An Occurrence connector is drawn from the Collaboration to the classifier.

Toolbox icon



Package Import



Description

A Package Import relationship is drawn from a source Package to a Package whose contents are to be imported. Private members of a target Package cannot be imported. The relationship is typically used in a Package diagram.

Toolbox icon

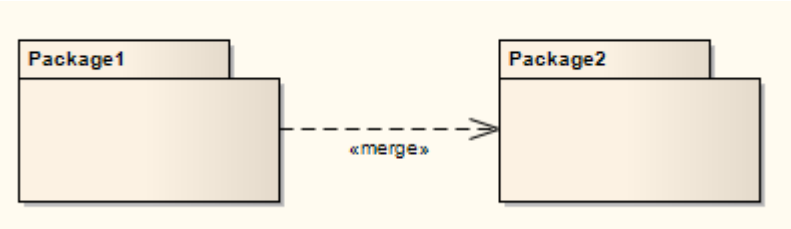
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.28-29) states:

A PackageImport is a DirectedRelationship between an importing Namespace and a Package, indicating that the importing Namespace adds the names of the members of the Package to its own Namespace. Conceptually, a Package import is equivalent to having an ElementImport to each individual member of the imported Namespace, unless there is a separately-defined ElementImport. If there is an

ElementImport for an Element, then this takes precedence over a potential import of the same Element via a PackageImport.

Package Merge



Description

In a Package diagram, a Package Merge indicates a relationship between two Packages whereby the contents of the target Package are merged with those of the source Package. Private contents of a target Package are not merged. The applicability of a Package Merge addresses any situation where multiple Packages contain identically-named elements, representing the same thing. A Package Merge merges all matching elements across its merged Packages, along with their relationships and behaviors. Note that a Package Merge essentially performs generalizations and redefinitions of all matching elements, but the merged Packages and their independent element representations still exist and are not affected.

The Package Merge serves a graphical purpose in Enterprise Architect, but creates an ordered Package relationship applied to related Packages (which can be seen under the 'Link' tab in the Package's 'Properties' dialog). Such relationships can be reflected in XMI exports or Enterprise Architect Automation Interface scripts for code generation or other Model Driven Architecture (MDA) interests.

Package Merge relationships are useful to reflect situations where existing architectures contain functionalities involving similar elements, which are merged in a developing architecture. Merging doesn't affect the merged objects, and supports the common situation of product progression.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.242) states:

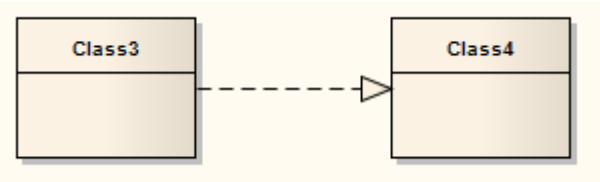
A PackageMerge is a directed relationship between two Packages that indicates that the contents of the target mergedPackage are combined into the source receivingPackage according to a set of rules defined below. It is very similar to Generalization in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both. Just as a subclass is not normally depicted with its inherited features, a receiving Package is not normally depicted with the merged elements from its mergedPackages.

Also, as with Generalization, a Package may not merge itself (directly or indirectly).

This capability is designed to be used when elements

defined in different Packages have the same name and are intended to represent the same concept. A given base concept may be merged for different purposes, with each purpose defined in a separate receiving Package. By selecting different receiving packages, it is possible to obtain a custom definition of a concept for a specific end.

Realization



Description

A source object implements or Realizes its destination object. Realize connectors are used in a Use Case, Component or Requirements diagram to express traceability and completeness in the model. A business process or Requirement is realized by one or more Use Cases, which in turn are realized by some Classes, which in turn are realized by a Component, and so on. Mapping Requirements, Classes and such across the design of your system, up through the levels of modeling abstraction, ensures the big picture of your system remembers and reflects all the little pictures and details that constrain and define it.

You can also define template binding parameters for a Realize connector between a binding Class and a parameterized Class.

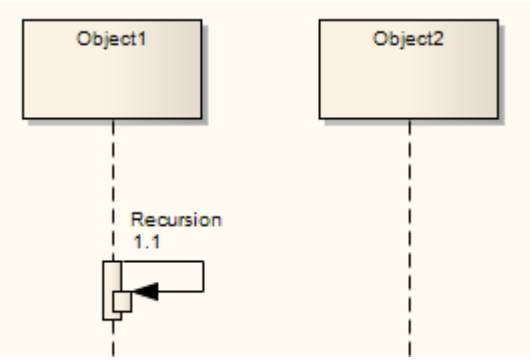
Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, pp.38-39) states:

Realization is a specialized Abstraction dependency between two sets of NamedElements, one representing a specification (the supplier) and the other representing an implementation of that specification (the client). Realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc. A Realization signifies that the set of clients is an implementation of the set of suppliers, which serves as the specification. The meaning of “implementation” is not strictly defined, but rather implies a more refined or elaborate form in respect to a certain modeling context. It is possible to sp

Recursion

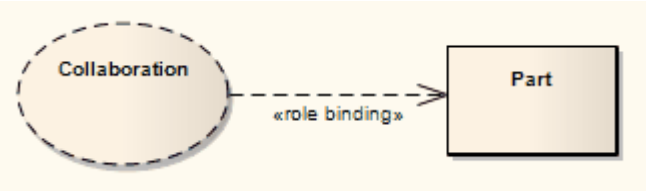


Description

A Recursion is a type of Message used in Sequence diagrams to indicate a recursive function.

Toolbox icon

Role Binding

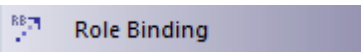


Description

Role Binding is the mapping between a Collaboration Use's internal roles and the respective Parts required to implement a specific situation, typically in a Composite Structure diagram. The associated Parts can have properties defined to enable the binding to occur, and the Collaboration to take place.

A Role Binding connector is drawn between a Collaboration and the classifier's fulfilling roles, with the Collaboration's internal binding roles labeled on the classifier end of the connector.

Toolbox icon



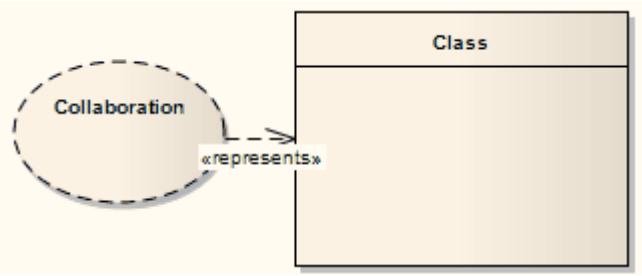
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.216) states:

The roleBindings are implemented using Dependencies

owned by the CollaborationUse. Each collaborationRole in the Collaboration is bound by a distinct Dependency and is its supplier. The client of the Dependency is a ConnectableElement that relates in some way to the context Classifier: it may be a direct collaborationRole of the context Classifier, or an element reachable by some set of references from the context Classifier. These roleBindings indicate which ConnectableElement from the context Classifier plays which collaborationRole in the Collaboration

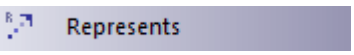
Represents



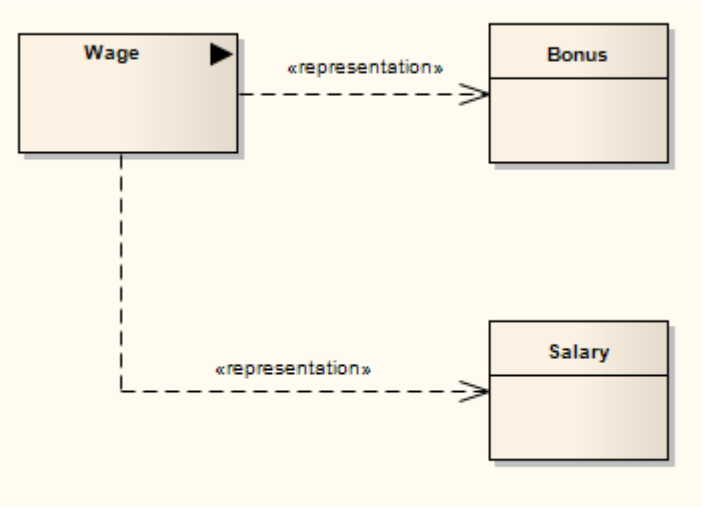
Description

The Represents connector indicates that a Collaboration is used in a classifier, typically in a Composite Structure diagram. The connector is drawn from the Collaboration to its owning classifier.

Toolbox icon



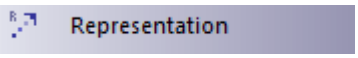
Representation



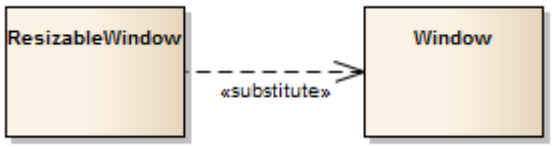
Description

The Representation relationship is a specialization of a Dependency, connecting Information Item elements that represent the same idea across models, typically in an Analysis diagram. For example, 'Bonus' and 'Salary' are both a representation of the Information Item 'Wage'.

Toolbox icon



Substitution

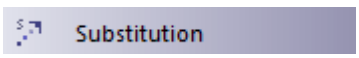


Description

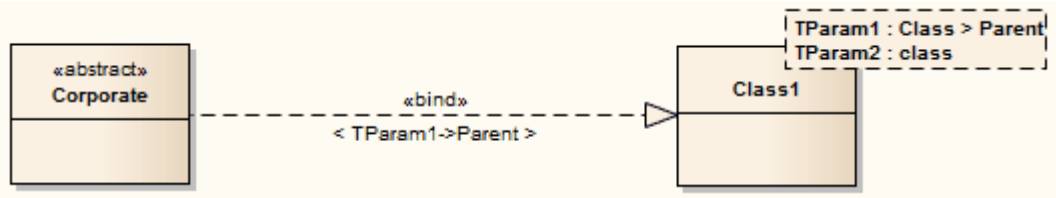
A Substitution is a relationship between two Classifiers, signifying that the substituting Classifier complies with the contract specified by the contract Classifier. This implies that instances of the substituting Classifier are runtime-substitutable, where instances of the contract Classifier are expected. In the example, the Class named ResizableWindow has a Substitution connector to the Class named Window, meaning that wherever you are asked for a window you can use a resizable window.

The Substitution relationship is a subtype of a Dependency relationship.

Toolbox icon



Template Binding



Description

You create a Template Binding connector between a binding Class and a parameterized Class. You then define a binding expression on that connector. However, if the binding Class requires a Generalization, Realization or Association relationship with the parameterized Class, you can define the binding expression on that relationship instead.

You can create a Template Binding connector using:

- The 'Template Binding' icon on the 'Class Relationships' page of the Diagram Toolbox
- The Quick Linker arrow next to the source Class element
- The 'Templates' dialog for the binding Class element; here, you create the Template Binding relationship by clicking the Add button under the 'Binding(s)' panel, specifying the connector type, and selecting the target parameterized Class from the 'Select <Item>' dialog

Each of these methods creates the connector itself. For the first two methods you then click on the connector to make it the focus of the Properties window, on which you select the 'Binding' tab to define parameter substitutions as the binding expression. The third method takes you to the same tab on

the 'Properties' dialog automatically.

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.24) states:

A template is a parameterized element ... used to generate other model elements using TemplateBinding relationships. The template parameters for the template signature specify the formal parameters that will be substituted by actual parameters (or the default) in a binding.

Parameter Substitution

Once a Template Binding (or other binding) relationship exists, you can add parameter substitutions to identify the formal parameters that are replaced, and the actual parameters that replace them, in the binding expression.

Access

Display the 'Binding' page of the connector's 'Properties' dialog or Properties window, using any of the methods outlined in this table.

Ribbon	Start > Application > Design > Properties > click on connector > Binding (Properties window) Design > Element > Editors > Properties > click on connector > Binding (Properties window)
Context Menu	On diagram | Right-click connector | Properties > Binding ('Properties' dialog)
Keyboard Shortcuts	Ctrl+2 > click on connector > Binding (Properties window)
Other	On diagram | Double-click on connector > Binding ('Properties' dialog)

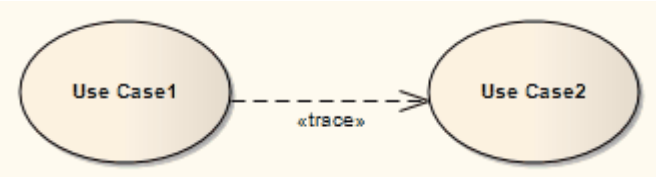
Define a parameter substitution

The 'Target' field identifies the target parameterized Class.

Step	Action
1	Click on the Add button below the 'Parameter Substitution(s)' panel. The next available row in the panel is enabled for editing, and the word '<none>' is displayed in the 'Formal' column.
2	Click on the field and on the drop-down arrow that is now displayed. A list of the template parameters from the target Class displays; click on the required parameter.
3	Click on the button in the corresponding 'Actual' field for the parameter. If the template parameter: • Does not have a constraint, a short context menu displays offering the choice of typing a free-text value into the 'Actual' field, or selecting a classifier from the 'Select Classifier' dialog • Has a constraint defined, the 'Select Classifier' dialog displays automatically, showing the

	available classifiers
4	Locate and select the required classifier to replace the parameter in the binding expression. If you do not define an Actual classifier and the template parameter has a default value defined, that default is used in the expression.
5	To edit existing parameter substitutions, click on them and make the required changes as indicated in steps 3 and 4.
6	Click on the Apply button and/or the OK button. The parameter substitutions display as a label underneath the connector.

Trace



Description

The Trace relationship is a specialization of an Abstraction, connecting model elements or sets of elements that represent the same concept across models. Traces are often used to track requirements and model changes, typically in a Traceability diagram, or in a Class, Use Case, Object or Composite Structure diagram.

As changes can occur in both directions, the order of this Trace is usually ignored. The relationship's properties can specify the trace mapping, but the trace is usually bi-directional, informal and rarely computable.

Toolbox icon

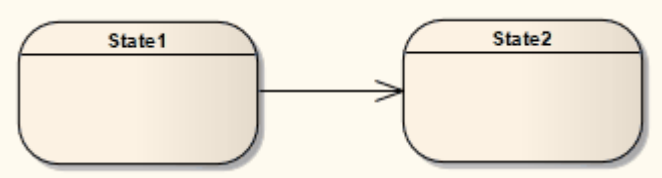
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.682) states:

Specifies a trace relationship between model elements or sets of model elements that represent the same concept in

different models. Traces are mainly used for tracking requirements and changes across models. As model changes can occur in both directions, the directionality of the dependency can often be ignored. The mapping specifies the relationship between the two, but it is rarely computable and is usually informal.

Transition



Description

If you need to define the logical movement from one State to another in a StateMachine diagram, you can drag a Transition connector from the Toolbox onto the diagram. You control the Transition through the connector 'Properties' dialog.

Field	Action
Guard	Type in the expression to be evaluated after an Event is dispatched but before the corresponding Transition is triggered. If the guard is true at that time, the Transition is enabled; otherwise, it is disabled.
Effect is a Behavior	Convert the 'Effect' field from a free-text field to the definition of a specific Activity or behavior. The 'Select <Item>' dialog displays, prompting you to select the Activity or behavior element from the model.

Effect	Either: • Type a description of the Effect of the Transition, or • If you have selected the 'Effect is a Behavior' checkbox, select an Activity or behavior to be performed during the Transition (to change this subsequently, click on the ... button to redisplay the 'Select <Item>' dialog)
Trigger Name	Specify the name of the trigger; either: • Type the name, or • Select an existing trigger in the model from the Select <Item> dialog, which you display by clicking on the ... button
Trigger Type	Specify the type of trigger: • Call - specifies that the event is a CallEvent, which sends a message to the associated object by invoking an operation • Change - specifies that the event is a ChangeEvent, which indicates that the transition is the result of a change in value of an attribute • Signal - specifies that the event is a SignalEvent, which corresponds to the

	receipt of an asynchronous signal instance • Time - corresponds to a TimeEvent; which specifies a moment in time Code generation for StateMachines expects a specification value for any of the four types.
Specification	Specify the event instigating the Transition; either: • Type the event (time or change), or • Select an existing specification in the model using the 'Select <Item>' dialog, which you display by clicking on the ... button
New	Clear the fields ready to begin defining a new trigger.
Save	Save the newly created or edited trigger.
Delete	Remove the selected trigger from the list.
<trigger list>	List the existing triggers, which might or might not have names and types, and which can include triggers created in older models.

Notes

- Fork and Join segments can have neither triggers nor guards
- You can identify hidden triggers and locate triggers in the Browser window, using the 'Find Triggers Associated' option on the Transition connector context menu; if one trigger exists for the Transition it is immediately highlighted in the Browser window, if more than one trigger exists the 'Element Usage' dialog displays - select the required trigger and click on the Open button to highlight the trigger in the Browser window
- You can define a self-Transition as an Internal Transition, and represent the connector and its properties in a compartment of the State element

Toolbox icon

OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.359) states:

A Transition represents an arc between exactly one source Vertex and exactly one Target vertex (the source and targets may be the same Vertex). It may form part of a compound transition, which takes the StateMachine from one steady

State configuration to another, representing the full response of the StateMachine to an occurrence of an Event that triggered it.

Internal Transition

If you need to define an internal Transition in a State, you can do so by creating an external self-Transition connector (where the Source and Target are the same State) and then changing the connector 'kind' property. The self-Transition connector is then removed from the diagram and the internal Transition displays in a compartment inside the State element.

Define an Internal Transition

Step	Action
1	In the Browser window, double-click on the StateMachine diagram containing the State element to open it.
2	On the State element, create a Transition connector issuing from and terminating in the element (a 'self Transition'). In the Diagram Toolbox, select the Transition connector, then click and release on the State element.
3	Right-click on the connector and select the 'Properties' option to display the 'Properties' dialog.

4	Select the 'Constraints' tab and define any guard, effect and trigger for the Transition.
5	Select the 'General' tab, then select the child tab 'Advanced'. Click on the drop-down arrow in the value field for the kind property and select 'internal'.
6	Click on the OK button. The Transitions display in the same compartment as internal activities (exit/, do/, entry/).

Notes

- To view or edit the properties of the internal Transition, double-click on the entry in the compartment within the State
- If you need multiple internal transitions, including those with the same Trigger but different guards, you create them separately with each Transition having its own guard
- You can create further transitions and internal triggers by clicking on the State element, displaying the Features window at the 'Internal Triggers' tab, right-clicking on the

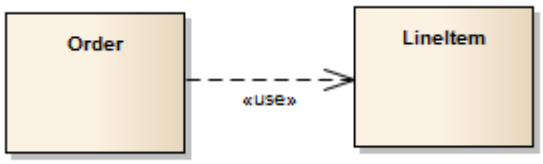
tab and selecting the 'New Internal Triggers' option

OMG UML Specification:

The OMG UML specification (UML Superstructure Specification, v2.4.1, p.362) states:

[A TransitionKind of internal] Implies that the Transition, if triggered, occurs without exiting or entering the source State (i.e., it does not cause a state change). This means that the entry or exit condition of the source State will not be invoked. An internal Transition can be taken even if the SateMachine is in one or more Regions nested within the associated State

Usage



Description

A 'Usage' is a Class diagram relationship in which one element requires another element for its full implementation or operation. The example diagram shows that the Class Order requires the Class LineItem for its full implementation.

The 'Usage' relationship is a subtype of a 'Dependency' relationship.

Toolbox icon

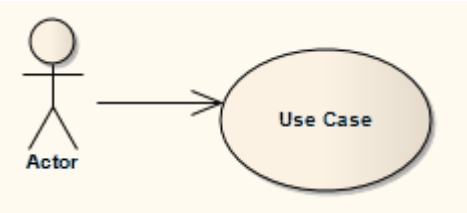
OMG UML Specification:

The OMG Unified Modeling Language specification, (v2.5.1, p.38) states:

A Usage is a Dependency in which one NamedElement requires another NamedElement (or set of NamedElements) for its full implementation or operation. The Usage does not specify how the client uses the supplier other than the fact

that the supplier is used by the definition or implementation of the client.

Use



Description

A Use relationship indicates that one element requires another to perform some interaction. The Use relationship does not specify how the target supplier is used, other than that the source client uses it in definition or implementation.

You typically use the Use relationship in Use Case diagrams to model how Actors use system functionality (Use Cases).

Notes

- It is more usual (and correct UML) to have an Association between an Actor and a Use Case
- The Usage relationship, used in Class diagrams, is a different relationship

Toolbox icon

UML Stereotypes

The UML supports stereotypes, which are an inbuilt mechanism for logically extending or altering the meaning, display, characteristics or syntax of basic UML model elements. You can apply stereotypes to a range of model element types, including:

- Elements (such as Classes and Objects)
- Relationships (such as Dependencies and Associations)
- Association Ends
- Attributes and Operations
- Operation Parameters

Different model elements have different stereotypes associated with them. You can create and use your own stereotypes in three different ways:

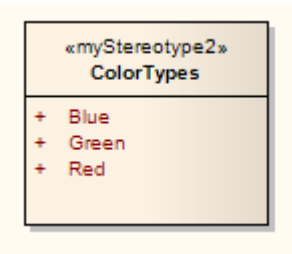
- To create a new object type based on a basic UML model element type, to be imported as part of a Profile into any model and made available for use through the Diagram Toolbox; examples of extended element types already provided in Enterprise Architect include a Table element (which is a stereotyped Class element) and Boundary, Control and Entity elements (which are stereotyped Object elements)
- To customize the appearance or property of an instance of a model element of a specific type; these stereotypes are applied only through the 'Properties' dialog of the object, within the model in which they are created, although you can transport custom stereotype definitions between

models as Reference Data

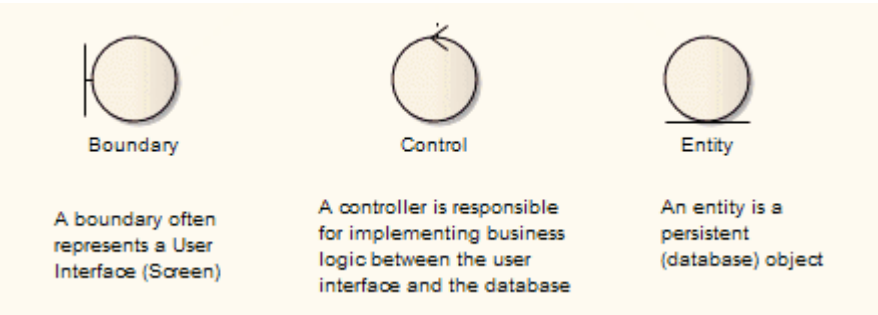
- As a simple label on an element, to identify the role or nature of the object that the element represents

For further definitions of stereotypes, see the OMG UML specification (*UML Superstructure Specification*, v2.1.1, section 18.3.8, pp. 667-672).

Where a stereotype does not affect appearance, it is generally indicated by name on the base UML object shape. In this example, «myStereotype2» is the stereotype name. Some of the built-in stereotypes are also represented by icons; see *Stereotype Visibility*.



Where the stereotype causes the element to be drawn differently or is used to define a new type of object, the element shape can be quite different, as illustrated by the three Robustness diagram stereotypes:



You apply a new appearance or shape by associating the stereotype with either a metafile (image file) and fill, border and text colors, or a Shape Script that defines the shape, dimensions and text of the object.

Apply Stereotypes

During the course of your modeling, you might decide that an existing object requires a stereotype. Enterprise Architect allows new stereotypes to be applied to objects by themselves or in combination with other stereotypes. You do this through the 'Stereotype' field on:

- The object's 'Properties' dialog or
- Properties window

Access

Open the Properties window or the 'Properties' dialog using one of the methods outlined here, then click on the button at the right of the 'Stereotype' field, and use the 'Stereotypes for <object name>' dialog (the Stereotype Selector).

Ribbon	Explore > Portals > Window > Properties > Properties Start > All Windows > Properties > General > Properties Design > Package > Manage > Properties > General
Context Menu	Right-click on Package or element | Properties > General
Keyboard	Alt+Enter ('Properties' dialog)

Shortcuts	Shift+Enter ('Properties' dialog) Ctrl+2 (Properties window)

Stereotype Selector

If you want to apply more than one stereotype to a UML object, from multiple sources such as Profiles or the Customized Stereotypes List, you can select the stereotypes from the 'Stereotypes for <object name>' dialog. This dialog also helps you to identify existing, valid individual stereotypes, and to create new stereotypes. The new stereotypes, at this point, are simple labels; if you want them to impose an Effect on the object, locate them on the 'Stereotypes' tab of the 'UML Types' dialog and define the Effect.

Access

Other	Display the 'Stereotype for <object name>' dialog by clicking on beside the 'Stereotype' field in the object's 'Properties' dialog or Properties window.

Select Stereotypes to Apply or Remove

Field/Button	Action
Perspective	Click on the drop-down arrow and select

	a Perspective name, to limit the stereotypes offered for selection to those available under that Perspective. If you want to examine stereotypes across the model from any Perspective, click on the button and select 'All'. The 'Stereotypes' column lists the available stereotypes; click on the checkbox against each stereotype to select. When you initially open the 'Stereotype for <object name>' dialog, the Perspective name in the field is the Perspective shown in the <perspective name> icon at the top right of the application screen. Changing the Perspective in the 'Stereotype for <object name>' dialog does not change the 'global' Perspective on the icon. If you want to re-set the 'Stereotype for <object name>' dialog Perspective to the global Perspective, simply click on the drop-down arrow and select 'Active'.
Profile	Click on the drop-down arrow and choose the required stereotype source - an integrated MDG Technology or the base EAUML, for example, or select the blank line for your Customized Stereotypes list.

	The field defaults to the last-selected Profile (if it is in the currently-set Perspective) or, if the element already has a stereotype, the Profile for that stereotype.
Stereotypes	Select the checkbox against each required stereotype. If you no longer want to use a stereotype, deselect the checkbox.
Apply to	Displays the types of object that the selected stereotype is assigned to.
New	Click on this button to create a new (but undefined) stereotype. A prompt displays for the stereotype name.
OK	Click on this button to apply the selection.
Cancel	Click on this button to cancel any selections and close the dialog.

Notes

- If you have selected more than one stereotype, the

'Properties' dialog lists them on separate lines of the 'Stereotype' field

- The appearance of a stereotype on an object in a diagram is influenced by the stereotype visibility settings on the 'Properties' dialog for the diagram

Stereotype Visibility

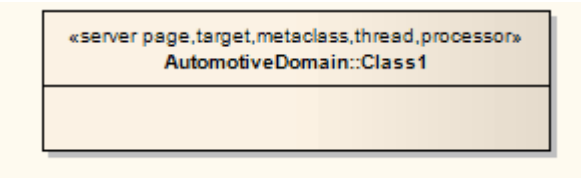
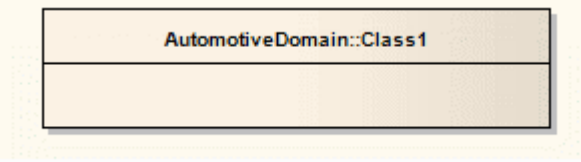
When you apply one or more stereotypes to an object, the display of that object in a diagram defaults to showing the stereotype names in a string within guillemets (« »); multiple names are separated by commas. Some stereotypes are associated with small icons that display in the top right corner of the element; these icons are built into the system, and cannot be deleted or added to. In both cases, you can modify the visibility of the text or icon stereotype indicators in a diagram, using the 'Properties' dialog for the diagram.

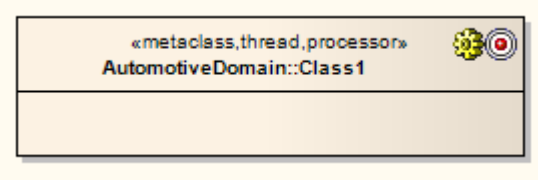
Access

Display the 'Properties' dialog for the diagram, then show the 'Elements' tab or the 'Features' tab, to select the visibility of stereotypes on elements or features respectively.

Ribbon	Design > Diagram > Manage > Properties > select 'Elements' or 'Features' tab
Context Menu	Right-click on diagram background | Properties > select 'Elements' or 'Features' tab
Other	Double-click diagram background > select 'Elements' or 'Features' tab

Set Stereotype Visibility Options

Field/Button	Action
Show Element Stereotypes	Select this checkbox on the 'Elements' tab to show all element stereotypes and keywords in the current diagram; for example (with 'Use Stereotype' icons not selected):  Deselect this checkbox to hide all element stereotype names, icons and keywords. 
Use Stereotype Icons	Select this checkbox on the 'Elements' tab to display icons instead of text, for those element stereotypes that have icons defined. Stereotypes that do not have associated icons are still represented by the stereotype names; for example.

	 The icons represent the stereotypes ?server page? and ?target?.
Show Stereotypes	Select this checkbox on the 'Features' tab to show all attribute and operation stereotypes in the current diagram. This option does not affect the display of element stereotypes.

Notes

- In the Browser window, the object name is preceded by the stereotype name(s) within guillemets, and multiple names are indicated by the first stereotype name followed by an ellipsis (...); you can hide the stereotype name by deselecting the Browser window 'Show Stereotypes' checkbox ('Start > Appearance > Preferences > Preferences > General')

Standard Stereotypes

This table identifies the standard stereotypes provided in the EABase base model, each enclosed by guillemets (« »).

Stereotypes

Stereotype	Base Class
«access»	Dependency
«become»	Flow
«call»	Usage
«copy»	Flow
«create»	Message
«derive»	Abstraction
«destroy»	Message
«document»	Abstraction
«executable»	Abstraction

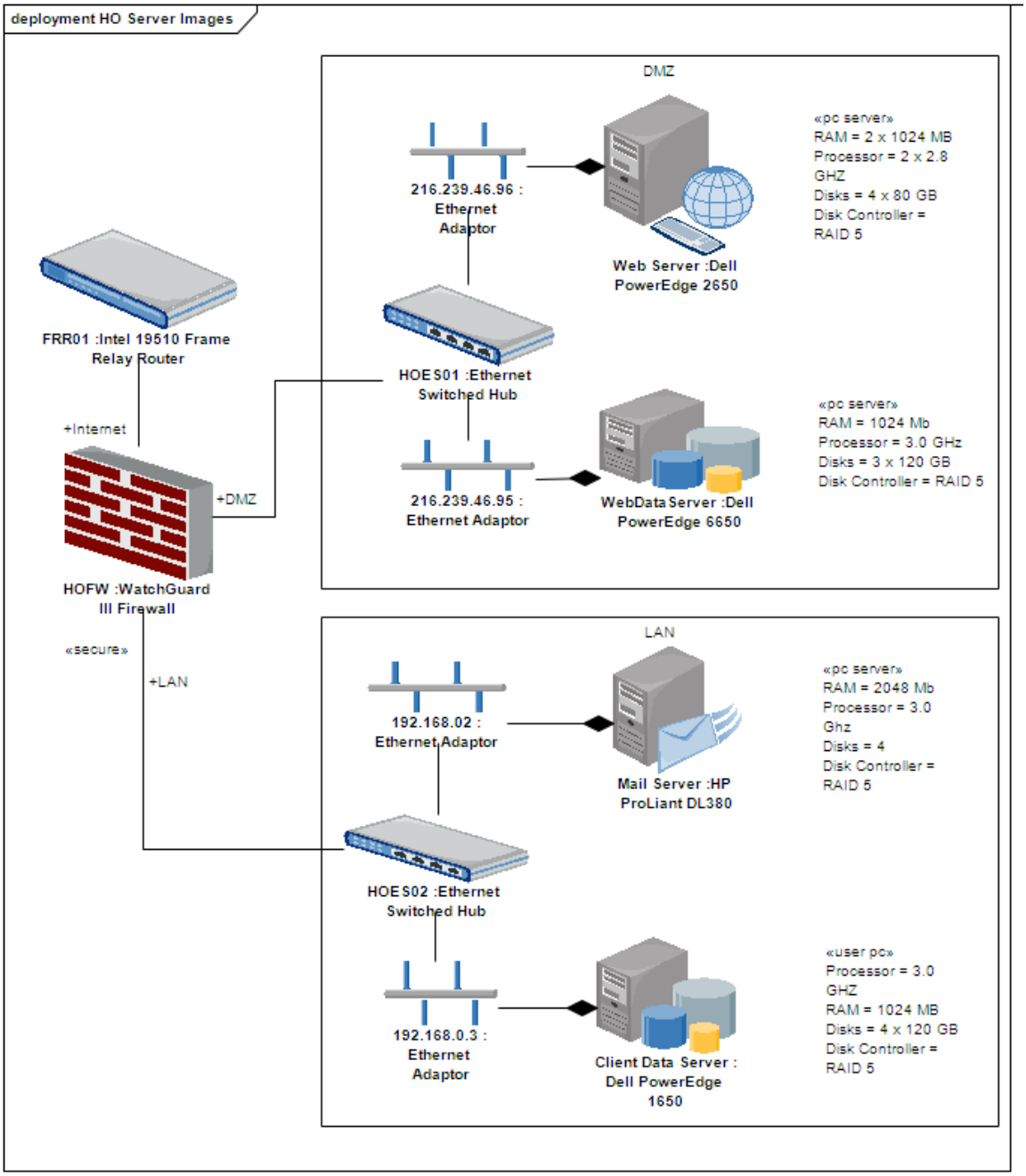
«facade»	Package
«file»	Abstraction
«framework»	Package
«friend»	Dependency
«global»	AssociationEnd
«implementation»	Class
«implementation»	Generalization
«import»	Dependency
«instantiate»	Usage
«invariant»	Constraint
«library»	Abstraction
«local»	AssociationEnd
«metaclass»	Class

«parameter»	AssociationEnd
«postcondition»	Constraint
«powertype»	Class
«precondition»	Constraint
«process»	Classifier
«refine»	Abstraction
«requirement»	Comment
«responsibility»	Comment
«self»	AssociationEnd
«send»	Usage
«stub»	Package
«table»	Abstraction

«thread»	Classifier
«trace»	Abstraction
«type»	Class
«utility»	Classifier

Stereotypes with Alternative Images

If you want to represent an element using an image (for example, depict a hardware component using a 3-D box, or even using an image of the unit itself), you can do so using a stereotype that has been associated with a metafile. When the stereotype is applied to a Class or other element that supports alternative graphical format, the element is drawn using the image instead of the standard UML shape. For example, in this Deployment diagram, the Component elements all have alternative images.



Notes

- You cannot change the representation of elements that include Lifelines, such as those in Sequence diagrams; the standard representation is important in the use and function of those elements

Custom Stereotypes

A custom Stereotype applies a different appearance or characteristic to a basic UML model component or feature. You can apply a custom stereotype in two different ways:

- To change the appearance or property of an instance of a model component of a specific type; these stereotypes are defined on the 'Stereotypes' tab of the 'UML Types' dialog and applied through the 'Properties' dialog of the object, within the model in which they are created, although you can transport custom stereotype definitions between models as Reference Data
- As a simple label on an element, to identify the role or nature of the object that an element represents; these stereotypes are simply names typed into the 'Stereotype' field of the object 'Properties' dialog, and do not affect the element display unless they are subsequently edited to have an effect

The more obvious changes you can make are to the shape, dimensions and appearance of the object, which you can apply by associating a metafile (image file) and customized colors with the stereotype, or by attaching a Shape Script to the stereotype. When you have defined and saved the stereotype, you can then apply it to any new or existing object of the base Class with which it is associated.

Access

Ribbon	Settings > Reference Data > UML Types > Stereotypes

Maintain custom stereotypes

Option	Action
Stereotype	Type or select the name of the stereotype.
Group name	(Optional) Type a plural name under which to group the stereotype features for attributes and operations; the name will be shown on diagrams in the attributes and operations compartments.
Base Class	Click on the drop-down arrow and select the name of a pre-existing object type so that the stereotyped element will inherit the base characteristics of that type.
Notes	(Optional, but recommended) Type any notes concerning the stereotype (not the elements to which the stereotype is to be applied).
New	Click on this button to clear the fields to

	create a new stereotype definition.
Save	Click on this button to save a new or edited stereotype definition.
Delete	Click on this button to delete a stereotype definition from the model.
Override Appearance	
None	Select to retain the default element appearance for this stereotype.
Metafile	Select to associate the stereotype with an image metafile (.emf or .wmf) to apply that image when the stereotype is used.
Shape Script	Select to associate the stereotype with a custom shape, created using the Shape Scripting language.
Assign	Click on this button to either: • Display the browser to locate the .emf or .wmf metafile to associate with the stereotype, or • Open the Shape Editor create the Shape Script to be associated with the

	stereotype
Edit	If a Shape Script is already associated with the stereotype, click on this button to open the Shape Editor to update the Shape Script.
Remove	Remove the associated metafile or Shape Script from the stereotype.
Default Colors	
Fill	Click on the drop-down arrow and select or define the default background color of the elements to be refined by the stereotype. This color will be applied to all occurrences of any element to which the stereotype has been applied; if the color is subsequently changed, the change is immediately applied to all occurrences of any element to which the stereotype was applied (as for changes to any other property of the stereotype). However, on elements created with the stereotype, the default color might be overridden by other color definitions of a higher priority that have been applied to

	the element.
Border	Click on the drop-down arrow and select or define the default color of the borders of the elements to be refined by the stereotype.
Font	Click on the drop-down arrow and select or define the default color of the text of the elements to be refined by the stereotype.
Reset	Reset the default colors to those of the base element with which the stereotype is associated.

Notes

- You can transport custom stereotype definitions between models, using the 'Settings > Model > Transfer > Export Reference Data' and 'Import Reference Data' ribbon options
- You can also create Stereotype elements that extend basic UML model element types to create new model element types; you can re-use these extended model elements in other projects, by incorporating them into a Profile (usually within an MDG Technology) and importing this into the various target projects

Extending UML

Sometimes a modeling problem cannot be adequately expressed using the base UML model elements or, similarly, an area of work falls into a specialized domain that requires a tailored modeling approach or program language support. To meet such requirements, you can extend the capabilities of UML to develop new modeling constructs, using MDG Technologies to combine and deploy a wide range or extension mechanisms such as:

- UML Profiles
- Stereotypes
- Shape Scripts
- Tagged Values
- Constraints
- Patterns
- Customized Code and Transformation Templates, and
- Grammars

Using the MDG Technology Creation Wizard, you can quickly and easily integrate the extensions into a technology and rapidly tailor UML and Enterprise Architect to address a particular modeling domain not explicitly defined in the original UML specification, but using extension mechanisms that are still part of the Specification.

Facilities

Facility	Description
Extending UML	Quickly and easily extend UML into a profile and technology using the MDG Technology Wizard.
Using MDG Technologies	Wrap your UML Profiles, code modules, scripts, Patterns, images, Tagged Value Types, report templates, Linked Document templates and Toolbox pages.
The MDG Technology SDK	Everything you require to build your own technology, such as Shape Scripts, Tagged Value Types, Code Template Frameworks, Grammar Frameworks and more.

Using UML Profiles

A UML profile is a light-weight extension mechanism that is part of the UML Standard. Using profiles, you can create a set of model constructs suitable for modeling a particular domain, platform or method. Enterprise Architect provides a flexible and intuitive mechanism for creating and deploying profiles. Standard UML constructs are augmented with stereotypes and Tagged Values to create new tailored elements suitable for the modeling purpose. A profile is simply a collection of these constructs with their stereotypes and associated Tagged Values. The stereotypes can be applied to elements, features, connectors and connector ends. A Profile is distributed and implemented using a Model Driven Generation (MDG) Technology.

The deployed technology automatically generates a page of elements and relationships in the Diagram Toolbox, for each of the UML profiles within the technology. When you drag the elements and connectors from the toolbox onto the current diagram, the stereotype, Tagged Values and default values, notes and metafile (if one is specified) are automatically applied to the new element. You can also drag and drop profile attributes and operations onto existing Classes, so that they are immediately added with the specified stereotype and Tagged Values.

Add Profile Objects to a Diagram

After a technology has been imported into your project, the profiled objects (elements and connectors) and features (attributes and operations) are available from the technology pages of the Diagram Toolbox. The way in which you add the Profile objects to a diagram is no different from the way in which you use the standard UML objects on the system.

Access

Ribbon	Design > Diagram > Toolbox : to display the 'Find Toolbox Item' dialog and specify <technology name>
Keyboard Shortcuts	Ctrl+Shift+3 : to display the 'Find Toolbox Item' dialog and specify <technology name>

Use the Profile Objects

Action	Description
Add a Profile-based	Click on the element in the Toolbox page and drag it onto the diagram.

element to a diagram	
Add a Profile-based connector to a diagram	Click on the connector in the Toolbox page, then click on the source element in the diagram and drag it to the target.
Add a Profile-based attribute or operation to a diagram	Click on the attribute or operation in the Toolbox page, and drag it onto the host element on the diagram. The system prompts you to enter a name for the feature.

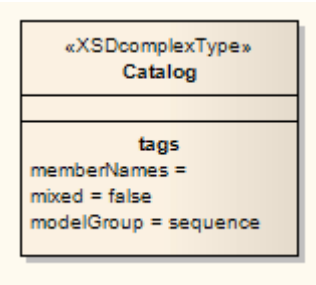
Tagged Values in Profiles

Stereotypes within a profiled element or connector can define one or more associated Tagged Values. When you drag a profiled element or connector from the Diagram Toolbox onto a diagram, any associated Tagged Values are automatically added to the new element or connector. Tagged Values in profiled objects are an excellent way to further extend the versatility of your UML modeling.

As an example, the UML Profile for XSD (XML Schema) provides the XSDComplexType stereotype to extend a Class; this stereotype has the Tagged Values:

- memberNames
- mixed and
- modelGroup

When you create a Complex Type element, the Tagged Values are added and are visible in the tags compartment of the element (including those that have no value set).



When you select the element, the Properties window displays all the associated tags.

- The values of tags imported in a Profile override the values of equivalent tags in the 'UML Types' dialog; if the initial value of the tag from the Profile is not set, the value

of the tag shown in the element will be blank, even if there are default values for the tag in the 'UML Types' dialog

- Tags that have default profile values are automatically set
- Where Tagged Values in the profiled element have a values section (for example, values="element | attribute | both" default="both") you can select the non-default values from a drop-down list
- Where no value exists, you can add a value as free text; you would do this for a profile tag that has no initial value, to use a default value from the 'UML Types' dialog

Synchronize Tagged Values and Constraints

When you create an element, attribute, operation or connector from a profiled object, the Tagged Values and constraints are added from the Profile stereotype. Subsequently, you might update the constraints or Tagged Values of a particular stereotype in the Profile, in which case the items already created in the model would not have those additional constraints or Tagged Value tags and notes.

Similarly, you might have manually added the stereotype to a set of objects, which automatically adds the Tagged Values but not the constraints associated with that stereotype, and now want the objects to receive the constraints.

You can apply the updated or missing Tagged Values and constraints using the Synchronize Stereotype function. This operates on any profiled element in your model, from any technology that is integrated with or imported into Enterprise Architect.

Access

Ribbon	Design > Diagram > Toolbox : to display the 'Find Toolbox Item' dialog and specify <technology name>| Right-click icon for profiled

	element/connector/feature | Synchronize Stereotype
Keyboard Shortcuts	Ctrl+Shift+3 : Specify <technology name> in the 'Find Toolbar Item' dialog | Right-click icon for profiled element/connector/feature | Synchronize Stereotype

Synchronize objects using the Technology Toolbox pages

Step	Action
1	On the 'Synch Profiled Elements' dialog, click on the OK button. All elements, features or connectors created with the selected profiled object icon are updated, across the model. The items that have been modified, and the changes that were made, are listed in the 'Actions' field.
2	When the update is complete, click on the Cancel button.

Alternative - Single Object Update

You can quickly synchronize the tags and constraints of a single element in a diagram. To do this:

Step	Action
1	Drag the updated profiled element from the Diagram Toolbox page onto the element in the diagram. A short context menu displays.
2	Select the 'Apply ?stereotype name?' menu option. The diagram element is updated with any tags and constraints from the profiled element that it does not already have.

Notes

- The 'Synchronize Stereotype' context menu option displays when a Diagram Toolbox icon represents a profiled element or a connector; it does not display for basic UML object icons
- You can review any changes by displaying the element 'Properties' dialog, opening the 'Tags' tab and clicking on an appropriate profiled element
- Removing a stereotype from an object automatically

removes any Tagged Values assigned by that stereotype

Extension Stereotypes

Enterprise Architect supports a formidable range of modeling languages and platforms that have defined sets of elements and connectors. However, Enterprise Architect provides the ability for the modeler to create an extensive set of other elements, typically by adding a stereotype to an existing element. You, as a modeller, are free to create your own new elements by using such facilities as stereotypes and Shape Scripts. It is common for communities of users to create and share a common set of stereotypes for a particular domain.

- Analysis Stereotypes
- Boundary
- Composite Elements
- Control
- Entity
- Event
- Feature
- Hyperlink
- Image
- N-Ary Association
- Packaging Component
- Process
- Requirements
- Risk

- Screen
- Task
- Test Case
- Database Tables
- UI Control Elements
- Web Stereotypes

Boundary

A Boundary is a stereotyped Object that models some system boundary, typically a user interface screen. You can also create a Boundary as a stereotyped Class. Boundary elements are used in analysis to capture user interactions, screen flows and element interactions (or 'collaborations').

A Boundary is used in the conceptual phase to capture users interacting with the system at a screen level (or some other boundary interface type). It is often used in Sequence and Robustness (Analysis) diagrams. It is the View in the Model-View-Controller Pattern.

Toolbox icon

Create a Boundary

There are two ways in which you can create a Boundary on a diagram.

Create a Boundary element as a stereotyped Class

Step	Action
1	Insert a new Class.
2	Right-click on the element and select the 'Properties' option; the 'Properties' dialog displays.
3	In the 'Stereotype' field, type the value 'boundary'.
4	Click on the Apply button and the OK button.
5	Press Ctrl+S to save the diagram.

Create a Boundary element as an Object

Ste	Action

p	
1	In the Diagram Toolbox, click on to display the 'Find Toolbox Item' dialog and specify 'Analysis'.
2	From the 'Analysis Elements' page, drag the 'Boundary' icon onto the diagram.

Control

A Control is a stereotyped Object that models a controlling entity or manager. A Control organizes and schedules other activities and elements, typically in Analysis (including Robustness), Sequence and Communication diagrams. It is the controller of the Model-View-Controller Pattern.

You can also create a Control as a stereotyped Class.

Toolbox icon

Create a Control Element

Create a Control element on a diagram as an Object

Step	Action
1	In the Diagram Toolbox, click on to display the 'Find Toolbox Item' dialog and specify 'Analysis'.
2	From the Analysis Elements page, drag the Control icon onto the diagram.

Create a Control element as a stereotyped Class

Step	Action
1	Insert a new Class.
2	Right-click on the element and select the 'Properties' option; the 'Properties' dialog displays.

3	In the 'Stereotype' field, type the value 'control'.
4	Click on the Apply and OK buttons.
5	Press Ctrl+S to save the diagram.

Entity

An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system. It is the Model in the Model-View-Controller Pattern.

You can also create an Entity as a stereotyped Class. See the *Create an Entity* topic.

Toolbox icon

Create an Entity

Create an Entity element on a diagram as an Object

Step	Action
1	In the Diagram Toolbox, click on to display the 'Find Toolbox Item' dialog and specify 'Analysis'.
2	From the Analysis Elements page, drag the Entity icon onto the diagram.

Create an Entity element as a stereotyped Class

Step	Action
1	Insert a new Class.
2	Right-click on the element and select the 'Properties' option; the 'Properties' dialog displays.

3	In the 'Stereotype' field, type the value 'entity'.
4	Click on the Apply and OK buttons.
5	Press Ctrl+S to save the diagram.

Hyperlink

You can place a Hyperlink element onto a diagram. This element is a type of text element, but one that can contain a pointer to a range of objects such as associated document files, web pages, Help, model features and even other Enterprise Architect model files. When you double-click on the element, Enterprise Architect executes the link.

To add a Hyperlink element, either:

- Drag the 'Hyperlink' icon from the 'Common' page of the Diagram Toolbox onto the diagram, or
- Click on the 'Hyperlink' icon in the UML Elements toolbar and then click on the diagram

Configure the Hyperlink

When you add the Hyperlink to the diagram, you immediately type in some link text, click off the element and then double-click on the element. The 'Hyperlink Details' dialog displays. If you want to display the information in a more readable layout, you can resize the dialog.

Field/Button	Action
Type	Click on the drop-down arrow and select the type of object to link to.

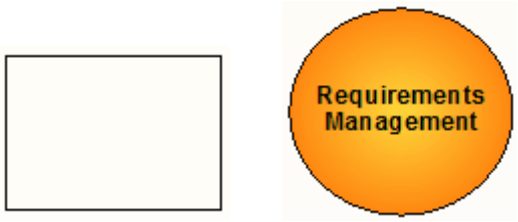
	In many cases, when you select the type a browser dialog displays for that type of object, from which you select the actual object to link to.
Action	This field is enabled when the dialog first displays with 'Type' defaulted to 'File', or if you select 'File' from the 'Type' drop-down list. The field defaults to the value 'Open', to display the file contents in read-only mode. If you want the user to be able to change the file contents, click on the drop-down arrow and select the value 'Edit'. The system automatically selects the appropriate editor. For example, if you hyperlink to a .rtf file, you can view the file in whichever internal viewer is appropriate; however, you cannot edit .rtf files in Enterprise Architect, so the file always opens in the Windows default .rtf editor.
Alias	This field displays the text you typed in as the link text when you created the element on the diagram. If you want to change this text, overtype it with the new text.

	If you do not provide an Alias, either the text defaults to the link itself, or (for certain link targets such as a Matrix Profile) the system generates a simple text instruction.
Hide Icon	If you prefer to display only the link text, without the icon, select this checkbox.
Notes	Type in any notes you might require to explain the hyperlink. These notes are not displayed in the element on the diagram. You can format the notes using the Notes toolbar.
Address	If a browser displayed on input to the 'Type' field, when you select the object to link to the object name or location displays in this field. (If the object is not accessed through a path or 'address', the field is generally not labeled.) If no browser displayed or if you want to change the linked object to another of the same type, either type in the object location or click on the button to display the appropriate browser, and select the target object.

Notes

- If required, you can create a number of empty hyperlinks to complete later; if you then double-click on an empty hyperlink, the 'Hyperlink Details' dialog displays and you can enter the details
- When hovering the cursor over the hyperlink, the standard hyperlink element buttons will also be displayed in a pop-up toolbar, allowing you to display the 'Properties' dialog, find the element in diagrams, find the element in the Browser window, or open the Linked Document attached to the element

Image

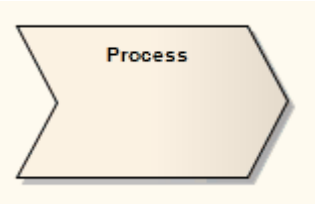


An Image is a System Boundary element that automatically displays first the Boundary 'Properties' dialog and then the 'Select Alternate Image' dialog to change its representation to an imported image. You can use it as an icon for an element or group of elements, or as a diagram background.

Image elements are available from the 'Common' page of the Toolbox.

Toolbox icon

Process

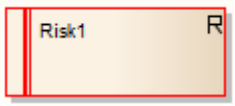


A Process is an Activity element with the stereotype process, which expresses the concept of a business process. Typically this involves inputs, outputs, workflow, goals and connections with other Processes. The Process element is typically used in Analysis diagrams.

Business processes typically range across many parts of the organization and span one or more systems.

Toolbox icon

Risk



Risk elements are not the same as the risks that you assign to an element through the Risks window. Such risks are internal to the selected element, whilst a Risk element can be associated with a number of elements, either in a logical group or totally separate.

Risk elements are available from the 'Requirements' page of the Toolbox.

Using the Risk element

A Risk is defined as the effect of uncertainty on objectives. In Project Management, it is necessary to try to identify risks and assess:

- The likelihood that they have a negative effect on a project and
- How large that effect is likely to be

Those risks with a high probability of occurrence and/or a large impact on the project can be mitigated.

A Risk Management process might consist of these five steps:

1. Identify risks and represent each with a Risk element.
2. Identify which elements (such as Components, Use Cases or Features) are vulnerable to each risk; you might decide to create «trace» dependencies from these elements to the

Risk elements.

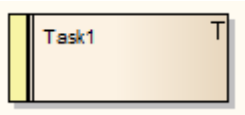
3. Assess the likelihood and magnitude of the risks.
4. Identify ways to mitigate the risks.
5. Prioritize the risk reduction measures based on their likelihood, magnitude and ease of mitigation.

Notes

- Risk elements can be displayed with or without an identifying 'R' in the top right corner of the element; to toggle the display of this letter, select or deselect the 'Show stereotype icon for requirements' checkbox on the 'Preferences' dialog, 'Objects' page

Toolbox icon

Task

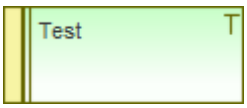


A Task element represents a task that must be performed in relation to an element. Through the Task element you can assign resources to the task itself, rather than just to the parent element.

You can create a hierarchy or tree structure of Task elements to break a large task into separate parts and assign different resources to each part.

Toolbox icon

Test Element



A Test element represents a step in the Basic, Alternate and Exception Paths of a Scenario created in a Use Case or other element. The Test element is generated within a Test Case element.

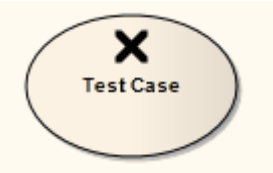
Each Test element has a status band at the left end, which is color coded to visually represent the value of the 'Status' field in the element properties. The element has an identifying 'T' in the top right corner, which you can hide if you prefer not to show it.

Toolbox icon

Notes

- To toggle display of the letter 'T' in the top right corner of the element, select or deselect the 'Show stereotype icon for requirements' checkbox on the 'Preferences' dialog, 'Objects' page

Test Case



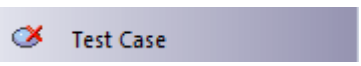
A Test Case is a stereotyped Use Case element. You might use it to extend the facilities of the Test Cases window, by applying element properties and capabilities to the tests of a feature represented by another element or - more appropriately - set of elements. That is, you can define in one go, in the Test Cases window for the Test Case element, the details of the tests that apply to each of several elements, instead of recording the details separately in each element.

Within the Test Case element properties you can define test requirements and constraints, and associate the test with test files. You can also link the element to Document Artifacts or (in the Corporate, Unified and Ultimate Editions) directly to a Linked Document, such as a Test Plan.

The Test Case element enables you to give greater visibility to tests, in the Browser window, Diagram List, Package Browser, Model Search, Relationship Matrix, Traceability window and reports.

The Test Case element is available through the 'Use Case' and 'Maintenance' pages of the Diagram Toolbox.

Toolbox icon



Design Patterns

A Design Pattern is a template for solving commonly recurring design problems. A Design Pattern consists of a series of elements and connectors that can be reused in a new context. The advantage of using these Patterns is they have been tested and refined in a number contexts and so are typically robust solutions to common problems.

Enterprise Architect provides extensive support for both creating and using Design Patterns. Patterns are typically created by experienced modelers who can see how to distil an abstract problem and solution from a concrete model. The Pattern user must be able to identify the correct Pattern to use and must select appropriate names for the elements of the Pattern in the context.

Patterns can be saved from any diagram, creating an XML file that describes the Pattern; these files can be imported into a repository as a resource that can then be used in any context.

Sparx-Created GoF Patterns

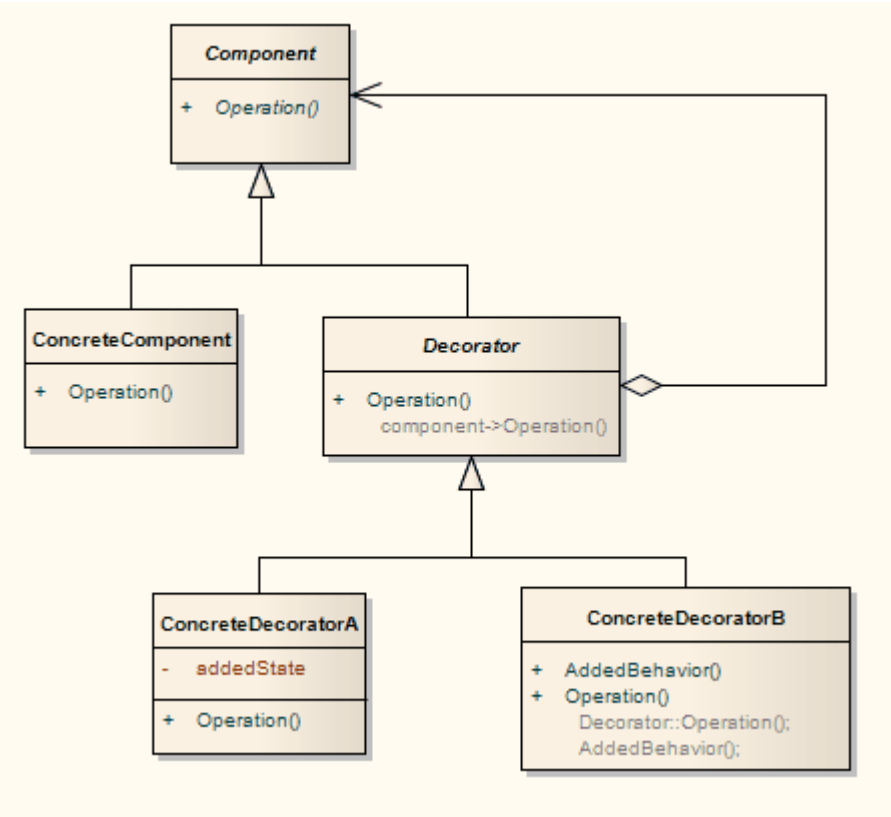
To help you start using Design Patterns in Enterprise Architect, Sparx Systems provides you with the Patterns originally published in the book *Design Patterns - Elements of Reusable Object-Oriented Software* by Gamma et al., referred to as the 'Gang of Four' or GoF Patterns. When the GoF Technology is enabled, you can access these Patterns through a set of Toolbox pages.

Notes

- You can transport all Patterns listed in the 'Resources' tab of the Browser window between projects, using the 'Settings > Model > Transfer > Export Reference Data' and 'Import Reference Data' ribbon options

Publish a Pattern

To publish a Design Pattern you first must model the Pattern as a diagram within Enterprise Architect. This example diagram was created from an example in the GoF book *Design Patterns - Elements of Reusable Object-Oriented Software* by Gamma et al.



Access

Ribbon	With diagram open: Specialize > Technologies > Publish Technology > Publish Diagram as Pattern

Define the Pattern File

Field/Button	Action
Pattern Name	Type the Pattern name.
Filename	Type a directory path and .XML filename to contain the published Pattern.
Category	Type the Category under which the Pattern should be listed in 'Patterns' (required).
Version	Type the Pattern version number.
Notes	Type any notes on the Pattern.
Actions	Select the appropriate checkboxes to select the actions for the elements that are contained in the Pattern; these actions are performed when the Pattern is used. The available actions are: • Create: Creates the Pattern element directly without modification • Merge: Merges the Pattern element

	with an existing element, enabling the existing element to take on the role of the selected Pattern element • Instance: Creates the Pattern element as an instance of an existing element • Type: Creates the Pattern element types as an existing element If your Pattern includes an Object element, you would use 'Instance' to set the classifier of the Object to one of the Classes in the diagram onto which you are dropping the Pattern. If your Pattern includes a Property (Port or Part) you would use 'Type' to set the type of the Property to one of the Classes in the diagram onto which you are dropping the Pattern.
OK	Click on this button twice to publish the Pattern. Once published, you can load the Pattern into Enterprise Architect, into the 'Resources' tab of the Browser window.

Notes

- In the Corporate, Unified and Ultimate Editions of

Enterprise Architect, if security is enabled you must have 'Manage Diagrams' permission to publish a diagram as a Pattern

- If your source diagram contains information flows, the 'Information Items Conveyed' and 'Information Flows Realized' data is not copied into the Pattern
- To change the name of one of the elements, double-click on the element to display the 'Edit' dialog; from this dialog you can also add comments detailing the element's purpose
- Patterns can not be published for Sequence diagrams

Save a Pattern as an Artifact

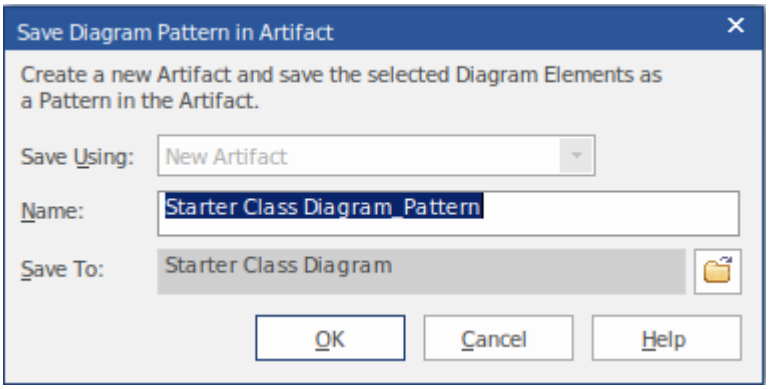
Enterprise Architect enables you to create a Pattern from selected elements in a diagram, and store the Pattern as an Artifact element in the model. You can then drop the Artifact into any diagram in the model to recreate the stored Pattern. This is similar to publishing a diagram as a Pattern, except that :

- Only the selected elements in the diagram are saved as a Pattern
- The Pattern is saved in an Artifact in the model and not as an XML file in the file system

Save Diagram Elements as a Pattern in a New Artifact

Follow the steps in this table.

Step	Action
1	Open the appropriate diagram, hold down the Ctrl key and click on each element that you want to include in the Pattern. You could also 'drag' across a set of elements in the diagram to save as a Pattern.
2	Right-click on one of the selected elements and click on the 'Save Selected

	Elements as Pattern' option. The 'Save Diagram Pattern in Artifact' dialog displays. 
3	If the 'Save Using' field does not show the value 'New Artifact', click on the drop-down arrow and select this value. If this is the first Pattern Artifact in the model, the field defaults to this value as the only value it can have.
4	The 'Name' field defaults to the name of parent diagram plus '_Pattern'. Either leave this name or overtype it with your preferred Pattern name.
5	The 'Save to' field defaults to the name of the diagram's parent Package. Either leave this Package name or click on the icon and browse for a different Package under which to create the Artifact.

6	Click on the OK button to generate the *DiagramPattern* stereotyped Artifact under the selected Package. The selected elements are saved as a Pattern within the Artifact.

Apply Pattern from Artifact to a Diagram

Follow the steps in this table.

Step	Action
1	Open the diagram into which you will paste the Pattern from the Artifact. The diagram must be in Graphical View, and not in Internal Specification View, Gantt View or List View.
2	Locate the required *DiagramPattern* stereotyped Artifact in the Browser window and drag it onto the open diagram. New elements and connectors are generated in the diagram from the Pattern.

Update a Pattern in an Artifact

There are two similar methods of updating a Pattern held in an Artifact.

Method	Description
1	Follow steps 1 and 2 in the *Save Diagram Elements as a Pattern in a New Artifact* table, then set the 'Save Using' field to 'Existing Artifact'. Click on the 'Name' field and on the name of the Pattern to update. The 'Save To' field grays out, as it uses the Package address of the existing Artifact. Click on the OK button; the Artifact is updated with the new Pattern of elements.
2	Open the diagram containing the elements to make up the Pattern. In the Browser window, click on the *DiagramPattern* Artifact to update with a new Pattern. In the diagram, select the required elements, then right-click and select the 'Save Selected Elements as Pattern' option. The 'Save Diagram Pattern in Artifact' dialog displays with the 'Save Using' field defaulted to 'Selected Artifact' and the other two fields grayed out. The 'Name'

	field shows the name of the selected Artifact. Click on the OK button; the selected Artifact is updated with the new Pattern of elements.

Notes

- In the Corporate, Unified and Ultimate Editions of Enterprise Architect, if security is enabled you must have 'Update Diagrams' permission to generate the Pattern from the *DiagramPattern* stereotyped Artifact into a diagram

Import a Model Pattern

Before being able to use a customized Pattern in your model, you must first import the Pattern XML file into the ModelPatterns directory in the Enterprise Architect install path; it is then available from the 'Resources' tab of the Browser window and optionally from the Toolbox.

Access

Use one of the methods outlined here to display the 'Resources' tab of the Browser window.

Within the 'Resources' tab of the Browser window, right-click on 'Model Patterns | Import Model Pattern'.

Ribbon	Start > All Windows > Design > General . Browse > Resources Explore > Portals > Windows > Explore > Resources
Keyboard Shortcuts	Alt+6

Import the Model Pattern

Ste	Action

p	
1	On the 'Import Model Pattern' dialog, type in or browse for the name of the XML file to import.
2	Select to import the file into either the model or the user APPDATA location.
3	Click on the OK button to import the Pattern. The imported Pattern is placed in the appropriate category as defined in the XML file; if the category does not already exist under 'Model Patterns', a new one is created.

Patterns in MDG Technologies

A number of technologies provide their own Patterns, and some technologies are designed principally as a vehicle for making specific Patterns available to the model, such as the technology for Gang of Four Patterns. Such Patterns are provided through the 'Resources' tab of the Browser window and the Diagram Toolbox pages for the technology. If you want to use such Patterns, check that the appropriate technology has been loaded and enabled in the model.

Use a Pattern

Using a Design Pattern, you can rapidly create template solutions for code structures that perform the same type of task in other situations, and use items defined in the Pattern with the model.

Access

Use one of the methods outlined here to display the 'Resources' tab of the Browser window.

Ribbon	Start > All Windows > Design > General > Browse > Resources Explore > Portals > Windows > Explore > Resources
Keyboard Shortcuts	Alt+6

Use a Pattern previously imported into the model

Step	Action

1	Open the diagram into which to add the Pattern.
2	Select the 'Resources' tab of the Browser window.
3	Expand the folder 'Patterns' and expand sub-folders as necessary, until the Pattern you require is located. You can view the Pattern details in read-only mode by right-clicking on the name and selecting the 'View Pattern Details' option.
4	Either: • Select the 'Add Pattern to Diagram' context menu option or • Drag and drop the Pattern from the 'Resources' tab of the Browser window onto the diagram The 'Add Pattern <pattern group> <pattern name> to Diagram' dialog displays.
5	Work through the dialog, making selections as required. Once the appropriate selections have been made, click on the OK button to import the Pattern into the model, recreating the original diagram with new GUIDs.

Change the default of the Pattern element

Step	Action
1	In the 'Add Pattern <pattern group> <pattern name> to Diagram' dialog, ensure the option 'Import as Package Fragment' is unchecked.
2	Select the individual element in the 'Pattern Elements' panel.
3	Click on the ... button at the end of the item row to display the 'Edit' dialog. The specific method for changing the element name is dependant upon the entry in the 'Action' column of the 'Pattern Elements' panel.
4	If the 'Action' entry for the element is 'Create', then in the 'Default' field in the 'Edit' dialog delete the existing value and type your own, user-defined value. Click on the OK button. The element default is updated on the 'Add Pattern...' dialog.
5	If the 'Action' entry for the element is 'Merge', in the 'Edit' dialog click on the ... button to browse to an existing element classifier.

	The 'Select <Item>' dialog displays.
6	Locate and select an existing element classifier. You can restrict the number of choices by selecting the elements from a specific namespace; to do this, click on the 'In Namespace' drop-down arrow and select a namespace.

Add Pattern Dialog

The 'Add Pattern <pattern group> <pattern name> to Diagram' dialog displays when you are using or editing a Design Pattern element.

Reference

Option	Action
Pattern Elements	Access the individual elements contained in the Pattern. From here you can: • Select the action for the individual element (Create, Merge, Instance or Type, as applicable for each element) by clicking on the drop-down arrow, or • Modify the default of the Pattern element or - for a merged element - choose the namespace, by clicking on the ... button on the right of the Default entry
Preview	Displays a preview of the Pattern.
Use Auto Names	Select this checkbox if you want to apply the element auto-naming convention defined for the project.

Element Notes	Display the comments that describe the element in the Pattern. Highlight an element in the 'Pattern Elements' panel to view the notes.
Import as Package Fragment	Select this option to import the Pattern on a new diagram instead of the current diagram. Click on the OK button. Enterprise Architect will : • Create a new Package, with the same name as that of the Pattern, under the currently selected Package in the Browser window • Create a new diagram, with the same name as that of the Pattern, under this Package • Import the Pattern into the new diagram

Notes

- When the option 'Import as Package Fragment' is selected, the 'Pattern Elements' section and 'Use Auto Names' option will become disabled