

The taxonomy of structure and behavior diagram

Structure diagrams show the static structure of the objects in a system. That is, they depict those elements in a specification that are irrespective of time. The elements in a structure diagram represent the meaningful concepts of an application, and may include abstract, real-world and implementation concepts.

Structure diagrams do not show the details of dynamic behavior, which are illustrated by behavioral diagrams. However, they may show relationships to the behaviors of the classifiers exhibited in the structure diagrams.

Behavior diagrams show the dynamic behavior of the objects in a system, including their methods, collaborations, activities, and state histories. The dynamic behavior of a system can be described as a series of changes to the system over time.

This taxonomy provides a logical organization for the various major kinds of diagrams. However, it does not preclude mixing different kinds of diagram types, as one might do when one combines structural and behavioral elements (e.g., showing a state machine nested inside an internal structure). Consequently, the boundaries between the various kinds of diagram types are not strictly enforced.

Variations of structure diagrams often focus on particular structural aspects, such as relationships between packages, showing instance specifications, or relationships between classes. There are no strict boundaries between different variations; it is possible to display any element you normally display in a structure diagram in any variation.

Class diagram

The following nodes and edges are typically drawn in a class diagram:

- Association
- Aggregation
- Class

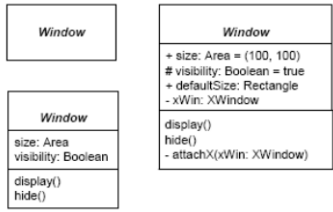
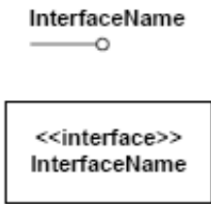
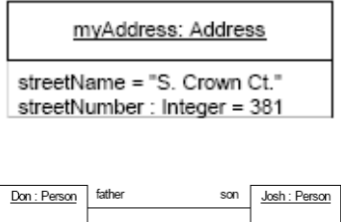
- Composition
- Dependency
- Generalization
- Interface
- InterfaceRealization
- Realization

Object diagram


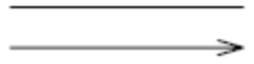
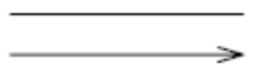


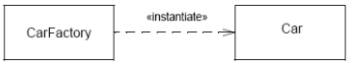

The following nodes and edges are typically drawn in an object diagram:



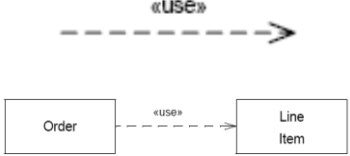
- InstanceSpecification
- Link (i.e., Association)

Graphic nodes included in structure diagrams

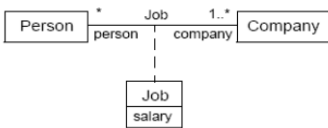
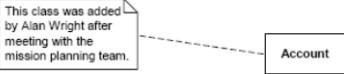
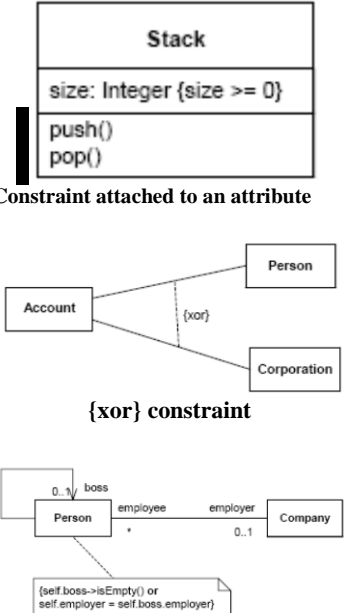
Node Type	Notation	Description
Class		Specifies a classification of objects, as well as, the features that characterize the structure and behavior of those objects.
Interface		An interface is a kind of classifier that represents a declaration of a set of coherent public features and obligations. An interface specifies a contract; any instance of a classifier that realizes the interface must fulfill that contract.
Instance Specification		<p>Instances of any classifier can be shown by prefixing the classifier name by the instance name followed by a colon and underlining the complete name string.</p> <p>An instance specification whose classifier is an association describes a link of that association.</p>

Graphic paths included in structure diagrams

Path Type	Notation	Description
Aggregation		An aggregation represents a whole/part relationship.
Association		<p>An association specifies a semantic relationship that can occur between typed instances.</p> <p>An open arrowhead on the end of an association indicates the end is navigable. A small x on the end of an association indicates the end is not navigable.</p> <p>Notations that can be placed near the end of the line as follows:</p> <ul style="list-style-type: none"> • name – name of association end • multiplicity • property string enclosed in curly braces <ul style="list-style-type: none"> ◦ {ordered} to show that the end represents an ordered set. ◦ {bag} to show that the end represents a collection that permits the same element to appear more than once. ◦ {sequence} or {seq} to show that the end represents a sequence
Link		An instance of an association
Composition		Composite aggregation is a strong form of aggregation that requires a part instance be included in at most one composite at a time. If a composite is deleted, all of its parts are normally deleted with it. Deleting an element will also result in the deletion of all elements of the subgraph below that element.
Dependency	  instantiate dependency	A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. The modification of the supplier may impact the client model elements.
Generalization		<p>A generalization is a taxonomic relationship between a more general classifier and a more specific classifier.</p> <p>Generalization hierarchies must be directed and acyclical.</p>



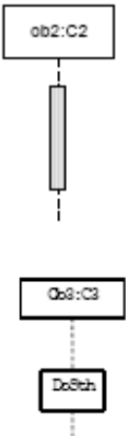
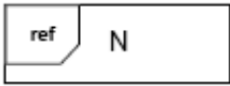
Path Type	Notation	Description
Interface Realization		<p>A specialized Realization relationship between a Classifier and an Interface. This relationship signifies that the realizing classifier conforms to the contract specified by the Interface.</p> <p>A classifier may implement a number of interfaces. The set of interfaces implemented by the classifier are its <i>provided</i> interfaces and signify the set of services the classifier offers to its clients.</p>
Realization		<p>Signifies that the client set of elements are an implementation of the supplier set, which serves as the specification.</p>
Usage	 <p>An Order class requires the Line Item class for its full implementation.</p>	<p>A usage is a relationship in which one element requires another element (or set of elements) for its full implementation or operation.</p>



Miscellaneous elements included in structure diagrams

Type	Notation	Description
Association Class		<p>Defines a set of features that belong to the relationship itself and not to any of the classifiers</p>
Comment		<p>A comment is a textual annotation that can be attached to a set of elements.</p> <p>A comment gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.</p> <p>The connection to each annotated element is shown by a separate dashed line.</p>
Constraint	 <p>Constraint attached to an attribute</p> <p>{xor} constraint</p> <p>Constraint in a note symbol</p>	<p>A constraint is a condition or restriction expressed in natural language text or in a machine readable language for the purpose of declaring some of the semantics of an element. One predefined language for writing constraints is OCL.</p>

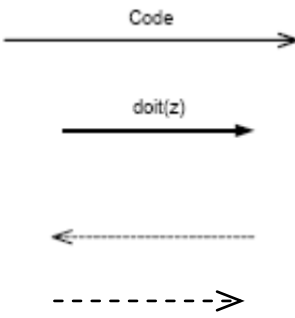
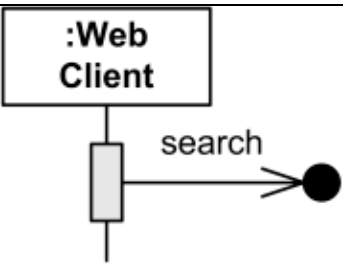
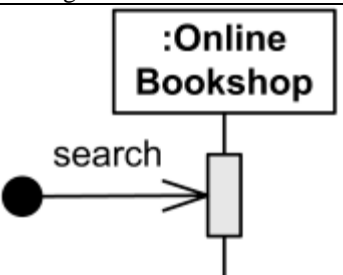
A **Sequence Diagram** describes an Interaction by focusing on the sequence of Messages that are exchanged, along with their corresponding Occurrence Specifications on the Lifelines.

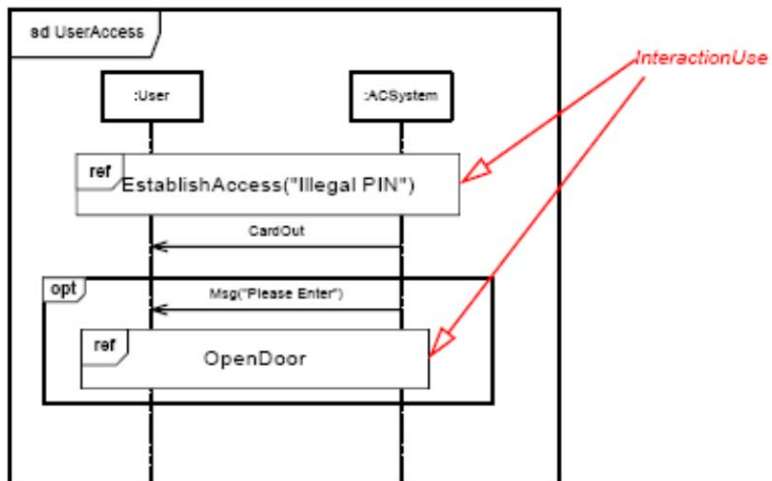
Graphic nodes included in sequence diagrams

Node Type	Notation	Description
Frame		<p>The notation shows a rectangular frame around the diagram with a name in a compartment in the upper left corner.</p> <p>The keyword sd followed by the Interaction name and parameters is in a pentagon in the upper left corner of the rectangle.</p>
Lifeline		<p>A lifeline represents an individual participant in the Interaction.</p> <p>A Lifeline is shown using a symbol that consists of a rectangle forming its “head” followed by a vertical line (which may be dashed) that represents the lifetime of the participant.</p> <p>The Lifeline head has a shape that is based on the classifier for the part that this lifeline represents. Often the head is a white rectangle containing the name.</p> <p>Information identifying the lifeline is displayed inside the rectangle in the following format: <i>[name] : <class_name></i> (<i>:class_name</i> is mandatory but <i>name</i> is optional).</p>
Execution Specification (Activation)		<p>An ExecutionSpecification is a specification of the execution of a unit of behavior or action within the Lifeline. The duration of an ExecutionSpecification is represented by the start ExecutionOccurrenceSpecification and the finish ExecutionOccurrenceSpecification.</p>
InteractionUse		<p>It is common to want to share portions of an interaction between several other interactions. An InteractionUse allows multiple interactions to reference an interaction that represents a common portion of their specification.</p> <p>The InteractionUse is shown as a CombinedFragment symbol where the operator is called <i>ref</i>.</p>

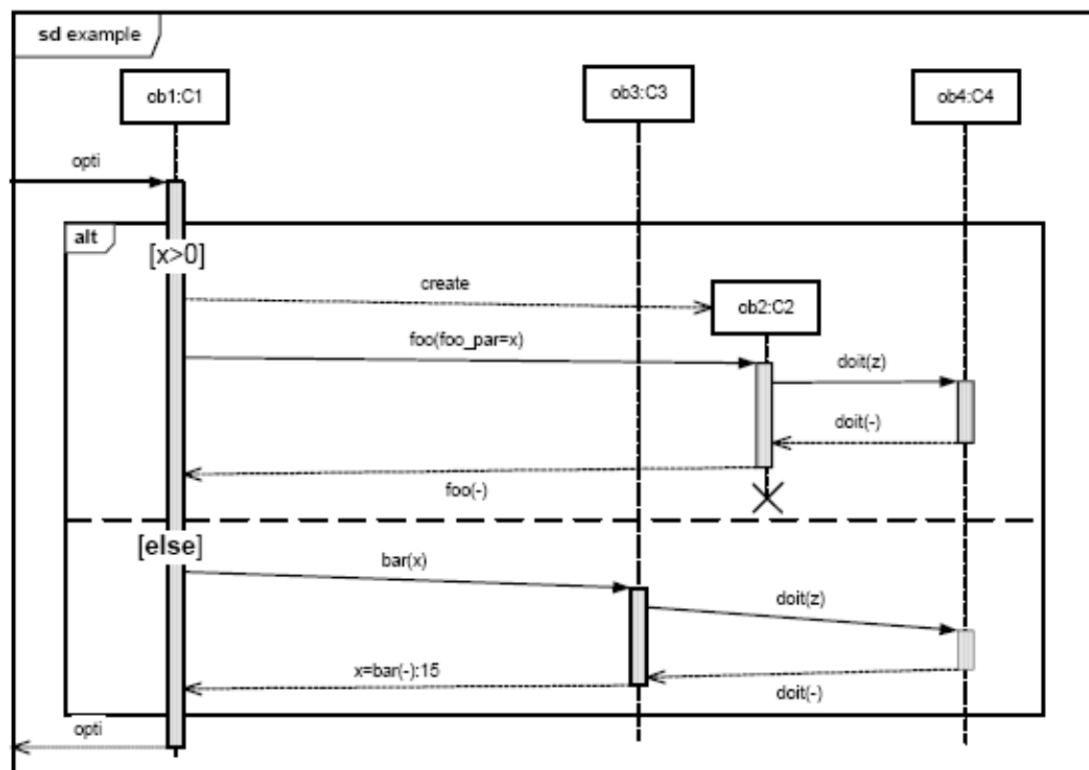
Node Type	Notation	Description
Combined Fragment		<p>A combined fragment is defined by an interaction operator and corresponding interaction operands.</p> <p>Through the use of CombinedFragments the user will be able to describe a number of traces in a compact and concise manner.</p> <p>alt designates 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.</p> <p>opt defines condition to a single call - the call will execute only if the supplied condition is true . Equivalent to an alt with only one trace.</p> <p>par defines that the calls within the fragment run in parallel.</p> <p>critical designates that the CombinedFragment represents a critical region, where the traces of the region cannot be interleaved by other OccurrenceSpecifications (on those Lifelines covered by the region).</p> <p>loop operand 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.</p> <p>break represents a breaking or exceptional scenario that is performed instead of the remainder of the enclosing interaction fragment.</p>
Destruction Event		<p>Destruction of the instance described by the lifeline containing the OccurrenceSpecification that references the destruction event.</p>

Graphic paths included in sequence diagrams

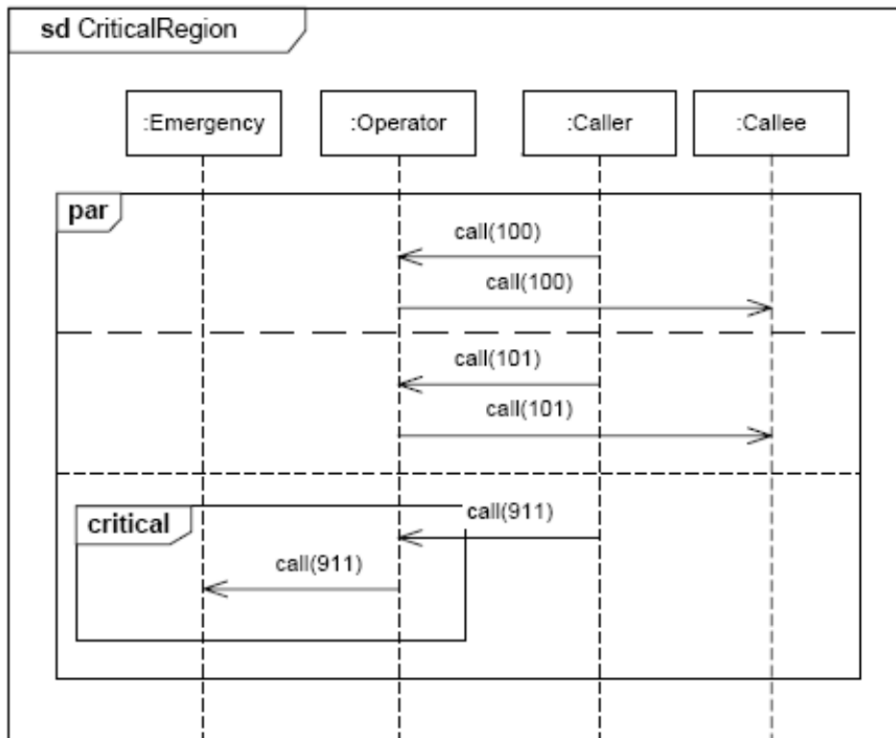
Path Type	Notation	Description
Message		<p>A Message defines a particular communication between Lifelines of an Interaction.</p> <p>A message is shown as a line from the sender message end to the receiver message end.</p> <p>Messages come in different variants depending on what kind of Message they convey. Here we show an asynchronous message, a call and a reply. These are all <i>complete</i> messages.</p> <ul style="list-style-type: none"> • Asynchronous Messages have an open arrow head • Synchronous Messages typically represent operation calls and are shown with a filled arrow head. • The reply message from a method has a dashed line. • Object creation Message has a dashed line with an open arrow. <p>Examples of syntax: <i>mymessage(14, - , 3.14, "hello") // second argument is undefined</i> <i>v=mymsg(16, variab):96 // this is a reply message carrying the return value 96 assigning it to v</i> <i>mymsg(myint=16) // the input parameter 'myint' is given the argument value 16</i></p>
Lost Message	 <p>Web Client sent search message which was lost.</p>	<p>Lost Message is a message where the sending event is known, but there is no receiving event. It is interpreted as if the message never reached its destination. Lost messages are denoted with as a small black circle at the arrow end of the message.</p>
Found Message	 <p>Online Bookshop gets search message of unknown origin.</p>	<p>Found Message is a message where the receiving event is known, but there is no (known) sending event. It is interpreted as if 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.</p> <p>Found messages are denoted with a small black circle at</p>



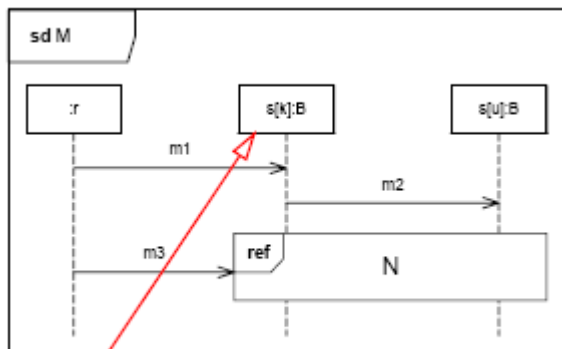
An *InteractionUse* referring the Interaction *EstablishAccess* with (input) argument “Illegal PIN.” Within the optional CombinedFragment there is another *InteractionUse* without arguments referring *OpenDoor*.



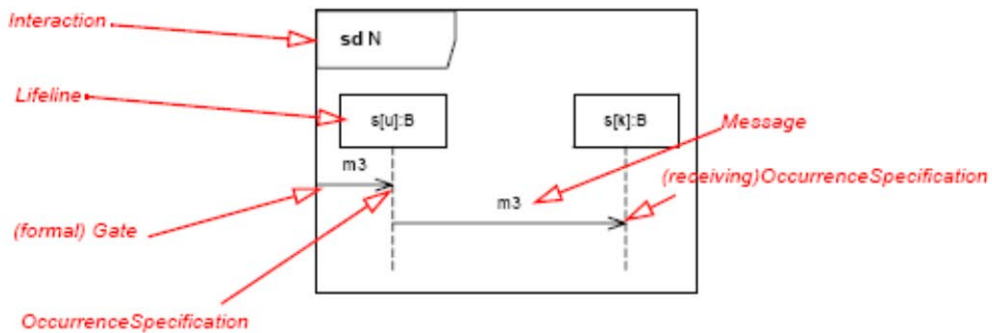
CombinedFragment

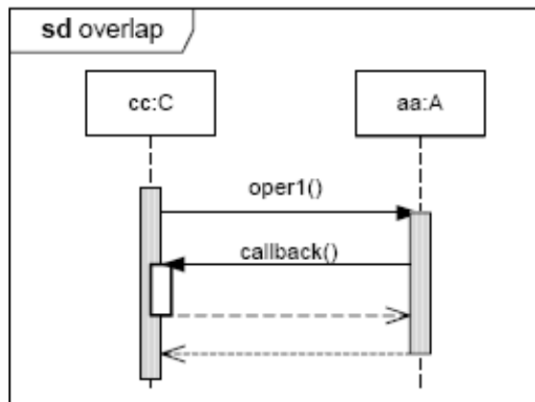


Critical Region - handling of a 911-call must be contiguously handled. The operator must make sure to forward the 911-call before doing anything else. The normal calls, however, can be freely interleaved.



Lifeline





Overlapping execution occurrences

Additional References :

<http://www.uml-diagrams.org/sequence-diagrams.html>