


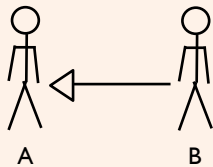
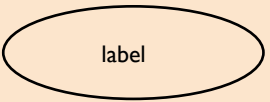
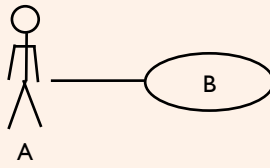
# **UML QUICK REFERENCE**

**SIMPLIFIED FOR OOP  
FOR ANALYSIS ACTIVITIES**

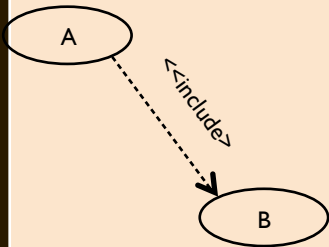
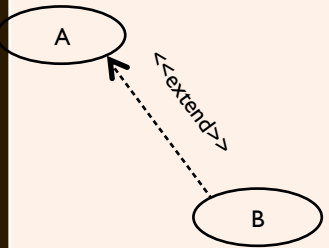
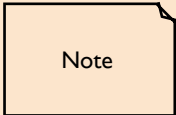
# GENERAL UML BEST PRACTICES FOR ANALYSIS

- Keep an “Analysis” perspective
  - Model the “real-world” problem or at least the world as the stakeholders with like it to be.
  - Don’t try to model software components at this point.
- Let your model help drive your exploration process
  - As discover something, like an actor, use case, association, etc., capture it in your model (in an appropriate diagram)
  - Then, let the “incompleteness” of the diagram trigger questions and drive further exploration.
  - For example, if you discover “B is a subpart of A”, then ask “how many B objects can a single A object contain?” or “Can a B object be part of more than one A object.
- Your entire model is like an “essay” that describes your understanding of the stakeholder’s problem or system
- Each diagram is like a paragraph in that essay
  - Have each diagram focus on one central idea or a few closely coupled ideas
  - Don’t take to say every in one diagram
- Use the right kind of diagram (modeling language) to express the key points that you are trying to communicate with that diagram
- Keep diagram readable
- Avoid line crossings
- Use color effectively

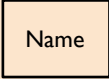
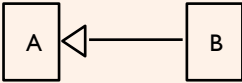
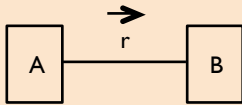
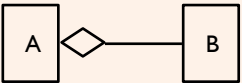
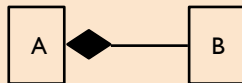
# USE CASE DIAGRAMS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Actor	Someone or something outside of the system being built that has a goal with respect to the system. Formally, it is a set of such people things that all have the same goals.	The name should accurately characterize a single instance of the set, e.g. “User”, “Student”, “Customer”, etc. The name should be a noun or noun phrase. The first letter of each word should be capitalized.
	Generalization/ Specialization	“B is a A”. Or, more formally, the set of B is a subset of the set of A. Therefore, every person or thing in the set B is also in the set A.	When drawing lots of generalization/ specialization relationships, it is okay and even preferred to overlap the lines on the triangle end, so it looks like the triangle has a tail that fans out into many legs.
	Use Case	A goal that an actor has with respect to the system.	The label should accurately and concisely characterize the goal and should be understandable “out of context” from the rest of the diagram”
	Actor “has” Use Case	“Actor A has Use Case B” or “Actor A has the goal represent by Use Case B” with respect to the system. The line represents a relationship between A and B.	Don’t put an adornments on the line. No arrowheads, for example. Note that this line is solid, not dashed.

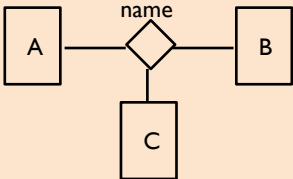
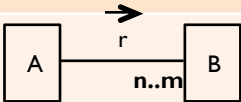
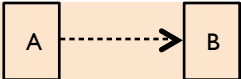
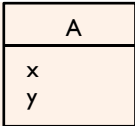
# USE CASE DIAGRAMS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Include Dependency	The goal A includes the sub-goal B. For the system to satisfy goal A, it must also satisfy goal B. An actor that has goal A, also has sub-goal B.	Use <i>Includes Dependencies</i> to break high-level goals down into sub-goals, but be carefully not to decompose the goals too far. Stop if the sub-goals start to sound like they are describing how the system will solve the problem instead of just what the actors' goals are.
	Extend Dependency	The goal B adds to or alters goal A in some special way. Goal B represents additional requirements for the system beyond goal A, that not a subpart of goal A in general.	Use <i>Extends Dependencies</i> to describe related goals, but under exceptional conditions or atypical situations. Note that the direction of the arrowhead goes from B (the extension) to the original goal A. The arrowhead always points in the direction of the dependency.
	Note	Any note that you want to add to the diagram. It can be connected to particular component with a dotted line.	Use notes to write ideas down as you become aware of them. Then, as you gather more information or figure out how to model the ideas, you may be able to move the ideas from the notes into other diagrams with more formal modeling components


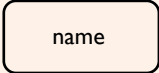
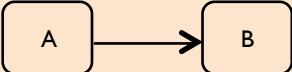
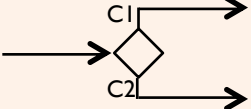
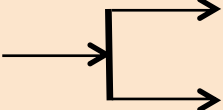
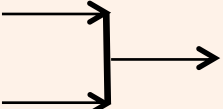
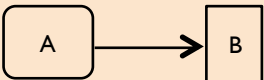
# CLASS DIAGRAMS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Class	A set of objects related to the application domain or system being built. From an analysis perspective, the objects in the set are “real world” objects.	The name should accurately characterize a single instance of the set, e.g. “Bank Account”. The name should be a noun or noun phrase. The first letter of each word should be capitalized
	Generalization / Specialization	“B is a A”. Or, more formally, the set of <i>B</i> is a subset of the set of <i>A</i> . Therefore, every object in the set <i>B</i> is also in the set <i>A</i> . Said another way, every <i>B</i> object is an <i>A</i> object.	When drawing lots of generalization/ specialization relationship, it is okay to overlap the lines on the triangle side, so it looks like the triangle has a tail that fans out into many legs.
	Named Binary Association	A set of <b>persistent links</b> between <i>A</i> and <i>B</i> objects with the meaning “ <i>A</i> <i>r</i> <i>B</i> ”, where “ <i>r</i> ” is some verb or preposition phrase that expresses how <i>A</i> objects can be linked to <i>B</i> objects	For “ <i>r</i> ”, use a verb phrase that convey the meaning of a single link as a clear and precise as possible. Don’t just use “has” – that coves no additional meaning besides what the line already tells the reader.
	Aggregation	“B is a subpart of A” or “A contains a B”. More formally, given an <i>A</i> object, it should be possible it pick it part and find some piece that is a <i>B</i> object. Note that <i>Aggregations</i> are actually just a special kinds of <i>Binary Association</i> .	Use <i>Aggregation</i> when something is really a subpart of something else, not just want you want to say “A has a B”. That later can be ambiguous. Named association are more expressive during analysis and are a better choice if the true semantics do not mean “subpart”.
	Composition	A <i>Composition</i> relation is specialization an <i>Aggregation</i> relation, where: <ul style="list-style-type: none"> <li>- If the <i>A</i> object is destroyed, then all of the <i>B</i> objects it contains are also destroyed</li> <li>- The relations is irreflexive, asymmetric, and transitive.</li> </ul>	Think about using <i>Composition</i> when the subpart object has to part of an aggregate object and is never part of more than one aggregate object.

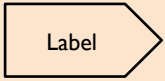


# CLASS DIAGRAMS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Named N-Nary Association	A set of <b>persistent links</b> between objects in sets A, B, and C, where the links have a meaning indicated by the name. There can be 3 more classes connected to the triangle in the center.	For “name”, use a complete sentence that include all the class names (A, B, C, etc.) conveys the meaning of single link as clearly and precisely as possible.
	Multiplicity Constraint	An A object can be related to between <i>n</i> and <i>m</i> B objects. Note that the “n..m” constraint is written on the B side.	Before finishing a model, complete and double check all multiplicity constraints. But, only put multiplicity constraints on associations. Do not put them on generalizations/specializations or dependencies.
	Dependency	An A object may depend on a B object. This does not represent persistent links between A and B objects, only potential dependencies.	Use sparingly in class diagrams during analysis, if at all. Some okay uses are to show how A objects might be “instances” of B objects, where B objects are templates or abstract types.
	Attributes	All of the objects in Class A can have attributes <i>x</i> and <i>y</i> . An attribute is a property or placeholder for some value. That value could be another object.	During analysis, be careful about defining attributes too early. They limit your expressiveness and do not encourage as many questions about the domain as associations. Be willing to convert attributes to associations to add detail.

# ACTIVITY DIAGRAMS

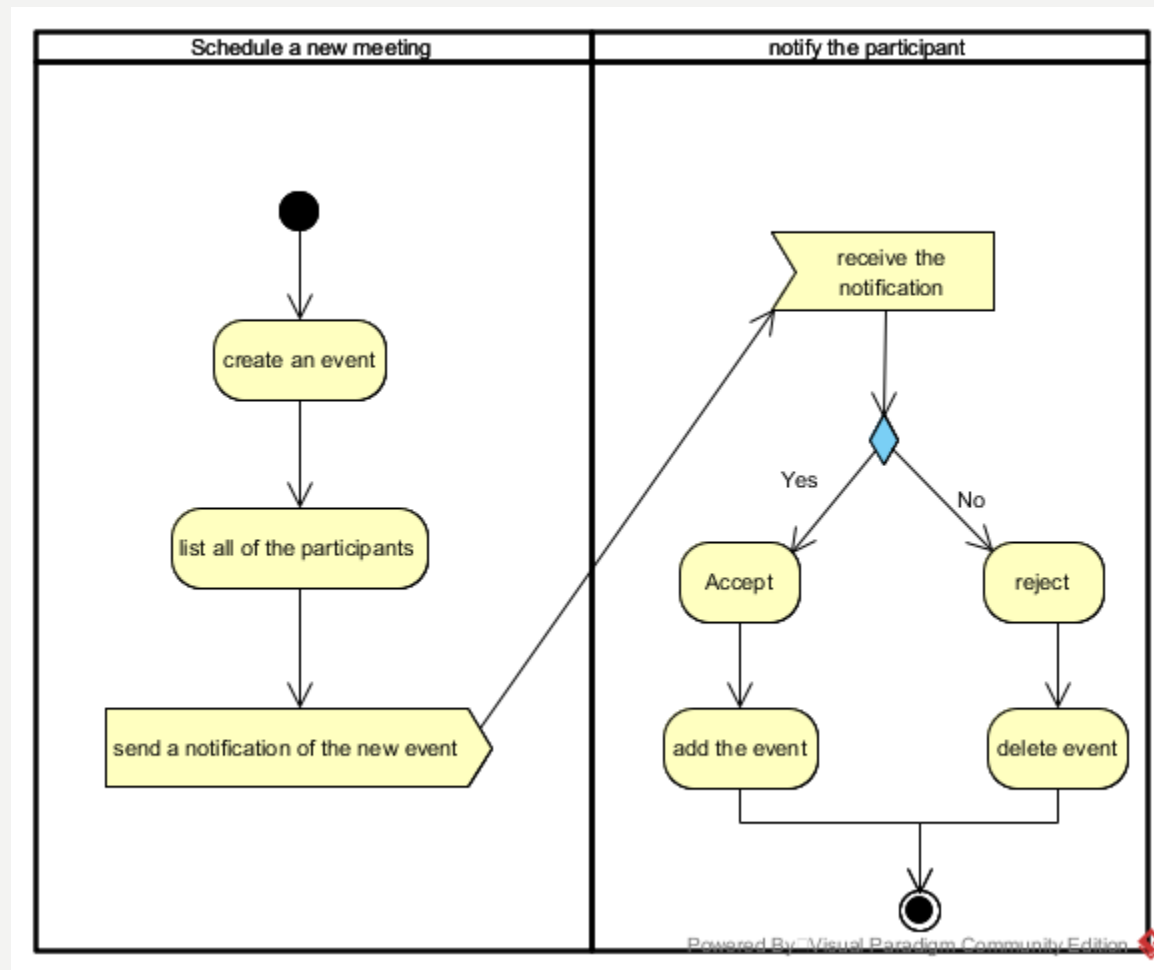
Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Initial Node	This is the <b>starting point</b> for the process	Try to have just one or at most one per parallel thread of activity
	Action Node	An <b>action</b> represents a step in the process or activity.	Have the name clearly and accurately communicate the purpose or nature of the action
	Control Flow	Action “A” is followed by Action “B”	For readability of the diagram, avoid line crossing.
	Decision Node	If C1 is true, then follow that control flow, <b>else if</b> C2 is true flow that control flow.	Be sure the conditions are mutual exclusive and complete.
	Fork Node	<b>Split a control flow into two or more concurrent flows.</b>	Use to create concurrent threads of activity in a process.
	Join Node	Join two or more concurrent control flows back <b>into one control flow</b> .	Use to bring concurrent threads of activity back together.
	Object Flow	<b>Action A creates or interacts with object B</b>	Use object flows to show how actions create, delete, or call methods of specific objects, as part of what they do.

# ACTIVITY DIAGRAMS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Send Signal Action	This is a special kind of action that represents the sending of a signal or event to some target object(s). This action completes immediately. In other words, it does not wait for target object(s) to receive and process the signal.	Use this to trigger behaviors within other objects or to trigger actions other swim lanes when those parallel thread of activity are asynchronous.
	Accept Signal Action	This is special kind of action that represent the receiving of signal or event. It can be connected of Send Signal Action via a dotted line, but doesn't have to be. The action complete when a signal is received	Use this to represent an action that listens and waits for a signal from some other parallel thread of activity.
	Final Node	This represent the <b>end of an activity or process.</b>	If there are parallel thread of activity in the process, the first one to reach the final node stops the process for all. So, use a join node before the final node if you want to synchronize termination of parallel threads.



# SCHEDULE A NEW MEETING

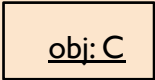
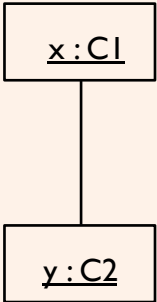
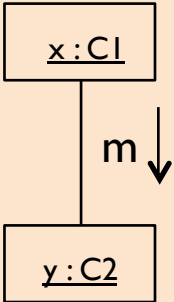


# INTERACTION DIAGRAMS

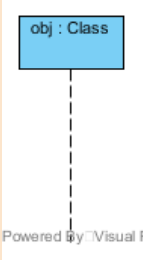
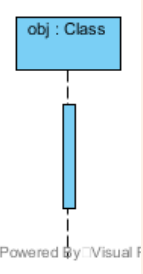
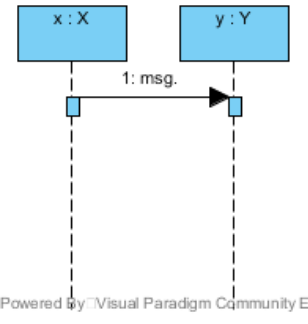
- An *interaction diagram* shows an interaction, which consists
  - of a set of objects
  - messages between those objects
  - The sequence or timing of those messages
  - their links (communication diagrams only)
- Kinds of interaction diagrams
  - Communication diagrams
  - Sequence diagrams

# INTERACTION DIAGRAMS: COMMUNICATION DIAGRAMS

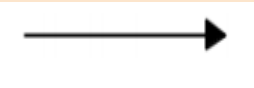


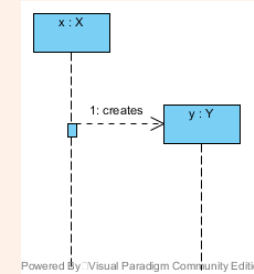


## From a Design Perspective

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	An object	An object, labeled “obj”, from some class, named “C”. The object label is optional and is only need if you need to refer that object in other places in the diagram.	Remember that all interaction diagrams (communication or sequence) model <b>from an object perspective</b> , not a class perspective
	A communication channel	There is a communication channel between x and y. That channel may be <b>the calling of methods</b> , a network communication channel, or some other from of communication. The diagram does not restrict or prescript the communication channel.	If x talks with y, think about how x becomes aware of y. Here are some possibilities: 1) x created y, 2) something else gave x knowledge of y in someway, or 3) x discovered y being querying something else. In other words, when defining the existence of a communication channel in a communication diagram, consider how it got created.
	A message being send from x to y	A message, m, is send from object x to object. A message can be a method call, signal or interrupt, inter-process communication, or other type of communication. Method call are the most common form or way of sending messages.	Typically, message m corresponds to a public method of object y


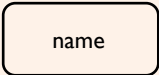
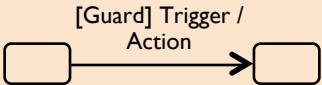
# INTERACTION DIAGRAMS: SEQUENCE DIAGRAMS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
 <p>Powered by Visual Paradigm</p>	An object with lifeline	A object, labeled “obj”, from some class, named “Class”. The object label is optional and is only need if you need to refer that object in other places in the diagram.	Remember that all interaction diagrams (communication or sequence) model from an object perspective, not a class perspective
 <p>Powered by Visual Paradigm</p>	Focus of control (execution occurrence)	an execution occurrence (shown as tall, thin rectangle on a lifeline) <b>represents the period during which an element is performing an operation.</b> The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively.	An Event is any point in an interaction where something occurs.
 <p>Powered by Visual Paradigm Community Edition</p>	A message being send from x to y	Messages (or signals) on a sequence diagram are specified using an arrow from the participant (message caller) that wants <b>to pass the message to the participant (message receiver)</b> that is to receive the message	A Message (or stimulus) is represented as an arrow going from the sender to the top of the focus of control (i.e., execution occurrence) of the message on the receiver’s lifeline

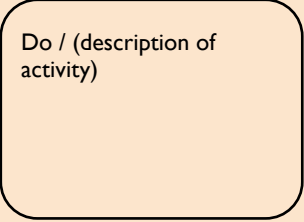
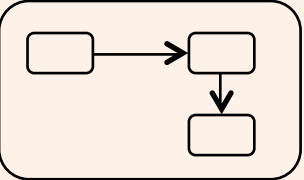
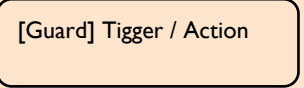
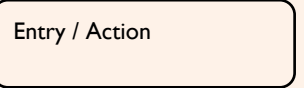
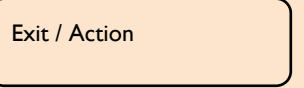

# SEQUENCE DIAGRAMS

Modeling Component	What it is?	What does it represent or mean
	Synchronous	A synchronous message between active objects <b>indicates wait semantics</b> ; the sender waits for the message to be handled before it continues. This typically shows a method call.
	Asynchronous	With an asynchronous flow of control, there <b>is no explicit return message</b> to the caller. An asynchronous message between objects <b>indicates no-wait semantics</b> ; the sender does not wait for the message before it continues. This allows objects to execute concurrently.
	Reply	This shows the return message from another message
	Create	This message results in the creation of a new object. The message could call a constructor for a class if you are working with Java, for example.
	Lost	A lost message occurs when <b>the sender of the message is known but there is no reception of the message</b> . This message allows advanced dynamic models to be built up by fragments without complete knowledge of all the messages in the system. This also allows the modeler to consider the possible impact of a message's being lost.
	Found	A found message indicates that although <b>the receiver of the message is known in the current interaction fragment, the sender of the message is unknown</b> .

# STATE CHARTS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Initial State	This state represent the “nothingness” previous to the object coming into existence. Inside a high-level state, it can represent the <b>starting state</b> for an object that enters the high-level state	Show at least one initial state in a state chart that represents an object’s behavior.
	State	Represents a <b>mode or operation or the abstraction</b> of condition for that object being described by the state chart	Use a Mealy or a Moore approach to identify the states of an object, but don’t mix the two in a single state chart. Also, use high-level states to represent complex modes of operation or conditions.
	Transition	<p>A transition represent a way in which an <b>object can move or change from one state to another</b>. A transition can include a <b>Guard, Trigger, and Action</b>.</p> <ul style="list-style-type: none"> <li>• The Guard <b>is a bool condition</b>, that if true, enables the transition.</li> <li>• The Trigger is <b>an event</b> that represent when the transition will be take.</li> <li>• The action <b>is something that take place during the transition</b>.</li> </ul> <p><b>A transition, once started, cannot be interrupted</b></p>	<p>Try to ensure that the guards and triggers leaving a state are mutually exclusive.</p> <p>If transition has an action, make sure it is one that <b>should not be interrupted</b> and will always complete.</p> <p>In transition actions should be simple and atomic.</p>

# STATE CHARTS

Modeling Component	What it is?	What does it represent or mean	Best practices in using this component
	Internal Activity	An object may perform non-trivial activities while in a state. These activities can be described <b>informally in text preceded by a “Do / “</b> or by an internal state chart. (See High-level State)	Use activities to represent tasks that can be interrupted.
	High-level State	A state can include <b>an internal state</b> chart that describes the activity an object performs while in a high-level state	Use high-level states to describe complex activities when a textual description is too informal.
	Internal Transition	Internal transition is just like a regular transition, except that the object doesn't change states. It doesn't leave and re-enter the state in which it is defined	Use internal transitions to describe simple actions that an object may perform while in a state.
	Entry Transition	This represents an <b>internal transition</b> which is fired (taken) as an object <b>enters a state</b>	Use entry transitions to describe actions that an object must perform while entering a state
	Exit Transition	This represents an <b>internal transition</b> which is fired (taken) as an object <b>leave a state</b>	Use exit transitions to describe actions that an object must perform before leaving a state
	Final State	This state represents the “nothingness” of when an object ceasing to exit	If an object can die (cease to exist), show at least one final state.