Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modeling tools include:[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)

[**Use Case Diagram**](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/use_case.htm) displays the relationship among actors and use cases.[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)

[**Class Diagram**](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/class.htm) models class structure and contents using design elements such as classes, packages and objects. It also displays relationships such as containment, inheritance, associations and others. [1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)

[**Interaction Diagrams**](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/interaction.htm)

* **Sequence Diagram**displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)
* **Collaboration Diagram**displays an interaction organized around the objects and their links to one another. Numbers are used to show the sequence of messages.[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)

[**State Diagram**](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/state.htm) displays the sequences of states that an object of an interaction goes through during its life in response to received stimuli, together with its responses and actions.[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)

[**Activity Diagram**](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/activity.htm)displays a special state diagram where most of the states are action states and most of the transitions are triggered by completion of the actions in the source states. This diagram focuses on flows driven by internal processing.[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)

[**Physical Diagrams**](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/physical.htm)

* **Component Diagram** displays the high level packaged structure of the code itself. Dependencies among components are shown, including source code components, binary code components, and executable components. Some components exist at compile time, at link time, at run times well as at more than one time.[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)
* **Deployment Diagram**displays the configuration of run-time processing elements and the software components, processes, and objects that live on them. Software component instances represent run-time manifestations of code units.[1](http://atlas.kennesaw.edu/%7Edbraun/csis4650/A&D/UML_tutorial/diagrams.htm#1)