An external entity in a Data Flow Diagram (DFD) represents an external system, organization, or person that interacts with the system being modeled. It is depicted as a rectangle and helps define the system boundaries by illustrating how data flows into or out of the system.

In a Data Flow Diagram (DFD), a process represents a function, task, or transformation that takes input data and produces output data. It is typically depicted as a circle or rectangle and helps illustrate the specific operations or computations within the system.

In a Data Flow Diagram (DFD), a data store represents a repository where data is stored and retrieved. It is depicted as a rectangle and signifies a database, file, or any storage system within the system being modeled. Data stores help show how data persists and is maintained in the system.

In a Data Flow Diagram (DFD), a data flow represents the movement of data between processes, external entities, and data stores within a system. It is depicted as an arrow and illustrates the path that data takes as it travels through the system. Data flows help depict the flow of information and the relationships between different components in the system.

In a Use Case Diagram, the use case subject is the system or entity being analyzed, representing the boundaries of the functionality you are modeling.

- Use Case: Describes a specific interaction between external entities (actors) and a system to achieve a goal. It represents a discrete unit of functionality provided by the system.
- Actor: Represents an external entity that interacts with the system by participating in one or more use cases. Actors can be individuals, other systems, or anything else that interacts with the system.
- Association: Represents a relationship between two or more actors and/or use cases. It is typically depicted by a solid line connecting the associated elements.
- Directed Association: Similar to association but with an arrow indicating the direction of the relationship. It shows the flow of communication between actors and use cases.
- Generalization (or Inheritance): Represents an "is-a" relationship between use cases or actors. It is depicted by a solid line with a triangle arrowhead pointing to the parent (superclass).

- Dependency: Represents a relationship where one element (e.g., a use case) depends on another. It is depicted by a dashed line and typically indicates that a change in one element may affect another.
- Include: Represents the inclusion of one use case by another. It is depicted by a dashed arrow from the including use case to the included use case.
- Extend: Represents the extension of a use case by another. It is depicted by a dashed arrow from the extending use case to the extended use case.

Statechart diagrams and activity diagrams are both UML (Unified Modeling Language) diagrams used in software engineering to model different aspects of a system's behavior. Here are the key differences between them:

1. **Focus of Modeling:**
   - **Statechart Diagrams:** Primarily model the dynamic behavior of a system in response to external stimuli by representing the different states an object can be in and the transitions between those states.
   - **Activity Diagrams:** Model the workflow or flow of activities within a system. They are used to illustrate the sequence of activities or steps that need to be performed.

2. **Elements of Modeling:**
   - **Statechart Diagrams:** Main elements include states, transitions, events, and actions. States represent conditions or situations, transitions depict the change from one state to another, events trigger transitions, and actions represent what happens during a state.
   - **Activity Diagrams:** Main elements include activities, actions, decisions, and control flow. Activities represent units of work, actions represent specific operations, decisions represent conditional branching, and control flow indicates the sequence of activities.

3. **Representation of Behavior:**
   - **Statechart Diagrams:** Emphasize the various states an object goes through and how transitions occur between these states in response to events.
   - **Activity Diagrams:** Emphasize the flow of activities and the sequence in which they are executed. They are particularly useful for modeling business processes or system workflows.

4. **Usage:**
   - **Statechart Diagrams:** Commonly used for modeling the behavior of individual objects or the dynamic aspects of a class.

- **Activity Diagrams:** Widely used for modeling high-level processes, workflows, and business processes. They provide a more holistic view of the system's behavior.

5. **Concurrency:**
   - **Statechart Diagrams:** Focus on the states and transitions of a single object. Concurrency is typically represented through parallel states.
   - **Activity Diagrams:** Explicitly model concurrency, showing parallel or concurrent activities and how they interact.

In summary, statechart diagrams focus on the states and transitions of individual objects, emphasizing the dynamic behavior, while activity diagrams focus on the flow of activities and processes within a system, providing a broader view of the system's behavior.