An accurate Object oriented requirements are the first step in developing a good system. Discuss, what are we meaning OO requirements, and how are they differ from regular system requirements? make sure to shed light on their representation by Use Case tool.

The traditional approach to requirements puts a heavy emphasis on data and the flow of data. For this reason, Data Flow Diagrams (DFDs) are used to chart transactions in terms of external agents initiating processes that interact with data stores. Context diagrams provide layers of detail into particular DFDs. This method of diagramming requirements appears to be subject to excessive complexity and logical errors.

Object Oriented requirements involve identifying "things" as classes that can encapsulate data and processes. Objects interact with one another, messaging each other with commands, data, or even objects. Objects can also have states, which aids in their reusability. Unlike DFD fragments, which are very context specific, a single object has a place in a variety of contexts.

OO requirements are diagramed in use cases, which are much simpler to read and understand in comparison to UML and DFD diagrams. Use Case diagrams are adapted into Activity and System Sequence Diagrams (SSDs), which focus on how classes of objects interact with one another. For an increased level of detail, an object may have a State Machine Diagram that contains the specifics of how it holds data or encapsulates functionality.

By encapsulating processes, data, and functionality into objects, the Systems Analyst is able to wrap up the details of these processes into "things," which are easier to understand. When someone reads the diagram, they don't have to know anything about how the objects themselves work, only that the objects exist and how they interact. When the details of an object's workings are needed, State Machine Diagrams provide that level of detail.