**1. What is a responsibility?**

The focus of object design is to identify classes and objects, decide what methods belong where and how these objects should interact. Responsibilities are related to the obligations of an object in terms of its behavior.

**2. Give a definition of responsibility.**

Responsibilities are assigned to classes during object design. Responsibilities related to “knowing” can often be inferred from the Domain Model

**3. What are the three kinds of responsibility?**

doing – doing something itself, initiating action in other objects, controlling and coordinating activities in other objects

knowing – knowing about private encapsulated data, about related objects, and things it can derive or calculate。

Guideline: for software domain objects, the domain model, because of the attributes and associations it illustrates, often inspires the relevant responsibilities related to "knowing".

**4. How are responsibilities used in *use case realization*?**

A use case realization describes the design for a given use case, in terms of collaborating objects. UML interaction diagrams are used to illustrate use case realizations. Each use case identifies a number of system events. These are shown in system sequence diagrams. The system events become the starting messages that enter the controllers for the domain, as shown in a domain layer interaction diagram.

**5. What is the relationship between *design for change* and the GRASP pattern *Protected Variations*?**

*Design for change* strives for minimizing impact or cost of change in the software, which is accomplished by PV. PV identifies points of predicted variation or instability; assign responsibilities to create a stable interface around them. It allows the variations and instability in objects, subsystems and systems not cause an undesirable impact on other elements. This allows modifications to be localized

**6. What is the relationship between** **Jacobson's *control object* and the GRASP pattern *Controller*?**

**Jacobson’s control object** : use case handlers as described in the controller pattern. UI Object (interface) is separate from Controller (implementation). UI Object is not allowed to be a controller

Controller pattern: UI objects and the UI layer should not have responsibility for fulfilling system events. System operations should be handled in the application logic or domain layers of objects rather than in the UI layer

**7. What are their relationship to the *Controller* of MVC?**

The controller in MVC differs from GRASP controller

MVC controller is part of the UI layer and controls the UI interaction and page flow

GRASP controller is part of the domain layer and controls or coordinates the handling of work request, essentially unaware of what UI technology is being used (Swing, web UI, etc)