**Checkpoints: Analysis Class**

* + The analysis class name is unique.
  + The class is used in at least one collaboration.
  + The class's brief description captures the purpose of the class and briefly summarizes its responsibilities.
  + The class represents a single set of cohesive responsibilities.
  + Responsibility names are descriptive and the responsibility descriptions are correct.
  + The responsibilities of the class are consistent with the expectations placed upon it by collaborations in which the class participates.
  + All classes needed to perform the use cases (excluding design classes) have been identified.
  + All actor-system interactions are supported by some boundary class.
  + No two classes possess the same responsibility.
  + Each analysis class represent a distinct set of responsibilities, consistent with the purpose of the class.
  + Relations between use cases (include, extend, generalization) are handled in a consistent way in the analysis model.
  + The complete lifecycle (creation, usage, deletion) of each analysis class is accounted for.
  + The class fulfills the responsibilities required of it, either directly or through delegation.
  + Classes collaborations are supported by appropriate associations.
  + All requirements on the class have been addressed.
  + If the class is a boundary class, all the requirements of the actor have been addressed (including input error).

## Checkpoints: Design Model

### General

* + The objectives of the model are clearly stated and visible.
  + The model is at an appropriate level of detail given the model objectives.
  + The model's use of modeling constructs is appropriate to the problem at hand.
  + The model is as simple as possible while still achieving the goals of the model.
  + The model appears to be able to accommodate reasonably expected future change.
  + The design is appropriate to the task at hand (neither too complex nor too advanced)
  + The design appears to be understandable and maintainable
  + The design appears to be implementable

### Layers

* + There are no more than seven (plus or minus two) layers.
  + The rationale for layer definition is clearly presented and consistently applied.
  + Layer boundaries are respected within the design.
  + Layers are used to encapsulate conceptual boundaries between different kinds of services and provide useful abstractions which makes the design easier to understand.

**Checkpoints: Data Model**

* + All persistent classes that use the database for persistency have been mapped to database structures.
  + Many-to-many relationships have an intersecting table.
  + Primary keys have been defined for each table, unless there is a performance reason not to define a primary key.
  + The storage and retrieval of data has been optimized.
    - If a relational database is used, tables have been demoralized (where necessary) to improve performance.
    - Where demoralization has been used, all update, insert and delete scenarios have been considered to ensure the demoralization does not degrade performance for those operations.
  + Indexes have been defined to optimize access.
  + The impact of index updates has been considered in the other table operations.
  + The distribution of data has been planned.
  + Data and referential integrity constraints have been defined.
  + A plan exists for maintaining validation constraints when the data rules change.
  + Stored procedures and triggers have been defined.
  + The persistence mechanism uses stored procedures and database triggers consistently.