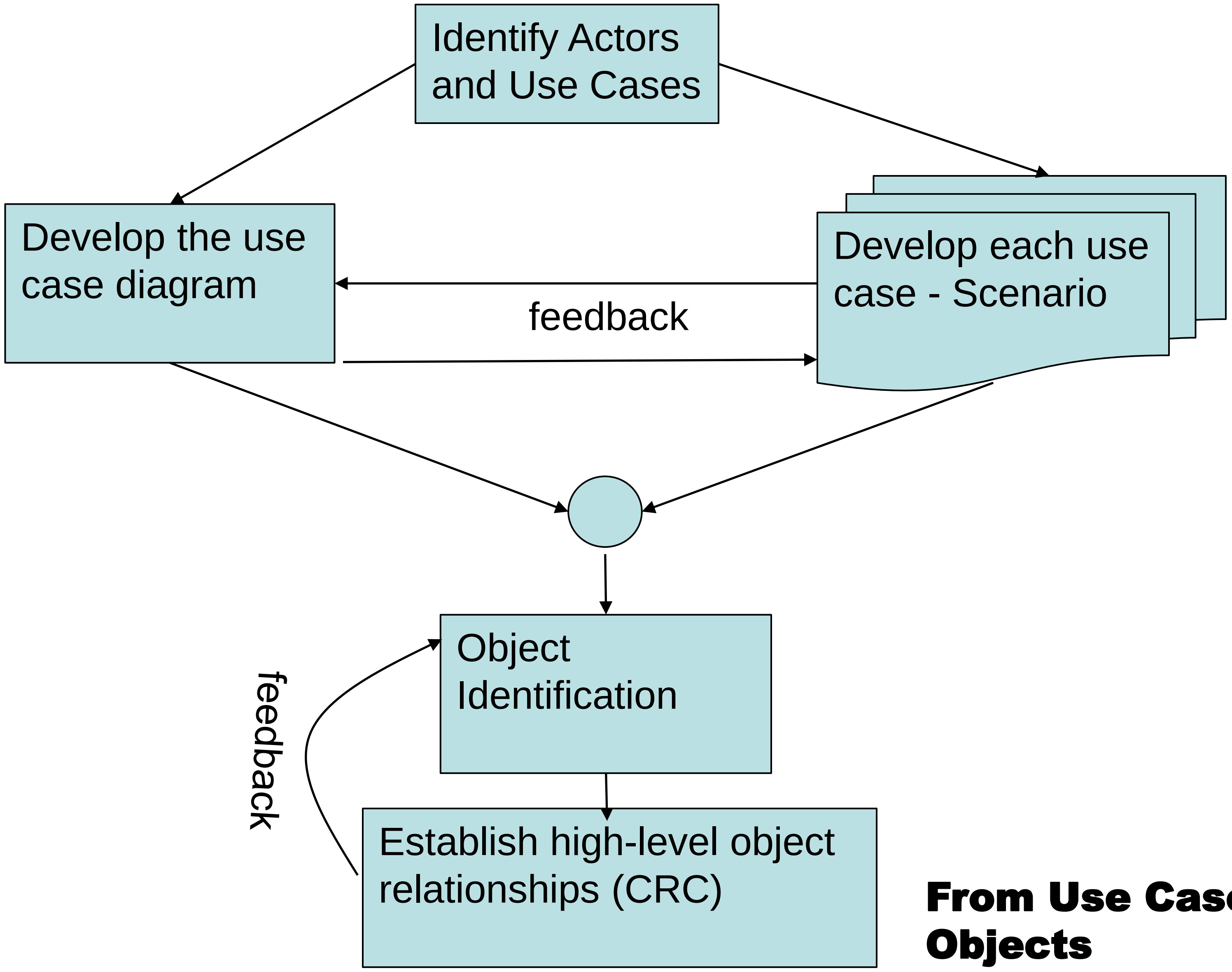
Identifying Objects - Approach


Identify Actors and Use Cases
Develop the use case diagram
Develop each use case - Scenario
feedback
Object Identification
feedback
Establish high-level object relationships (CRC)

From Use Cases to Objects

Categorization of Classes

- it is sometimes helpful during early design to categorize potential classes
  - boundary
  - entity
  - control
- this categorization represents a separation of concerns that supports maintainability and minimizes complexity

Boundary Class

- serves as an interface between the system anc its environment
  - reduces complexity (minimizes communication and coordination between other classes and the actors by providing an interface neutral form of communication)
  - helps maintainability (normally only boundary class will nee to change when interface changes)
- types of boundary classes include:
  - user interfaces (one boundary class per window)
  - other system interfaces

Entity Class

- entity classes are independent of the system’s interfaces
- correspond to real-world entities that the system manipulates

Control Class

- responsible for coordinating boundary and entity classes
  - receives or handles system events
- often one control class per use case
  - control classes encapsulate behavior of use case
- control object is created at beginning of use case and is destroyed at the end of the use case
  - this means persistence must be considered when use cases can span a single execution of the system
  - control classes implement the Controller Pattern

Ira A. Fulton Schools of

Identifying Object Classes

- Structures
- External systems
- Devices
- Roles
- Operating procedures
- Places
- Organizations
- Things that are manipulated by the system to be built

Analysis Classes

- *External entities* (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.
- *Things* (e.g, reports, displays, letters, signals) that are part of the information domain for the problem.
- *Occurrences or events* (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.
- *Roles* (e.g., manager, engineer, salesperson) played by people who interact with the system.
- *Organizational units* (e.g., division, group, team) that are relevant to an application.
- *Places* (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.
- *Structures* (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects.

Defining Operations

- operations that manipulate data in some way (adding, deleting)
- operations that perform a computation
- operations that inquire about the state of an object
- operations that monitor an object for the occurrence of a controlling event.

- Grammatical parse – to select those operations/verbs are isolated.

UML: Class Responsibility Collaborator diagrams and Class Diagrams

- CRC diagram represents the responsibilities (functions) of each class and other collaborating classes to achieve those functionalities
- Class diagrams describes static structure of the system, objects/classes including attributes and operations, and relationship among them.
- Class Diagrams are used in analysis and design; the level of detail expressed is dependent upon the modeling being done.