CS 3773
Software Engineering
Lecture 6

Dr. Mark Robinson
Office: NPB 3.350

Requirements Analysis Models

- Different views/perspectives of requirements artifacts
- Benefits:
  - Can give a clear picture of what customer wants
    - Helps find gaps, conflicts, inconsistencies in requirements
  - Provides a bridge to design phase
  - Can make it easier for customer to validate

Types of Analysis Models

- 4 primary types:
  - **Scenario**: use cases and diagrams, user stories, usage scenarios, sequence diagrams
  - **Structural**: object details, class diagrams, ERDs
  - **Behavioral**: state diagrams
  - **Flow** : DFDs, activity diagrams

The UML

- Unified Modeling Language
- Developed by Rational Software Corp. in mid 90's
- A standardization of software modeling notation
  - Adapt,simplify,expand,merge widely used notations of the day (Booch, Rumbaugh, Jacobsen)
- Widely used today (at least some parts of the UML)

Modeling Guidelines

- Stay high-level and abstract
- Don’t get technical unless client requires it
- Loose coupling
- Models should be useful for all SHs
- Keep it simple
  - Complex models should be decomposed

Use Case Diagrams

- Simple model for analyzing which actors (humans and external systems) can access what functionality
- Can be used to guide further requirements elaboration and for access control testing
  - Can take the place of a list of user roles and use cases

Use Case Diagrams

Actor

External entity/actor

Use Case/Functional Req.

System Boundary

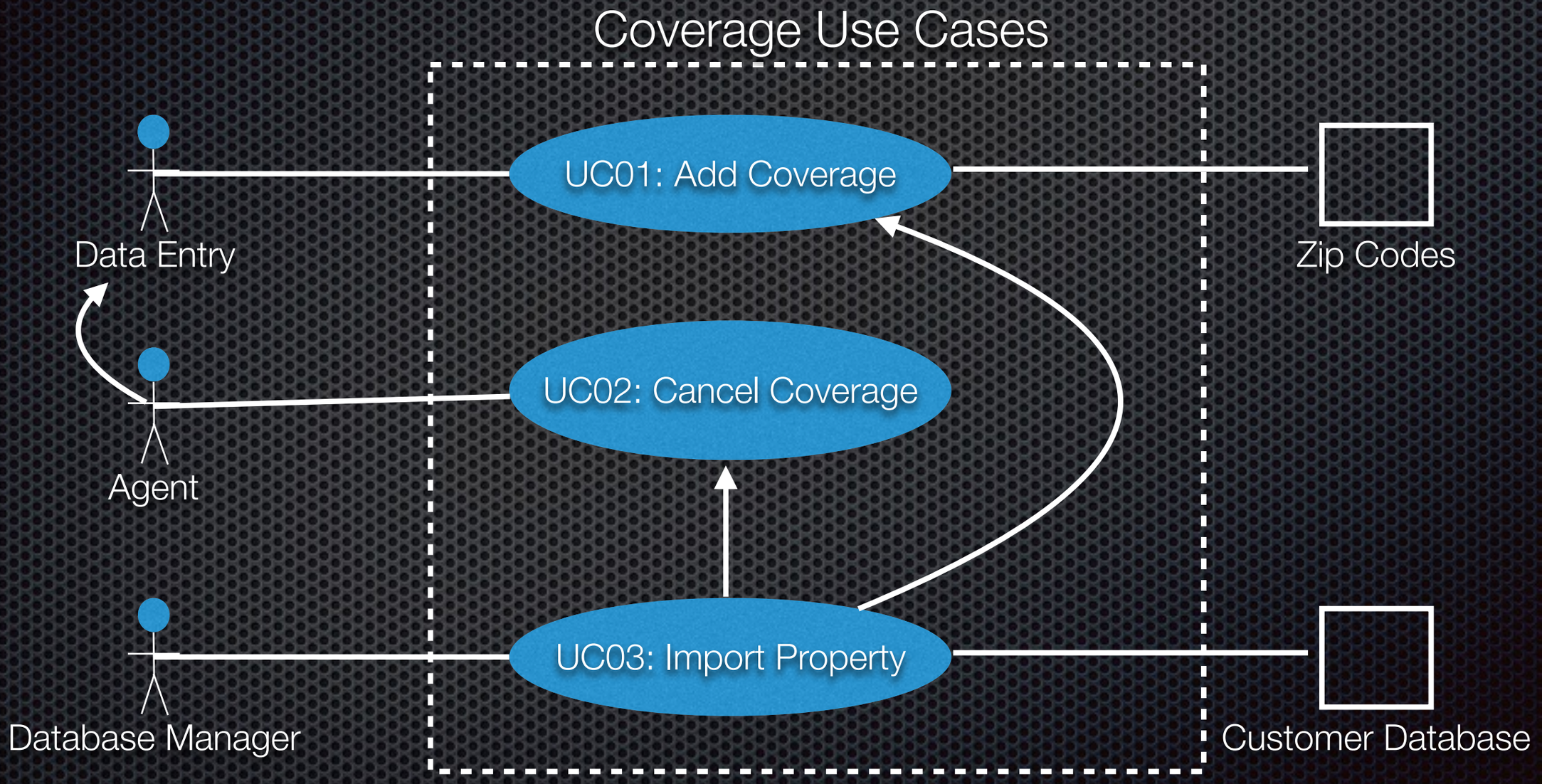
Dependency or Generalization/Inheritence

Association

Add Coverage UC Detail

	Use Case
Name	AddPropertyCoverage
ID	UC01
Description	Adds a property and coverage to the system
Actors	Agent, Data Entry, DB Manager (users); Zip code db
Preconditions	Actor logged in and has access to function
Main Flow	1. user selects Add Coverage function 2. the system shows a blank property coverage form 3. user enters address, zip, insured value, coverage dates 4. zip code db fills in city, state, and county 5. user submits property coverage to system
Postconditions	system creates new property coverage user returns to menu choices
Alternative Flows	NotLoggedIn during Preconditions NoAccess after Step 1 InvalidZipCode after Step 3 InvalidInsuredValue before Step 5 InvalidCoverageDate before Step 5

Insurance UC Diagram Example

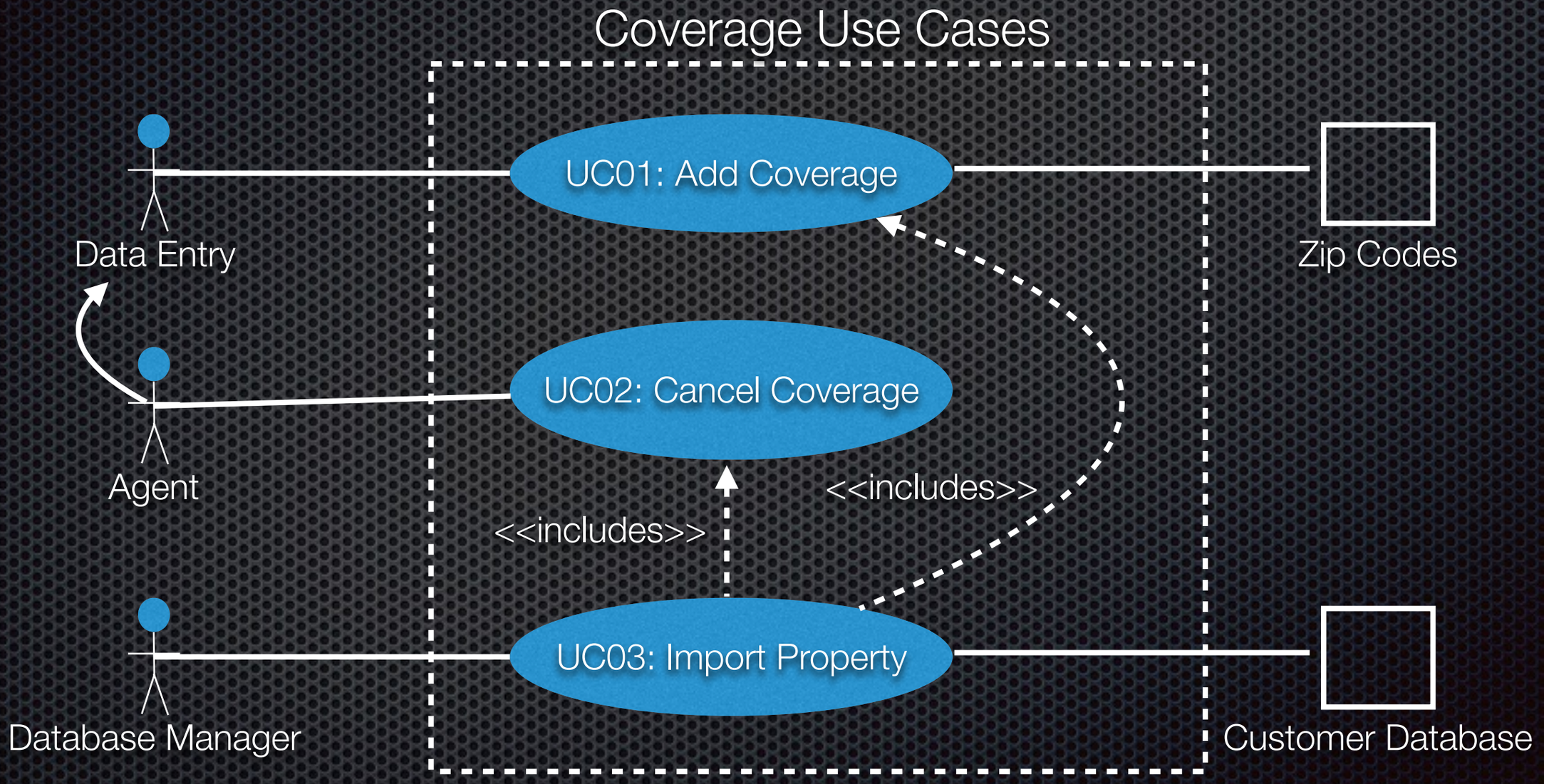


Use Case Functional Relationships

INCLUDES allows use cases to reuse (or call) another use case's functonality.
E.g., Import Property Coverage calls Add Coverage (reuses its functionality)

EXTENDS allows conditional behavior to be part of the base Use Case.
E.g., If a user adds coverage with value > 100,000 then a manager must approve the transaction

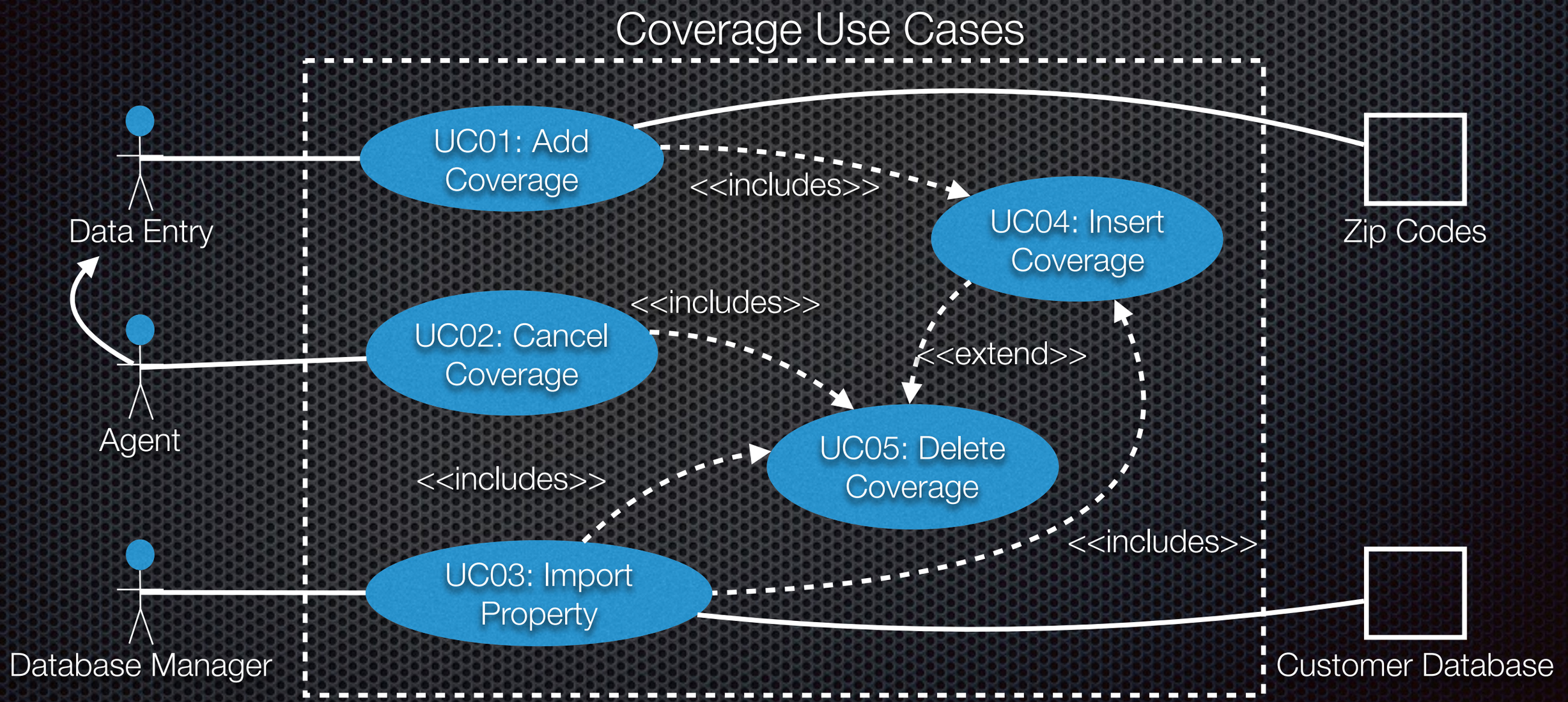
Insurance UC Diagram Example



Import Property Detail

	Use Case
Name	ImportPropertyCoverage
ID	UC03
Description	Imports 1 or more properties into system for coverage
Actors	DB Manager (users); Customer db
Preconditions	Actor logged in and has access to function
Main Flow	1. user selects Import Property Data function 2. Customer db sends list of properties for coverage 3. each property is add or cancel 3.1 if add, include(UC01 **AddPropertyCoverage**) 3.2 if cancel, include(UC02 **CancelPropertyCoverage**) 4. system sends import summary to user
	...

Revised UC Diagram



InsertCoverage Detail

	Use Case
Name	InsertCoverage
ID	UC04
Description	Inserts a new coverage record (replaces prev. coverage)
Actors	none
Preconditions	called by AddPropertyCoverage or ImportProp.Data
Main Flow	1. if prev. coverage exists 1.1 extend(UC05 **DeleteCoverage**) 2. the system creates a new coverage record and calculates premium
Postconditions	new property coverage records created
Alternative Flows	...

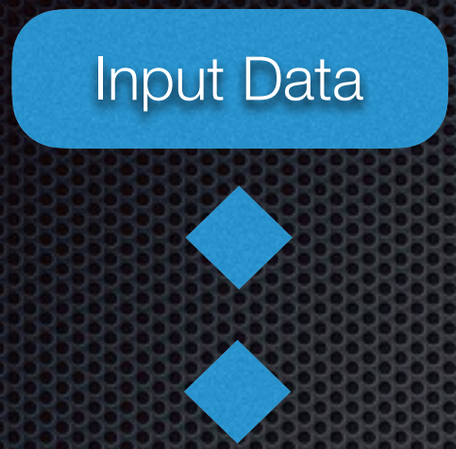
AddProperty Detail

	Use Case
Name	AddPropertyCoverage
ID	UC01
Description	Adds a property and coverage to the system
Actors	Agent, Data Entry, DB Manager (users); Zip code db
Preconditions	User logged in and has access to function
Main Flow	1. user selects Add Coverage function 2. the system shows a blank property coverage form 3. user enters address, zip, insured value, coverage dates 4. zip code db fills in city, state, and county **5. include(UC04 InsertCoverage)**
Postconditions	system creates new property coverage user returns to menu choices
Alternative Flows	NotLoggedIn during Preconditions NoAccess after Step 1 InvalidZipCode after Step 3 InvalidInsuredValue before Step 5 InvalidCoverageDate before Step 5

Activity Diagrams

- UML flow model for things like:
  - Use case main and alternate flows
  - Operations/functions
  - Objects/classes
  - Business processes
  - Anything that has a flow/sequence of transformation

Node Types



Action node: performs work; all inputs required for node to generate output

Decision node: branches based on guard condition

Merge node: merges multiple paths together; any input causes merge node to output

Initial node: flow starts here

Final node: flow ends here

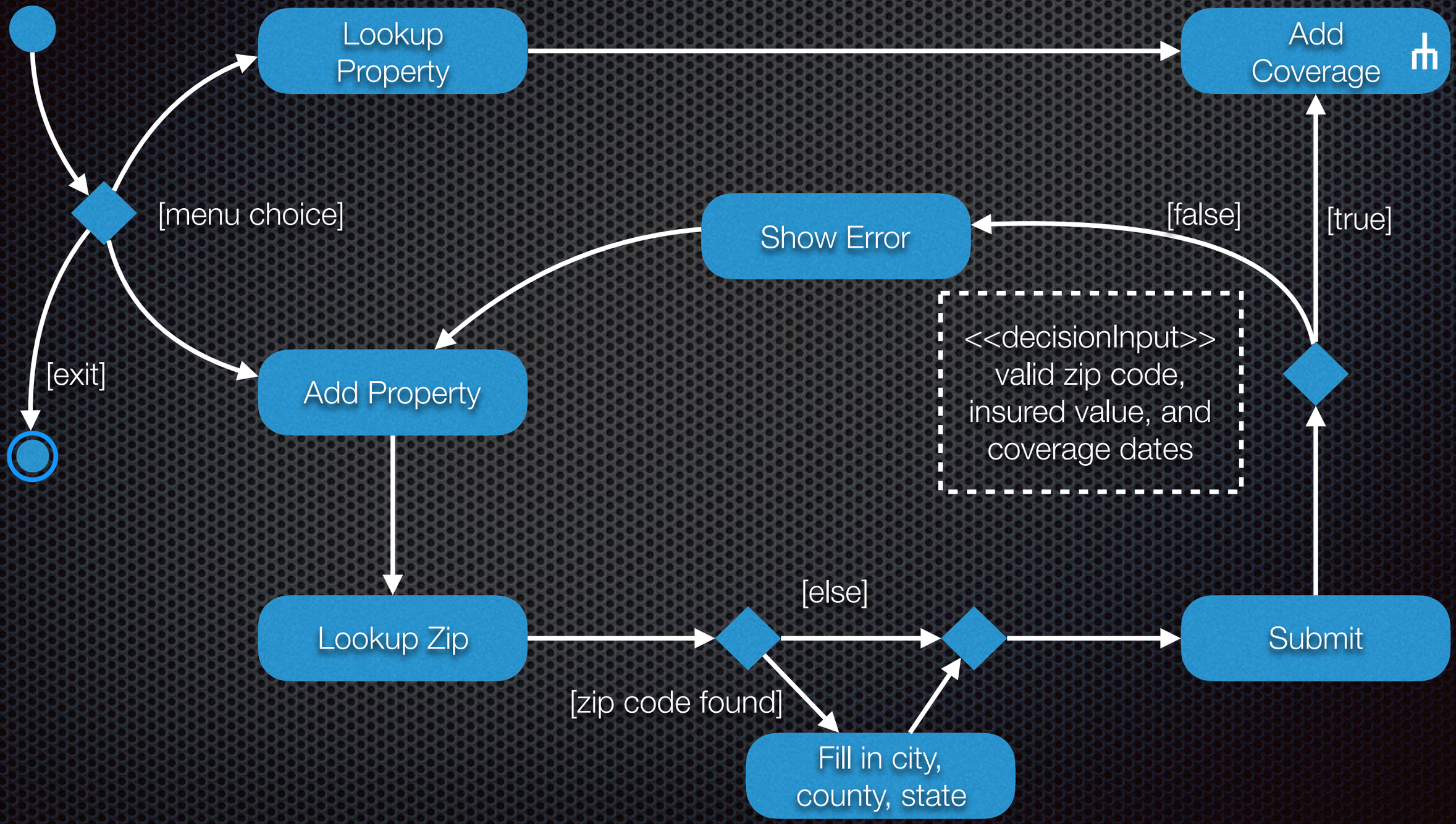
Go to a different activity diagram

[zip code found]

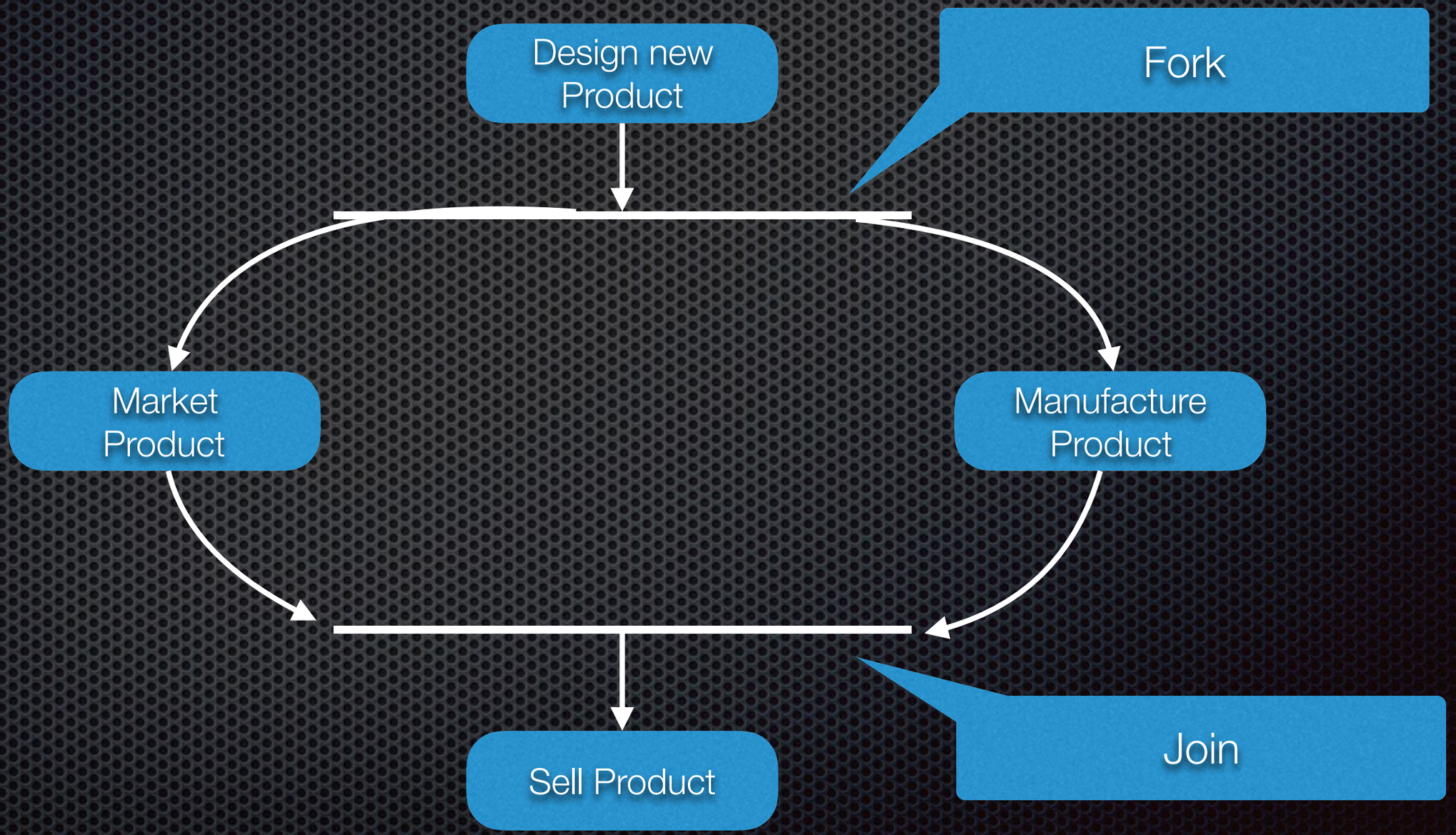
Guard condition: condition that affects decision branch

Decision condition: more complex guard condition

Activity Diagram Example 1



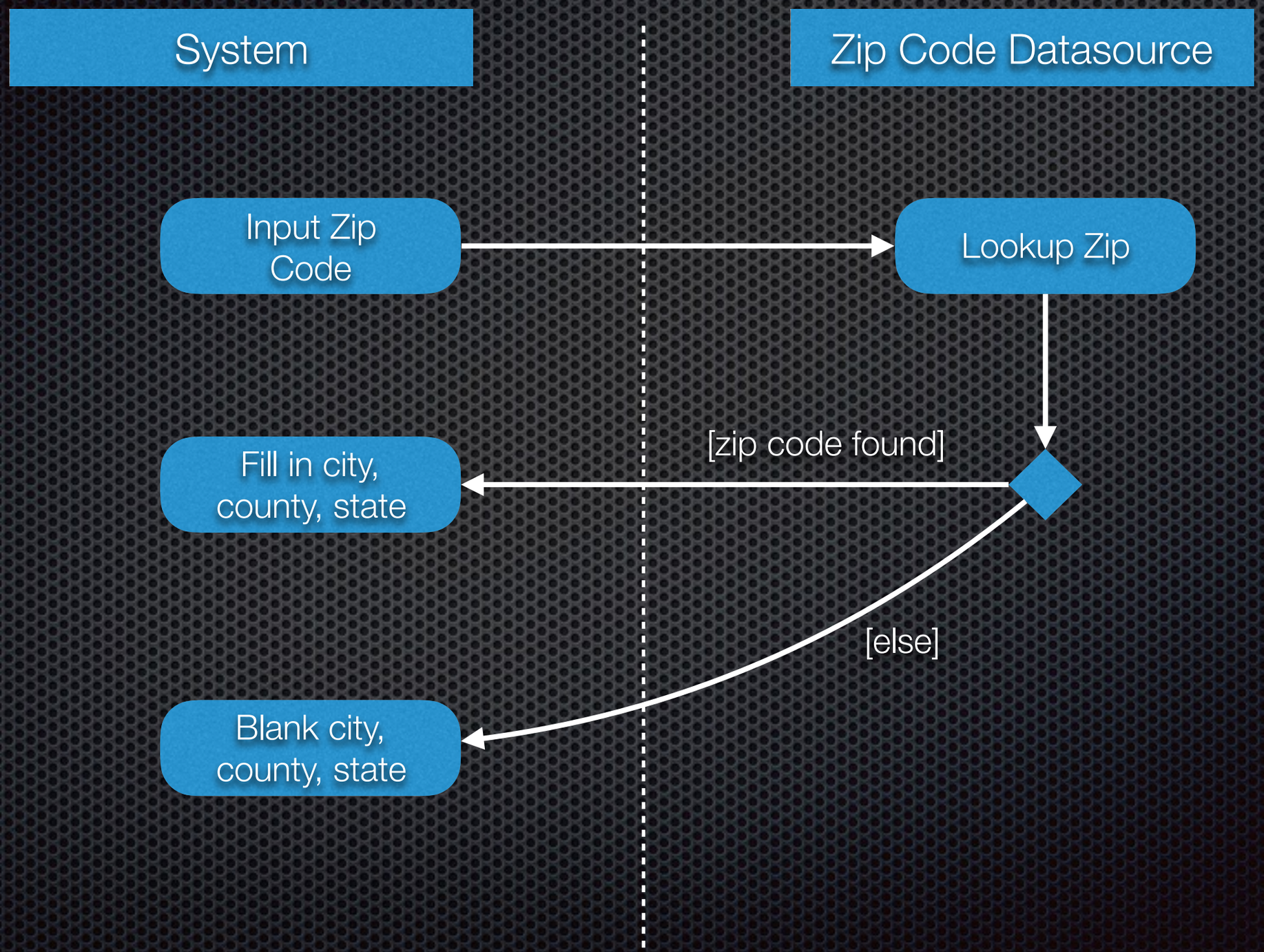
Concurrency Example



Activity Diagram Swimlanes (Partitions)

- Group related actions by:
  - Use case (diagram shows interaction between use cases)
  - Classes (i.e., data objects)
  - Entity/user role
  - Business units (e.g., Production and Warehouse)

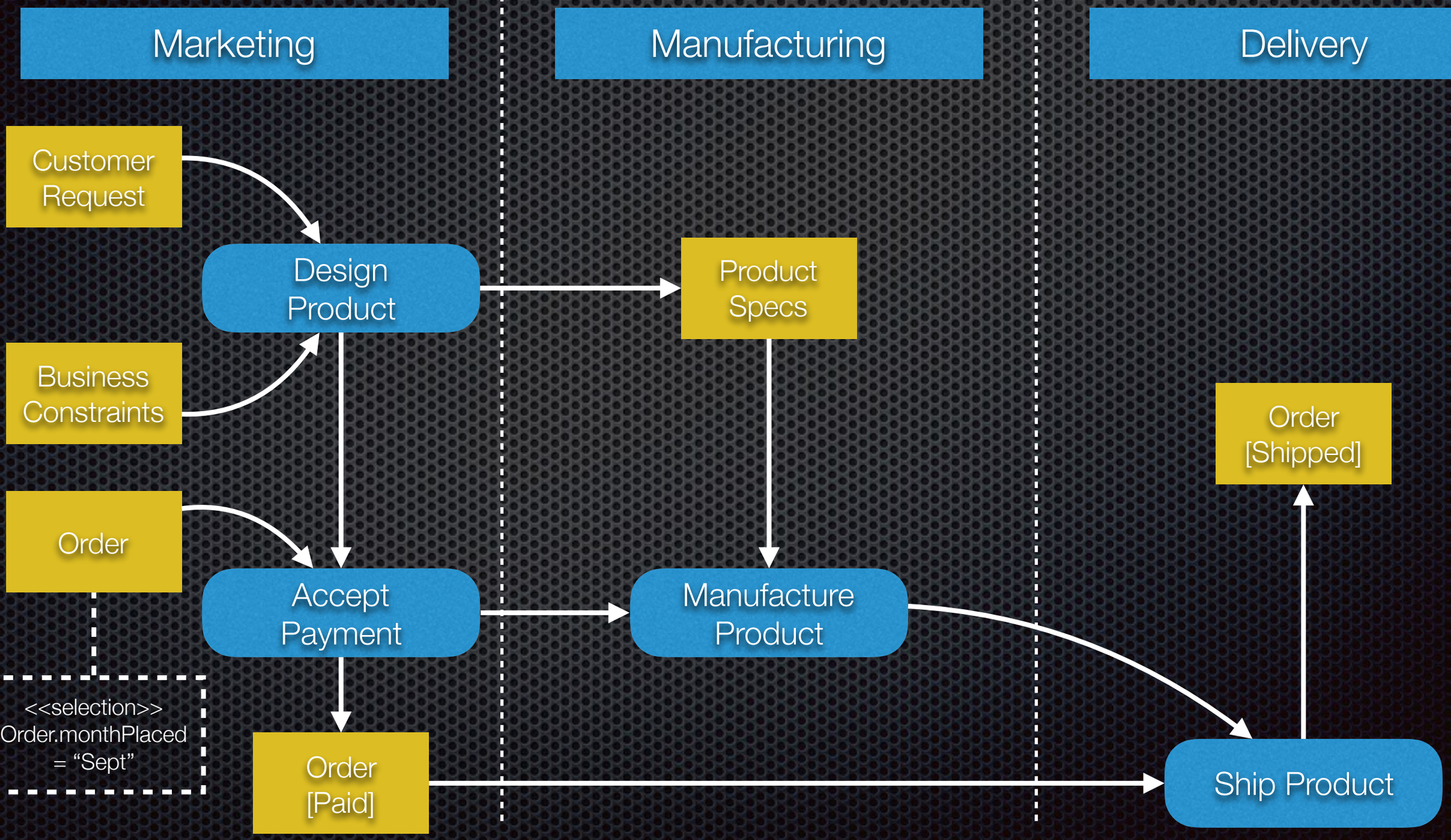
Swim lane Example



Objects in Activity Diagrams

- Represent instances of an object as it is transformed
- Object flow is object's movement through the model
- Objects are created and consumed by action nodes
- Can show state or buffer semantics with Object
- Can be inputs or output of activity diagram

Activity Diagram Object Example



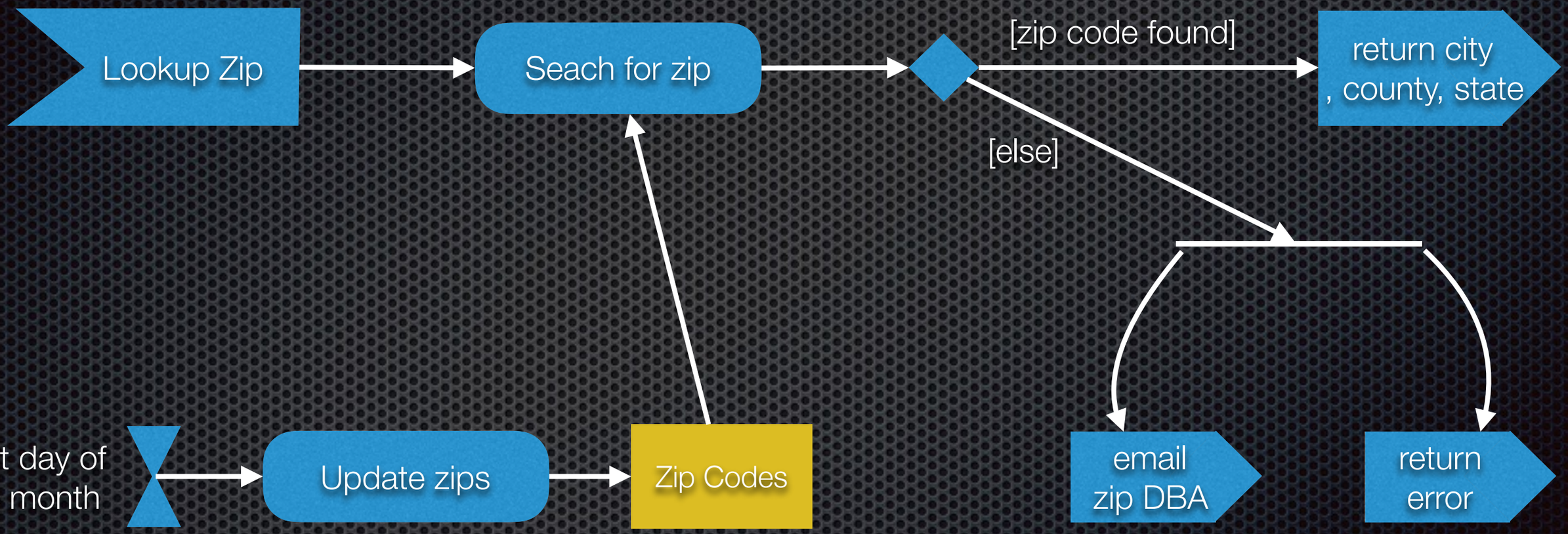
Advanced Activity Diagram Symbols

Receive Signal: responds to an external signal and initiates flow

Time Event: generates flow at specified moment , interval, or period

Send Signal: sends a signal to a receiving Activity Diagram

Advanced AD Example



Sequence Diagrams

- Show time-based sequence of events between entities and data objects
- Provide more detailed view of a use case scenario
- Lifeline: represents a participant in an interaction
  - e.g., actor, class/object, subsystem
- Similar to activity diagrams with swimlanes

Sequence Diagram Notation

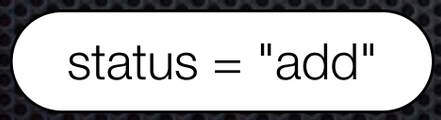
Lifelines (system/component names start with ":")

Synchronous Message/call
(can also <<create>> a new lifeline)

Asynchronous Message/call

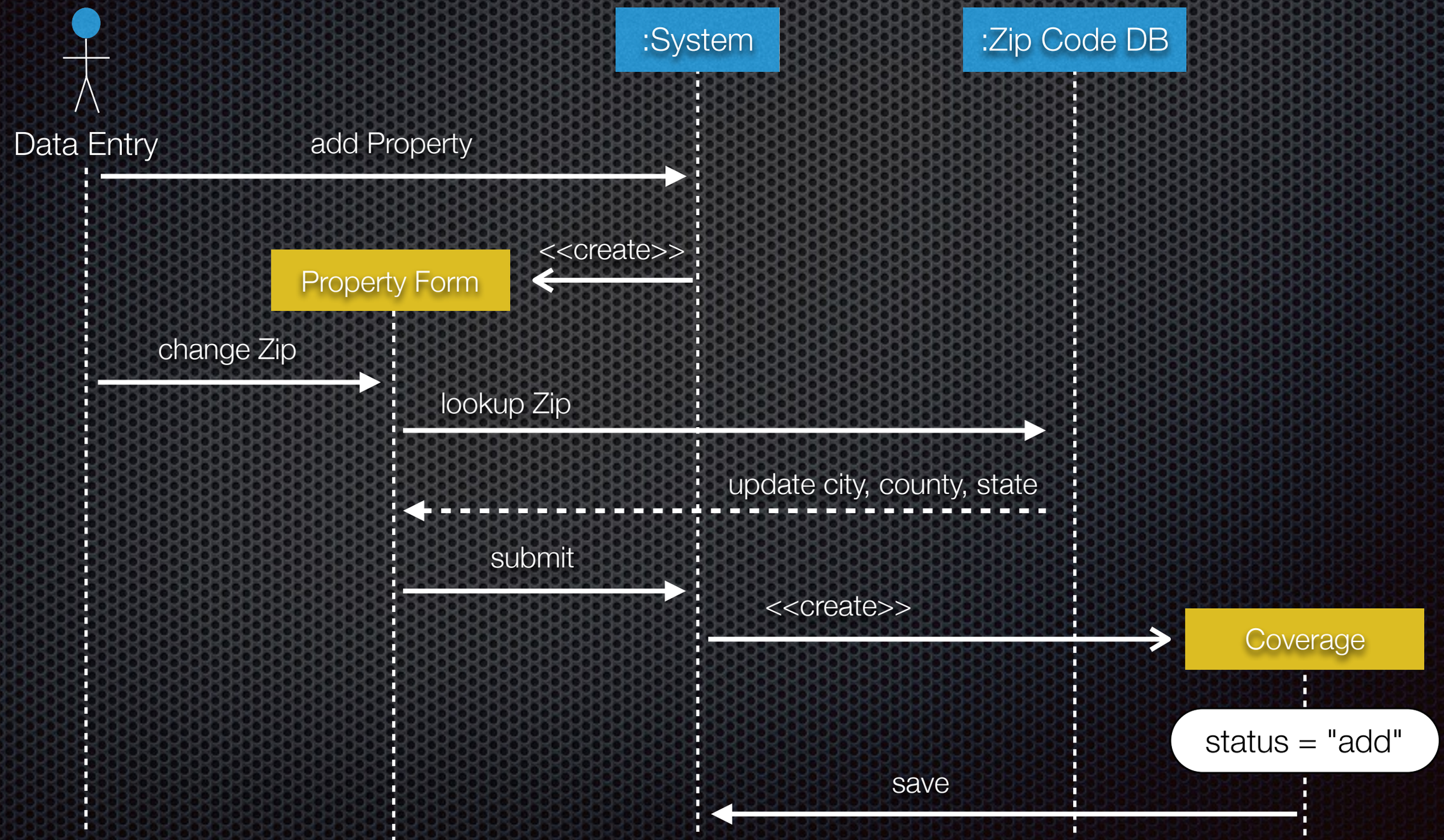
Response/return

Object



Object state

Sequence Diagram Example



A Larger Example

We want a software program so that our users can publish and read technical articles concerning basket weaving.

New users have to fill out a registration form and pick a user name and password. Users can suspend their account at any time, which suppresses further notifications of new articles, and can resume at any time.

Certain members can become authors by requesting and being approved by an admin. A similar request and approval process is also used for reviewers. Authors cannot be reviewers.

Articles are due on the 10th of each month and then are reviewed by the reviewers. Authors submit their articles for review. An author may recall their article at any point before the article is published. Once an article is submitted for review, reviewers will review the article and approve or reject it for publishing. Articles must be approved or rejected by the 15th of each month. Approved articles will then be published on the 16th by the system admin.

All members can comment on any article at any time. Members can also delete any comment he/she has made. All users are notified on the 16th when new articles are available.