Software Engineering

CS 780
School of Computing
WSU

OBJECT-ORIENTED ANALYSIS

Overview

- The analysis workflow
- Extracting the entity classes
- The elevator problem case study
  - Object-oriented analysis
  - Functional modeling
  - Entity class modeling
  - Dynamic modeling
- The test workflow: Object-oriented analysis
- Extracting the boundary and control classes

Overview (contd)

- The MSG Foundation case study
  - The initial functional model
  - The initial class diagram
  - The initial dynamic model
  - Extracting the entity classes
  - Extracting the boundary classes
  - Extracting the control classes
  - Use-case realization
  - Incrementing the class diagram
  - The test workflow

Overview (contd)

- The specification document in the Unified Process
- More on actors and use cases
- CASE tools for the object-oriented analysis workflow
- Challenges of the object-oriented analysis workflow

Object-Oriented Analysis

- OOA is a semiformal analysis technique for the object-oriented paradigm
  - There are over 60 equivalent techniques
  - Today, the Unified Process is the only viable alternative
- During this workflow
  - The classes are extracted
- Remark
  - The Unified Process assumes knowledge of class extraction

13.1 The Analysis Workflow

- The analysis workflow has two aims
  - Obtain a deeper understanding of the requirements
  - Describe them in a way that will result in a maintainable design and implementation

The Analysis Workflow (contd)

- There are three types of classes:
  - Entity classes
  - Boundary classes
  - Control classes

The Analysis Workflow (contd)

- Entity class
  - Models long-lived information
- Examples:
  - **Account Class**
  - **Investment Class**

The Analysis Workflow (contd)

- Boundary class
  - Models the interaction between the product and the environment
  - A boundary class is generally associated with input or output

- Examples:
  - **Investments Report Class**
  - **Mortgages Report Class**

The Analysis Workflow (contd)

- Control class
  - Models complex computations and algorithms
- Example:
  - **Estimate Funds for Week Class**

UML Notation for These Three Class Types

- Stereotypes (extensions of UML)

Figure 13.1

13.2 Extracting the Entity Classes

- Perform the following three steps incrementally and iteratively

  - Functional modeling
    - » Present scenarios of all the use cases (a *scenario* is an instance of a use case)
  - Class modeling
    - » Determine the entity classes and their attributes
    - » Determine the interrelationships and interactions between the entity classes
    - » Present this information in the form of a *class diagram*
  - Dynamic modeling
    - » Determine the operations performed by or to each entity class
    - » Present this information in the form of a *statechart*

A product is to be installed to control n elevators in a building with m floors. The problem concerns the logic required to move elevators between floors according to the following constraints:

1. Each elevator has a set of m buttons, one for each floor. These illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is canceled when the corresponding floor is visited by the elevator

2. Each floor, except the first and the top floor, has two buttons, one to request an up-elevator, one to request a down-elevator. These buttons illuminate when pressed. The illumination is canceled when an elevator visits the floor, then moves in the desired direction

3. If an elevator has no requests, it remains at its current floor with its doors closed

13.4 Functional Modeling: The Elevator Problem Case Study

- A use case describes the interaction between
  - The product, and
  - The actors (external users)

Use Cases

- For the elevator problem, there are only two possible use cases
  - `Press an Elevator Button`, and
  - `Press a Floor Button`

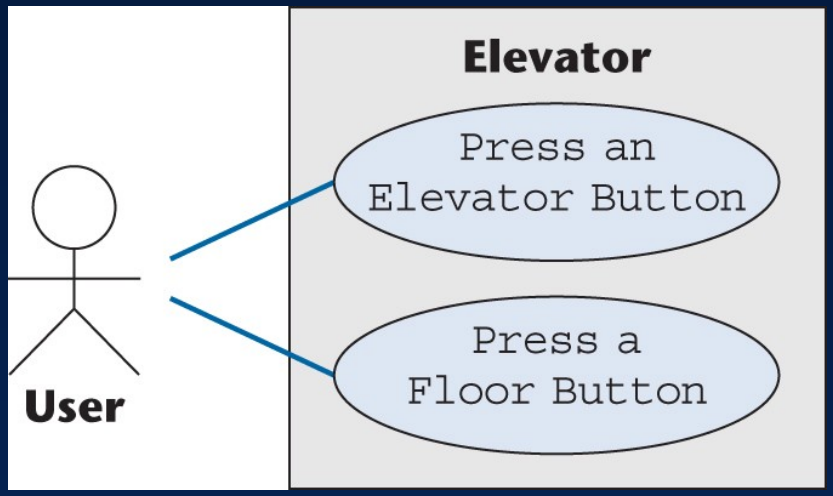


Figure 13.2

Scenarios

- A use case provides a generic description of the overall functionality
- A scenario is an instance of a use case
- Sufficient scenarios need to be studied to get a comprehensive insight into the target product being modeled

Normal Scenario: Elevator Problem

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The elevator doors open.
5. The timer starts.
   User A enters the elevator.
6. User A presses the elevator button for floor 7.
7. The elevator button for floor 7 is turned on.
8. The elevator doors close after a timeout.
9. The Up floor button is turned off.
10. The elevator travels to floor 7.
11. The elevator button for floor 7 is turned off.
12. The elevator doors open to allow User A to exit from the elevator.
13. The timer starts.
    User A exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 9 with User B.

Figure 13.3

Exception Scenario: Elevator Problem

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 1.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The elevator doors open.
5. The timer starts.
   User A enters the elevator.
6. User A presses the elevator button for floor 1.
7. The elevator button for floor 1 is turned on.
8. The elevator doors close after a timeout.
9. The Up floor button is turned off.
10. The elevator travels to floor 9.
11. The elevator button for floor 9 is turned off.
12. The elevator doors open to allow User B to exit from the elevator.
13. The timer starts.
    User B exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 1 with User A.

Figure 13.4

- Extract classes and their attributes
  - Represent them using a UML diagram
- One alternative: Deduce the classes from use cases and their scenarios
  - Possible danger: Often there are many scenarios, and hence
  - Too many candidate classes
- Other alternatives:
  - CRC cards (if you have domain knowledge)
  - Noun extraction

13.5.1 Noun Extraction

- A two-stage process

- Stage 1. Concise problem definition
  - Describe the software product in single paragraph
  - Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed

Noun Extraction (contd)

- Stage 2. Identify the nouns
  - Identify the nouns in the informal strategy
  - Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed

- Use the nouns as candidate classes

Noun Extraction (contd)

- Nouns
  - button, elevator, floor, movement, building, illumination, request, door
  - floor, building, door are outside the problem boundary — exclude
  - movement, illumination, request are abstract nouns — exclude (they may become attributes)
- Candidate classes:
  - **Elevator Class** and **Button Class**
- Subclasses:
  - **Elevator Button Class** and **Floor Button Class**

First Iteration of Class Diagram

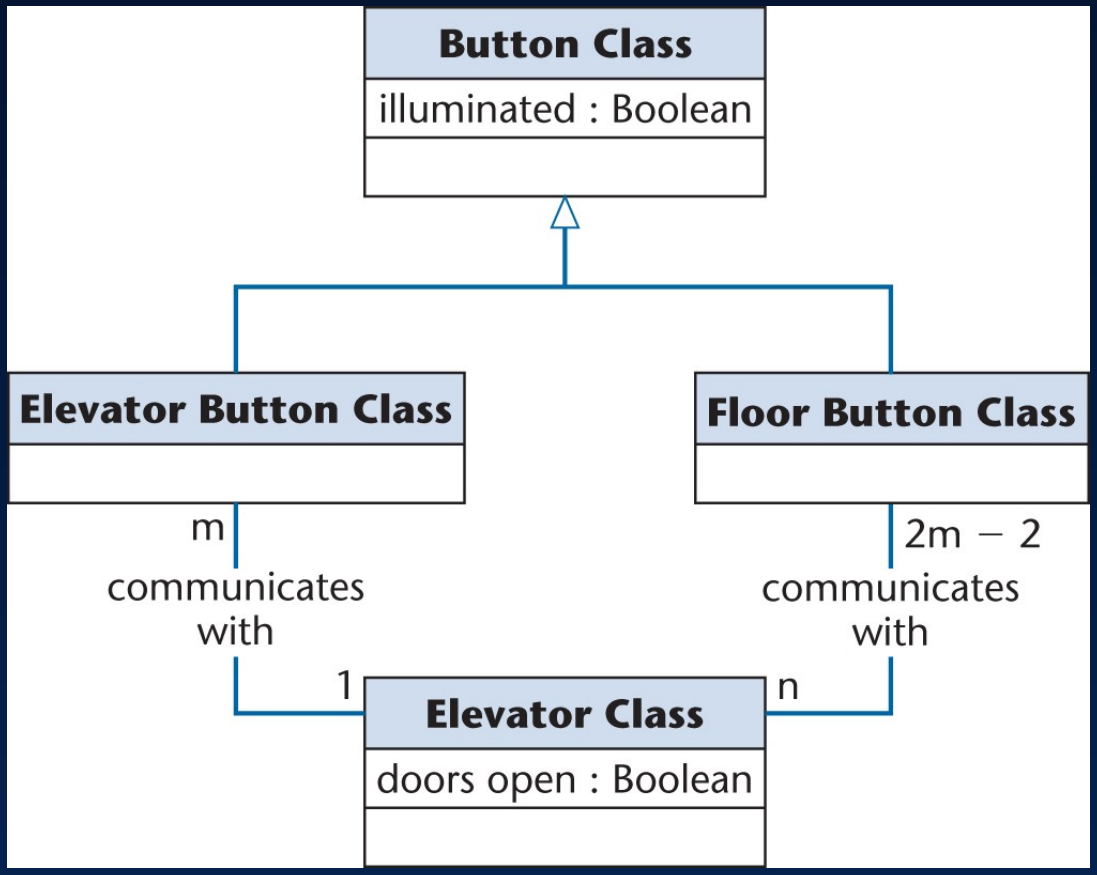


Figure 13.5

- Problem
  - Buttons do not communicate directly with elevators
  - We need an additional class: **Elevator Controller Class**

Second Iteration of Class Diagram

- All relationships are now 1-to-n
  - This makes design and implementation easier

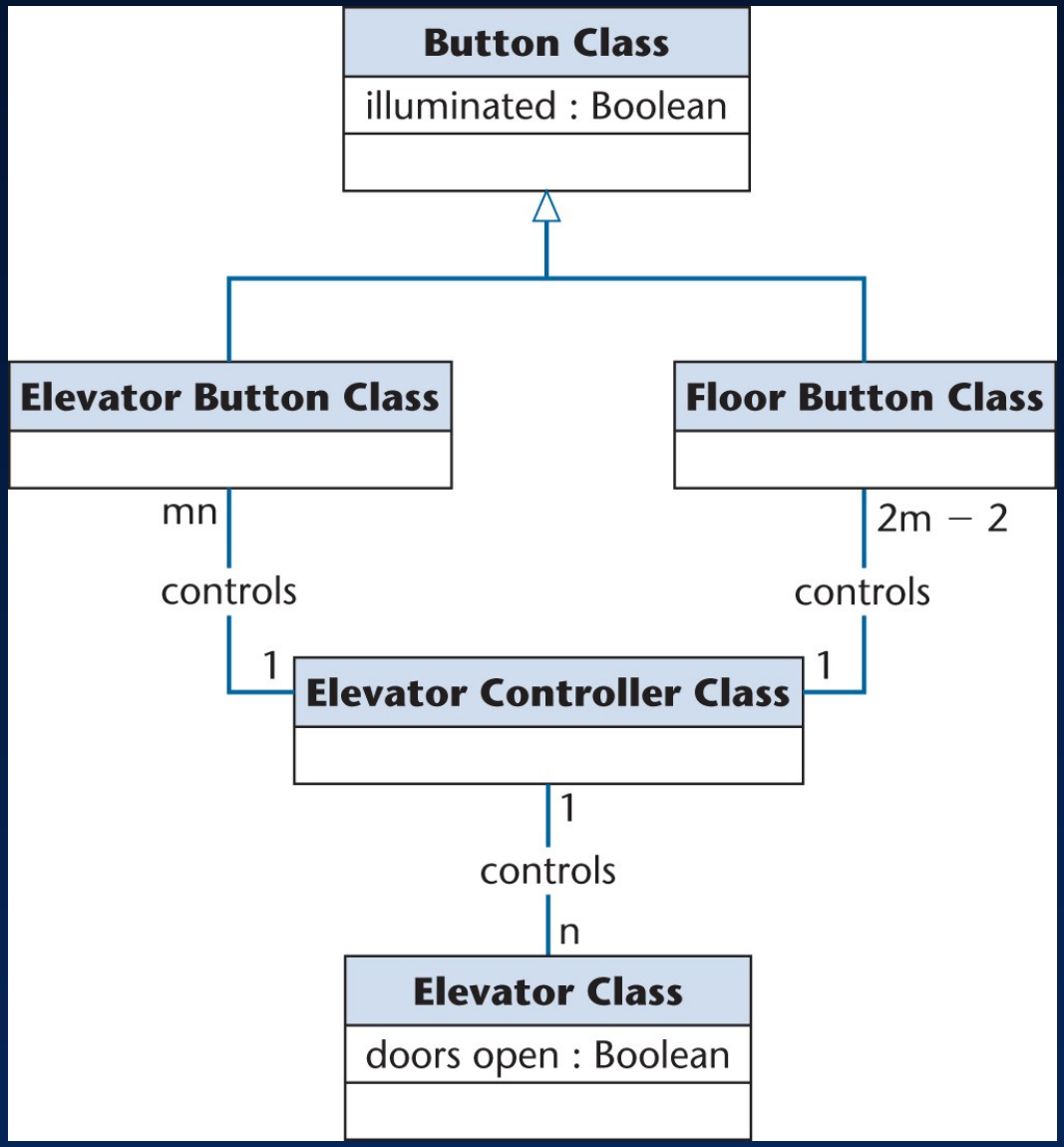


Figure 13.6

Exercise

Please extract entity classes for the following system and draw a class diagram:

Consider an automated teller machine (ATM). The user puts a card into a slot and enters a four-digit personal identification number (PIN). If the PIN is incorrect, the card is ejected. Otherwise, the user may perform the following operations on up to four different bank accounts:

(i) Deposit any amount. A receipt is printed showing the date, amount deposited, and account number.

(ii) Withdraw up to $200 in units of $20 (the account may not be overdrawn). In addition to the money, the user is given a receipt showing the date, amount withdrawn, account number, and account balance after the withdrawal.

(iii) Determine the account balance. This is displayed on the screen.

(iv) Transfer funds between two accounts. Again, the account from which the funds are transferred must not be overdrawn. The user is given a receipt showing the date, amount transferred, and the two account numbers.

(v) Quit. The card is ejected.

13.5.2 CRC Cards

- Used since 1989 for OOA
- For each class, fill in a card showing
  - Name of Class
  - Functionality (Responsibility)
  - List of classes it invokes (Collaboration)
- Now CRC cards are automated

CLASS **Elevator Controller Class**
RESPONSIBILITY 1. Send message to **Elevator Button Class** to turn on button 2. Send message to **Elevator Button Class** to turn off button 3. Send message to **Floor Button Class** to turn on button 4. Send message to **Floor Button Class** to turn off button 5. Send message to **Elevator Class** to move up one floor 6. Send message to **Elevator Class** to move down one floor 7. Send message to **Elevator Doors Class** to open 8. Start timer 9. Send message to **Elevator Doors Class** to close after timeout 10. Check requests 11. Update requests
COLLABORATION 1. **Elevator Button Class** (subclass) 2. **Floor Button Class** (subclass) 3. **Elevator Doors Class** 4. **Elevator Class**

CRC Cards (contd)

- Strength
  - When acted out by team members, CRC cards are a powerful tool for highlighting missing or incorrect items
- Weakness
  - If CRC cards are used to identify entity classes, domain expertise is needed

13.6 Dynamic Modeling: The Elevator Problem Case Study

- Produce a UML statechart
- State, event, and predicate are distributed over the statechart

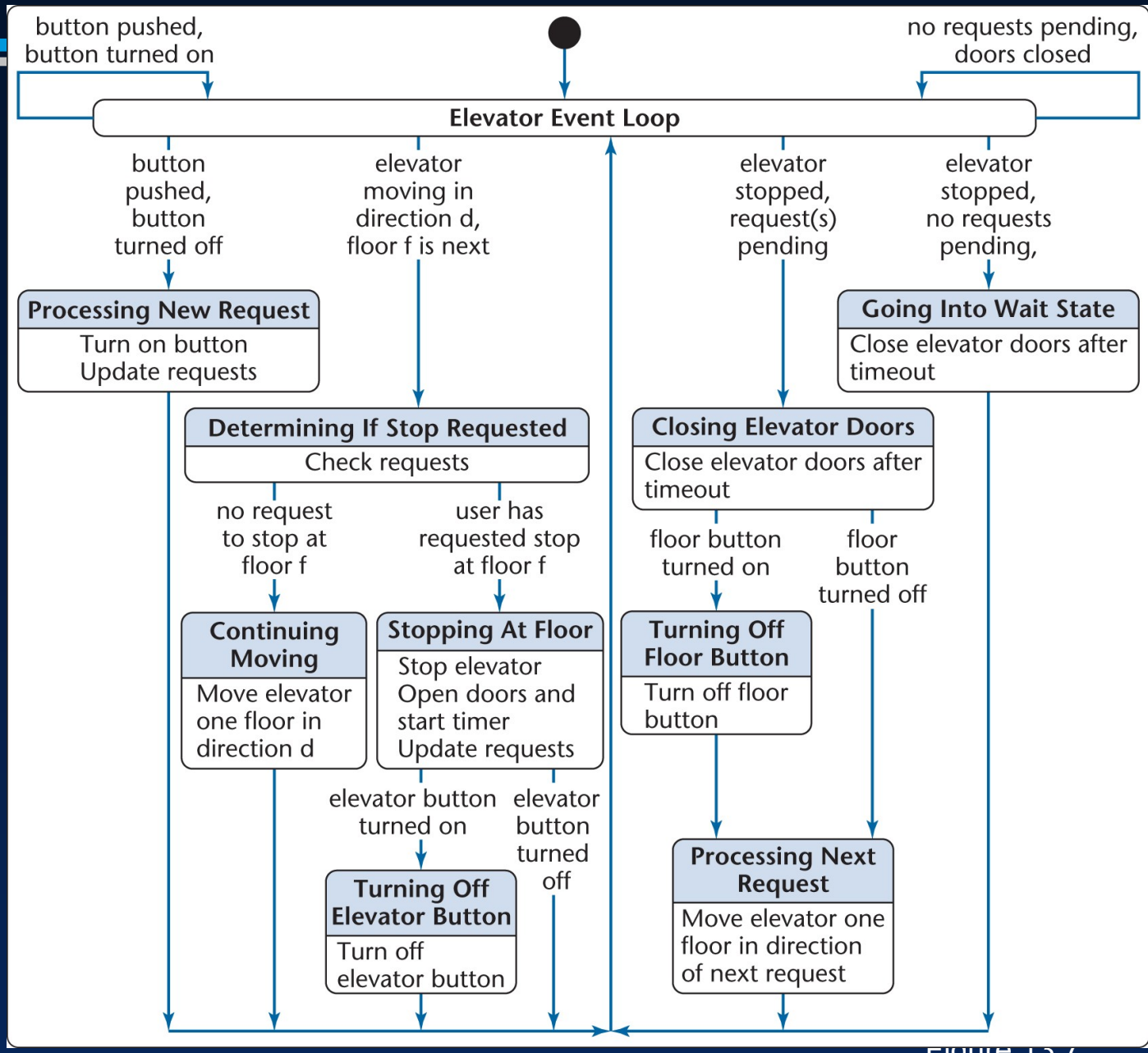


Figure 13.7

Dynamic Modeling: Elevator Problem (contd)

- This UML statechart is a Finite State Machine.
- This is shown by considering specific scenarios
- In fact, a statechart is constructed by modeling the events of the scenarios

13.7 The Test Workflow: Object-Oriented Analysis

- CRC cards are an excellent testing technique

CLASS **Elevator Controller Class**
RESPONSIBILITY 1. Turn on elevator button 2. Turn off elevator button 3. Turn on floor button 4. Turn off floor button 5. Move elevator up one floor 6. Move elevator down one floor 7. Open elevator doors and start timer 8. Close elevator doors after timeout 9. Check requests 10. Update requests
COLLABORATION 1. **Elevator Button Class** 2. **Floor Button Class** 3. **Elevator Class**

Figure 13.8

CRC Cards

- Consider responsibility
  - 1. Turn on elevator button
- This is totally inappropriate for the object-oriented paradigm
  - Responsibility-driven design has been ignored
  - Information hiding has been ignored
- Responsibility
  1. Turn on elevator button

  should be
  1. Send message to **Elevator Button Cla**

CRC Cards (contd)

- Also, a class has been overlooked
- The elevator doors have a *state* that changes during execution (class characteristic)
  - Add class **Elevator Doors Class**
  - Safety considerations
- Modify the CRC card

Second Iteration of the CRC Card

CLASS **Elevator Controller Class**
RESPONSIBILITY 1. Send message to **Elevator Button Class** to turn on button 2. Send message to **Elevator Button Class** to turn off button 3. Send message to **Floor Button Class** to turn on button 4. Send message to **Floor Button Class** to turn off button 5. Send message to **Elevator Class** to move up one floor 6. Send message to **Elevator Class** to move down one floor 7. Send message to **Elevator Doors Class** to open 8. Start timer 9. Send message to **Elevator Doors Class** to close after timeout 10. Check requests 11. Update requests
COLLABORATION 1. **Elevator Button Class** (subclass) 2. **Floor Button Class** (subclass) 3. **Elevator Doors Class** 4. **Elevator Class**

Figure 13.9

CRC Cards (contd)

- Having modified the class diagram, reconsider the
  - Use-case diagram (no change)
  - Class diagram (see the next slide)
  - Statecharts
  - Scenarios (see the slide after the next slide)

Third Iteration of Class Diagram

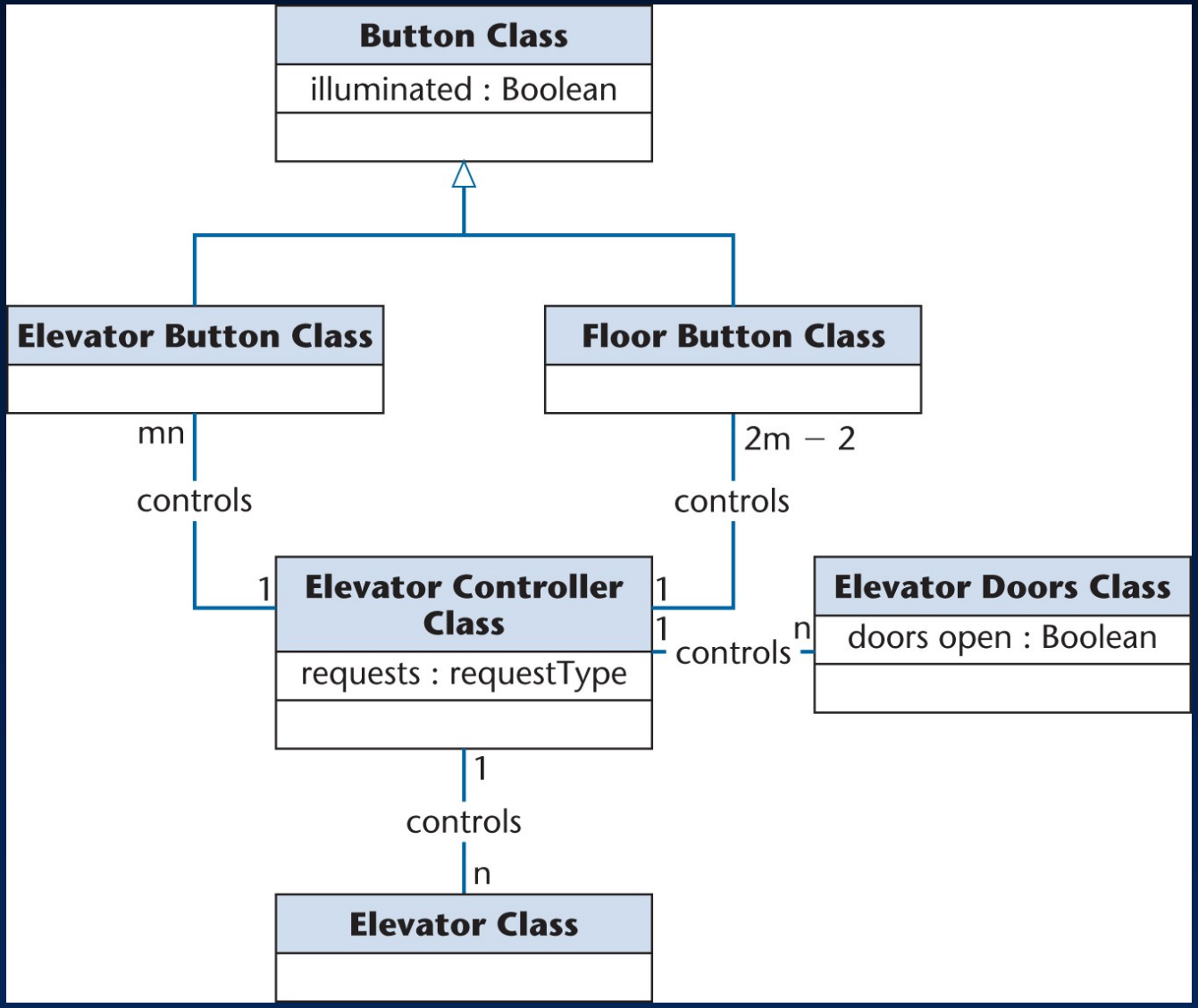


Figure 13.10

Second Iteration of the Normal Scenario:

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The floor button informs the elevator controller that the floor button has been pushed.
3. The elevator controller sends a message to the Up floor button to turn itself on.
4. The elevator controller sends a series of messages to the elevator to move itself up to floor 3. The elevator contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
5. The elevator controller sends a message to the elevator doors to open themselves.
6. The elevator controller starts the timer.

   User A enters the elevator.
7. User A presses elevator button for floor 7.
8. The elevator button informs the elevator controller that the elevator button has been pushed.
9. The elevator controller sends a message to the elevator button for floor 7 to turn itself on.
10. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
11. The elevator controller sends a message to the Up floor button to turn itself off.
12. The elevator controller sends a series of messages to the elevator to move itself up to floor 7.
13. The elevator controller sends a message to the elevator button for floor 7 to turn itself off.
14. The elevator controller sends a message to the elevator doors to open themselves to allow User A to exit from the elevator.
15. The elevator controller starts the timer.

    User A exits from the elevator.
16. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
17. The elevator controller sends a series of messages to the elevator to move itself up to floor 9 with User B.

Figure 13.11

Serious Problem with Third Iteration of Class Diagram

- Return to Figure 13.10 (third iteration of class diagram)
- **Elevator Controller Class** is running everything
- This is an example of a so-called “God class”
  - A class that is exposed to too much information, and
  - has too much control
- This is a well-known antipattern

Distributed Decentralized Architecture

- Distribute the control
  - Instead of having one central elevator controller

Fourth Iteration of Class Diagram

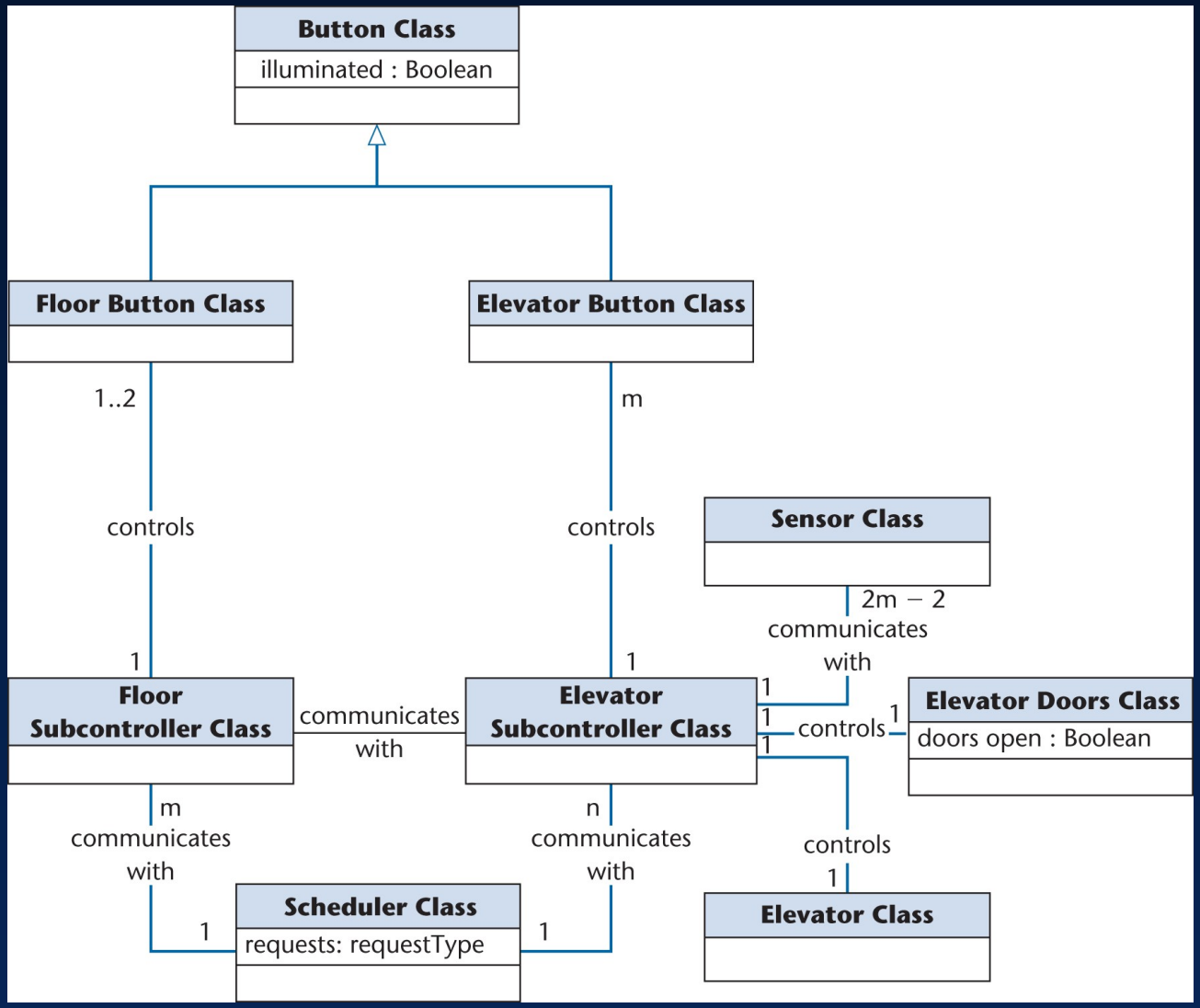
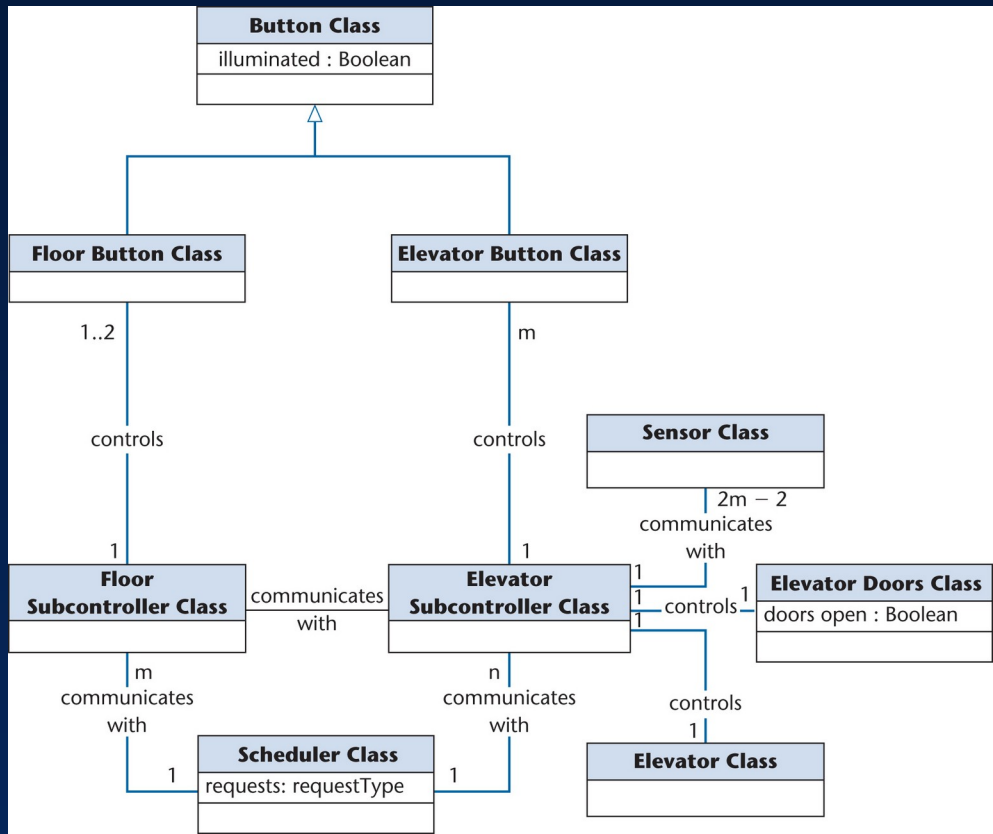


Figure 13.12

Distributed Decentralized Architecture

- Each of the n elevators now has its own elevator subcontroller
- Each of the m floors now has its own floor subcontroller
- The (m + n) subcontrollers all communicate with a scheduler, which processes requests



Distributed Decentralized Architecture (contd)

- A **Floor Button Class** object is controlled by its corresponding **Floor Subcontroller Class** object

- An **Elevator Button Class** object is controlled by its corresponding **Elevator Subcontroller Class** object

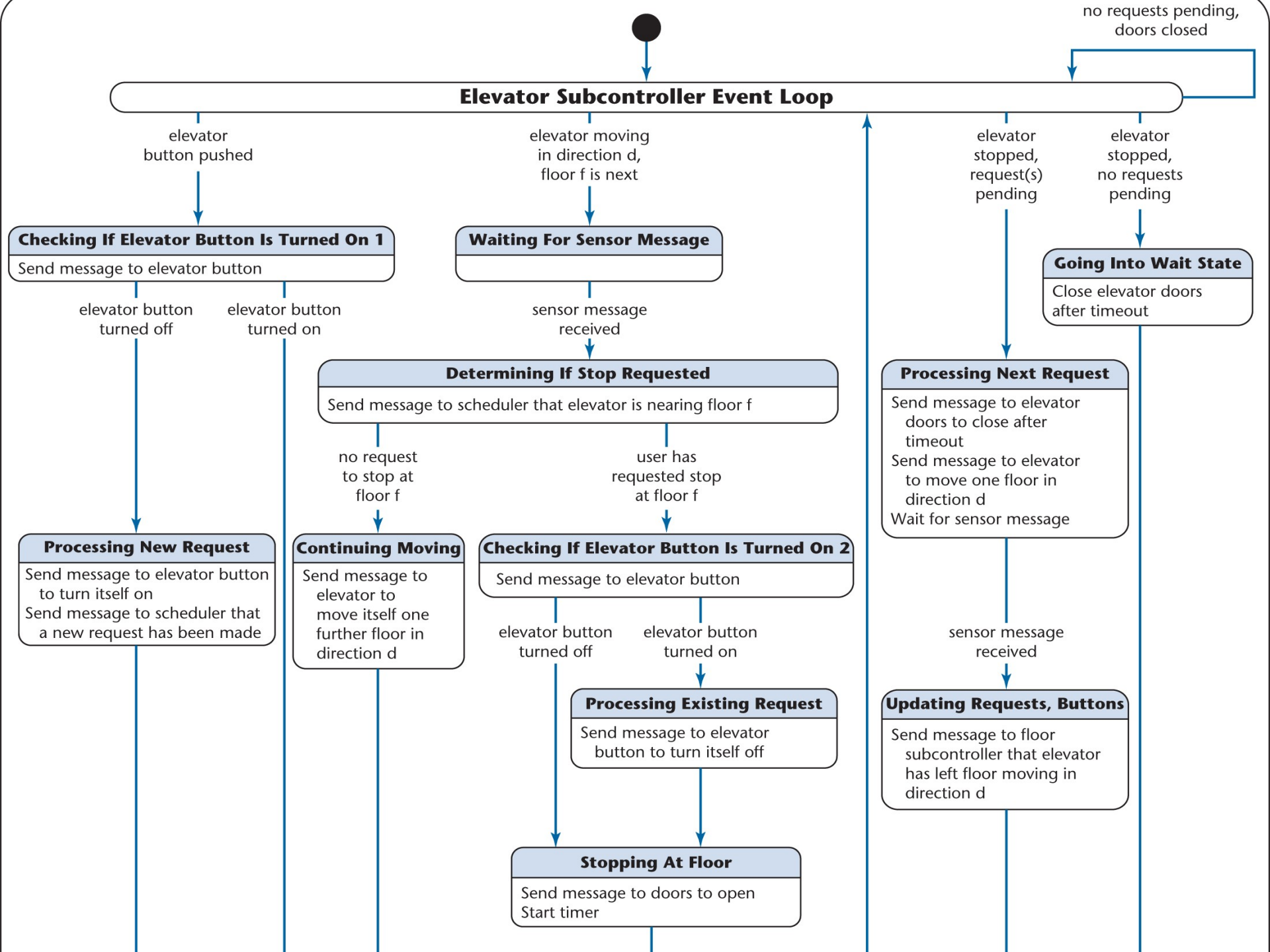
Distributed Decentralized Architecture (contd)

- There is a sensor just above and just below each floor in each elevator shaft

- When an **Elevator Class** object nears or leaves a floor
  - The corresponding **Sensor Class** object informs the corresponding **Elevator Subcontroller Class** object

Distributed Decentralized Architecture (contd)

- The UML diagrams now need to be updated to reflect the fourth iteration of the class diagram


Elevator Subcontroller Event Loop
elevator button pushed
elevator moving in direction d, floor f is next
elevator stopped, request(s) pending
elevator stopped, no requests pending
no requests pending, doors closed
Checking If Elevator Button Is Turned On 1
Send message to elevator button
Waiting For Sensor Message
Going Into Wait State
Close elevator doors after timeout
elevator button turned off
elevator button turned on
sensor message received
Determining If Stop Requested
Send message to scheduler that elevator is nearing floor f
Processing Next Request
Send message to elevator doors to close after timeout
Send message to elevator to move one floor in direction d
Wait for sensor message
no request to stop at floor f
user has requested stop at floor f
Processing New Request
Send message to elevator button to turn itself on
Send message to scheduler that a new request has been made
Continuing Moving
Send message to elevator to move itself one further floor in direction d
Checking If Elevator Button Is Turned On 2
Send message to elevator button
sensor message received
elevator button turned off
elevator button turned on
Processing Existing Request
Send message to elevator button to turn itself off
Updating Requests, Buttons
Send message to floor subcontroller that elevator has left floor moving in direction d
Stopping At Floor
Send message to doors to open
Start timer

First Iteration of CRC Card for **Elevator Subcontroller Class**

CLASS **Elevator Subcontroller Class**
RESPONSIBILITY 1. Send message to **Elevator Button Class** to check if it is turned on 2. Send message to **Elevator Button Class** to turn itself on 3. Send message to **Elevator Button Class** to turn itself off 4. Send message to **Elevator Doors Class** to open themselves 5. Start timer 6. Send message to **Elevator Doors Class** to close themselves after timeout 7. Send message to **Elevator Class** to move itself up one floor 8. Send message to **Elevator Class** to move itself down one floor 9. Send message to **Scheduler Class** that a request has been made 10. Send message to **Scheduler Class** that a request has been satisfied 11. Send message to **Scheduler Class** to check if the elevator is to stop at the next floor 12. Send message to **Floor Subcontroller Class** that elevator has left floor
COLLABORATION 1. **Elevator Button Class** (subclass) 2. **Sensor Class** 3. **Elevator Doors Class** 4. **Elevator Class** 5. **Scheduler Class** 6. **Floor Subcontroller Class**

Figure 13.14

13.8 Extracting the Boundary and Control Classes

- Each
  - Input screen,
  - Output screen, and
  - Report

  is modeled by its own boundary class

- Each nontrivial computation is modeled by a control class

OOA: Elevator Problem (contd)

- The object-oriented analysis is now fine
- We should rather say:
  - The object-oriented analysis is fine *for now*
- We may need to return to the object-oriented analysis workflow during the object-oriented design workflow

Exercises

Modify the scenario of Figure 13.11 to reflect the fourth iteration of the class diagram of the elevator problem case study (Figure 13.12).

Develop a statechart for the **Button Class** shown in Figure 13.12.

Develop a statechart for the **Elevator Class** shown in Figure 13.12.

Develop a statechart for the **Elevator Doors Class** shown in Figure 13.12.

Construct a CRC card for the **Floor Subcontroller Class** shown in Figure 13.12.

Exercise

Please draw state chart and CRC cards for the following system:

Consider an automated teller machine (ATM). The user puts a card into a slot and enters a four-digit personal identification number (PIN). If the PIN is incorrect, the card is ejected. Otherwise, the user may perform the following operations on up to four different bank accounts:

(i) Deposit any amount. A receipt is printed showing the date, amount deposited, and account number.

(ii) Withdraw up to $200 in units of $20 (the account may not be overdrawn). In addition to the money, the user is given a receipt showing the date, amount withdrawn, account number, and account balance after the withdrawal.

(iii) Determine the account balance. This is displayed on the screen.

(iv) Transfer funds between two accounts. Again, the account from which the funds are transferred must not be overdrawn. The user is given a receipt showing the date, amount transferred, and the two account numbers.

(v) Quit. The card is ejected.

13.9 The Initial Functional Model: MSG Foundation

- Recall the seventh iteration of the use-case diagram

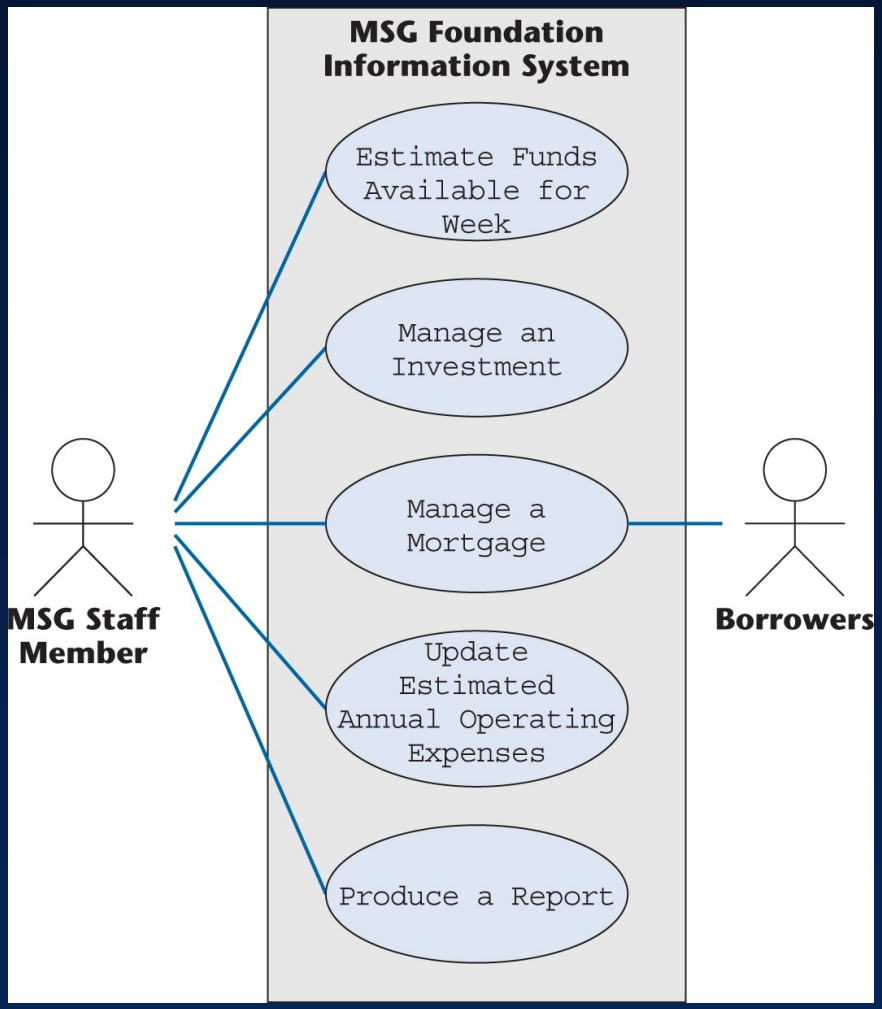


Figure 13.15

Use Case Manage a Mortgage

- One possible extended scenario

An MSG Foundation staff member wants to update the annual real-estate tax on a home for which the Foundation has provided a mortgage.

1. The staff member enters the new value of the annual real-estate tax.
2. The information system updates the date on which the annual real-estate tax was last changed.

Possible Alternative

A. The staff member enters the mortgage number incorrectly.

Figure 13.16

Use Case `Manage a Mortgage` (contd)

- A second extended scenario

There is a change in the weekly income of a couple who have borrowed money from the MSG Foundation. They wish to have their weekly income updated in the Foundation records by an MSG staff member so that their mortgage payments will be correctly computed.

1. The staff member enters the new value of the weekly income.
2. The information system updates the date on which the weekly income was last changed.

Possible Alternatives

A. The staff member enters the mortgage number incorrectly.
B. The borrowers do not bring documentation regarding their new income.

Figure 13.17

Use Case Estimate Funds Available for Week

- One possible scenario

An MSG Foundation staff member wishes to determine the funds available for mortgages this week.

1. For each investment, the information system extracts the estimated annual return on that investment. It sums the separate returns and divides the result by 52 to yield the estimated investment income for the week.
2. The information system then extracts the estimated annual MSG Foundation operating expenses and divides the result by 52.
3. For each mortgage:
   - 3.1 The information system computes the amount to be paid this week by adding the principal and interest payment to $\frac{1}{52}$nd of the sum of the annual real-estate tax and the annual homeowner's insurance premium.
   - 3.2 It then computes 28 percent of the couple's current gross weekly income.
   - 3.3 If the result of Step 3.1 is greater than the result of Step 3.2, then it determines the mortgage payment for the week as the result of Step 3.2, and the amount of the grant for this week as the difference between the result of Step 3.1 and the result of Step 3.2.
   - 3.4 Otherwise, it takes the mortgage payment for this week as the result of Step 3.1, and there is no grant for the week.
4. The information system sums the mortgage payments of Steps 3.3 and 3.4 to yield the estimated total mortgage payments for the week.
5. It sums the grant payments of Step 3.3 to yield the estimated total grant payments for the week.
6. The information system adds the results of Steps 1 and 4 and subtracts the results of Steps 2 and 5. This is the total amount available for mortgages for the current week.
7. Finally, the software product prints the total amount available for new mortgages during the current week.

Figure 13.18

Use Case `Produce a Report`

- One possible scenario

An MSG staff member wishes to print a list of all mortgages.

1. The staff member requests a report listing all mortgages.

Figure 13.19

Use Case `Produce a Report` (contd)

- Another possible scenario

An MSG staff member wishes to print a list of all investments.

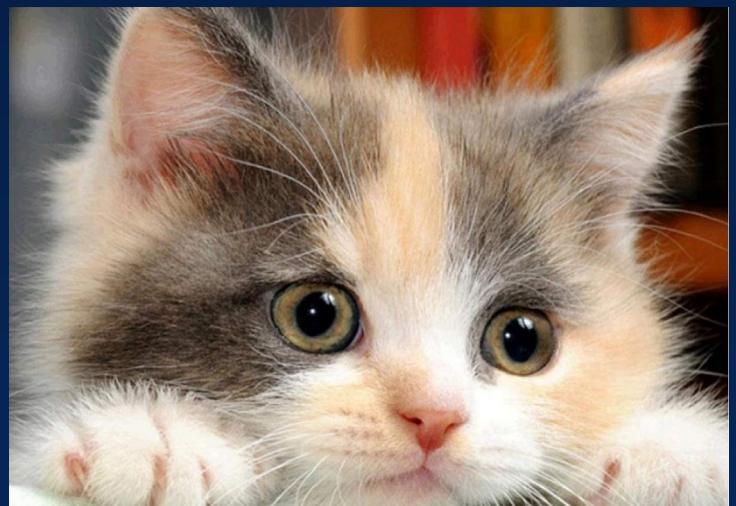
1. The staff member requests a report listing all investments.

Figure 13.20

13.10 The Initial Class Diagram: MSG Foundation

- The aim of entity modeling step is to extract the entity classes, determine their interrelationships, and find their attributes

- Usually, the best way to begin this step is to use the two-stage noun extraction method



Noun Extraction: MSG Foundation

- Stage 1: Describe the information system in a single paragraph
  - Weekly reports are to be printed showing how much money is available for mortgages. In addition, lists of investments and mortgages must be printed on demand.

Noun Extraction: MSG Foundation (contd)

- Stage 2: Identify the nouns in this paragraph
  - Weekly reports are to be printed showing how much money is available for mortgages. In addition, lists of investments and mortgages must be printed on demand.

- The nouns are report, money, mortgage, list, and investment

Noun Extraction: MSG Foundation (contd)

- Nouns report and list are not long lived, so they are unlikely to be entity classes (report will surely turn out to be a boundary class)
- money is an abstract noun
- This leaves two candidate entity classes
  - **Mortgage Class** and **Investment Class**

First Iteration of the Initial Class Diagram

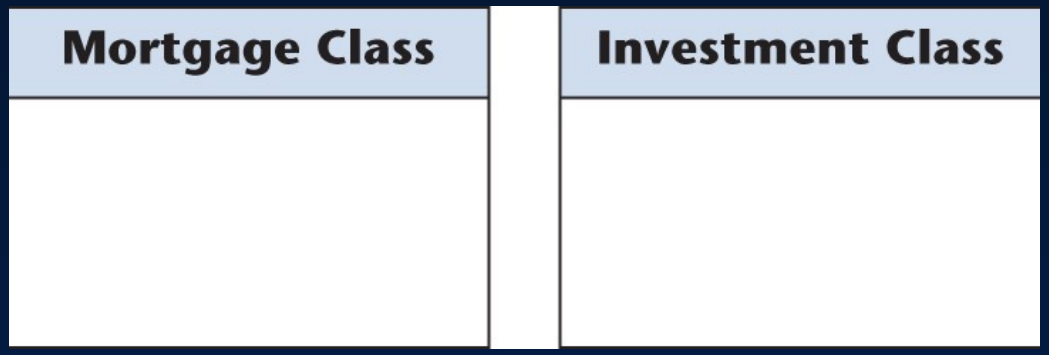


Figure 13.21

Second Iteration of the Initial Class Diagram

- Operations performed on the two entity classes are likely to be very similar
  - Insertions, deletions, and modifications
  - All members of both entity classes have to be printed on demand

- **Mortgage Class** and **Investment Class** should be subclasses of a superclass called **Asset Class**

Second Iteration of Initial Class Diagram (contd)

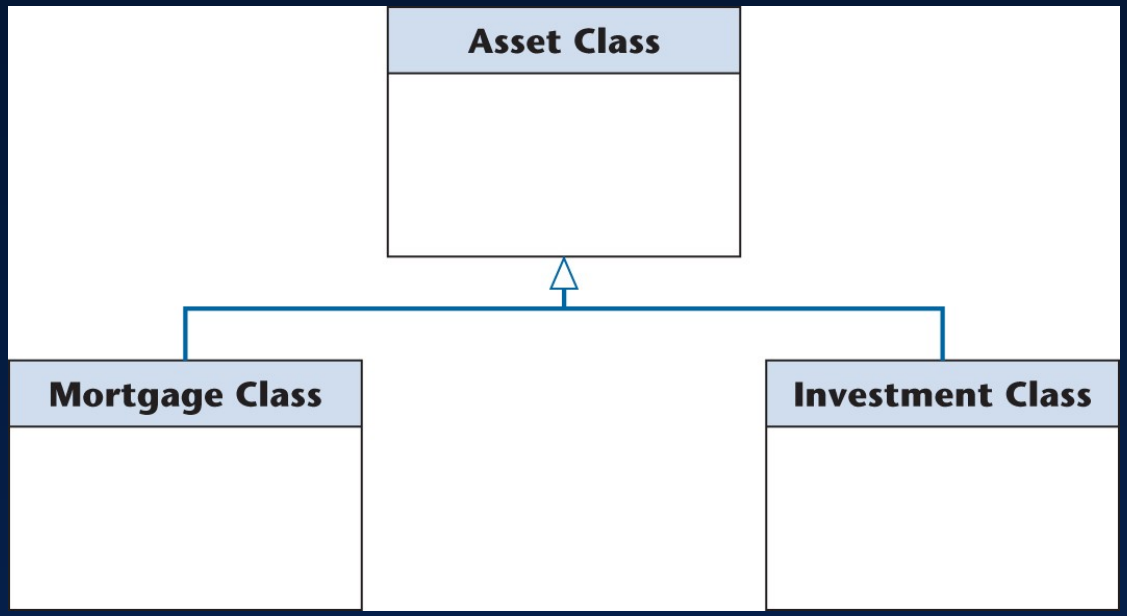


Figure 13.22

Back to the Requirements Workflow

- The current five use cases include `Manage a Mortgage` and `Manage an Investment`
- These two can now be combined into a single use case, `Manage an Asset`

Eighth Iteration of the Use-Case Diagram

- The new use case is shaded

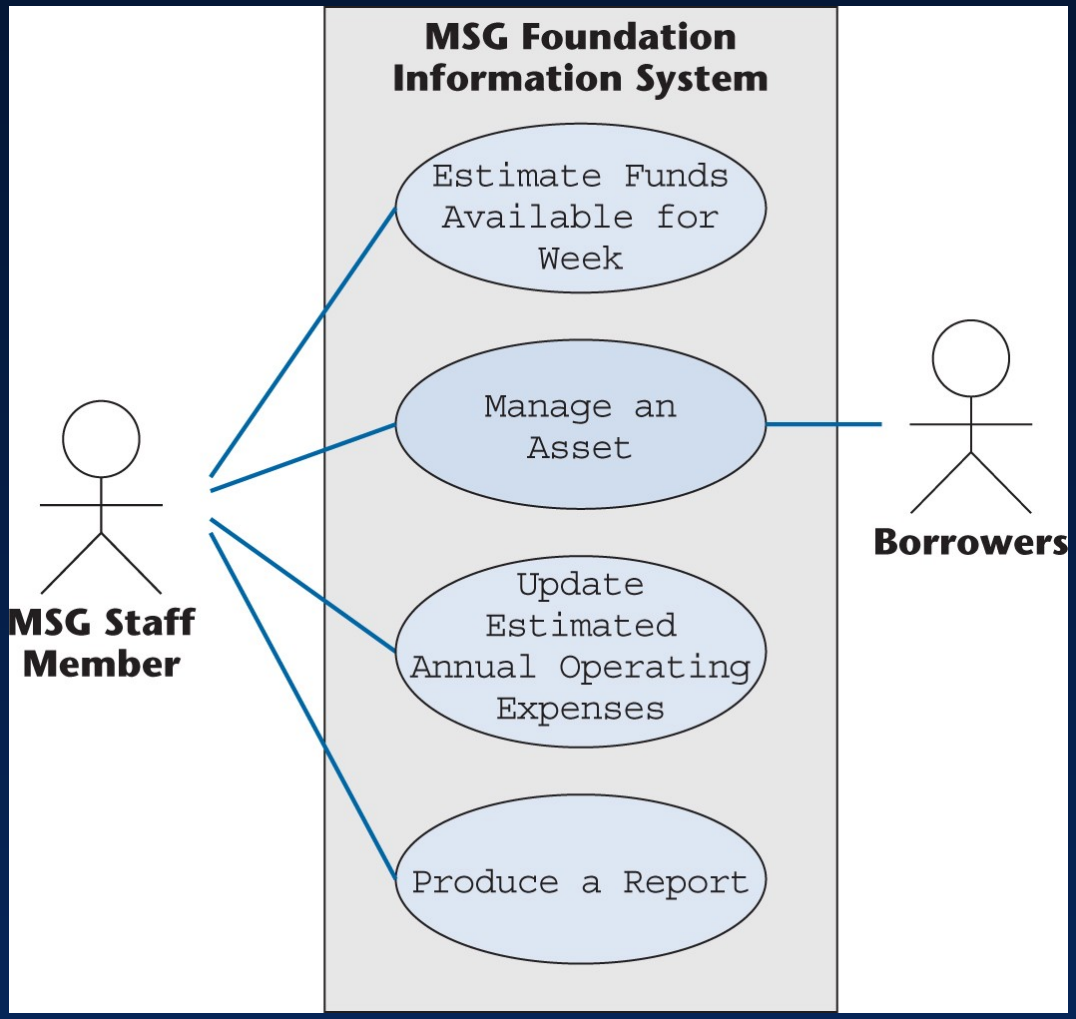


Figure 13.23

Initial Class Diagram: MSG Foundation (contd)

- Finally, we add the attributes of each class to the class diagram
  - For the MSG Foundation case study, the result is shown on the next slide
- The empty rectangle at the bottom of each box will later be filled with the operations of that class

Second Iteration of Initial Class Diagram (contd)

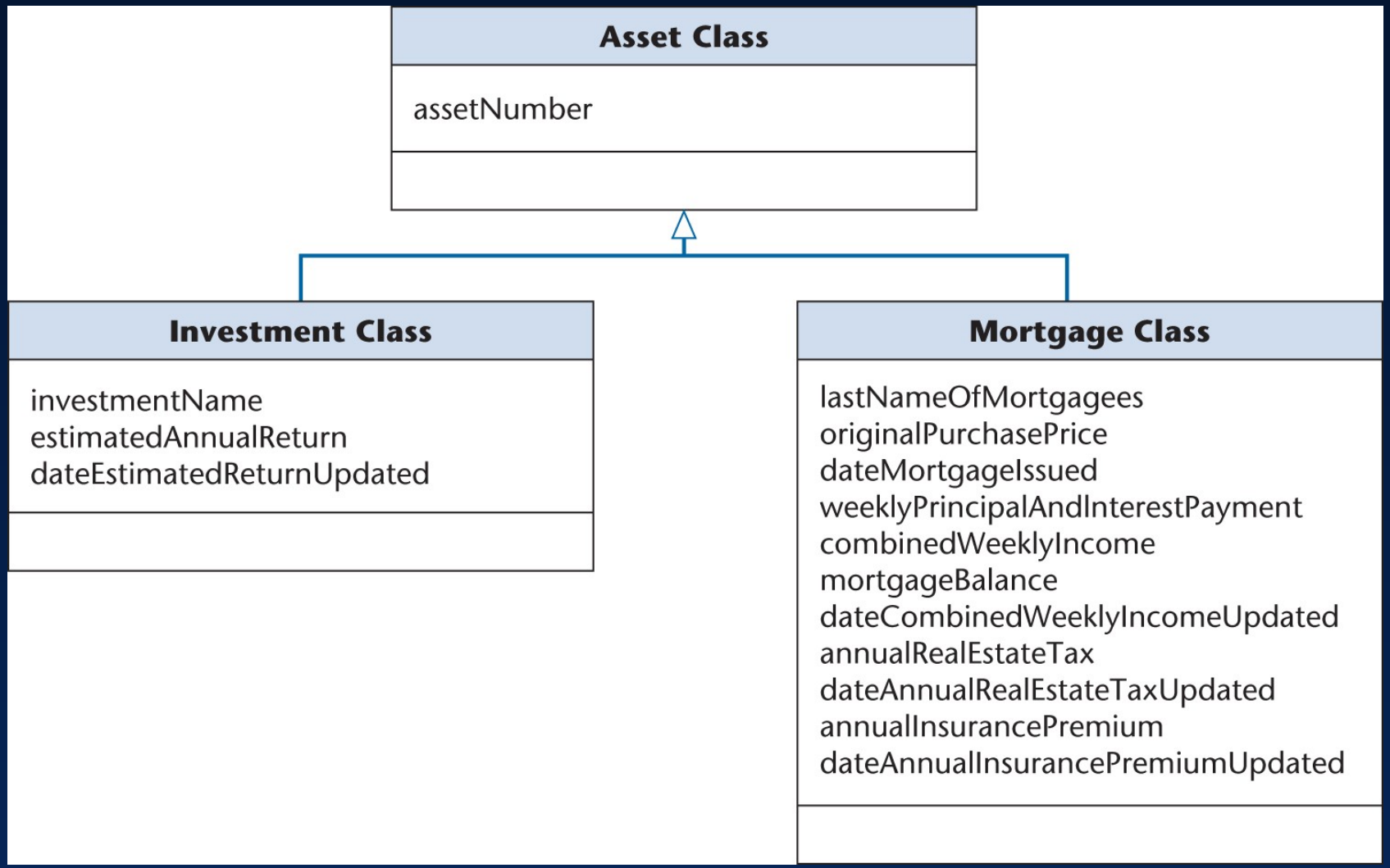


Figure 13.24

13.11 The Initial Dynamic Model: MSG Foundation

- Dynamic modeling is the third step in extracting the entity classes
- A statechart is constructed that reflects all the operations performed by or to the software product
- The operations are determined from the scenarios

Initial Dynamic Model: MSG Foundation (contd)

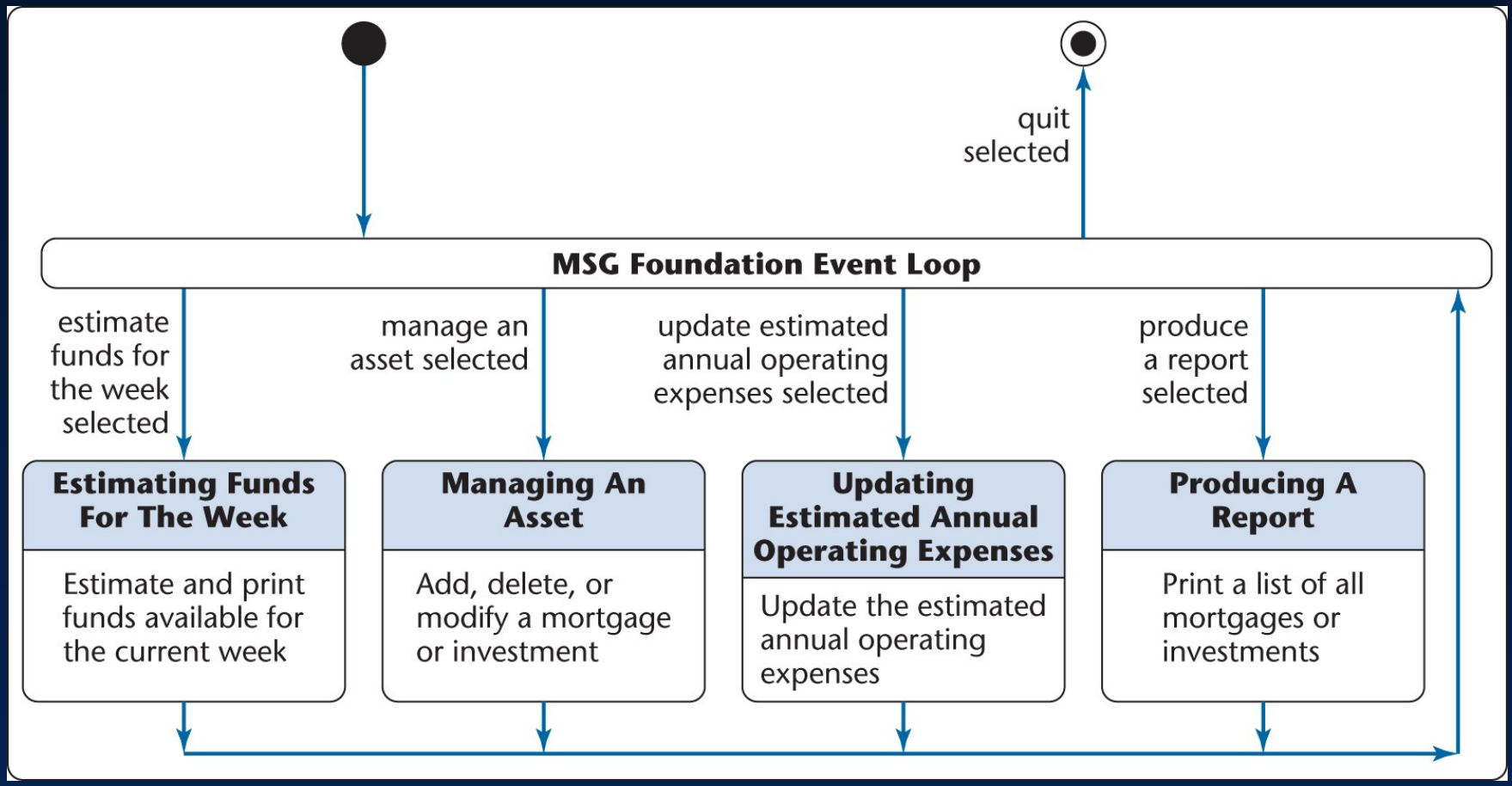


Figure 13.25

Initial Dynamic Model: MSG Foundation (contd)

- The statechart reflects the operations of the complete MSG Foundation information system
  - The solid circle on the top left represents the initial state, the starting point of the statechart
  - The white circle containing the small black circle on the top right represents the final state
  - States other than the initial and final states are represented by rectangles with rounded corners
  - The arrows represent possible transitions from state to state

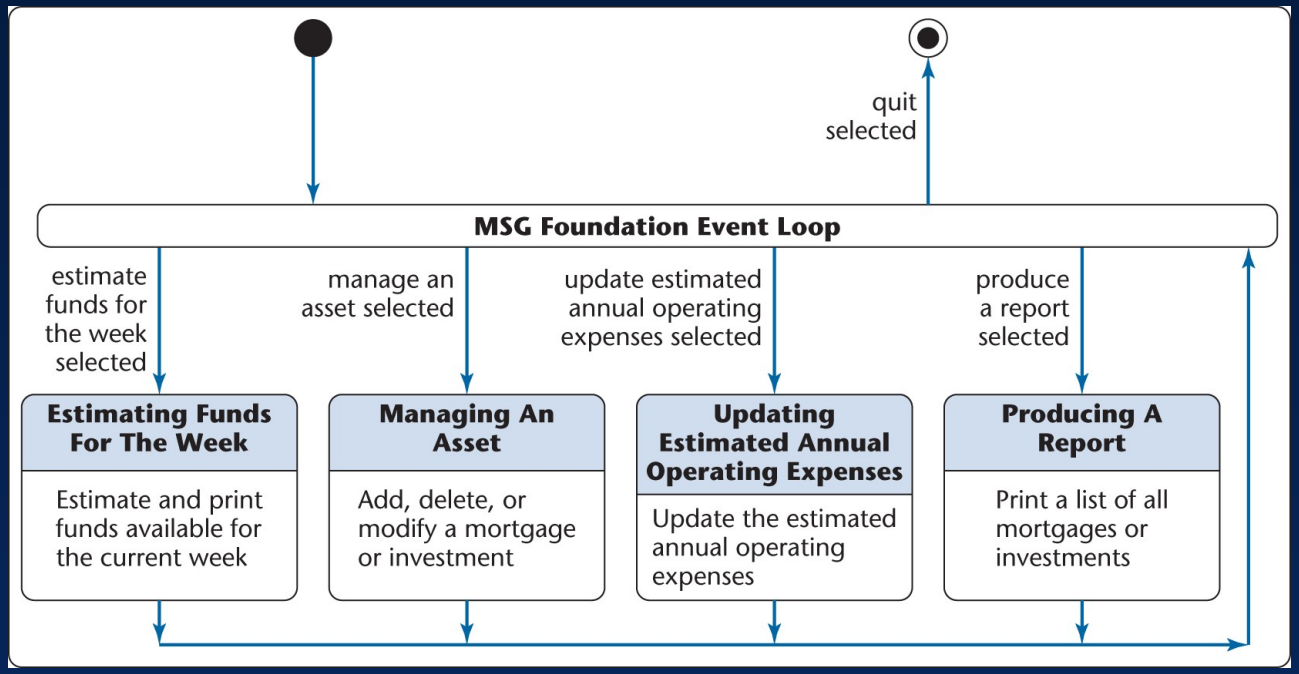
Initial Dynamic Model: MSG Foundation (contd)

- In state **MSG Foundation Information System Loop**, one of five events can occur

- An MSG staff member can issue one of five commands:
  - estimate funds for the week
  - manage an asset
  - update estimated annual operating expenses
  - produce a report, or
  - quit

Initial Dynamic Model: MSG Foundation (contd)

- These possibilities are indicated by the five events
  - estimate funds for the week selected
  - manage an asset selected
  - update estimated annual operating expenses selected
  - produce a report selected, and
  - quit selected
- An event causes a transition between states



Initial Dynamic Model: MSG Foundation (contd)

- An MSG staff member selects an option by clicking on the menu

Figure 13.26

Initial Dynamic Model: MSG Foundation (contd)

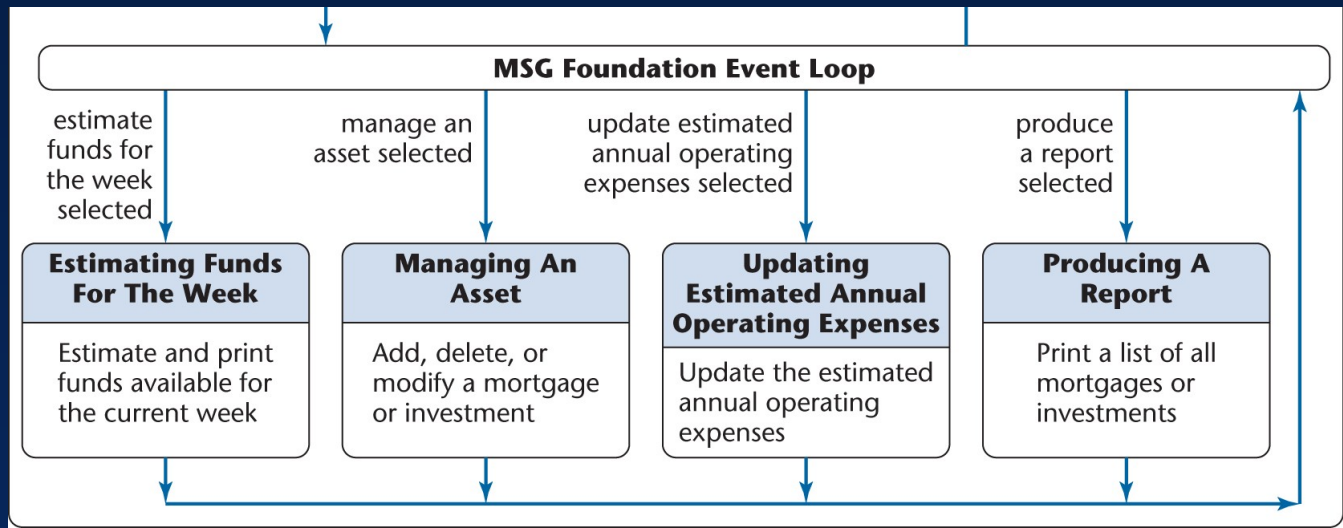
- Equivalent textual user interface that can run on any computer

```
                    MAIN MENU
MARTHA STOCKTON GREENGAGE FOUNDATION
   1.  Estimate funds available for week
   2.  Manage an asset
   3.  Update estimated annual operating expenses
   4.  Produce a report
   5.  Quit
Type your choice and press <ENTER>:
```

Figure 13.27

13.12 Revising the Entity Classes: MSG Foundation

- The initial functional model, the initial class diagram, and the initial dynamic model are completed
  - Checking them reveals a fault
- In the initial statechart, consider state **Updating Estimated Annual Operating Expenses** with operation Update the estimated annual operating expenses
  - This operation has to be performed on the current value of the estimated annual operating expense



Revising the Entity Classes: MSG Foundation (contd)

- But where is the value of the estimated annual operating expenses to be found?
- Currently there is only one class (**Asset Class**) and its two subclasses
  - Neither is appropriate for storing the estimated annual operating expenses

Revising the Entity Classes: MSG Foundation (contd)

- The only way a value can be stored on a long-term basis is as an attribute of an instance of that class or its subclasses

- Another entity class is needed for storing the estimated annual operating expenses
  - **MSG Application Class**

Third Iteration of the Initial Class Diagram: MSG Foundation

- **MSG Application Class** has other attributes as well
  - Attributes that do not appertain to the assets

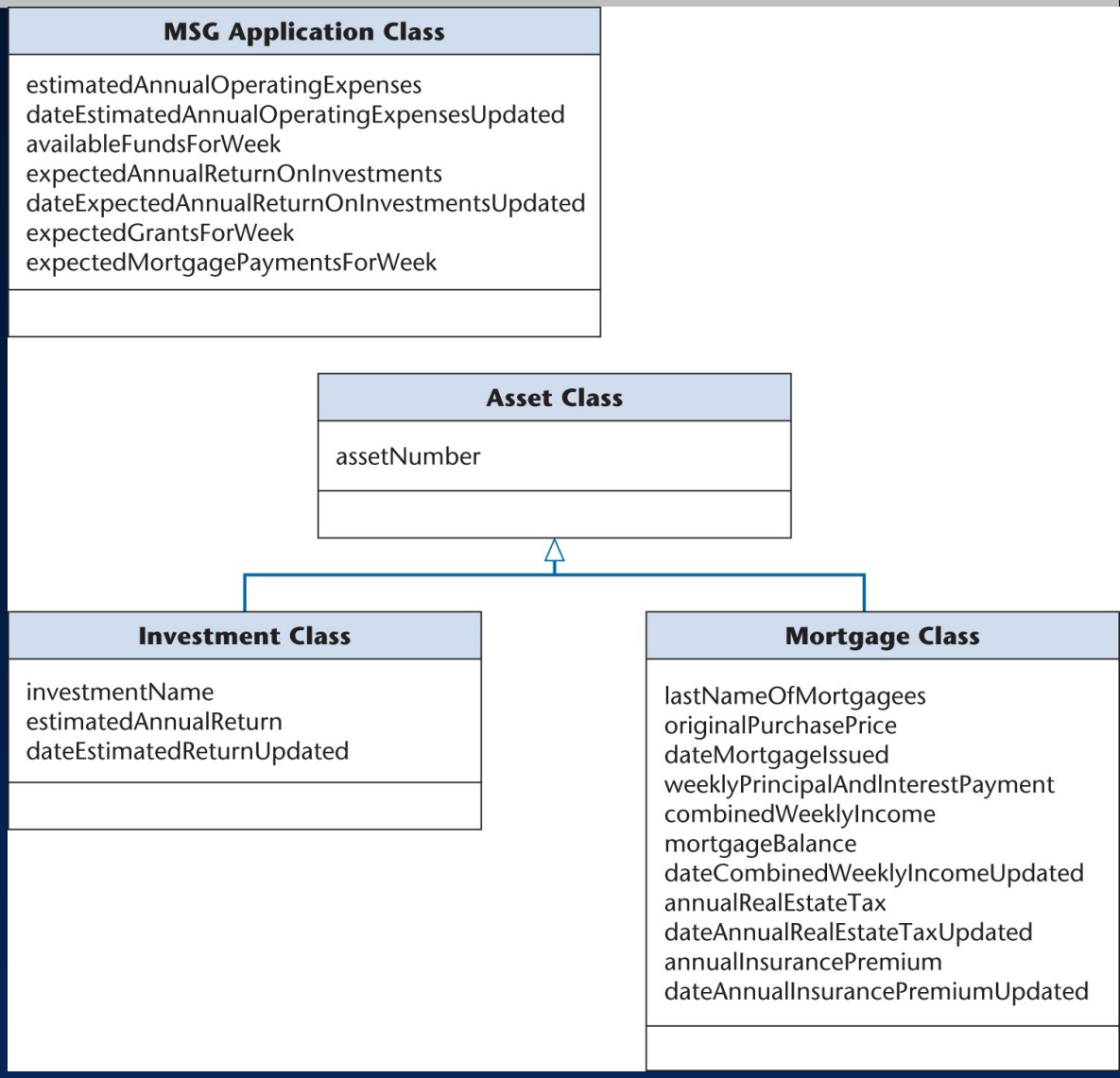


Figure 13.28

Third Iteration of the Initial Class Diagram: MSG Foundation

- The class diagram redrawn to show the prototypes

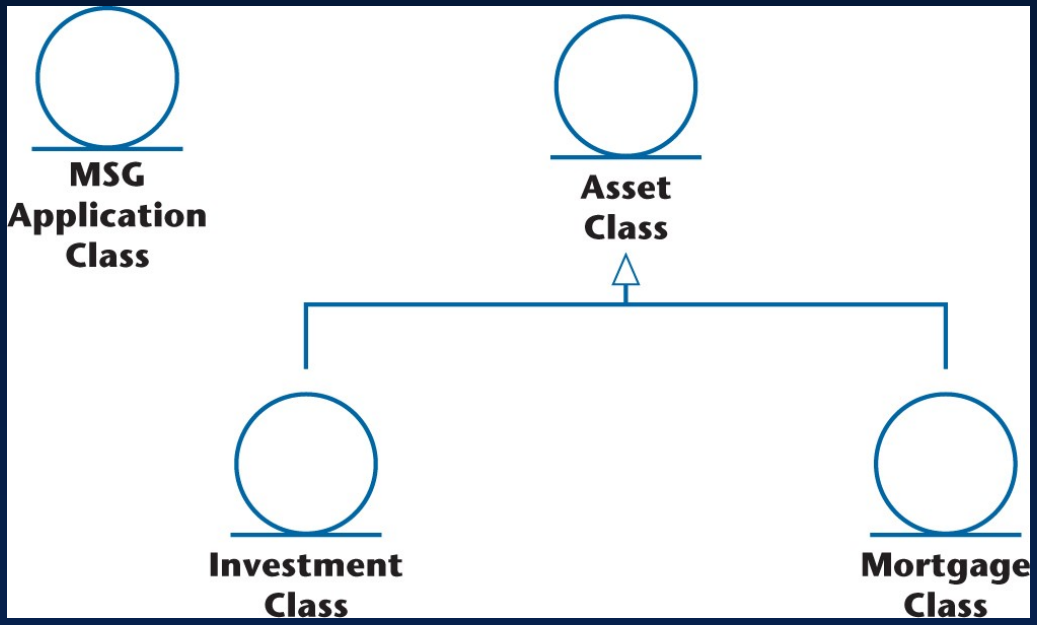


Figure 13.29

13.13 Extracting the Boundary Classes: MSG Foundation

- It is usually easy to extract boundary classes
  - Each input screen, output screen, and printed report is generally modeled by a boundary class
- One screen should be adequate for all four MSG Foundation use cases
  - » `Estimate Funds Available for Week`
  - » `Manage an Asset`
  - » `Update Estimated Annual Operating Expenses`
  - » `Produce a Report`

- Accordingly there is one initial boundary class
  - **User Interface Class**

Extracting Boundary Classes: MSG Foundation (contd)

- Three reports have to be printed
  - The estimated funds for the week report
  - The listing of all mortgages
  - The listing of all investments

- Each of these has to be modeled by a separate boundary class
  - **Estimated Funds Report Class**
  - **Mortgages Report Class**
  - **Investments Report Class**

Extracting Boundary Classes: MSG (contd)

- Here are the four initial boundary classes

User Interface Class
Estimated Funds Report Class
Mortgages Report Class
Investments Report Class

Figure 13.30

13.14 Extracting the Control Classes: MSG Foundation

- Each computation is usually modeled by a control class
- The MSG Foundation case study has just one
  - Estimate the funds available for the week
- There is one initial control class

Estimate Funds for Week Class

Figure 13.31

Class Extraction (contd)

- The description of class extraction is complete

Exercise

Please extract entity, boundary and control classes for the following system and draw a class diagram:

Consider an automated library circulation system. Every book has a bar code, and every borrower has a card bearing a bar code. When a borrower wishes to check out a book, the librarian scans the bar codes on the book and the borrower's card, and enters C at the computer terminal. Similarly, when a book is returned, it is again scanned and the librarian enters R. Librarians can add books (+) to the library collection or remove them (−). Borrowers can go to a terminal and determine all the books in the library by a particular author (the borrower enters A= followed by the author's name), all the books with a specific title (T= followed by the title), or all the books in a particular subject area (S= followed by the subject area). Finally, if a borrower wants a book currently checked out, the librarian can place a hold on the book so that, when it is returned, it will be held for the borrower who requested it (H= followed by the number of the book).

13.15 Use-Case Realization: The MSG Foundation Case Study

- The process of extending and refining use cases is called *use-case realization*

Use-Case Realization (contd)

- The realization of a specific scenario of a use case is depicted using an interaction diagram
  - Either a sequence diagram or collaboration diagram
- Consider use case `Estimate Funds Available for Week`
- We have previously seen
  - The use case
  - The description of the use case

13.15.1 Estimate Funds Available for Week Use Case

- Use-case diagram

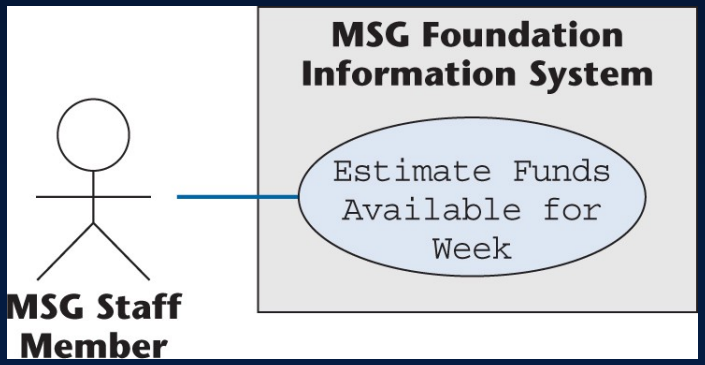


Figure 13.32

Estimate Funds Available for Week Use Case (contd)

- Description of use case

Brief Description

The Estimate Funds Available for Week use case enables an MSG Foundation staff member to estimate how much money the Foundation has available that week to fund mortgages.

Step-by-Step Description

1. For each investment, extract the estimated annual return on that investment. Summing the separate returns and dividing the result by 52 yields the estimated investment income for the week.
2. Determine the estimated MSG Foundation operating expenses for the week by extracting the estimated annual MSG Foundation operating expenses and dividing by 52.
3. For each mortgage:
   - 3.1 The amount to be paid this week is the total of the principal and interest payment and $\frac{1}{52}$nd of the sum of the annual real-estate tax and the annual homeowner's insurance premium.
   - 3.2 Compute 28 percent of the couple's current gross weekly income.
   - 3.3 If the result of Step 3.1 is greater than the result of Step 3.2, then the mortgage payment for this week is the result of Step 3.2, and the amount of the grant for this week is the difference between the result of Step 3.1 and the result of Step 3.2.
   - 3.4 Otherwise, the mortgage payment for this week is the result of Step 3.1, and there is no grant this week.
4. Summing the mortgage payments of Steps 3.3 and 3.4 yields the estimated total mortgage payments for the week.
5. Summing the grant payments of Step 3.3 yields the estimated total grant payments for the week.
6. Add the results of Steps 1 and 4 and subtract the results of Steps 2 and 5. This is the total amount available for mortgages for the current week.
7. Print the total amount available for new mortgages during the current week.

Figure 13.33

Estimate Funds Available for Week Use Case (contd)

- Class diagram (classes that enter into the use case)

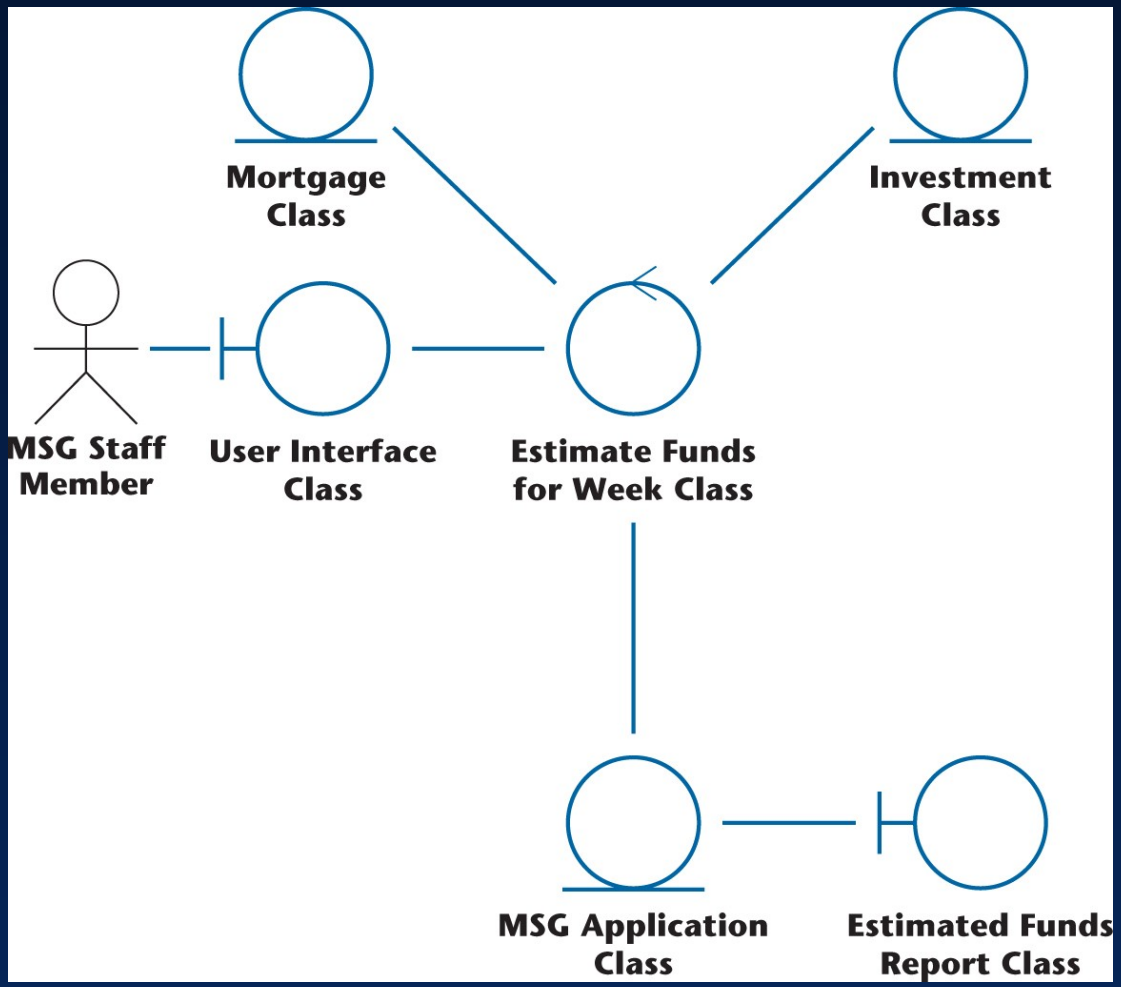


Figure 13.34

Estimate Funds Available for Week Use Case (contd)

- The six classes that enter into this use case are:
  - **User Interface Class**
    - » This class models the user interface
  - **Estimate Funds for Week Class**
    - » This control class models the computation of the estimate of the funds that are available to fund mortgages during that week
  - **Mortgage Class**
    - » This class models the estimated grants and payments for the week

Estimate Funds Available for Week Use Case (contd)

- **Investment Class**
  - This class models the estimated return on investments for the week
- **MSG Application Class**
  - This class models the estimated operating expenses for the week
- **Estimated Funds Report Class**
  - This class models the printing of the report

Estimate Funds Available for Week Use Case (contd)

- Scenario (one possible instance of the use case)

An MSG Foundation staff member wishes to determine the funds available for mortgages this week.

1. For each investment, the information system extracts the estimated annual return on that investment. It sums the separate returns and divides the result by 52 to yield the estimated investment income for the week.
2. The information system then extracts the estimated annual MSG Foundation operating expenses and divides the result by 52.
3. For each mortgage:
   - 3.1 The information system computes the amount to be paid this week by adding the principal and interest payment to $\frac{1}{52}$nd of the sum of the annual real-estate tax and the annual homeowner's insurance premium.
   - 3.2 It then computes 28 percent of the couple's current gross weekly income.
   - 3.3 If the result of Step 3.1 is greater than the result of Step 3.2, then it determines the mortgage payment for the week as the result of Step 3.2, and the amount of the grant for this week as the difference between the result of Step 3.1 and the result of Step 3.2.
   - 3.4 Otherwise, it takes the mortgage payment for this week as the result of Step 3.1, and there is no grant for the week.
4. The information system sums the mortgage payments of Steps 3.3 and 3.4 to yield the estimated total mortgage payments for the week.
5. It sums the grant payments of Step 3.3 to yield the estimated total grant payments for the week.
6. The information system adds the results of Steps 1 and 4 and subtracts the results of Steps 2 and 5. This is the total amount available for mortgages for the current week.
7. Finally, the software product prints the total amount available for new mortgages during the current week.

Figure 13.35

Estimate Funds Available for Week Use Case (contd)

- A working information system uses objects, not classes
  - Example: A specific mortgage cannot be represented by **Mortgage Class** but rather by an object, a specific instance of **Mortgage Class**

- Such an object is denoted by **: Mortgage Class**

Estimate Funds Available for Week Use Case (contd)

- A class diagram shows the classes in the use case and their relationships
  - It does not show the objects nor the sequence of messages as they are sent from object to object
- Something more is needed

Estimate Funds Available for Week Use Case (contd)

- Collaboration diagram (of the realization of the scenario of the use case)

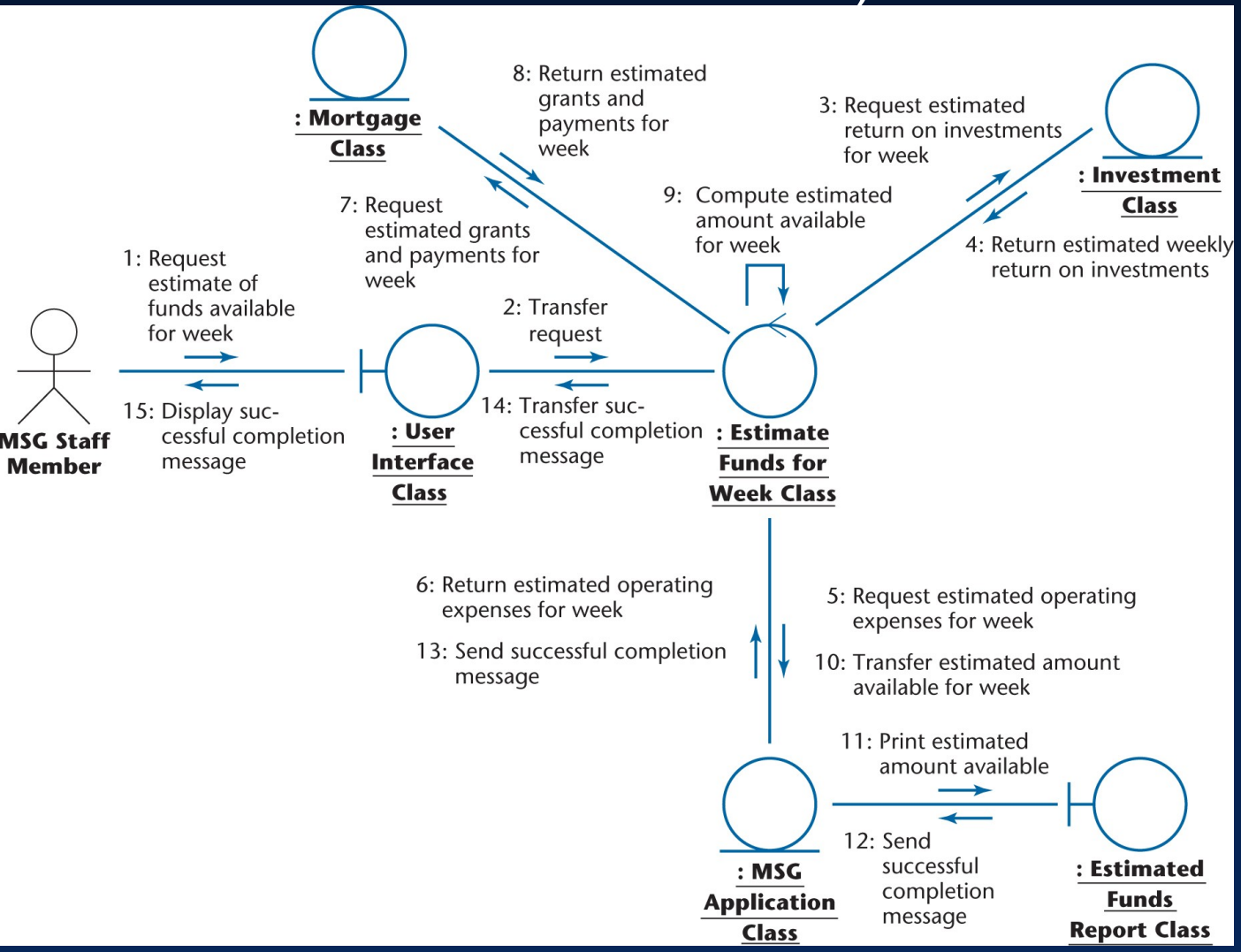
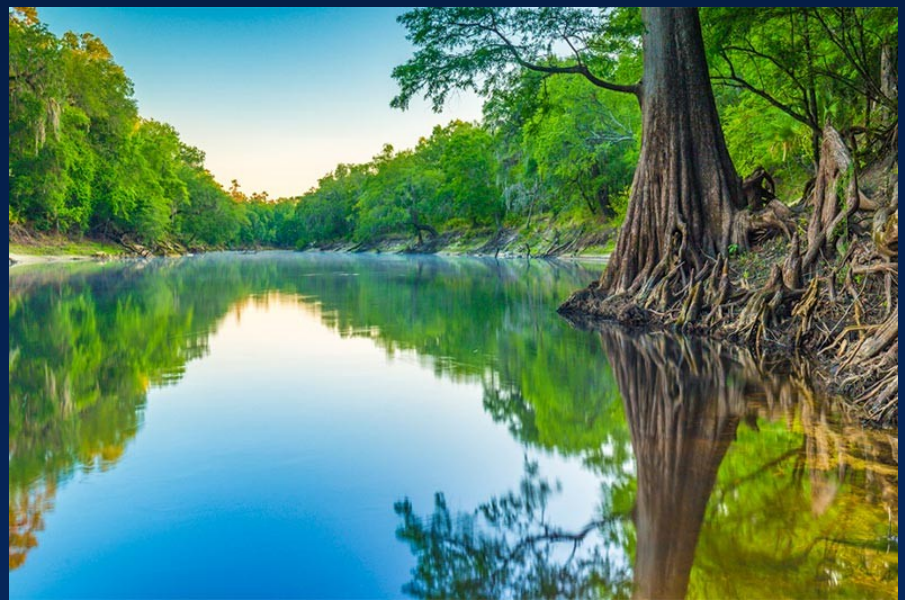


Figure 13.36

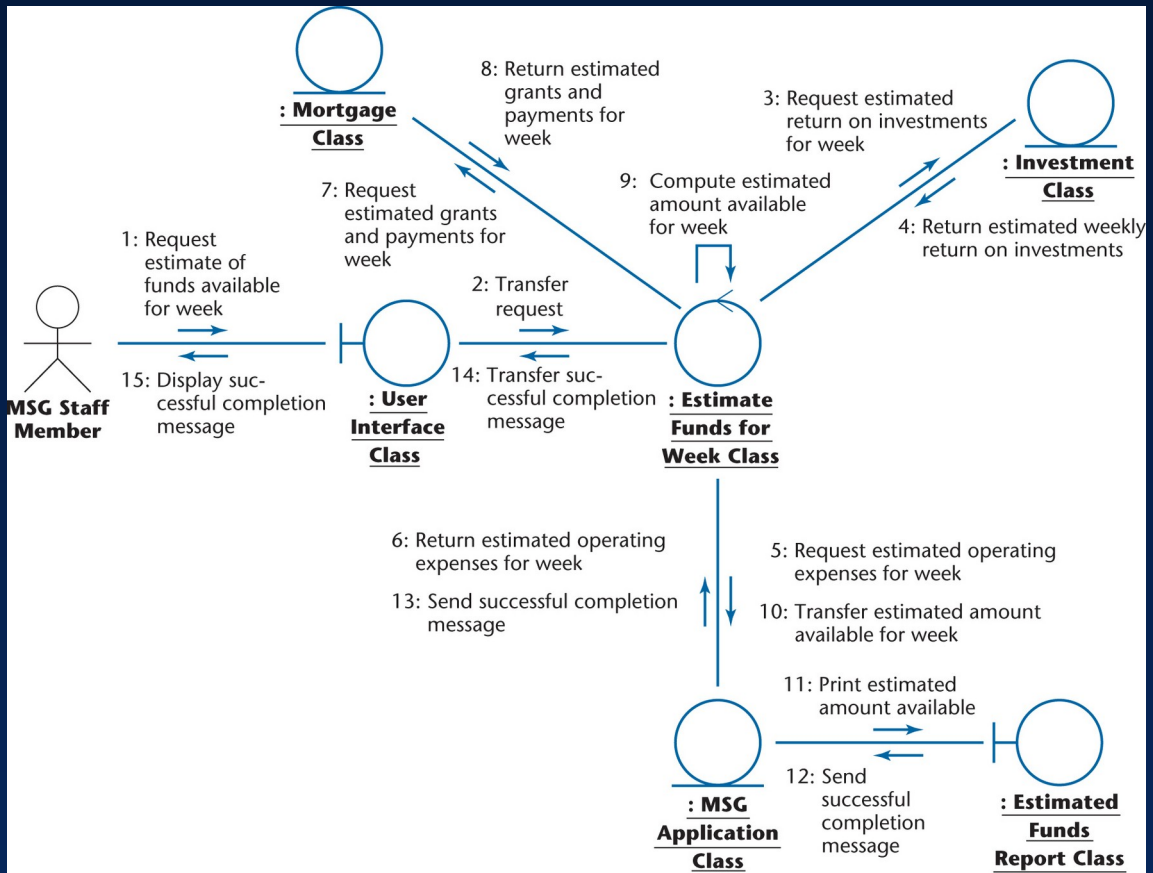
- The collaboration diagram shows the objects as well as the messages, numbered in the order in which they are sent in the specific scenario



Estimate Funds Available for Week Use Case (contd)

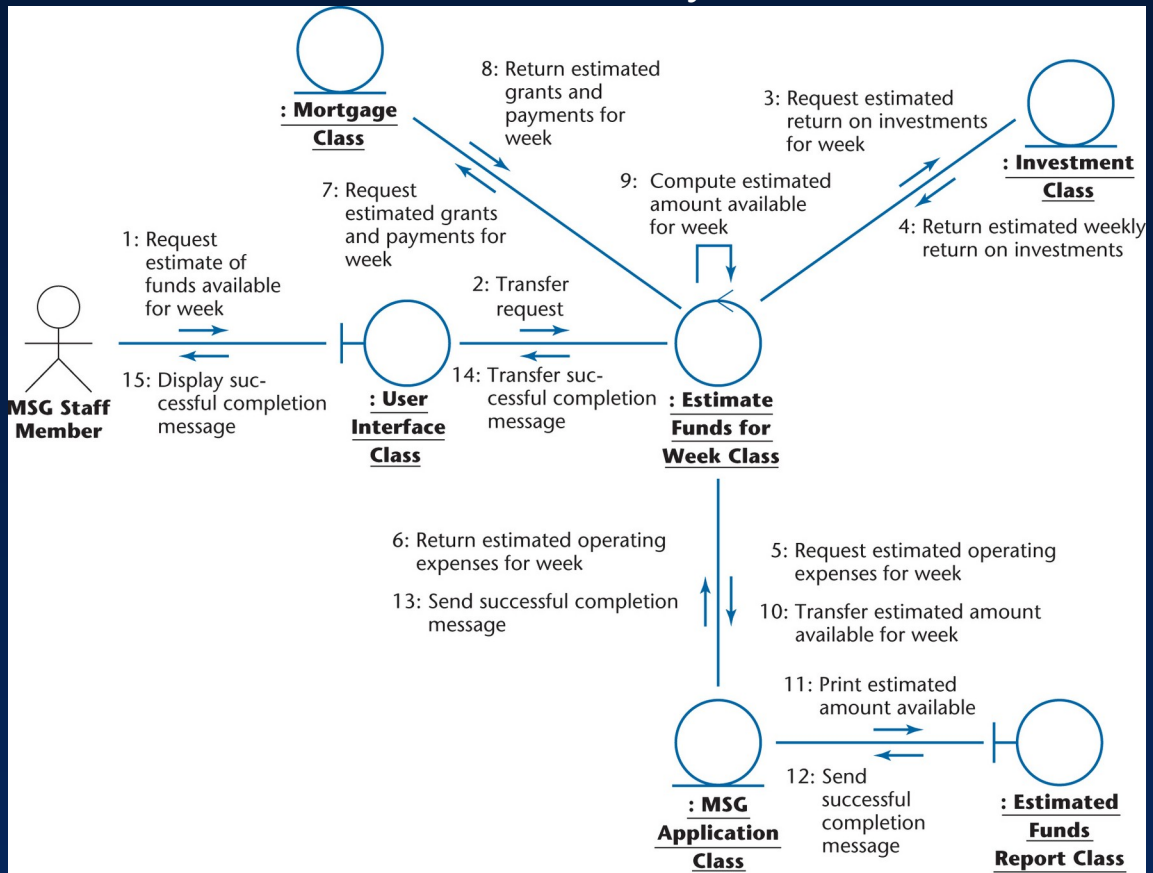
- Item 1:
  - The staff member wants to compute the funds available for the week
  - In the collaboration diagram, this is modeled by message
    - » 1: Request estimate of funds available for week

    from **MSG Staff Member** to **: User Interface Class**, an instance of **User Interface Class**



Estimate Funds Available for Week Use Case (contd)

- Item 2
  - This request is passed on to **: Estimate Funds for Week Class**, an instance of the control class that actually performs the calculation
  - This is modeled by message
    - » 2: Transfer request
- Four separate financial estimates are now determined by **: Estimate Funds for Week Class**

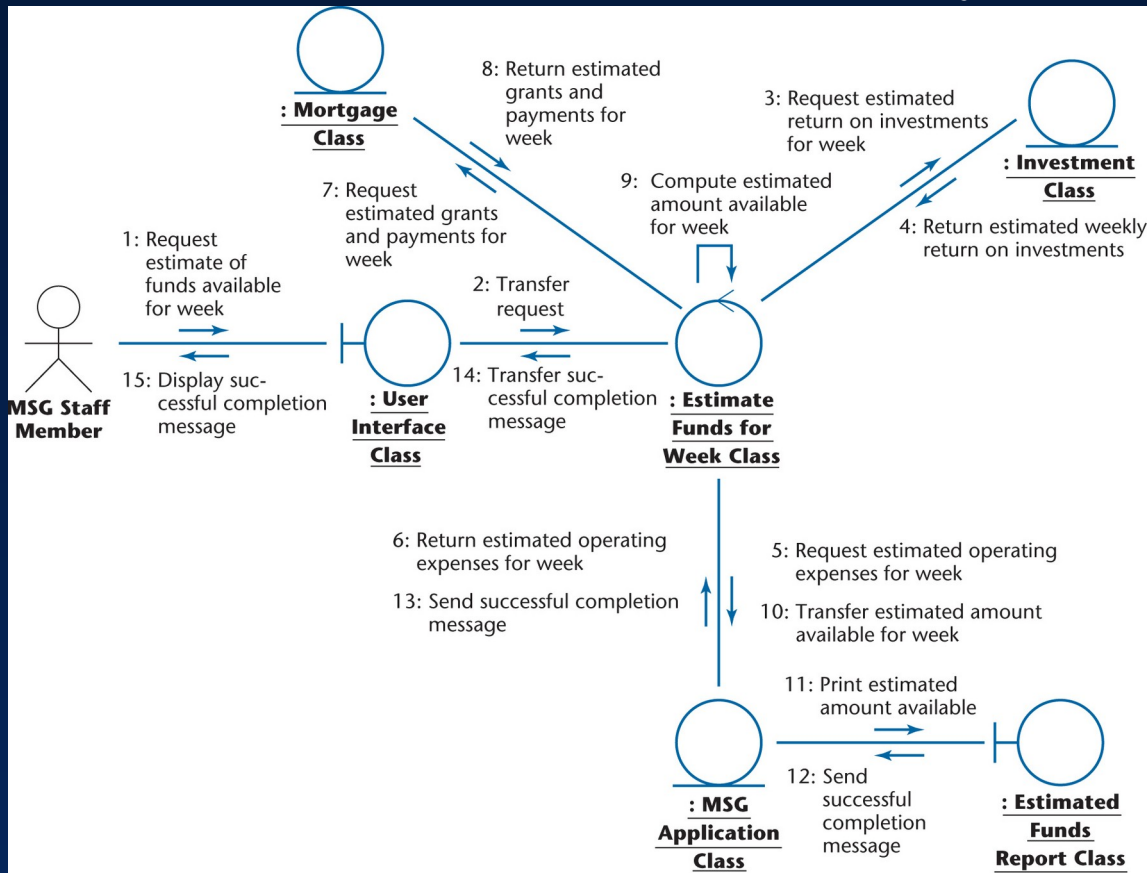


Estimate Funds Available for Week Use Case (contd)

- Item 3
  - In Step 1 of the scenario, the estimated annual return on investments is summed for each investment and the result divided by 52
  - This extraction of the estimated weekly return is modeled by message
    - » 3: Request estimated return on investments for week

    from **: Estimate Funds for Week Class** to **: Investment Class** followed by message
    - » 4: Return estimated weekly return on investments

    in the other direction

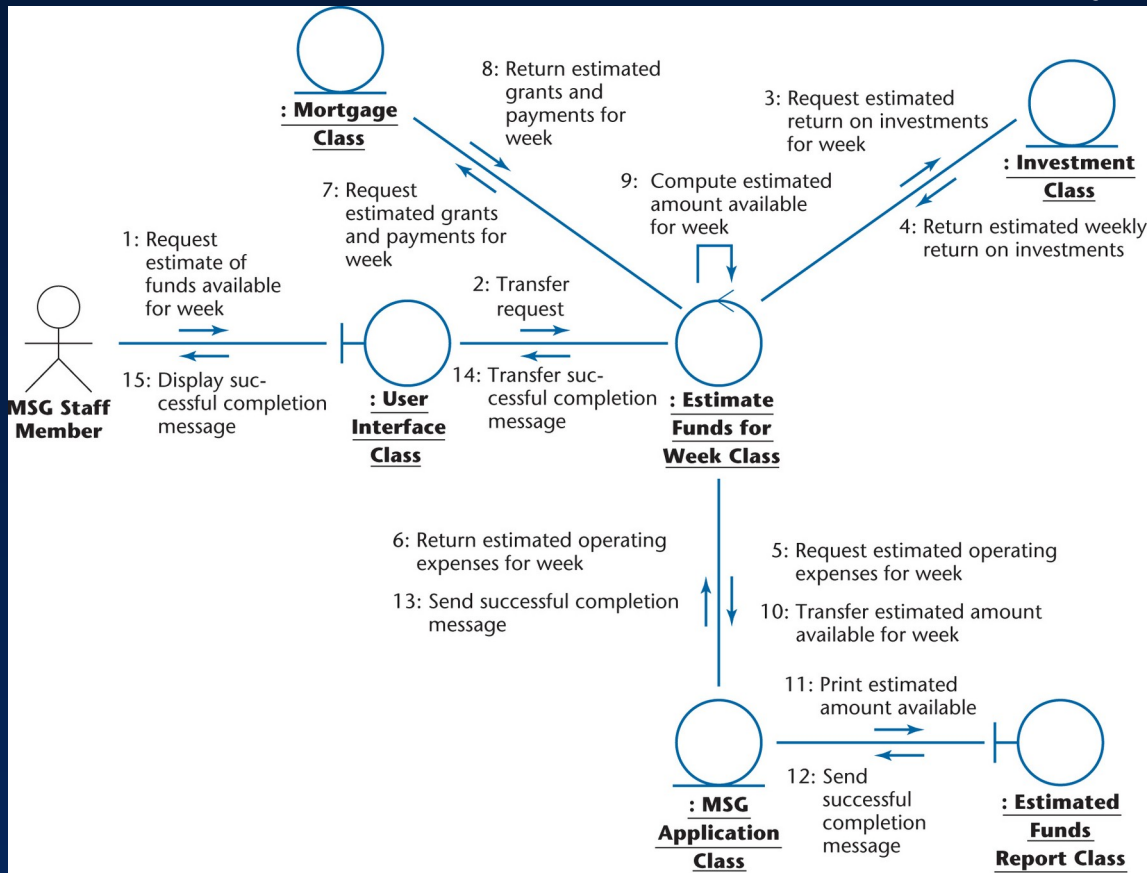


Estimate Funds Available for Week Use Case (contd)

- Item 4
  - In Step 2 of the scenario, the weekly operating expenses are estimated by taking the estimated annual operating expenses and dividing by 52
  - This extraction of the weekly expenses is modeled by message
    - » 5: Request estimated operating expenses for week

    from **: Estimate Funds for Week Class** to **: MSG Application Class** followed by message
    - » 6: Return estimated operating expenses for week

    in the other direction

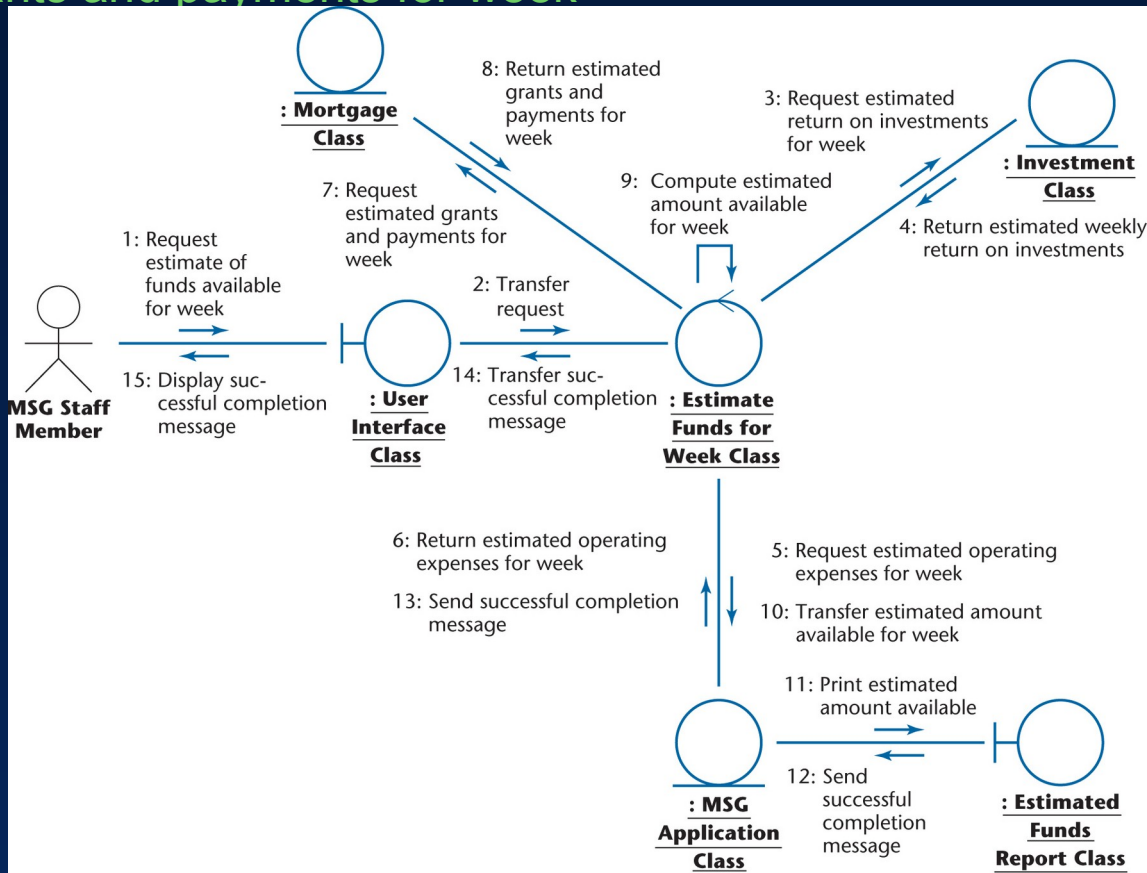


Estimate Funds Available for Week Use Case (contd)

- Item 5
  - In Steps 3, 4, and 5 of the scenario, two estimates are determined
    - » the estimated grants for the week, and
    - » the estimated payments for the week
  - This is modeled by message
    - » 7: Request estimated grants and payments for week

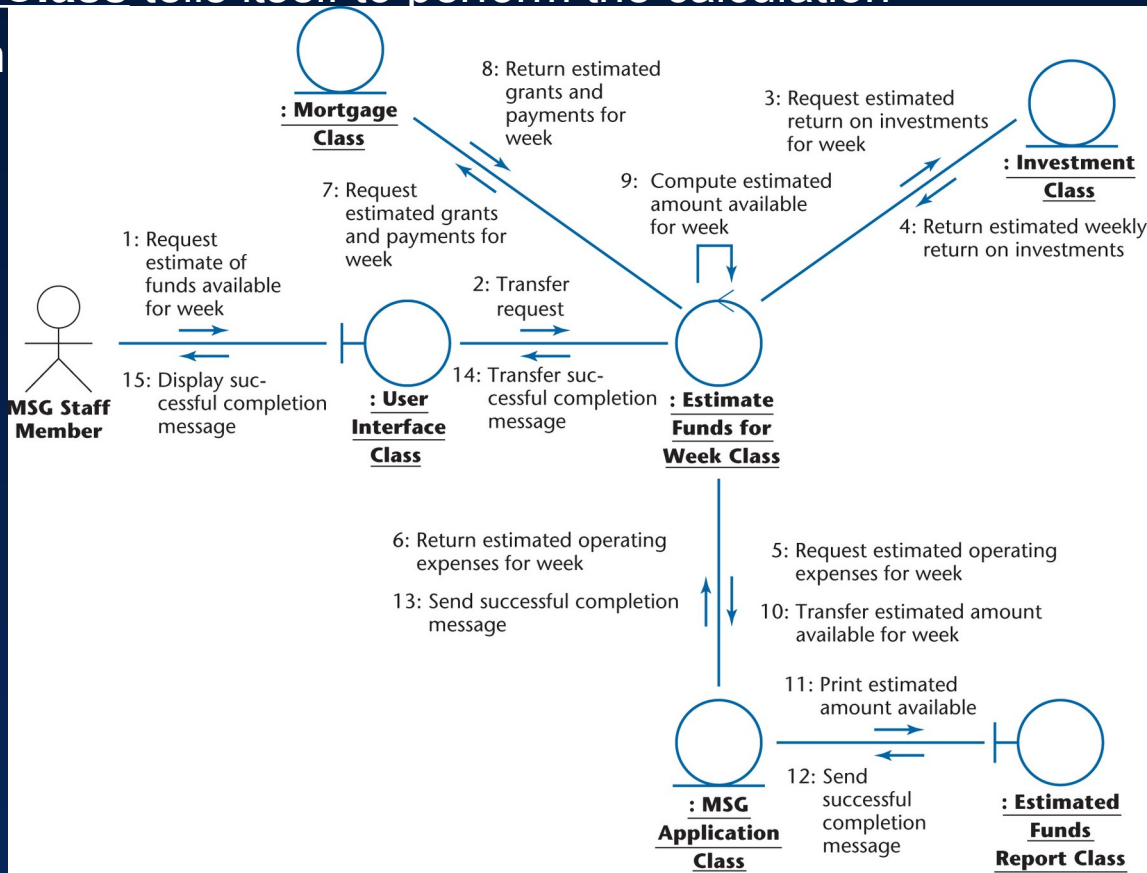
    from **: Estimate Funds for Week Class** to **: Mortgage Class**, and by message
    - » 8: Return estimated grants and payments for week

    in the other direction



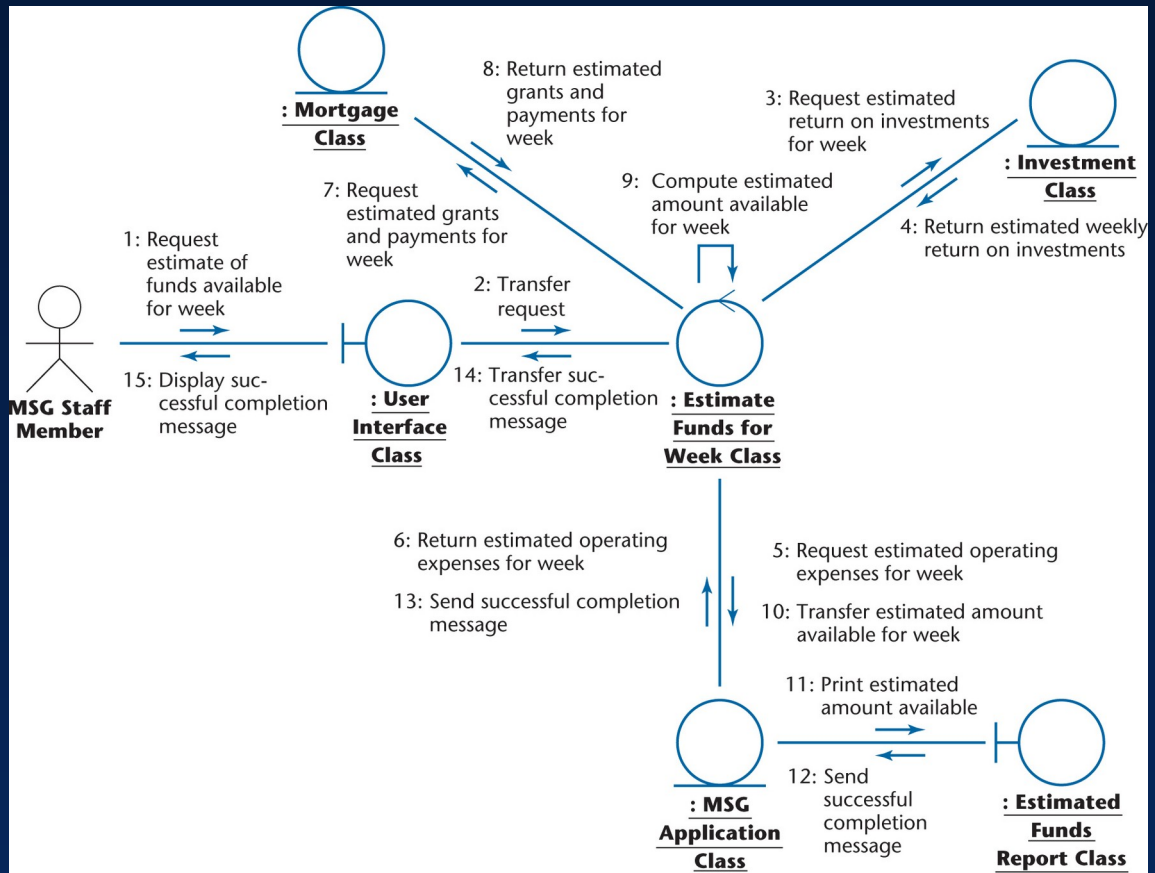
Estimate Funds Available for Week Use Case (contd)

- Item 6
  - Now the arithmetic computation of Step 6 of the scenario is performed
  - This is modeled by message
    - 9: Compute estimated amount available for week
  - This is a self call
  - **: Estimate Funds for Week Class** tells itself to perform the calculation
  - The result of the computation is stored in **: MSG Application Class** by message
    - 10: Transfer estimated amount available for week



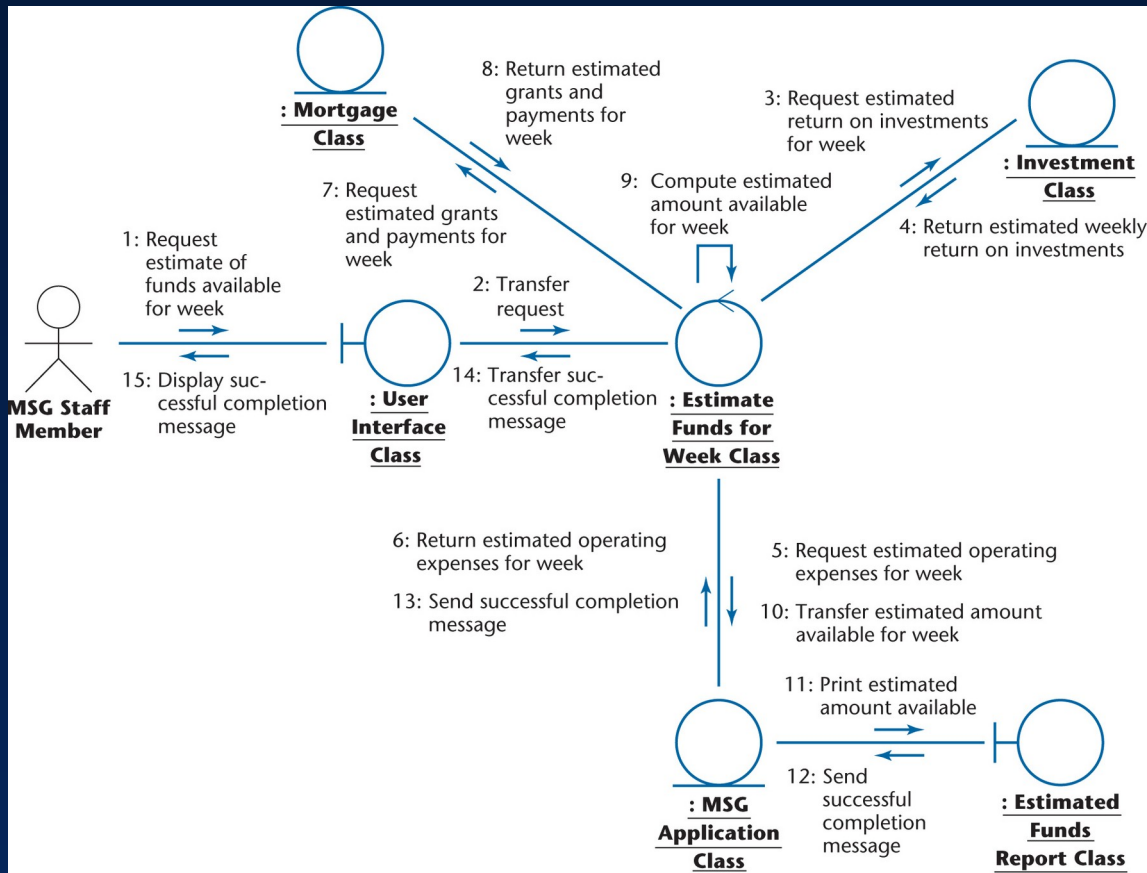
Estimate Funds Available for Week Use Case (contd)

- Item 7
  - The result is printed in Step 7 of the scenario
  - This is modeled by message
    - » 11: Print estimated amount available
  - from **: MSG Application Class** to **: Estimated Funds Report Class**



Estimate Funds Available for Week Use Case (contd)

- Item 8
  - Finally, an acknowledgment is sent to the MSG staff member that the task has been successfully completed
  - This is modeled by messages
    - 12: Send successful completion message
    - 13: Send successful completion message
    - 14: Transfer successful completion message, and
    - 15: Display successful completion message



Exercise

Please draw collaboration diagrams for the following system:

Consider an automated library circulation system. Every book has a bar code, and every borrower has a card bearing a bar code. When a borrower wishes to check out a book, the librarian scans the bar codes on the book and the borrower's card, and enters C at the computer terminal. Similarly, when a book is returned, it is again scanned and the librarian enters R. Librarians can add books (+) to the library collection or remove them (−). Borrowers can go to a terminal and determine all the books in the library by a particular author (the borrower enters A= followed by the author's name), all the books with a specific title (T= followed by the title), or all the books in a particular subject area (S= followed by the subject area). Finally, if a borrower wants a book currently checked out, the librarian can place a hold on the book so that, when it is returned, it will be held for the borrower who requested it (H= followed by the number of the book).

- No client will approve the specification document without understanding it
- Accordingly, a written description of the collaboration diagram is needed, the flow of events

Estimate Funds Available for Week Use Case (contd)

- The flow of events of the collaboration diagram of the realization of the scenario of the use case

An MSG staff member requests an estimate of the funds available for mortgages for the week (1, 2). The information system estimates the return on investments for the week (3, 4), the operating expenses for the week (5, 6), and the grants and payments for the week (7, 8). Then it estimates (9), stores (10), and prints out (11–15) the funds available for the week.

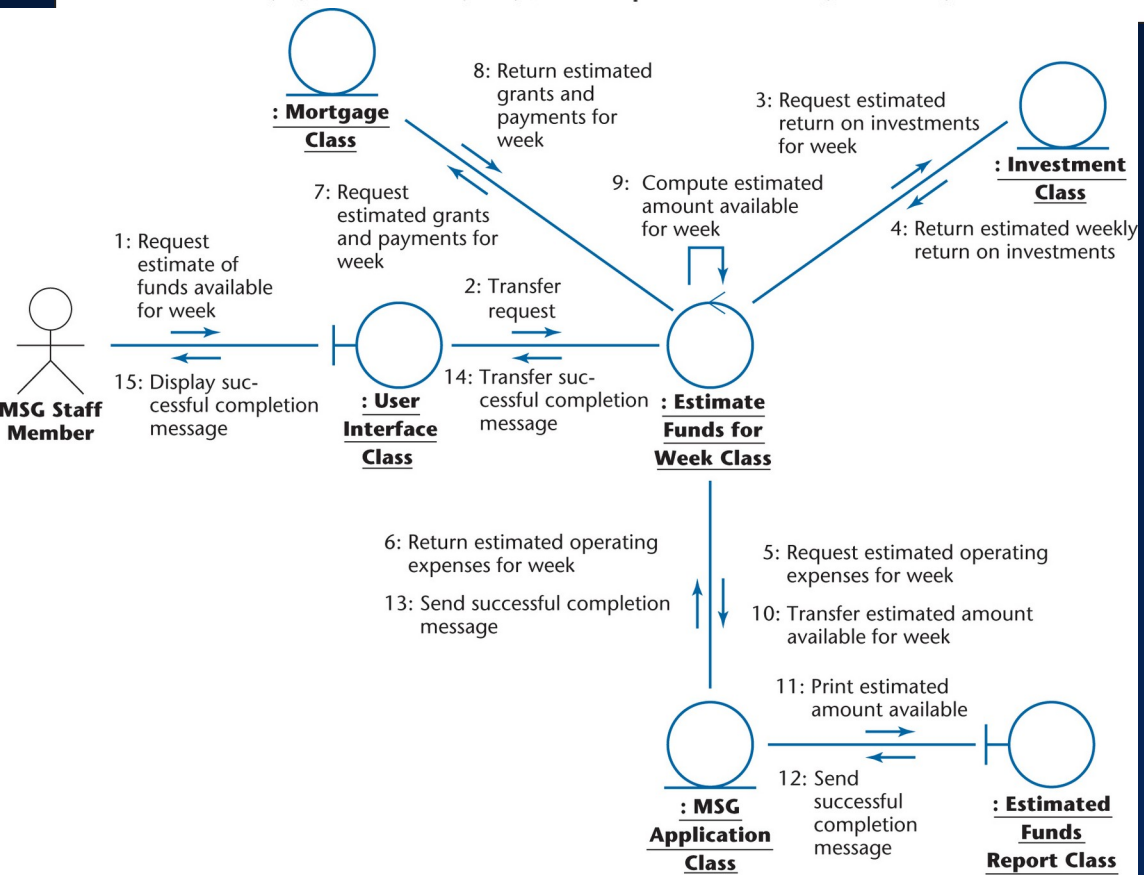


Figure 13.37

Estimate Funds Available for Week Use Case (contd)

- Sequence diagram equivalent to the collaboration diagram (of the realization of the use case)

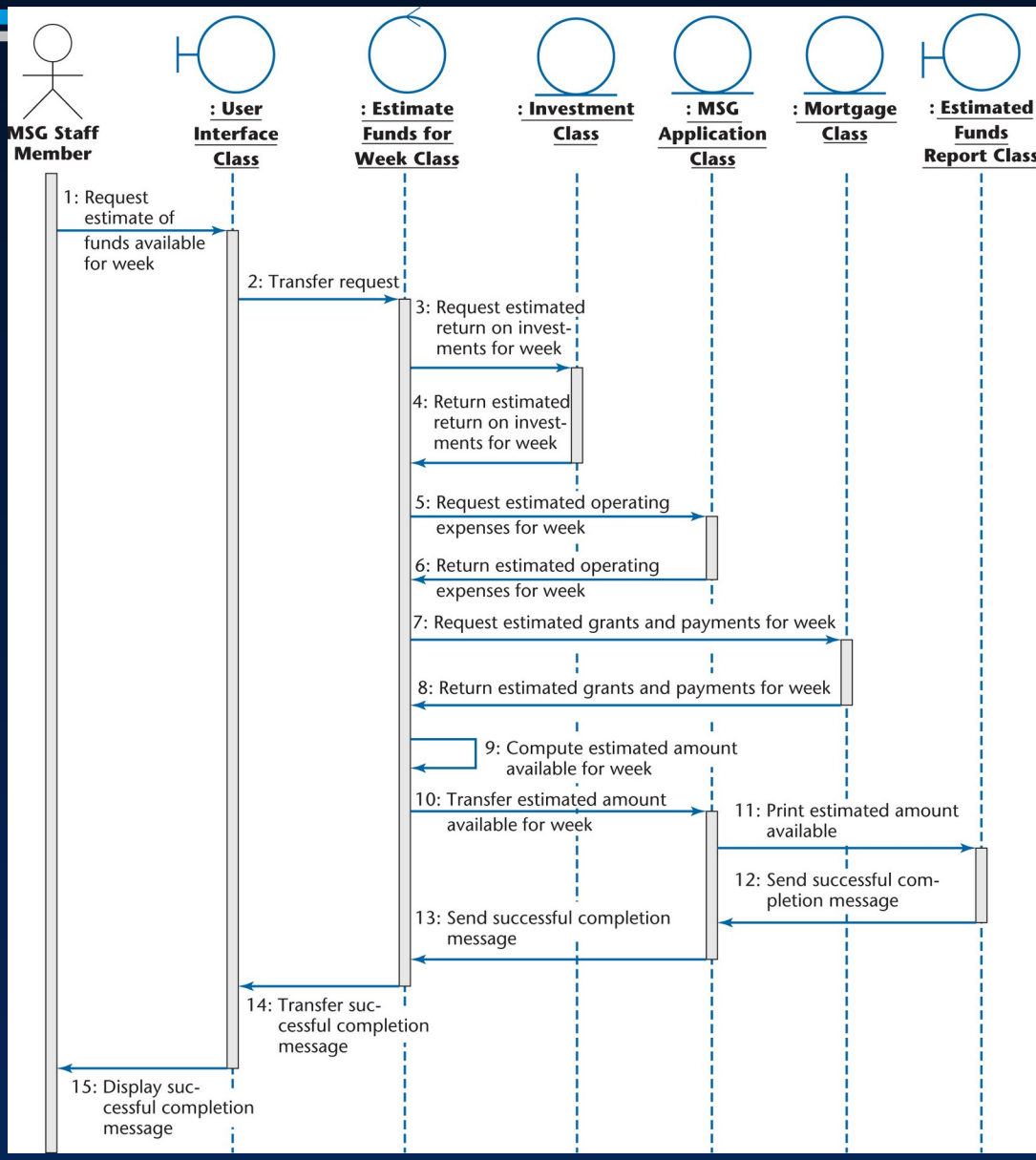


Figure 13.38

Interaction Diagrams

- The strength of a sequence diagram is that it shows the flow of messages and their order unambiguously
  - When transfer of information is the focus of attention, a sequence diagram is superior to a collaboration diagram
- A collaboration diagram is similar to a class diagram
  - When the developers are concentrating on the classes, a collaboration diagram is more useful than the equivalent sequence diagram

13.15.2 Manage an Asset Use Case

- Use case

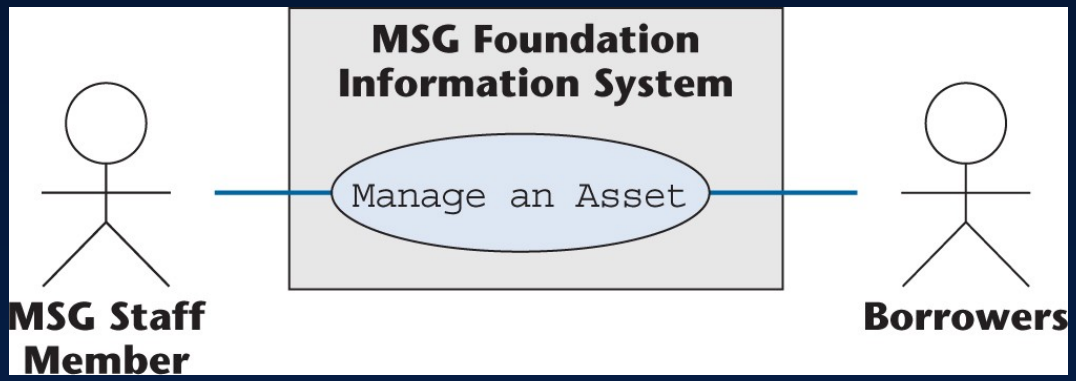


Figure 13.39

Manage an Asset Use Case (contd)

- Description of use case

Brief Description

The Manage an Asset use case enables an MSG Foundation staff member to add and delete assets and manage the portfolio of assets (investments and mortgages). Managing a mortgage includes updating the weekly income of a couple who have borrowed money from the Foundation.

Step-by-Step Description

1. Add, modify, or delete an investment or mortgage, or update the borrower's weekly income.

Figure 13.40

Manage an Asset Use Case (contd)

- Class diagram showing the classes that realize the use case

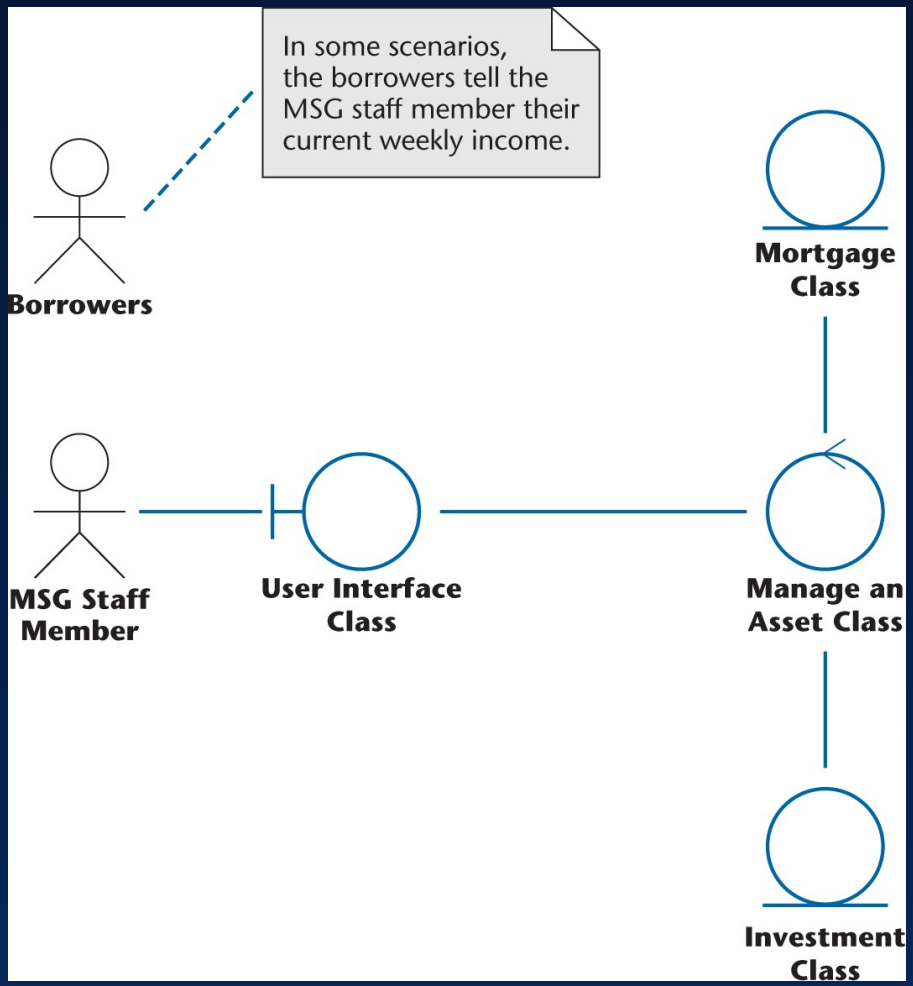


Figure 13.41

Manage an Asset Use Case (contd)

- One scenario of the use case

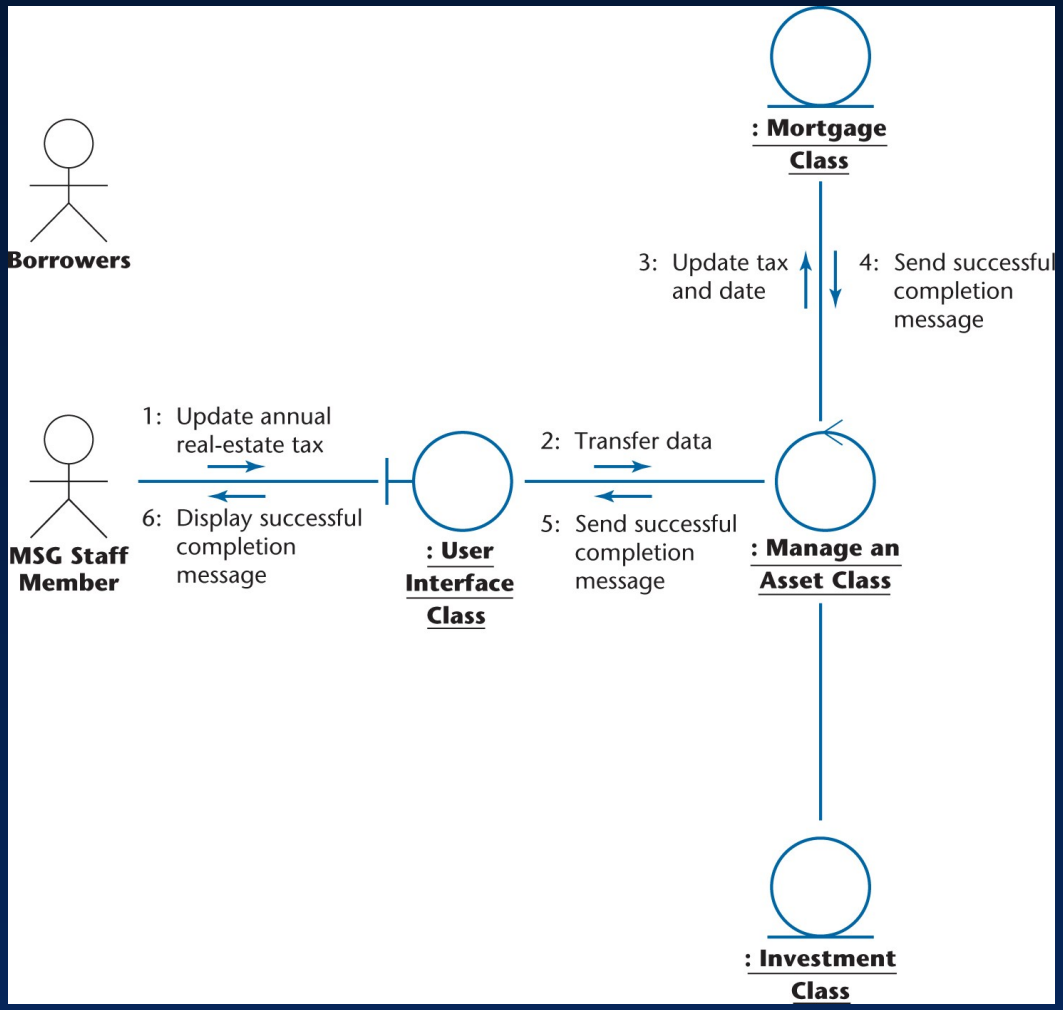
An MSG Foundation staff member wants to update the annual real-estate tax on a home for which the Foundation has provided a mortgage.

1. The staff member enters the new value of the annual real-estate tax.
2. The information system updates the date on which the annual real-estate tax was last changed.

Figure 13.42

Manage an Asset Use Case (contd)

- Collaboration diagram of the realization of the scenario of the use case



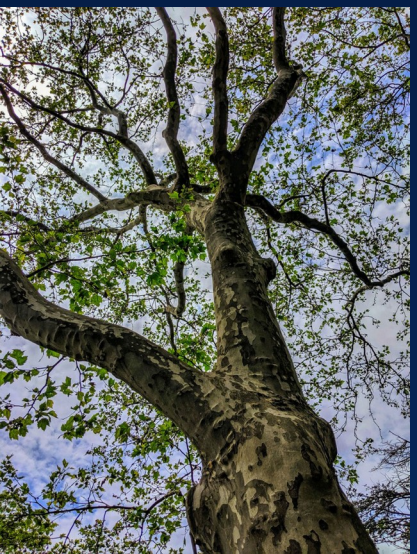


Figure 13.43

Manage an Asset Use Case (contd)

- Object **: Investment Class** does not play an active role in this collaboration diagram
  - This scenario does not involve an investment, only a mortgage
- Actor **Borrowers** does not play a role in this use case, either

Manage an Asset Use Case (contd)

- Sequence diagram equivalent to the collaboration diagram (of the realization of the scenario of the use case)

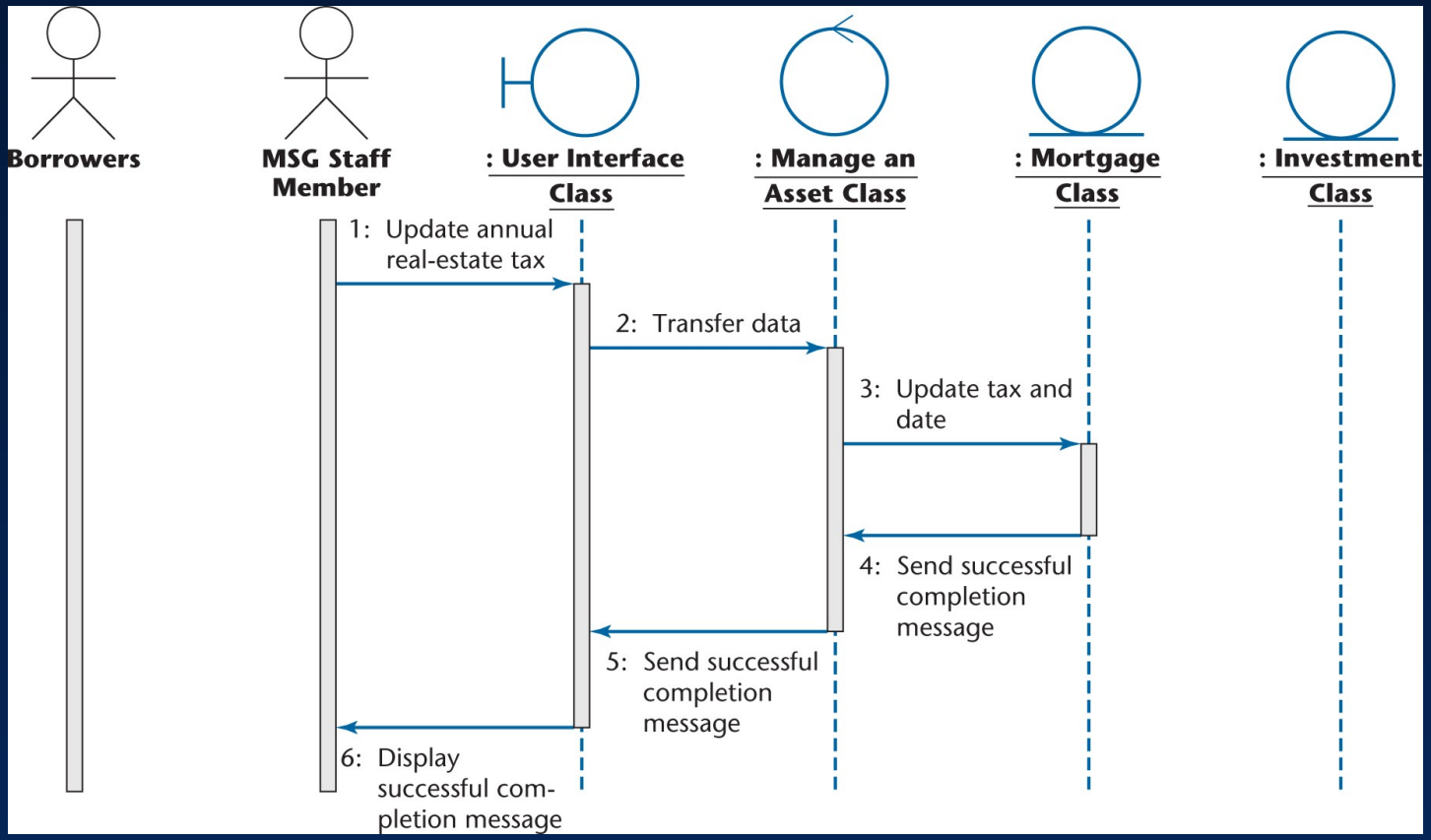


Figure 13.44

Manage an Asset Use Case (contd)

- A different scenario of the use case

There is a change in the weekly income of a couple who have borrowed money from the MSG Foundation. They wish to have their weekly income updated in the Foundation records by an MSG staff member so that their mortgage payments will be correctly computed.

1. The staff member enters the new value of the weekly income.
2. The information system updates the date on which the weekly income was last changed.

Figure 13.45

Manage an Asset Use Case (contd)

- Collaboration diagram of the realization of the scenario of the use case

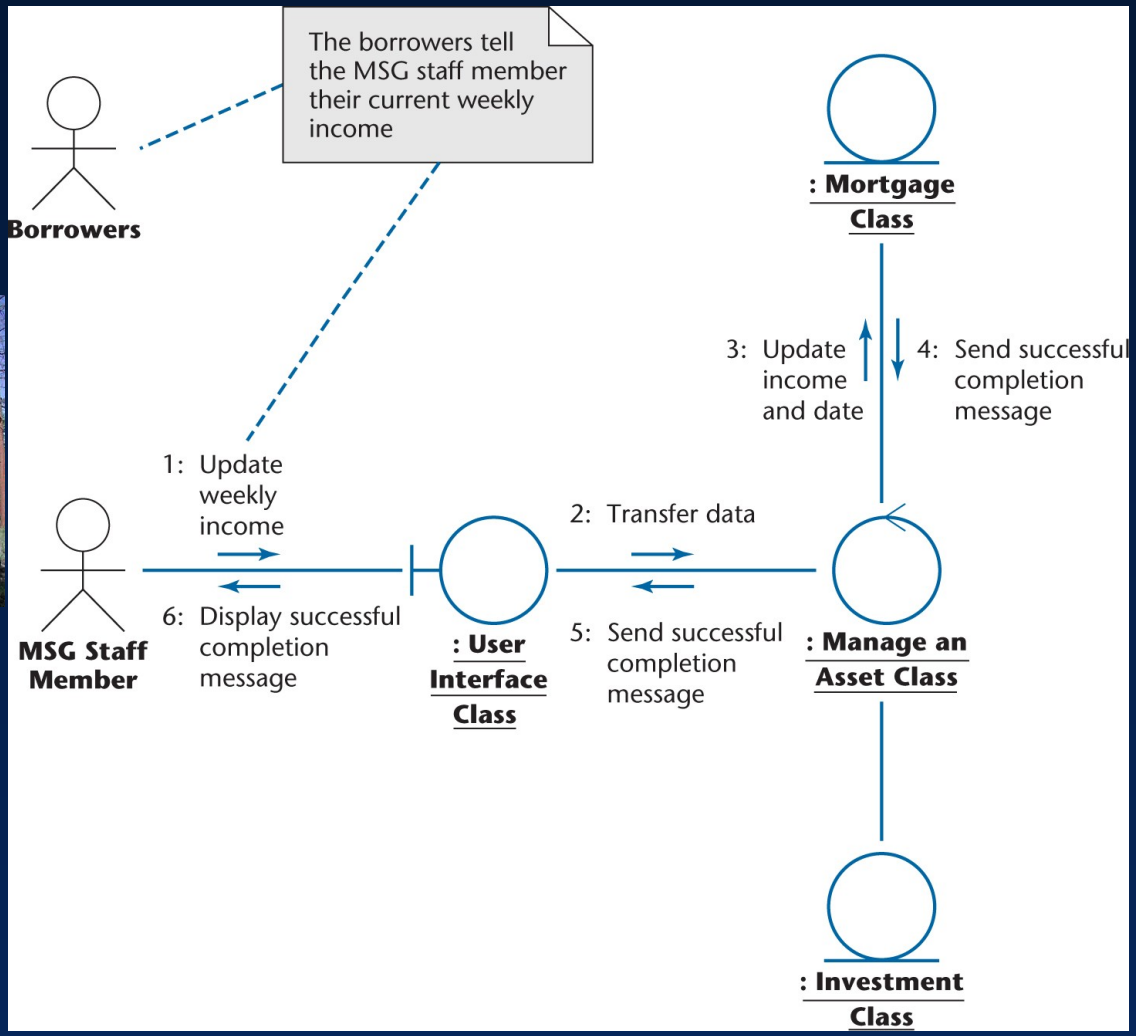


Figure 13.46

`Manage an Asset` Use Case (contd)

- At the request of the borrowers, the MSG staff member updates the weekly income of a couple
- The scenario is initiated by the **Borrowers**
- Their data are entered into the software product by the **MSG Staff Member**
  - This is stated in the note in the collaboration diagram

Manage an Asset Use Case (contd)

- Sequence diagram equivalent to the collaboration diagram (of the realization of the scenario of the use case)

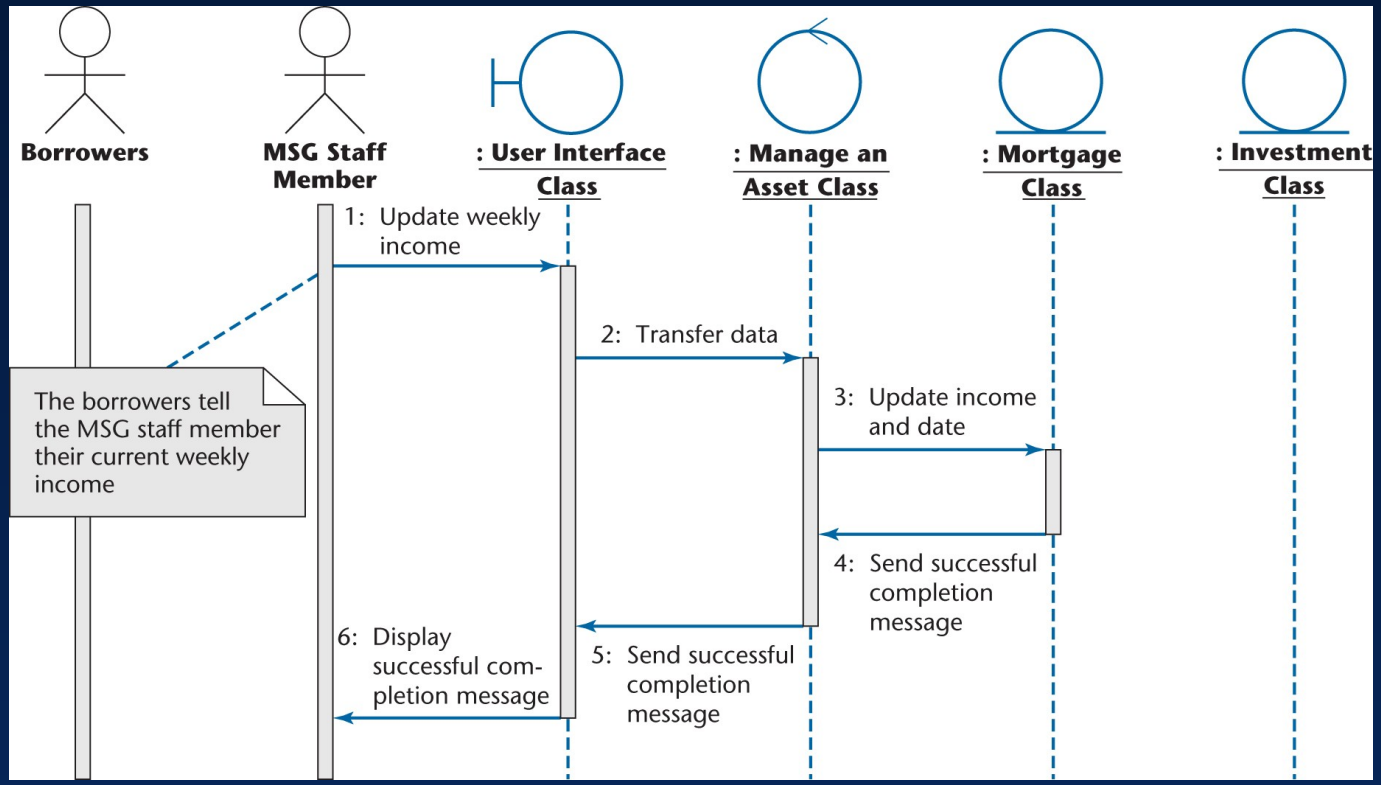


Figure 13.47

Manage an Asset Use Case (contd)

- Two different scenarios of the same use case have been presented
- The use case is the same
  - The class diagram is therefore the same
- However, the collaboration (and sequence) diagrams reflect the differences between the two scenarios

Manage an Asset Use Case (contd)

- Boundary class **User Interface Class** appears in all the realizations
  - The same screen will be used for all commands of the information system
- Revised menu

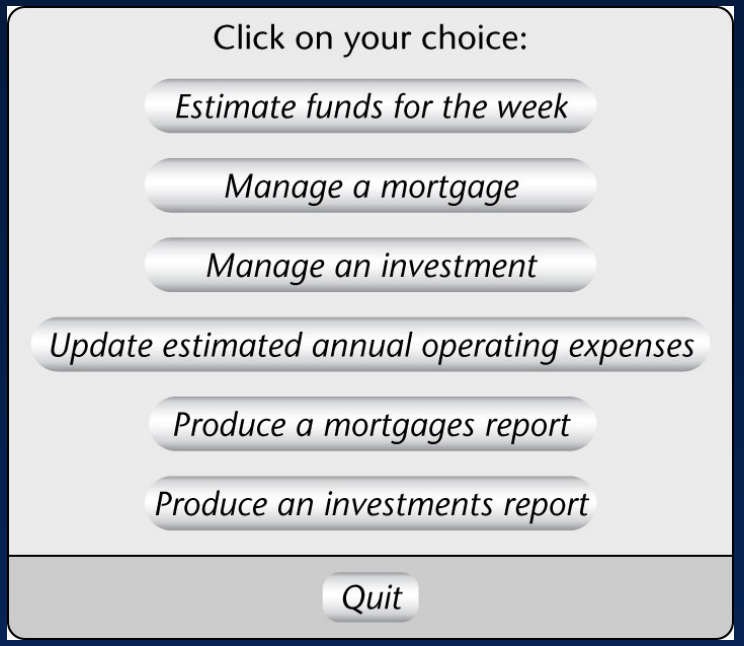


Figure 13.48

Manage an Asset Use Case (contd)

- Corresponding textual interface

```
                    MAIN MENU
MARTHA STOCKTON GREENGAGE FOUNDATION
   1. Estimate funds available for week
   2. Manage a mortgage
   3. Manage an investment
   4. Update estimated annual operating expenses
   5. Produce a mortgages report
   6. Produce an investments report
   7. Quit
Type your choice and press <ENTER>:
```

Figure 13.49

- Class diagram

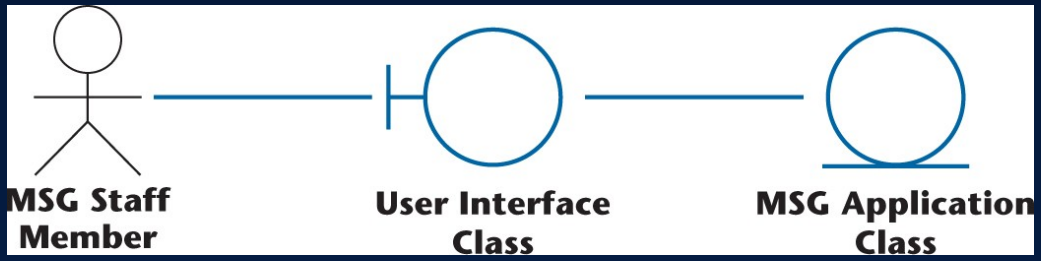


Figure 13.50

Update Annual Operating Expenses Use Case (contd)

- Collaboration diagram of a realization of a scenario of the use case

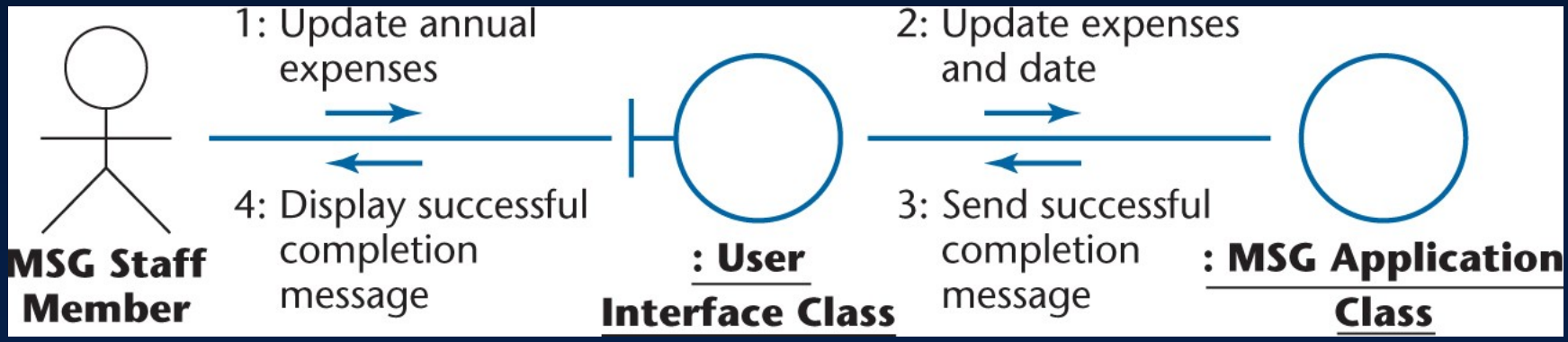


Figure 13.51

- Equivalent sequence diagram

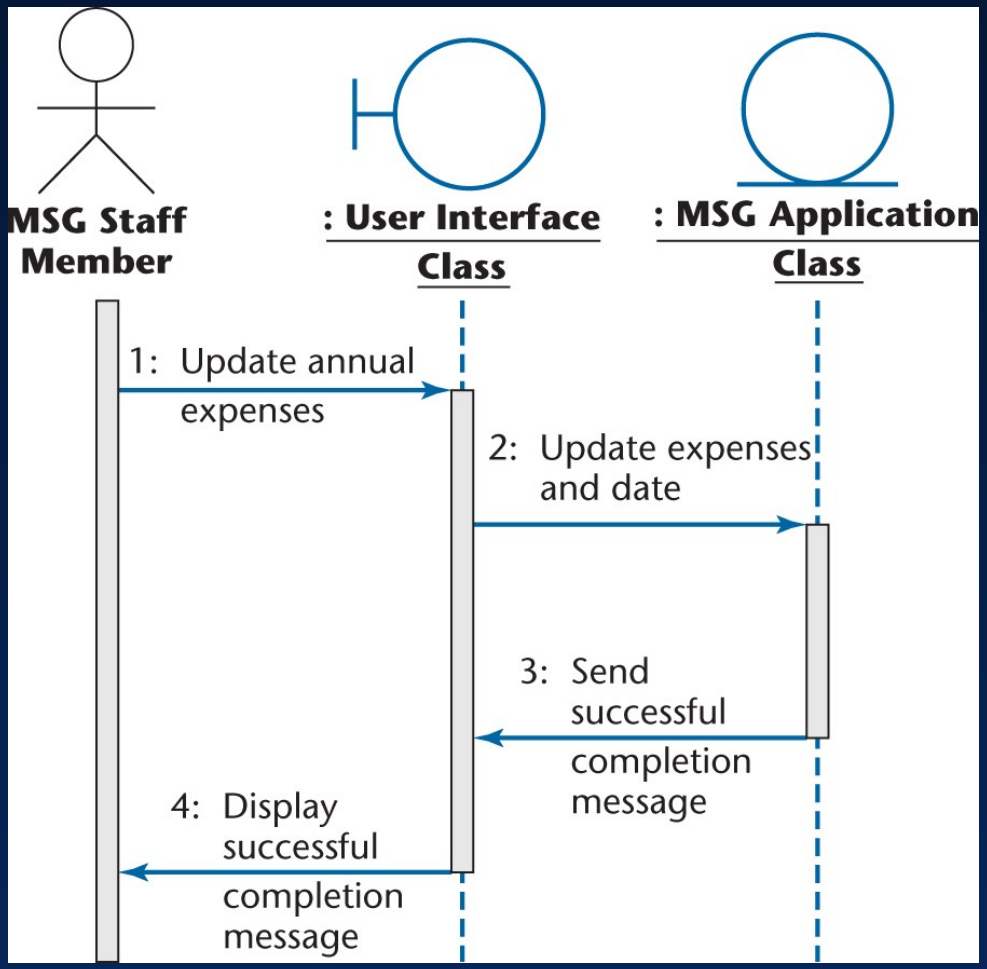


Figure 13.52

13.15.4 Produce a Report Use Case

- Use case

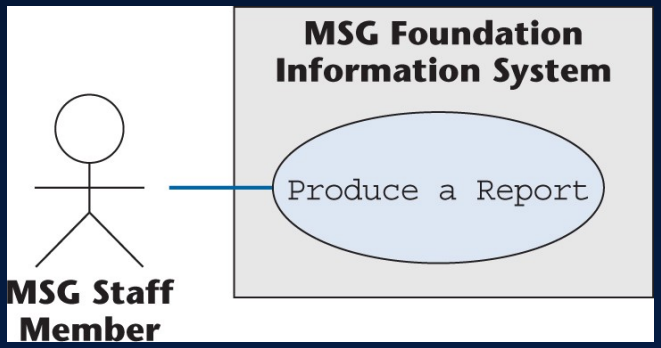


Figure 13.53

Produce a Report Use Case (contd)

- Description of use case

Brief Description

The Produce a Report use case enables an MSG Foundation staff member to print a listing of all investments or all mortgages.

Step-by-Step Description

1. The following reports must be generated:
   - 1.1 Investments report—printed on demand:
     The information system prints a list of all investments. For each investment, the following attributes are printed:
     - Item number
     - Item name
     - Estimated annual return
     - Date estimated annual return was last updated
   - 1.2 Mortgages report—printed on demand:
     The information system prints a list of all mortgages. For each mortgage, the following attributes are printed:
     - Account number
     - Name of mortgagees
     - Original price of home
     - Date mortgage was issued
     - Principal and interest payment
     - Current combined gross weekly income
     - Date current combined gross weekly income was last updated
     - Annual real-estate tax
     - Date annual real-estate tax was last updated
     - Annual homeowner's insurance premium
     - Date annual homeowner's insurance premium was last updated

Figure 13.54

Produce a Report Use Case (contd)

- Class diagram

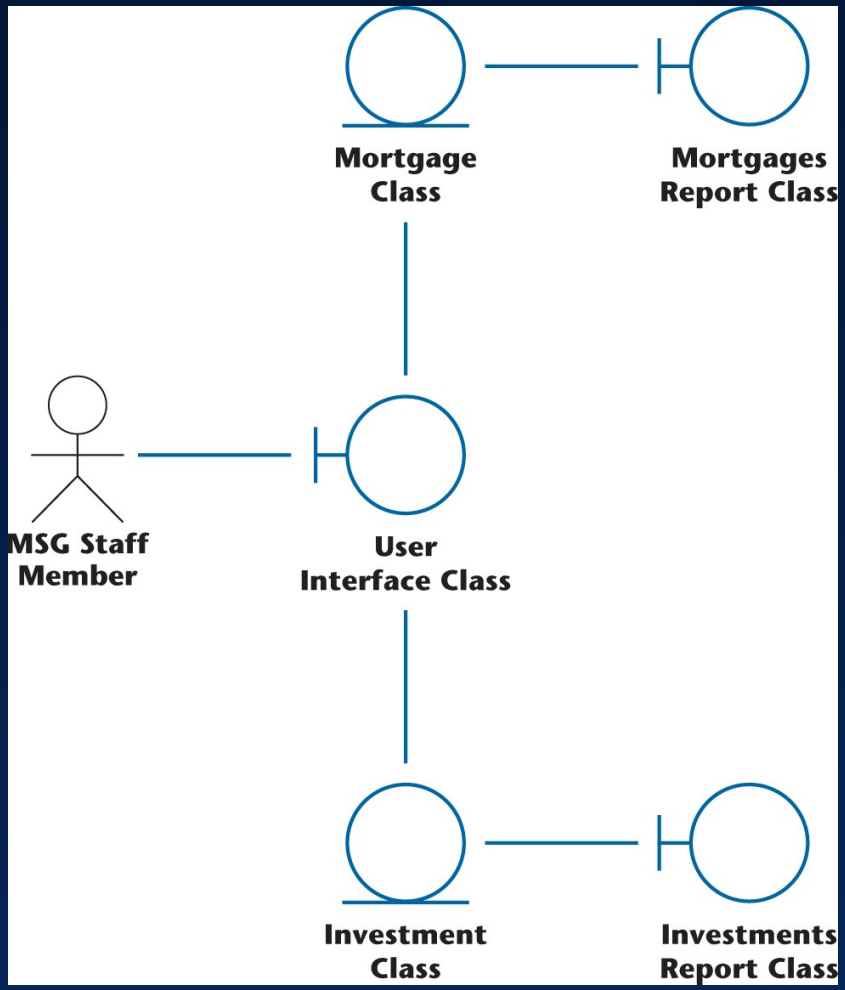


Figure 13.55

Produce a Report Use Case (contd)

- One scenario of the use case

An MSG staff member wishes to print a list of all mortgages.

1. The staff member requests a report listing all mortgages.

Figure 13.56

Produce a Report Use Case (contd)

- Collaboration diagram
  - Mortgages (but not investments) are involved

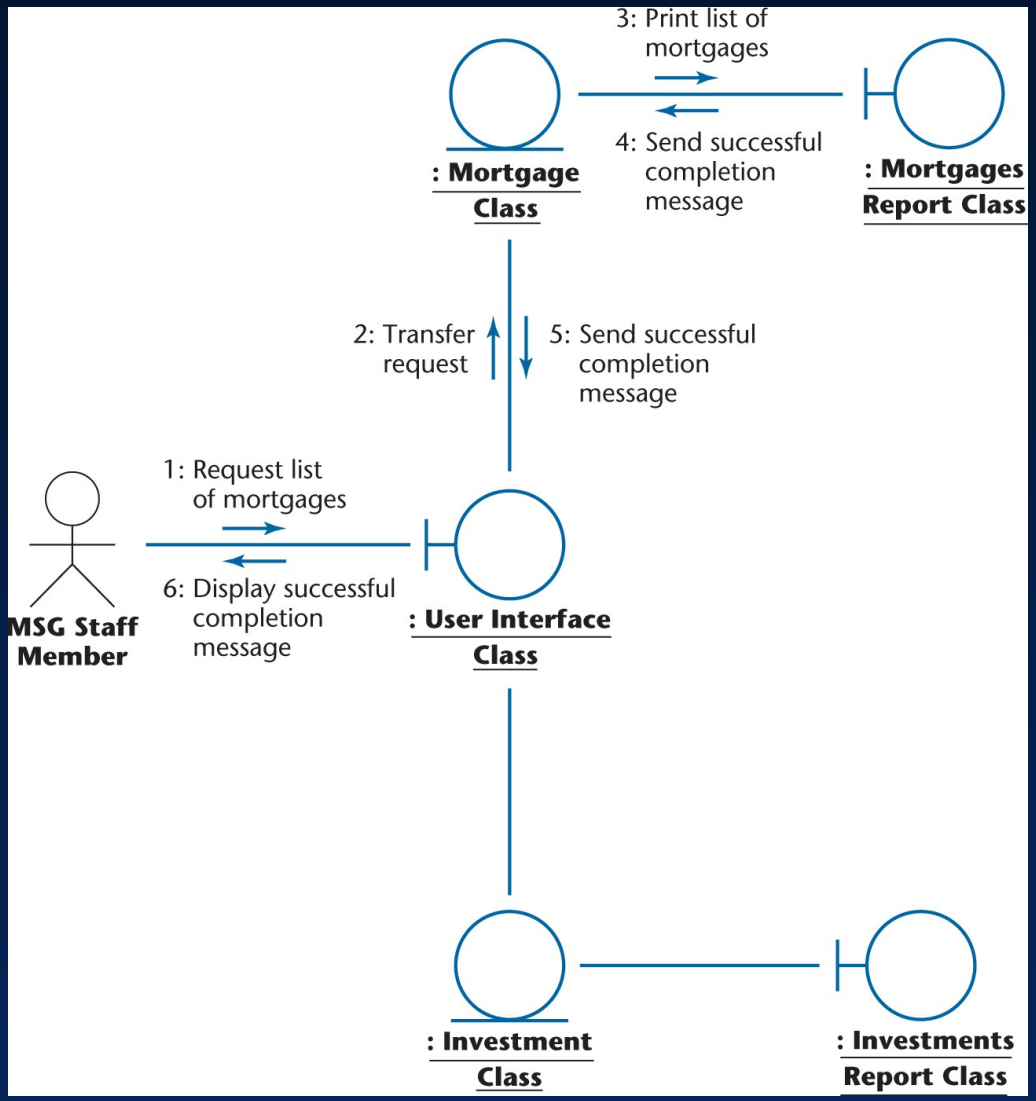


Figure 13.57

Produce a Report Use Case (contd)

- Sequence diagram

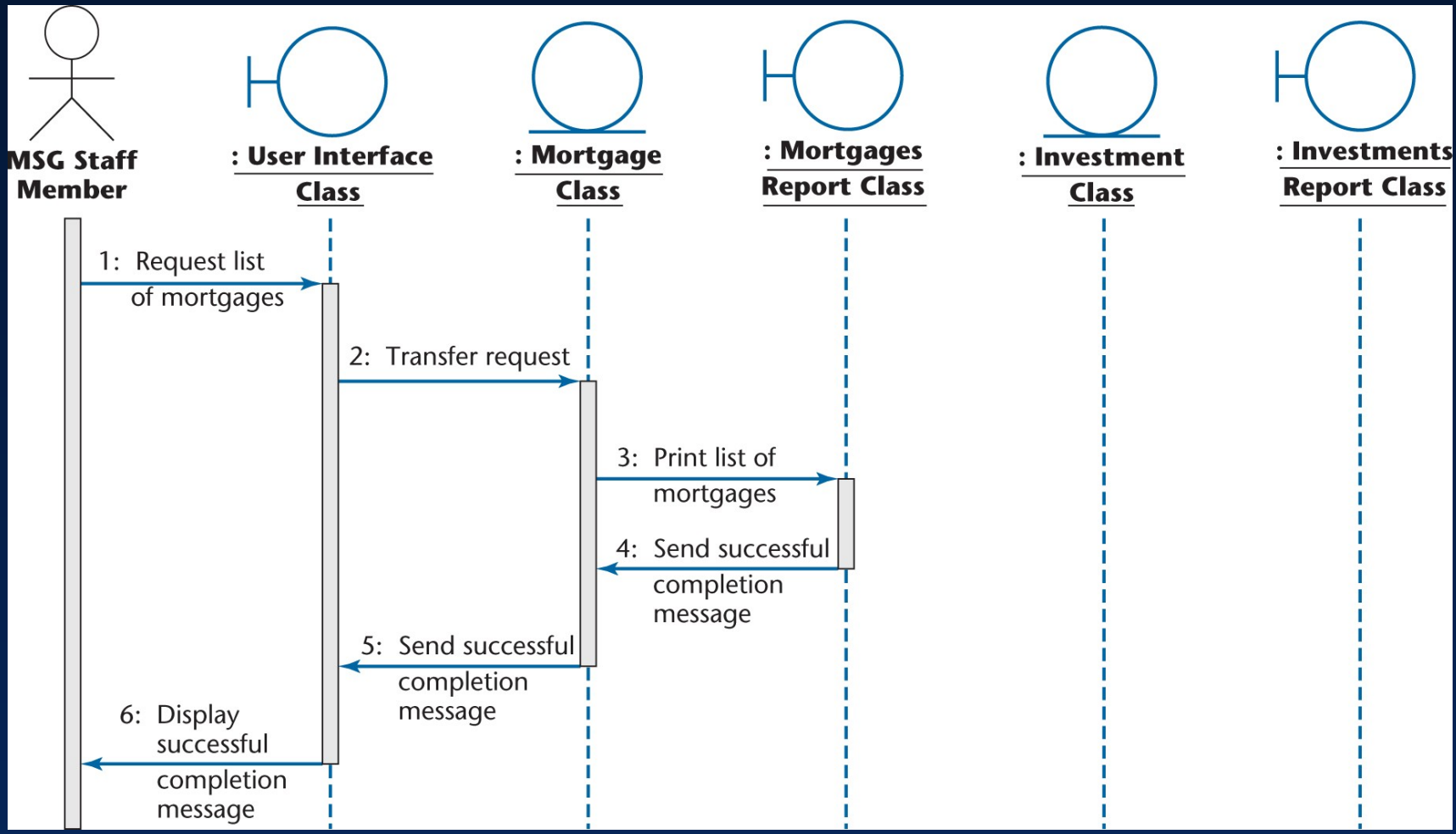


Figure 13.58

Produce a Report Use Case (contd)

- A second scenario (listing all investments) of the use case

An MSG staff member wishes to print a list of all investments.

1. The staff member requests a report listing all investments.

Figure 13.59

Produce a Report Use Case (contd)

- Collaboration diagram for second scenario
  - This time, investments (but not mortgages) are involved

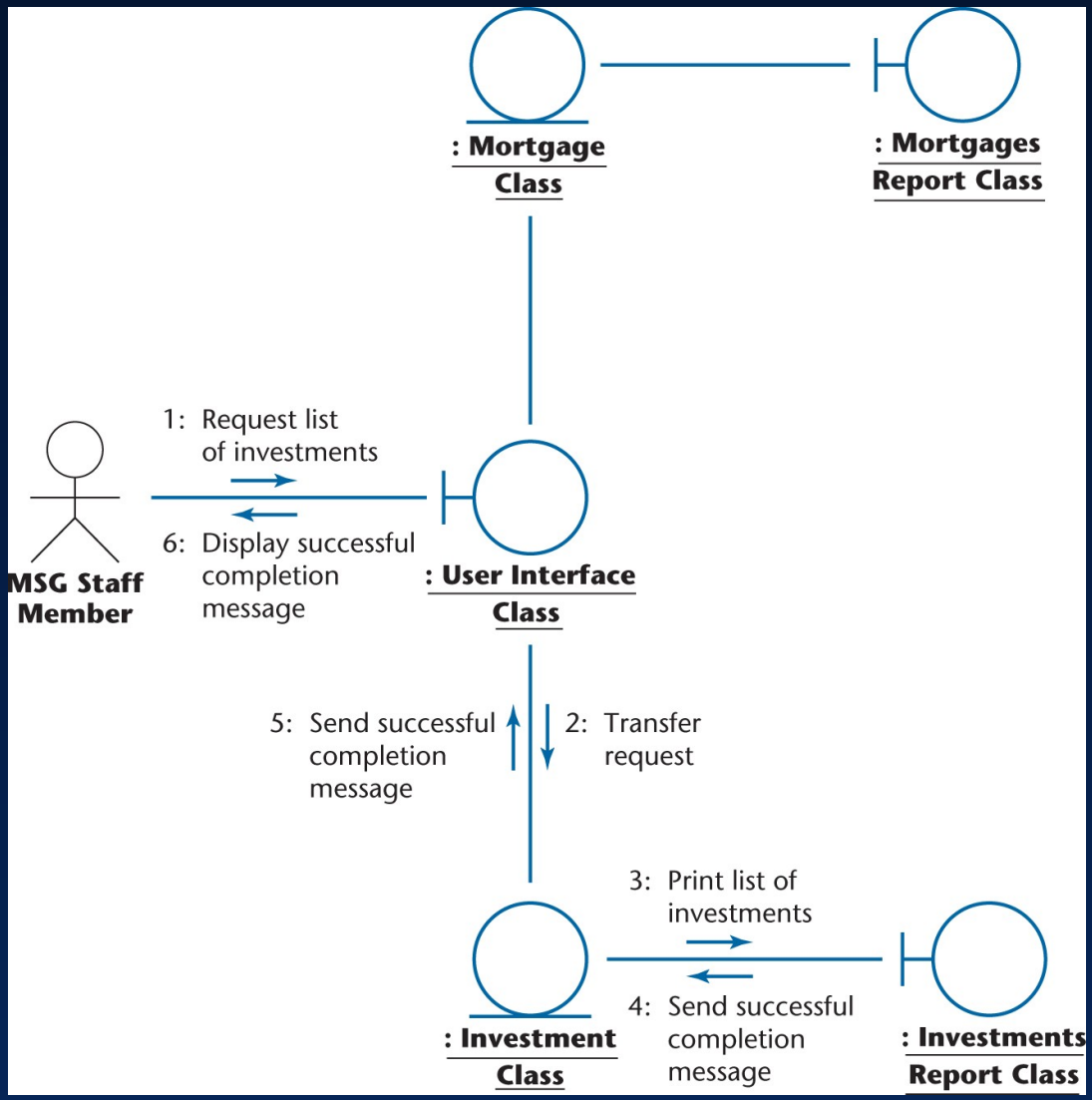


Figure 13.60

Produce a Report Use Case (contd)

- Sequence diagram for second scenario

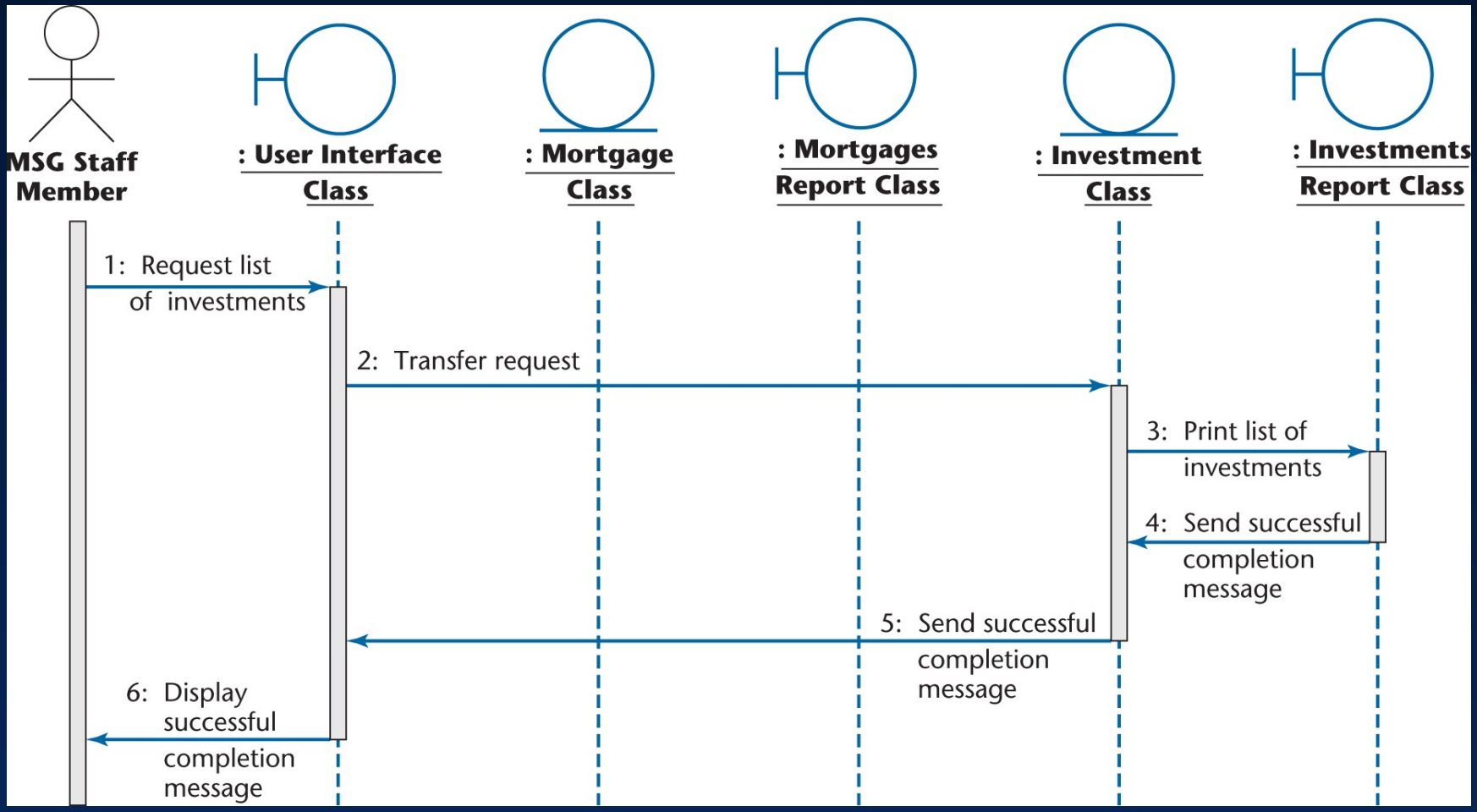


Figure 13.61

13.16 Incrementing the Class Diagram: The MSG Foundation

- In the course of realizing the various use cases
  - Interrelationships between classes become apparent
- Accordingly, we now combine the realization class diagrams

Combining the Realization Class Diagrams

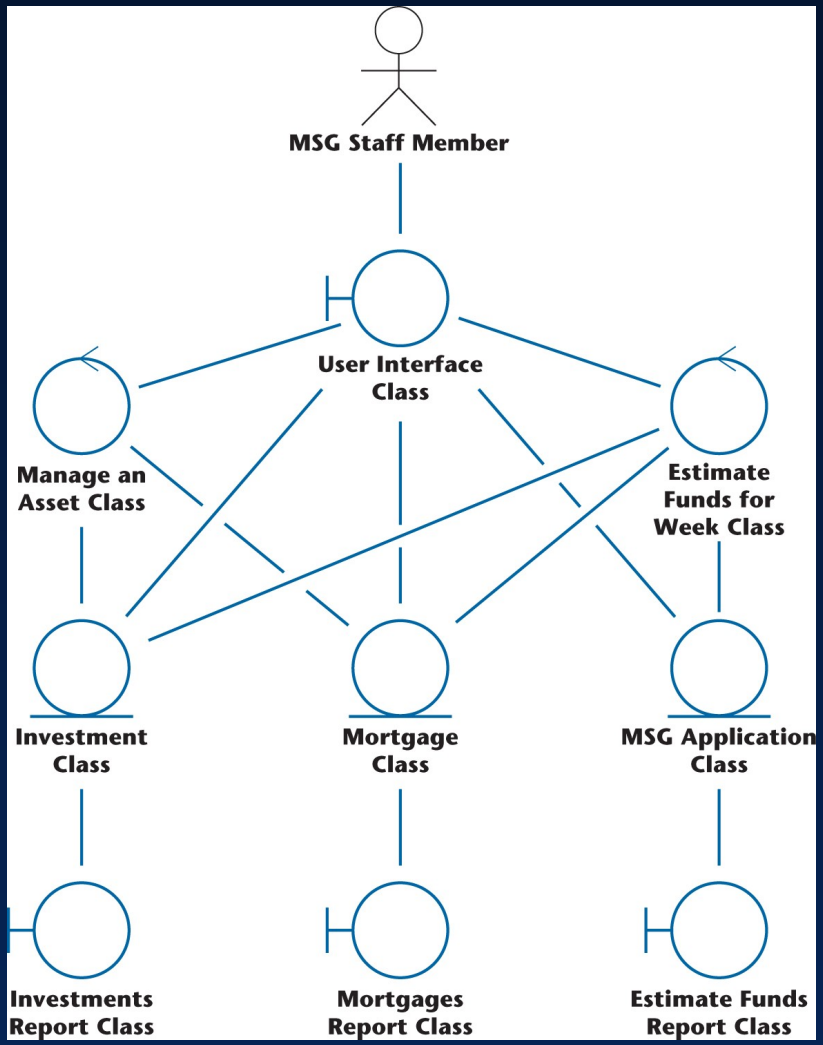
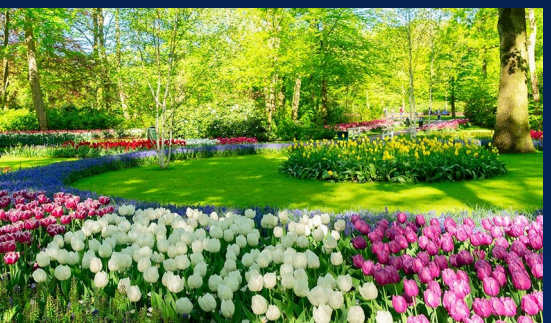


Figure 13.62



Fourth Iteration of the Class Diagram

- Fifth iteration + realization class diagram

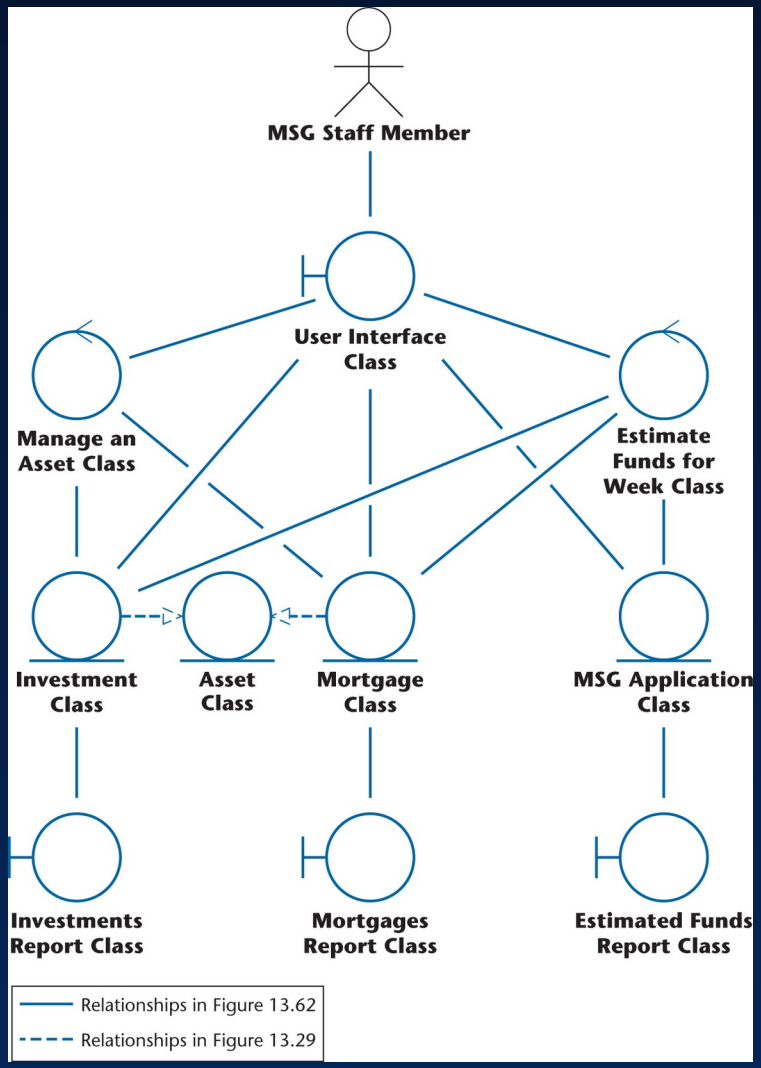


Figure 13.63

How to Perform Object-Oriented Analysis

How to Perform Object-Oriented Analysis — Box 13.1

- **Iterate**

  Perform functional modeling.
  Perform entity class modeling.
  Perform dynamic modeling.

- **Until** the entity classes have been satisfactorily extracted.
- Extract the boundary classes and control classes.
- Refine the use cases.
- Perform use-case realization.

Exercises

- Please give the flow of events of the interaction diagrams of Figure 13.46 and 13.47.

Exercise

Please draw sequence diagrams for the following system:

Consider an automated library circulation system. Every book has a bar code, and every borrower has a card bearing a bar code. When a borrower wishes to check out a book, the librarian scans the bar codes on the book and the borrower's card, and enters C at the computer terminal. Similarly, when a book is returned, it is again scanned and the librarian enters R. Librarians can add books (+) to the library collection or remove them (−). Borrowers can go to a terminal and determine all the books in the library by a particular author (the borrower enters A= followed by the author's name), all the books with a specific title (T= followed by the title), or all the books in a particular subject area (S= followed by the subject area). Finally, if a borrower wants a book currently checked out, the librarian can place a hold on the book so that, when it is returned, it will be held for the borrower who requested it (H= followed by the number of the book).