Object-Oriented Software Engineering

WCB/McGraw-Hill, 2008

THE ANALYSIS WORKFLOW

Overview

- The specification document
- Informal specifications
- Correctness proof mini case study redux
- The analysis workflow
- Extracting the entity classes
- The elevator problem
- Functional modeling: The elevator problem case study
- Entity class modeling: The elevator problem case study

Overview (contd)

- Dynamic modeling: The elevator problem case study
- The test workflow: The elevator problem case study
- Extracting the boundary and control classes
- The initial functional model: The MSG Foundation case study
- The initial class diagram: The MSG Foundation case study

Overview (contd)

- The initial dynamic model: The MSG Foundation case study
- Revising the entity classes: The MSG Foundation case study
- Extracting the boundary classes: The MSG Foundation case study
- Extracting the control classes: The MSG Foundation case study
- Use-case realization: The MSG Foundation case study

Overview (contd)

- Incrementing the class diagram: The MSG Foundation case study
- The software project management plan: The MSG Foundation case study
- The test workflow: The MSG Foundation case study
- The specification document in the Unified Process
- More on actors and use cases
- CASE tools for the analysis workflow
- Challenges of the analysis workflow

The Specification Document Must Be

- Informal enough for the client
  - The client is generally not a computer specialist
- Formal enough for the developers
  - It is the sole source of information for drawing up the design
- These two requirements are mutually contradictory

11.1 The Specification Document

- The specification document is a contract between the client and the developers
- Typical constraints
  - Deadline
  - Parallel running
  - Portability
  - Reliability
  - Rapid response time
- For real-time software
  - Hard real-time constraints must be satisfied

Specification Document (contd)

- Acceptance criteria
  - It is vital to spell out a series of tests
- If the product passes the tests, it is deemed have satisfied its specifications
- Some acceptance criteria are restatements of constraints

Solution Strategy

- A general approach to building the product
- Find strategies without worrying about constraints
  - Then modify the strategies in the light of the constraints, if necessary
- Keep a written record of all discarded strategies, and why they were discarded
  - To protect the analysis team
  - To prevent unwise new “solutions” during postdelivery maintenance

11.2 Informal Specifications

- Informal specifications are written in a natural language
  - Examples: English, Mandarin, Kiswahili, Hindi

- Example

  "If the sales for the current month are below the target sales, then a report is to be printed, unless the difference between target sales and actual sales is less than half of the difference between target sales and actual sales in the previous month, or if the difference between target sales and actual sales for the current month is under 5%"

The Meaning of This Specification

- The sales target for January was $100,000, but actual sales were only $64,000 (36% below target)
  - Print the report
- The sales target for February was $120,000, the actual sales were only $100,000 (16.7% below target)
  - The percentage difference for February (16.7%) is less than half of the previous month's percentage difference (36%), so do not print the report

The Meaning of This Specification (contd)

- The sales target for March was $100,000, the actual sales were $98,000 (2% below target)
  - The percentage difference is under 5%, so do not print the report

But the Specifications Do Not Say This

- "[D]ifference between target sales and actual sales"
  - There is no mention of percentage difference in the specifications
- The difference in January was $36,000, the difference in February was $20,000
  - Not less than half of $36,000, so the report is printed
- "[D]ifference … [of] 5%"
  - Again, no mention of percentage

But the Specifications Do Not Say This (contd)

- Ambiguity—should the last clause read "percentage difference … [of] 5%" or "difference … [of] $5,000" or something else entirely?

- The style is poor
  - The specifications should state when the report should be printed …
  - … Rather than when it should not be printed

Informal Specifications (contd)

- Claim
  - This cannot arise with professional specifications writers
- Refutation
  - Text processing case study

11.3 Correctness Proof Case Study

- Naur text-processing problem

  Given a text consisting of words separated by `blank` or by `newline` characters, convert it to line-by-line form in accordance with the following rules:

  (1) Line breaks must be made only where the given text contains a `blank` or `newline`;

  (2) Each line is filled as far as possible, as long as

  (3) No line will contain more than `maxpos` characters

Episode 1

- 1969 — Naur Paper
- Naur constructed a procedure (25 lines of Algol 60), and informally proved its correctness

Episode 2

- 1970 — Reviewer in *Computing Reviews*
  - The first word of the first line is preceded by a `blank` unless the first word is exactly `maxpos` characters long

Episode 3

- 1971 — London found 3 more faults
  - Including: The procedure does not terminate unless a word longer than `maxpos` characters is encountered

Episode 4

- 1975 — Goodenough and Gerhart found 3 further faults
  - Including: The last word will not be output unless it is followed by a `blank` or `newline`
- Goodenough and Gerhart then produced a new set of specifications, about four times longer than Naur's

Episode 5

- 1985 — Meyer detected 12 faults in Goodenough and Gerhart's specifications

Episode 5

- Goodenough and Gerhart's specifications
  - Were constructed with the greatest of care
  - Were constructed to correct Naur's specifications
  - Went through two versions, carefully refereed
  - Were written by experts in specifications,
  - With as much time as they needed,
  - For a product about 30 lines long

- So, what chance do we have of writing fault-free specifications for a real product?

Episode 6

- 1989 — Schach found a fault in Meyer's specifications
  - Item (2) of Naur's original requirement ("each line is filled as far as possible") is not satisfied

Informal Specifications (contd)

- Conclusion
  - Natural language is *not* a good way to specify a product

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

11.4 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 11.1

11.5 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*

11.6 The Elevator Problem

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

11.7 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`

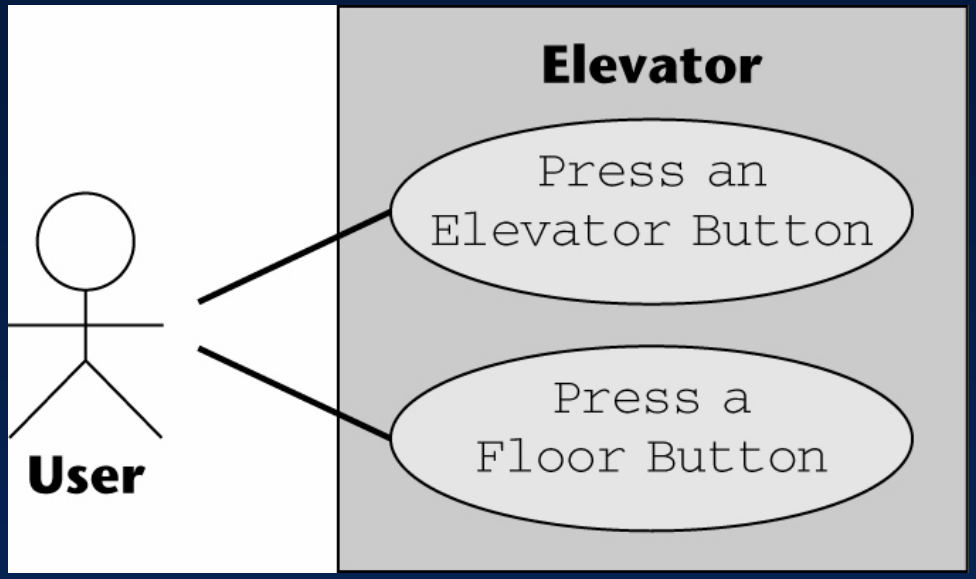

Figure 11.2

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 11.3

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 11.4

11.8 Entity Class Modeling : The Elevator Problem Case Study

- 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

11.8.1 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

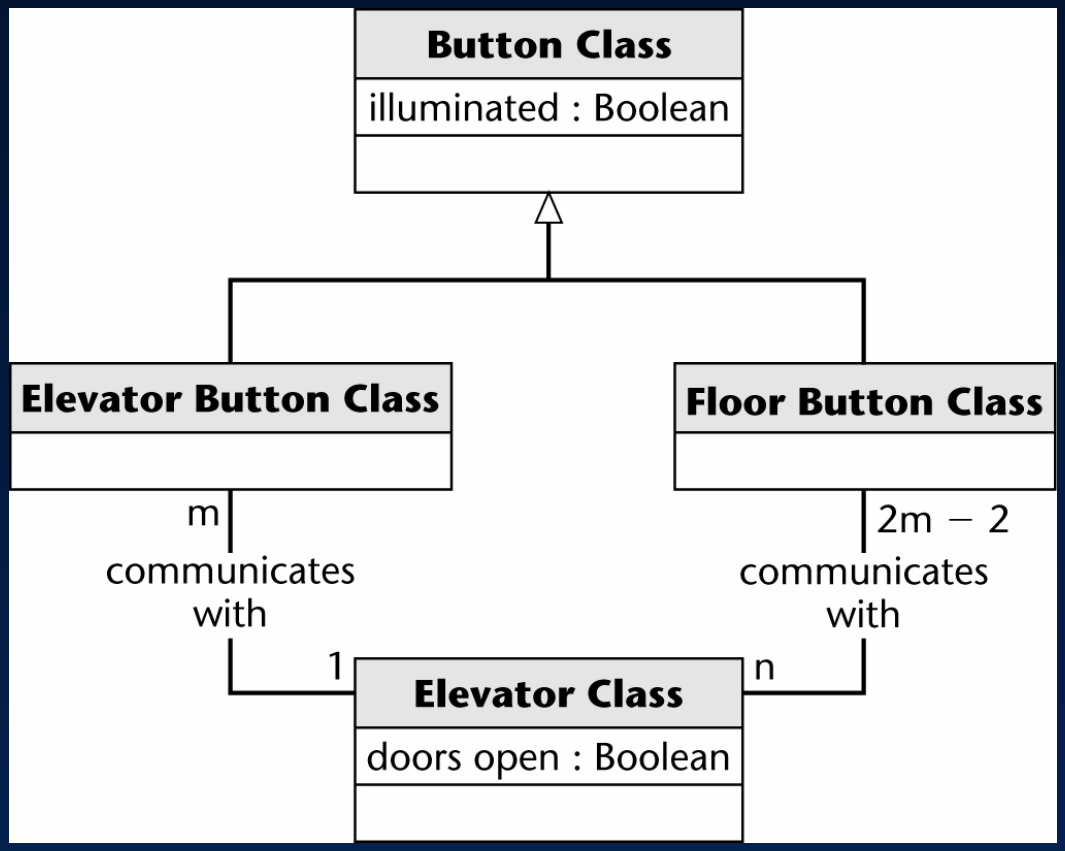

Figure 11.5

- 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

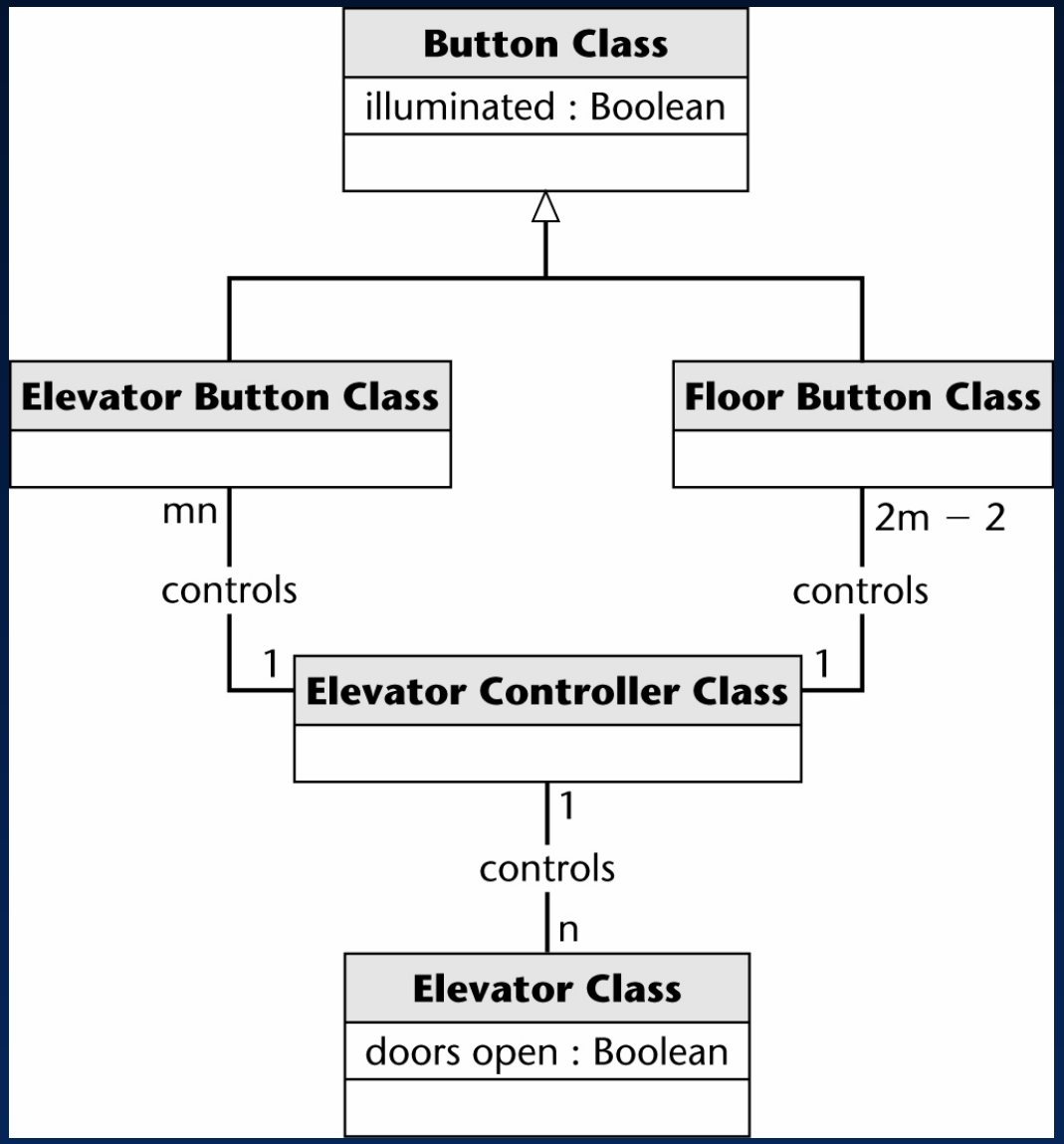

Figure 11.6

11.8.2 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 (Communication)
- Now CRC cards are automated (CASE tool component)

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

11.9 Dynamic Modeling: The Elevator Problem Case Study

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

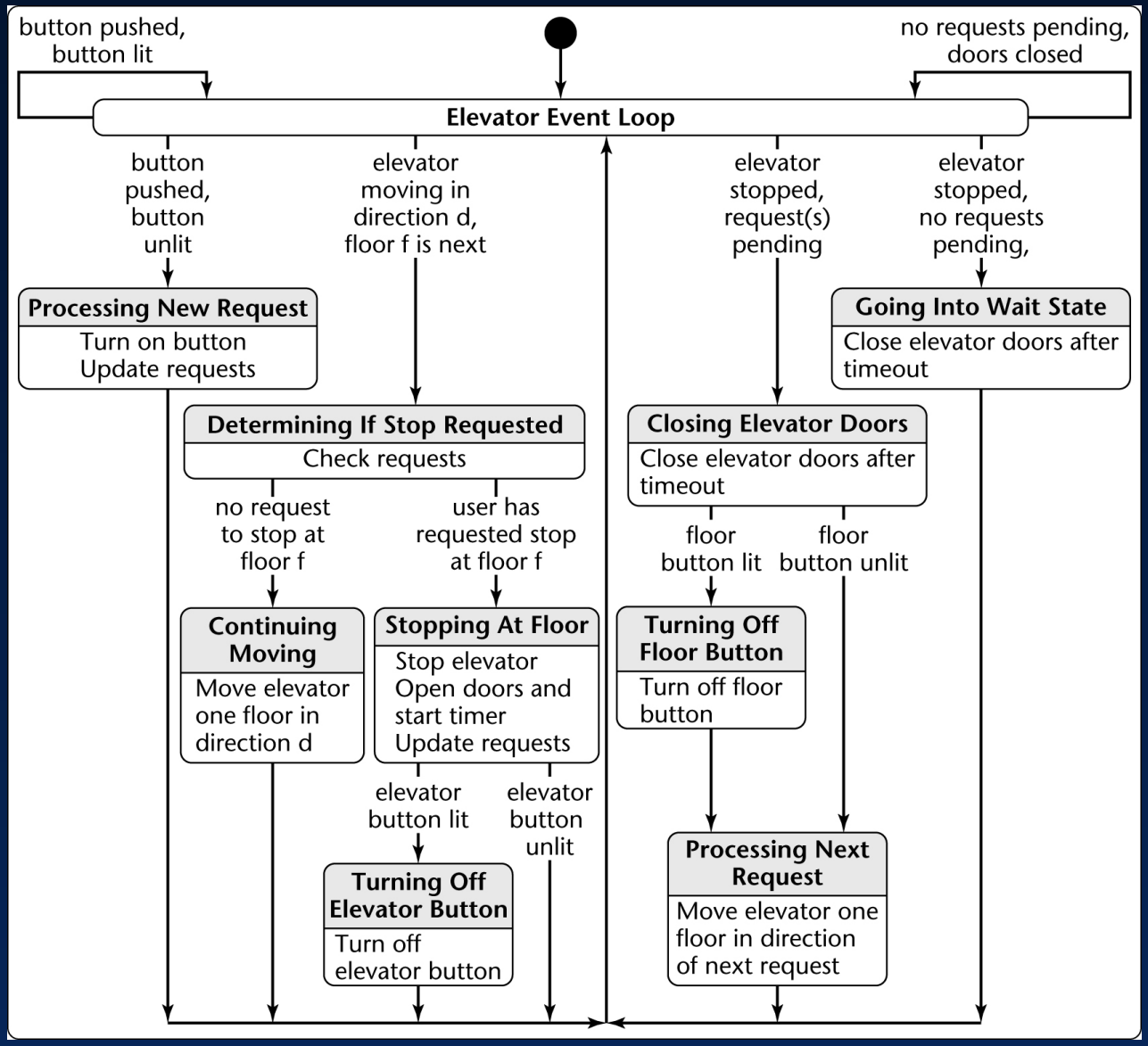

Figure 11.7

Dynamic Modeling: Elevator Problem (contd)

- This UML statechart is equivalent to the state transition diagram of Figures 11.15 through 11.17
- This is shown by considering specific scenarios
- In fact, a statechart is constructed by modeling the events of the scenarios

11.10 The Test Workflow: Elevator Problem

- 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 11.8

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 Class** to turn itself on`

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 11.9

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

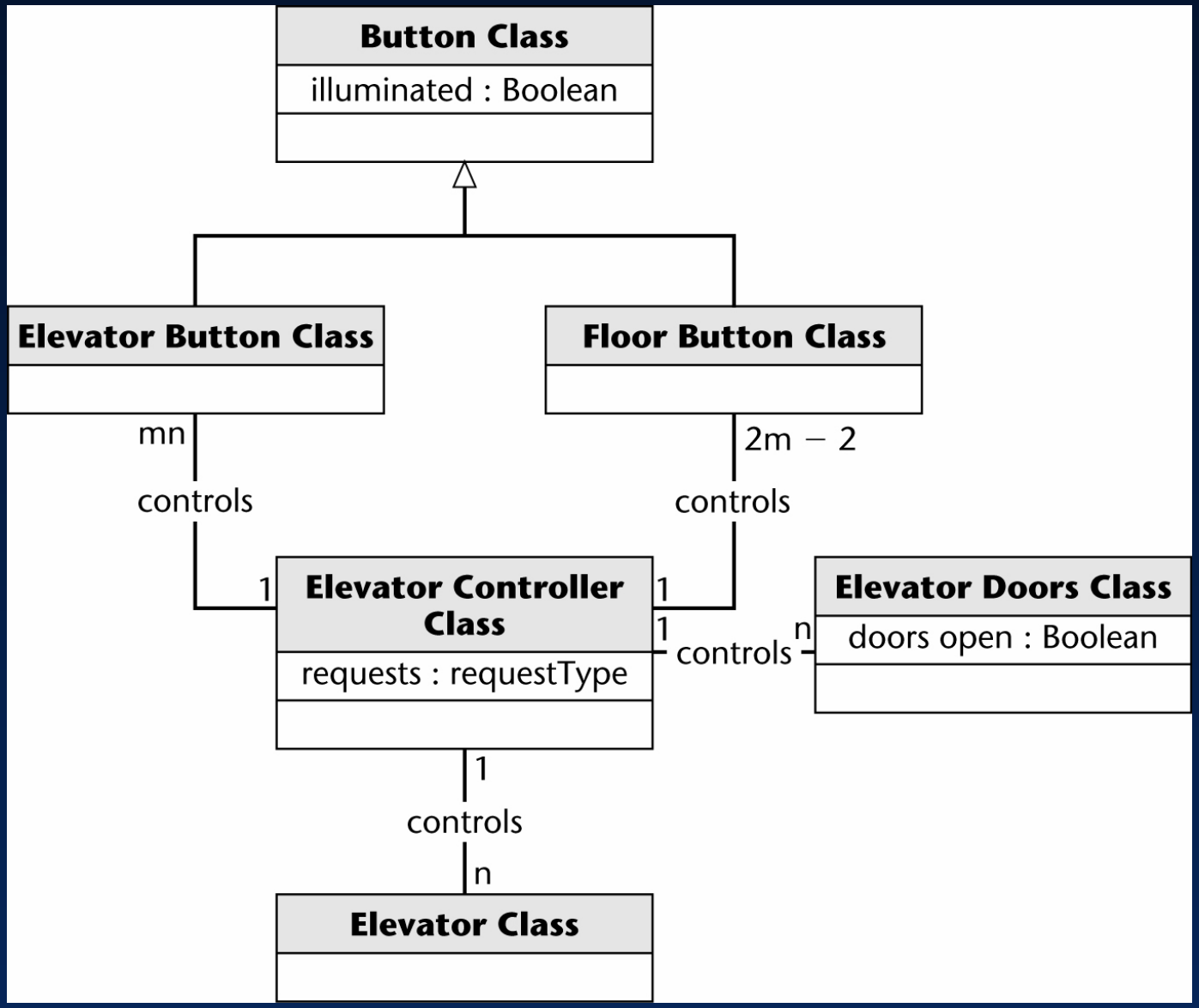

Figure 11.10

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 11.11

The Analysis Workflow: Elevator Problem (contd)

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

11.11 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

11.12 The Initial Functional Model: MSG Foundation

- Recall the seventh iteration of the use-case diagram

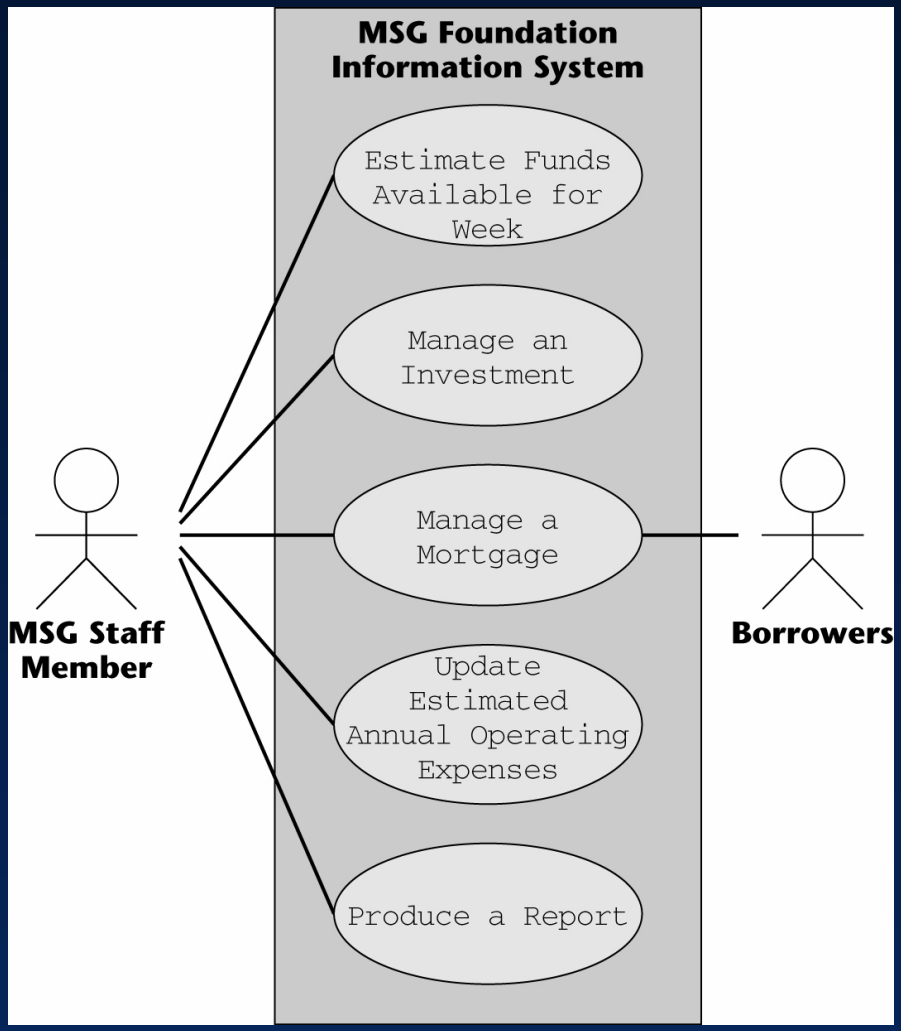

Figure 11.12

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 software product updates the date on which the annual real-estate tax was last changed.

Possible alternatives:

A. The staff member enters the mortgage number incorrectly.

Figure 11.13

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 software product 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 11.14

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 software product 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 software product then extracts the estimated annual MSG Foundation operating expenses and divides the result by 52.
3. For each mortgage:
   - 3.1 The software product 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 software product 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 software product 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 11.15

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 11.16

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 11.17

11.13 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 software product 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

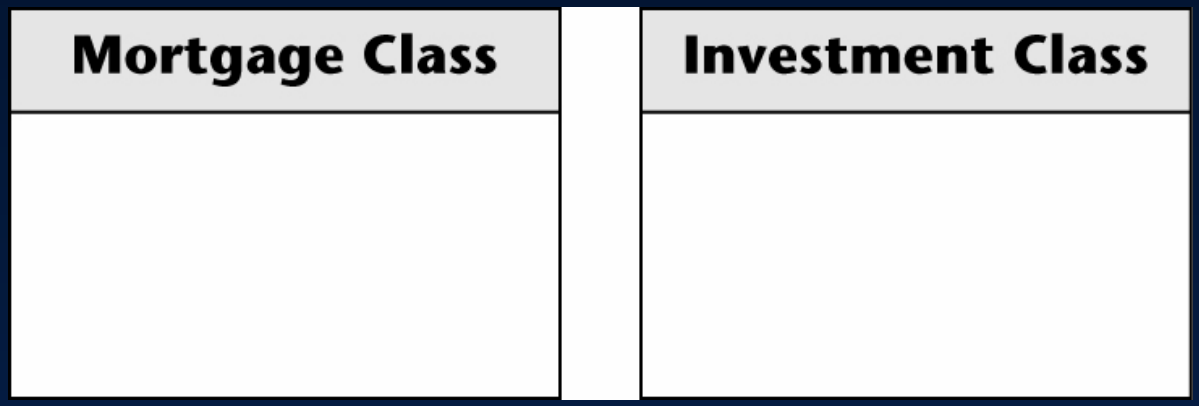

Figure 11.18

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)

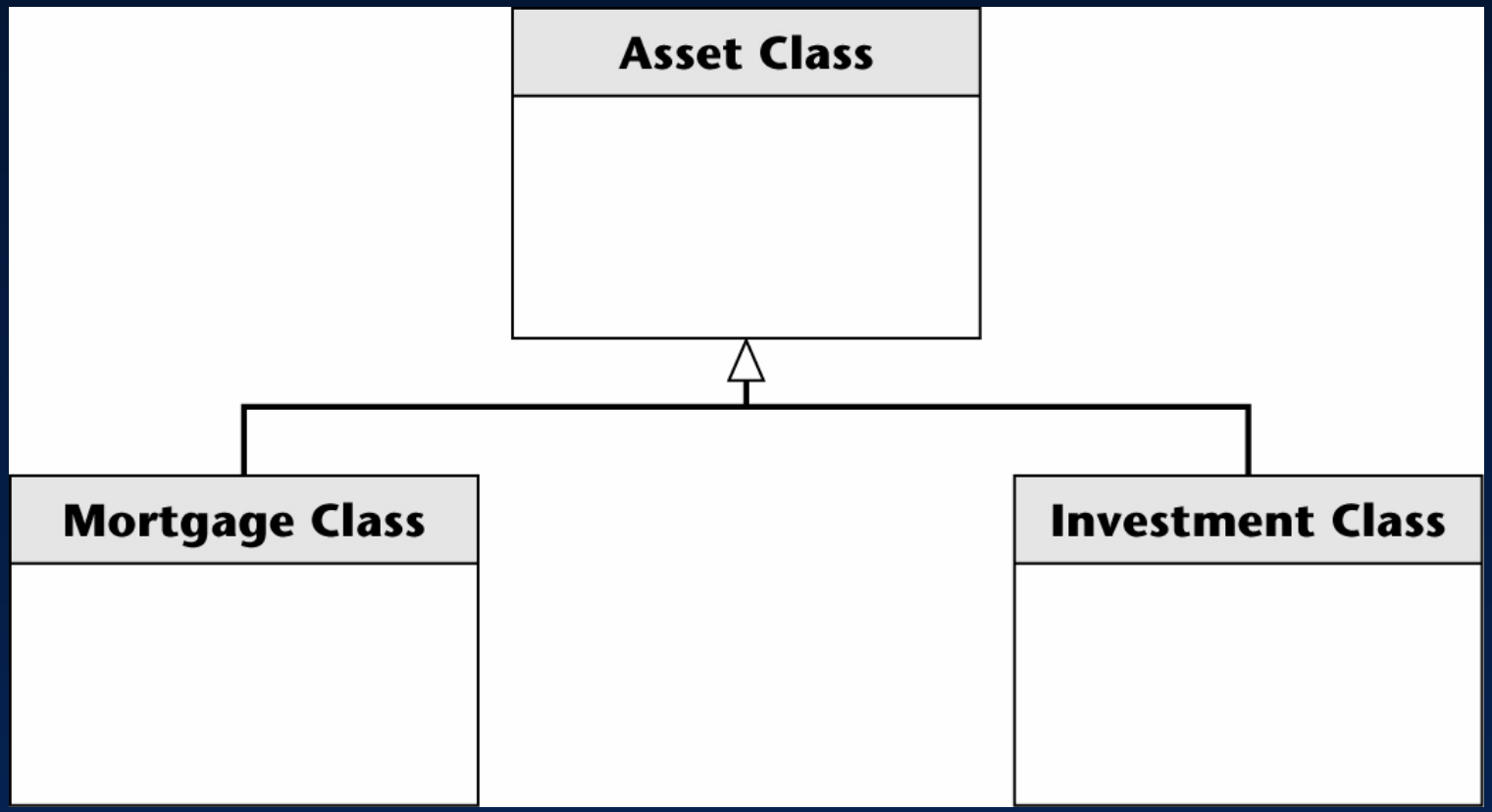

Figure 11.19

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

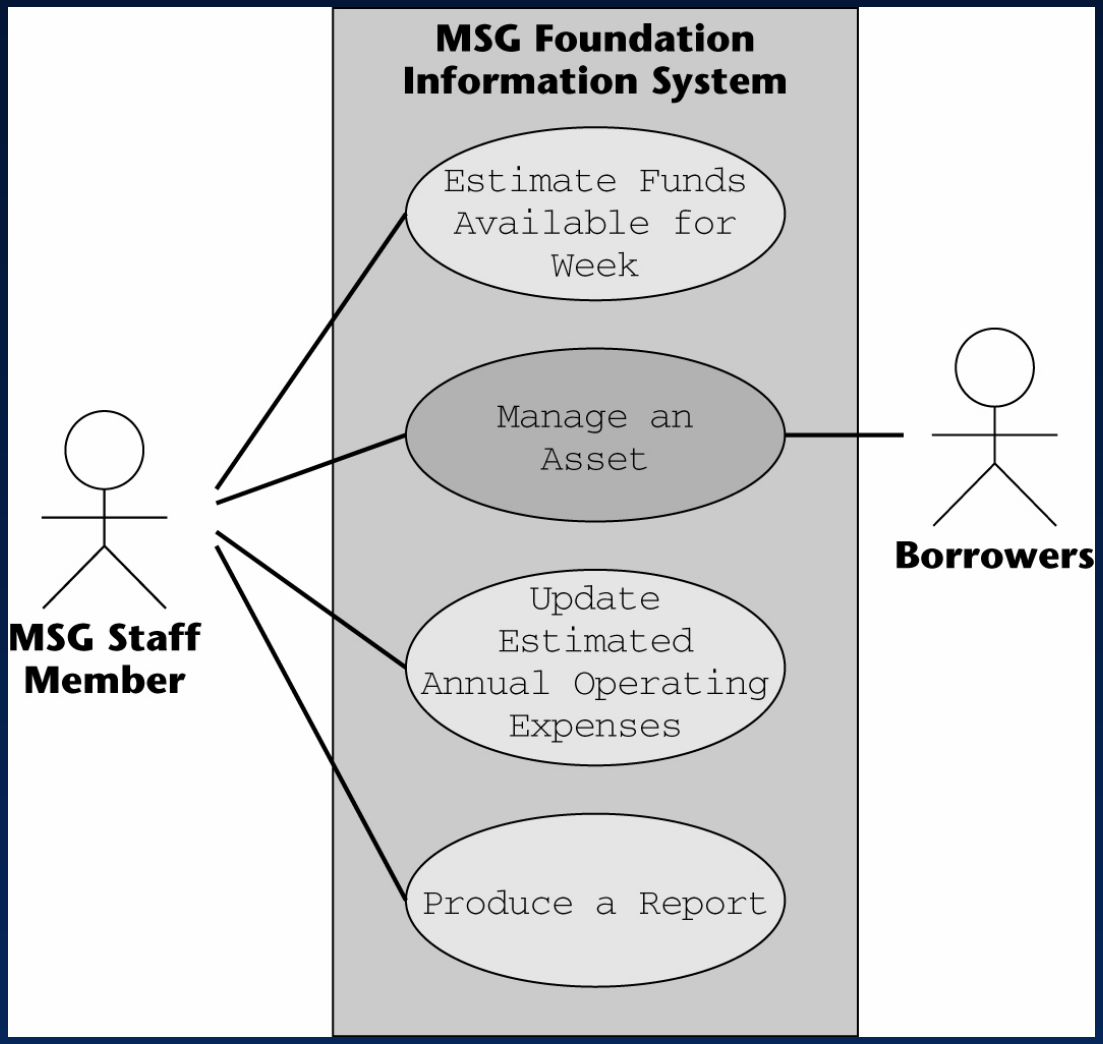

Figure 11.20

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)

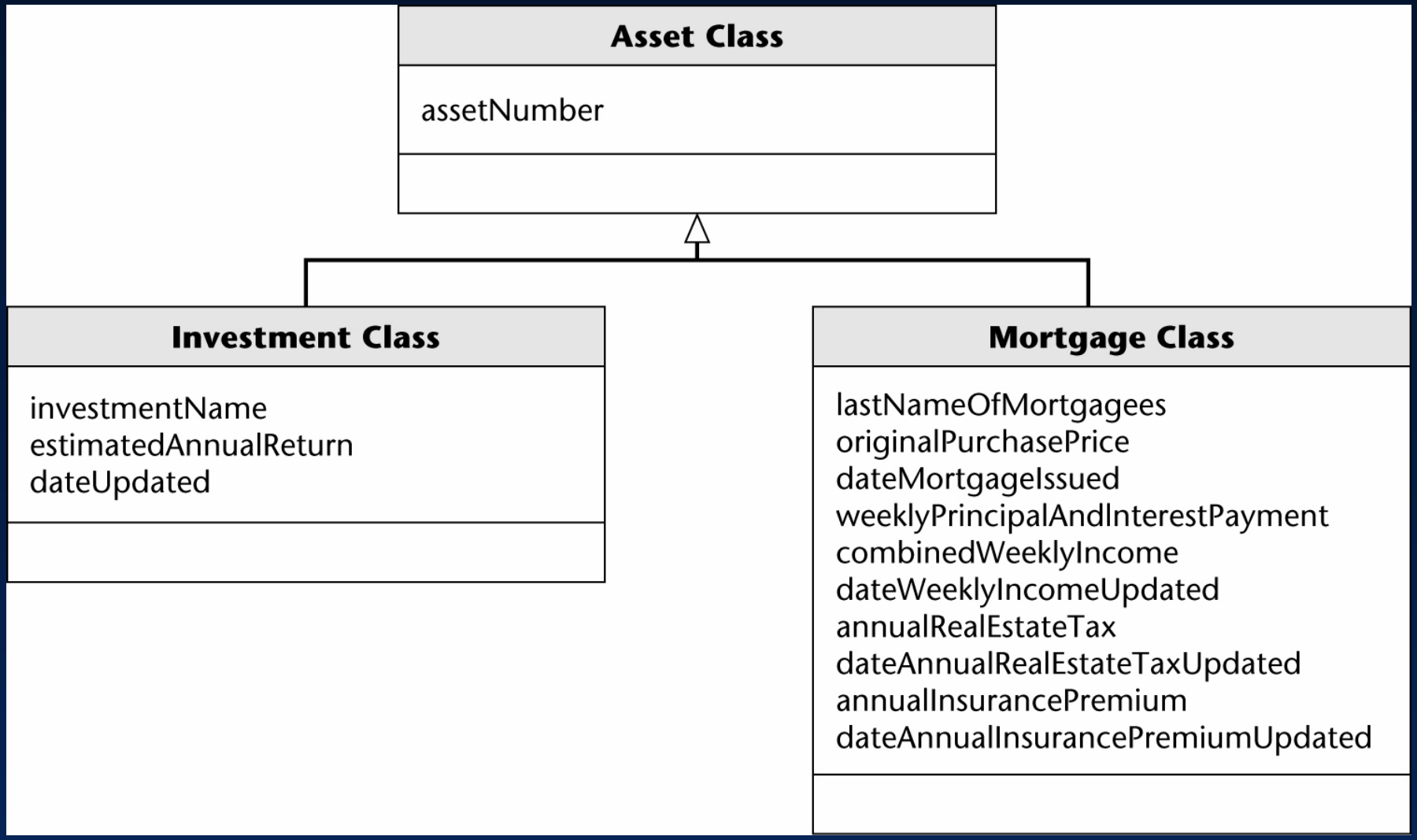

Figure 11.21

Iteration and Incrementation

- The phrase “iterate and increment” also includes the possibility of having to decrement what has been developed to date
  - A mistake may have been made, and backtracking is needed
  - As a consequence of reorganizing the UML models, one or more artifacts may have become superfluous

11.14 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)

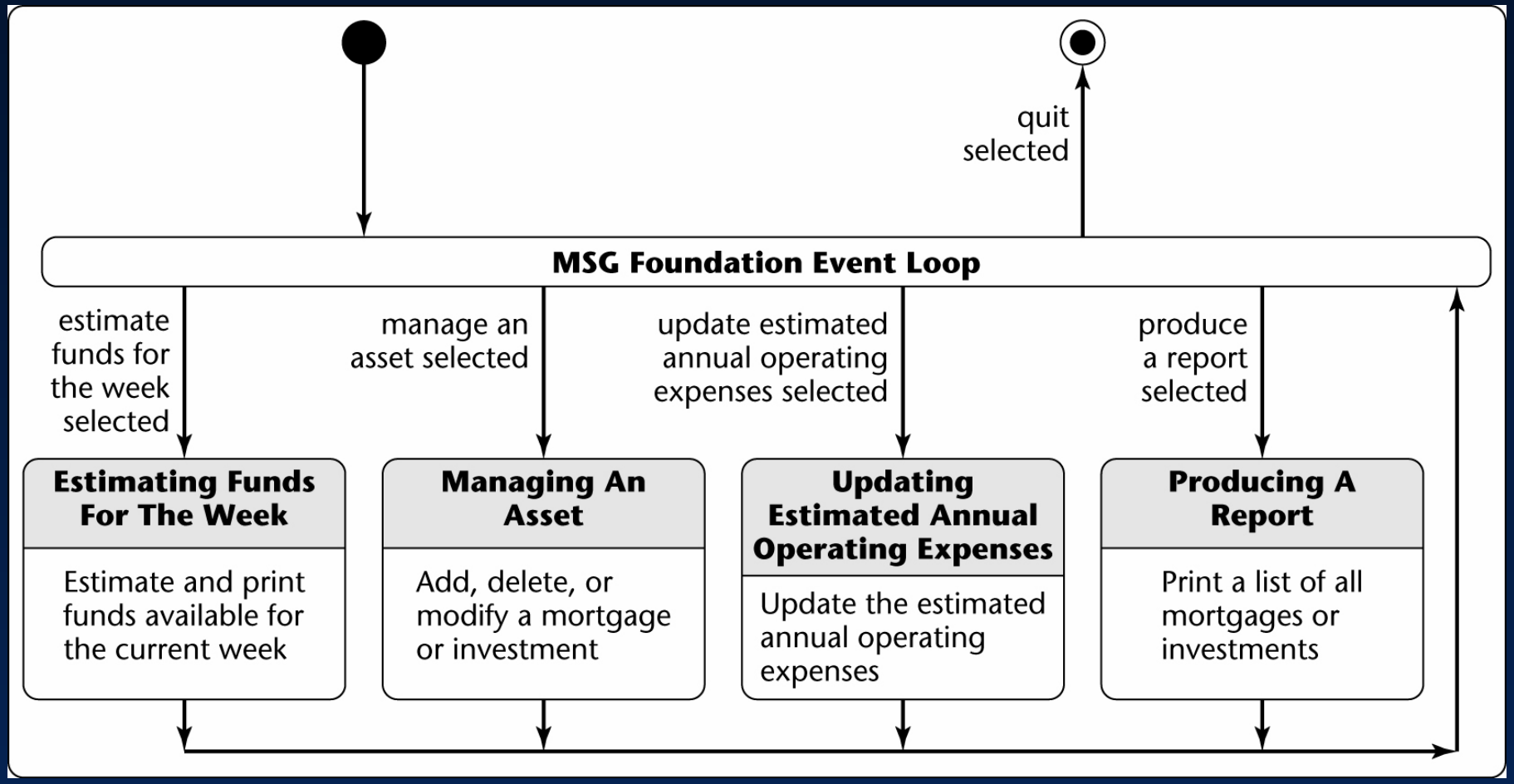

Figure 11.22

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

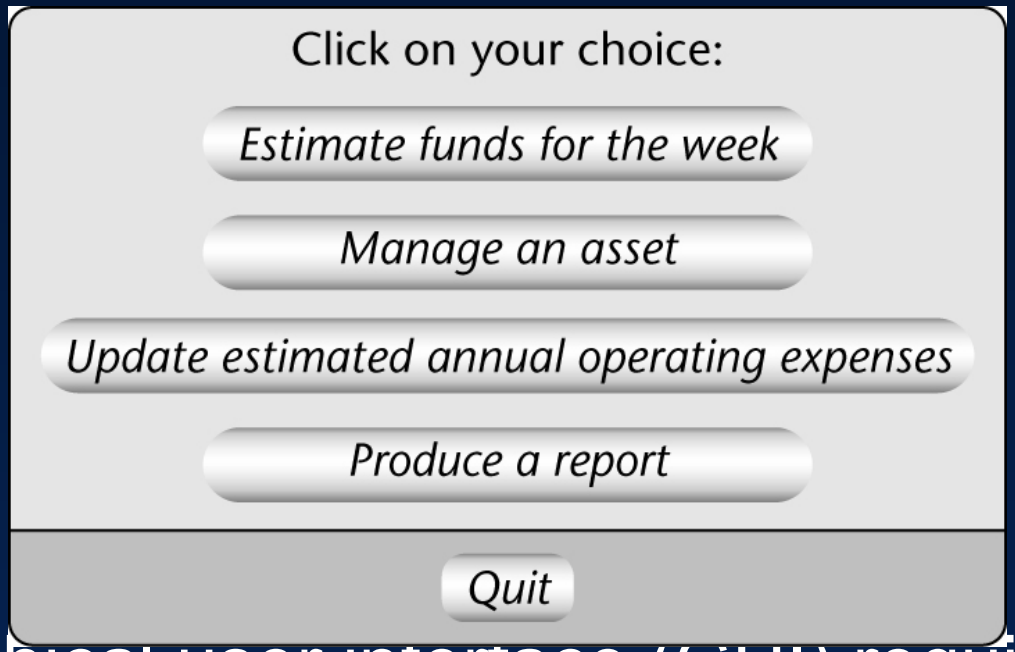

Figure 11.23

- This graphical user interface (GUI) requires special software

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 11.24

11.15 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 **Update 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

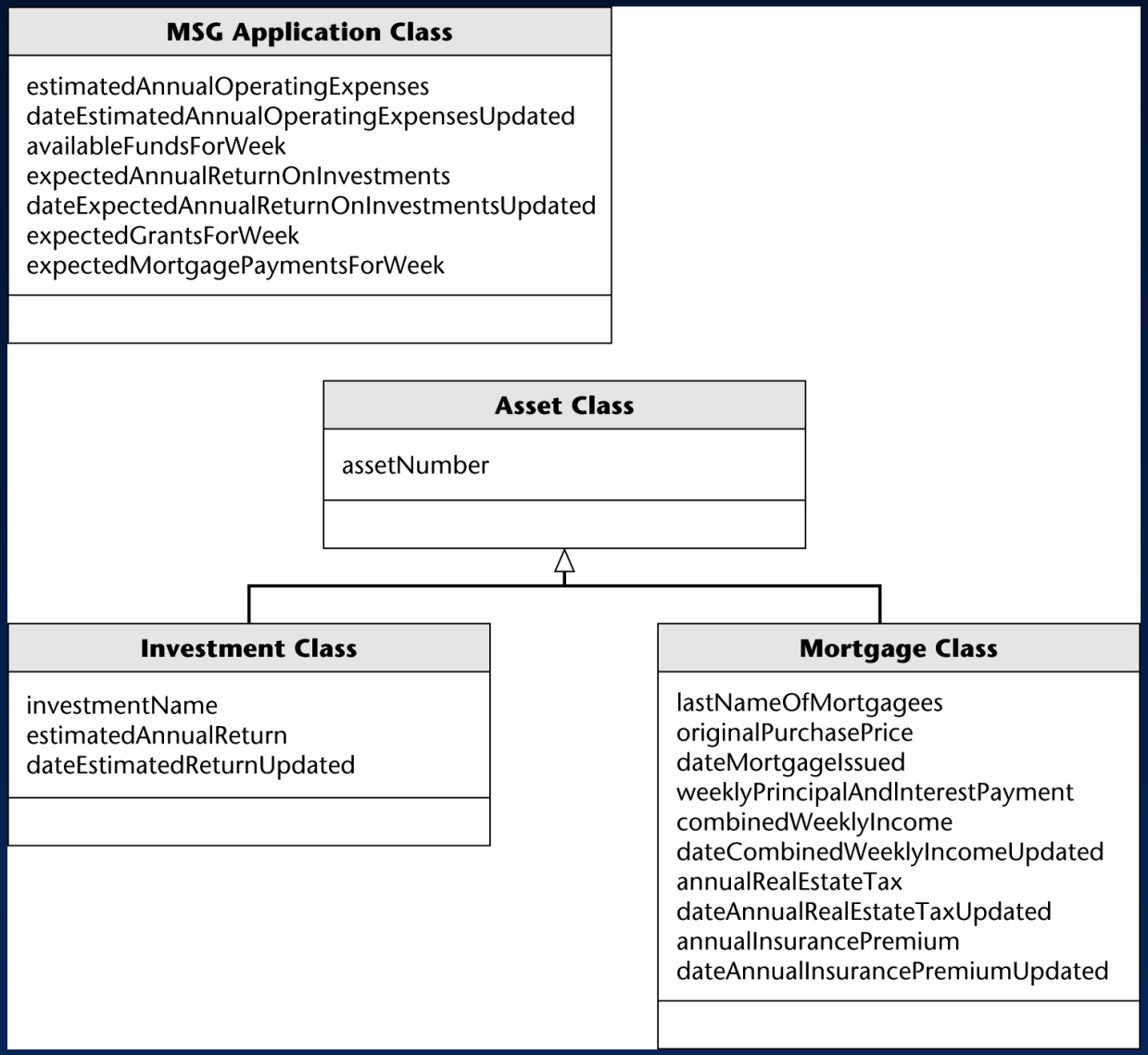

Figure 11.25

Third Iteration of the Initial Class Diagram: MSG Foundation

- The class diagram redrawn to show the prototypes

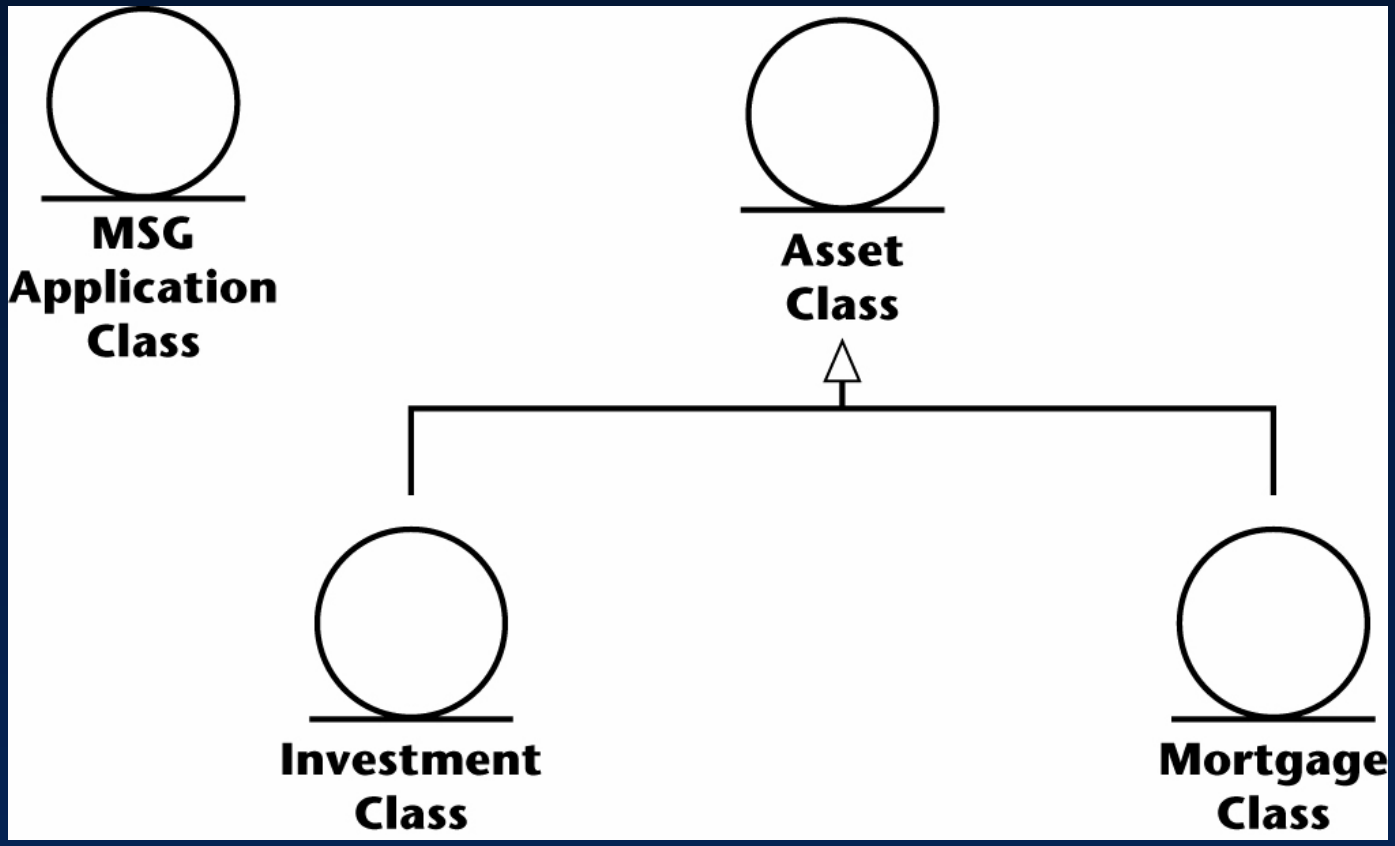

Figure 11.26

11.16 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

Figure 11.27

Initial Boundary Classes: MSG Foundation (contd)

- There are three reports:
  - The purchases report
  - The sales report
  - The future trends report
- The content of each report is different
  - Each report therefore has to be modeled by a separate boundary class

11.17 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

Figure 11.28

Class Extraction (contd)

- The description of class extraction is complete
- We now therefore return to the Unified Process