Appendix C

Requirements Workflow: The MSG Foundation Case Study

The requirements workflow for the MSG Foundation case study appears in Chapter 10.

Appendix D

Structured Systems Analysis: The MSG Foundation Case Study

Step 1. Draw the Data Flow Diagram See Figure 12.9.

Step 2. Decide What Sections to Computerize and How Computerize the complete pilot project online. However, if the weekly computation regarding availability of funds to purchase homes turns out to be time consuming, it may be better to perform it the night before it is required.

Step 3. Put in the Details of the Data Flows

investment_details	
investment_number	(12 characters)
investment_name	(25 characters)
expected_return	(9 + 2 digits)
date_expected_return_updated	(8 characters)
mortgage_details	
mortgage_number	(12 characters)
mortgage_name	(21 characters)
price	(6 + 2 digits)
date_mortgage_issued	(8 characters)
weekly_income	(6 + 2 digits)
date_weekly_income_was_updated	(8 characters)
annual_property_tax	(5 + 2 digits)

annual_insurance_premium	(5 + 2 digits)
mortgage_balance	(6 + 2 digits)
available_funds_for_week	(9 + 2 digits)
annual_operating_expenses	(9 + 2 digits)
update_request	(1 character)

Step 4. Define the Logic of the Processes

compute_availability_of_funds_and_generate_funds_report

Determine the expected income for the week by adding the expected_return of each investment in INVESTMENT_DATA.

Determine the expected mortgage payments for the week by adding the expected mortgage payment of each mortgage in MORTGAGE_DATA.

Determine the expected grants for the week by adding the expected grant for each mortgage in MORTGAGE_DATA.

Compute available_funds_for_week =
expected income for the week
− annual_operating_expenses / 52
+ expected mortgage payments for the week
− expected grants for the week

Display/print available_funds_for_week

generate_listing_of_investments

For each investment in INVESTMENT_DATA
Print investment_details

generate_listing_of_mortgages

For each mortgage in MORTGAGE_DATA
Print mortgage_details

perform_selected_update

Use the value of update_request to determine whether MORTGAGE_DATA, INVESTMENT_DATA, or EXPENSES_DATA are to be updated.

Perform the update.

Step 5. Define the Data Stores

EXPENSES_DATA
annual_operating_expenses [defined in Step 3]

INVESTMENT_DATA
investment_details [defined in Step 3]

MORTGAGE_DATA
mortgage_details [defined in Step 3]

All files are sequential, and hence there is no DIAD.

Step 6. Define the Physical Resources

EXPENSES DATA
Sequential file
Stored on disk

INVESTMENT DATA
Sequential file
Stored on disk

MORTGAGE DATA
Sequential file
Stored on disk

Step 7. Determine the Input/Output Specifications Input screens are designed for the following processes:

update_investment, update_mortgage, update_annual_operating_expenses, compute_availability_of_funds_and_generate_funds_report

The following reports are displayed:

list_of_investments, list_of_mortgages, available_funds_for_week

The screens and reports of the rapid prototype will be used as a basis for the preceding. The exact format of all screens and reports is subject to approval by the MSG Foundation.

Step 8. Perform Sizing Approximately 4 megabytes of storage are needed for the software. Each investment object requires approximately 50 bytes of storage. Each mortgage object requires approximately 90 bytes of storage. The storage requirements can be computed on the basis of the number of investments and mortgages owned by the MSG Foundation.

Step 9. Determine the Hardware Requirements

Desktop computer with hard disk, running Linux.
Zip drive for backups.
Laser printer for printing reports.

Appendix E

Analysis Workflow: The MSG Foundation Case Study

The analysis workflow is presented in Chapter 12.

Appendix F

Software Project Management Plan: The MSG Foundation Case Study

The plan presented here is for development of the MSG product by a small software organization consisting of three individuals: Almaviva, the owner of the company, and two software engineers, Bartolo and Cherubini.

1 Overview.

1.1 Project Summary.

1.1.1 Purpose, Scope, and Objectives. The objective of this project is to develop a software product that will assist the Martha Stockton Greengage (MSG) Foundation in making decisions regarding home mortgages for married couples. The product will allow the client to add, modify, and delete information regarding the Foundation's investments, operating expenses, and individual mortgage information. The product will perform the required calculations in these areas and produce reports listing investments, mortgages, and weekly operating expenses.

1.1.2 Assumptions and Constraints. Constraints include the following:

The deadline must be met.

The budget constraint must be met.

The product must be reliable.

The architecture must be open so that additional functionality may be added later.

The product must be user-friendly.

1.1.3 Project Deliverables. The complete product, including user manual, will be delivered 10 weeks after the project commences.

1.1.4 Schedule and Budget Summary. The duration, personnel requirements, and budget of each workflow are as follows:

Requirements workflow (1 week, two team members, $3740)
Analysis workflow (2 weeks, two team members, $7480)
Design workflow (2 weeks, two team members, $7480)
Implementation workflow (3 weeks, three team members, $16,830)
Testing workflow (2 weeks, three team members, $11,220)

The total development time is 10 weeks, and the total internal cost is $46,750.

1.2 Evolution of the Project Management Plan. All changes in the project management plan must be agreed to by Almaviva before they are implemented. All changes should be documented to keep the project management plan correct and up to date.

2 Reference Materials. All artifacts will conform to the company's programming, documentation, and testing standards.

3 Definitions and Acronyms. MSG—Martha Stockton Greengage; the MSG Foundation is our client.

4 Project Organization.

4.1 External Interfaces. All the work on this project will be performed by Almaviva, Bartolo, and Cherubini. Almaviva will meet weekly with the client to report progress and discuss possible changes and modifications.

4.2 Internal Structure. The development team consists of Almaviva (owner), Bartolo, and Cherubini.

4.3 Roles and Responsibilities. Bartolo and Cherubini will perform the design workflow. Almaviva will implement the class definitions and report artifacts, Bartolo will construct the artifacts to handle investments and operating expenses, and Cherubini will develop the artifacts that handle mortgages. Each member is responsible for the quality of the artifacts he or she produces. Almaviva will oversee integration and the overall quality of the software product and will liaise with the client.

5 Managerial Process Plans.

5.1 Start-up Plan.

5.1.1 Estimation Plan. As previously stated, the total development time is estimated to be 10 weeks and the total internal cost to be $46,750. These figures were obtained by expert judgment by analogy, that is, by comparison with similar projects.

5.1.2 Staffing Plan. Almaviva is needed for the entire 10 weeks, for the first 5 weeks in only a managerial capacity and the second 5 weeks as both manager and programmer. Bartolo and Cherubini are needed for the entire 10 weeks, for the first 5 weeks as systems analysts and designers, and for the second 5 weeks as programmers and testers.

5.1.3 Resource Acquisition Plan. All necessary hardware, software, and CASE tools for the project are already available. The product will be delivered to the MSG Foundation installed on a desktop computer that will be leased from our usual supplier.

5.1.4 Project Staff Training Plan. No additional staff training is needed for this project.

5.2 Work Plan.

5.2.1–2 Work Activities and Schedule Allocation.

Week 1.	(Completed) Met with client, and determined requirements artifacts. Inspected requirements artifacts.
Weeks 2, 3.	(Completed) Produced analysis artifacts, and inspected analysis artifacts. Showed artifacts to client, who approved them. Produced software project management plan, and inspected software project management plan.
Weeks 4, 5.	Produce design artifacts, inspect design artifacts.
Weeks 6–10.	Implementation and inspection of each class, unit testing and documentation, integration of each class, integration testing, product testing, and documentation inspection.

5.2.3 Resource Allocation. The three team members will work separately on their assigned artifacts. Almaviva's assigned role will be to monitor the daily progress of the other two, oversee implementation, be responsible for overall quality, and interact with the client. Team members will meet at the end of each day and discuss problems and progress. Formal meetings with the client will be held at the end of each week to report progress and determine if any changes need to be made. Almaviva will ensure that schedule and budget requirements are met. Risk management will also be Almaviva's responsibility.

Minimizing faults and maximizing user-friendliness will be Almaviva's top priorities. Almaviva has overall responsibility for all documentation and has to ensure that it is up to date.

5.2.4 Budget Allocation. The budget for each workflow is as follows:

Requirements workflow	$ 3,740
Analysis workflow	7,480
Design workflow	7,480
Implementation workflow	16,830
Testing workflow	11,220
Total	$46,750

5.3 Control Plan. Any major changes that affect the milestones or the budget have to be approved by Almaviva and documented. No outside quality assurance personnel are involved. The benefits of having someone other than the individual who carried out the development task do the testing will be accomplished by each person testing another person's work products.

Almaviva will be responsible for ensuring that the project is completed on time and within budget. This will be accomplished through daily meetings with the team members. At each meeting, Bartolo and Cherubini will present the day's progress and problems.

Almaviva will determine whether they are progressing as expected and whether they are following the specification document and the project management plan. Any major problems faced by the team members will immediately be reported to Almaviva.

5.4 Risk Management Plan. The risk factors and the tracking mechanisms are as follows:

There is no existing product with which the new product can be compared. Accordingly, it will not be possible to run the product in parallel with an existing one. Therefore, the product should be subjected to extensive testing.

The client is assumed to be inexperienced with computers. Therefore, special attention should be paid to the analysis workflow and communication with the client. The product has to be made as user-friendly as possible.

Because of the ever-present possibility of a major design fault, extensive testing will be performed during the design workflow. Also, each of the team members will initially test his or her own code and then test the code of another member. Almaviva will be responsible for integration testing and in charge of product testing.

The information must meet the specified storage requirements and response times. This should not be a major problem because of the small size of the product, but it will be monitored by Almaviva throughout development.

There is a slim chance of hardware failure, in which case another machine will be leased. If there is a fault in the compiler, it will be replaced. These are covered in the warranties received from the hardware and compiler suppliers.

5.5 Project Close-out Plan. Not applicable here.

6 Technical Process Plans.

6.1 Process Model. The Unified Process will be used.

6.2 Methods, Tools, and Techniques. The workflows will be performed in accordance with the Unified Process. The product will be implemented in Java.

6.3 Infrastructure Plan. The product will be developed using ArgoUML running under Linux on a personal computer.

6.4 Product Acceptance Plan. Acceptance of the product by our client will be achieved by following the steps of the Unified Process.

7 Supporting Process Plan

7.1 Configuration Management Plan. CVS will be used throughout for all artifacts.

7.2 Testing Plan. The testing workflow of the Unified Process will be performed.

7.3 Documentation Plan. Documentation will be produced as specified in the Unified Process.

7.4–5 Quality Assurance Plan and Reviews and Audits Plan. Bartolo and Cherubini will test each other's code, and Almaviva will conduct integration testing. Extensive product testing will then be performed by all three.

7.6 Problem Resolution Plan. As stated in 5.3, any major problems faced by the team members will immediately be reported to Almaviva.

7.7 Subcontractor Management Plan. Not applicable here.

7.8 Process Improvement Plan. All activities will be conducted in accord with the company plan to advance from CMM level 2 to level 3 within 2 years.

8. Additional Plans. Additional components:

Security. A password will be needed to use the product.

Training. Training will be performed by Almaviva at time of delivery. Because the product is straightforward to use, 1 day should be sufficient for training. Almaviva will answer questions at no cost for the first year of use.

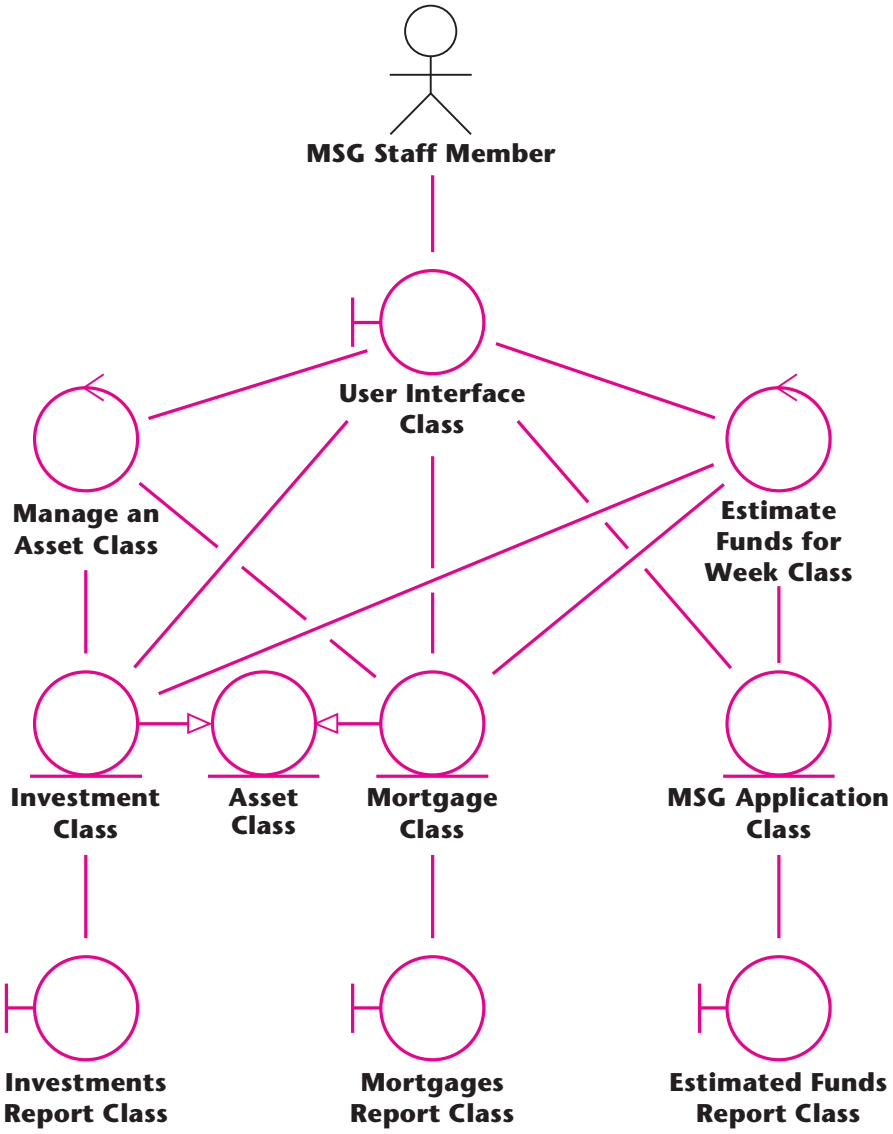
Maintenance. Corrective maintenance will be performed by the team at no cost for a period of 12 months. A separate contract will be drawn up regarding enhancement.

Appendix G

Design Workflow: The MSG Foundation Case Study

This appendix contains the final version of the class diagram for the MSG Foundation case study (Figure G.1). The overall class diagram is followed by UML diagrams for the 10 component classes, in alphabetical order. These UML diagrams show the attributes and the methods. As explained in Section 17.2, the UML visibility prefixes are – for **private**, + for **public**, and # for **protected**. The attributes and methods are shown in a PDL for Java. Accordingly, there is no **Date Class** (see Section 14.8).

FIGURE G.1
The final class diagram for the MSG Foundation case study.



«entity class» **Asset Class**
assetNumber : string
+ getAssetNumber ( ) : string + setAssetNumber (a : string) : void + **abstract** read (fileName : RandomAccessFile) : void + **abstract** obtainNewData ( ) : void + **abstract** performDeletion ( ) : void + **abstract** write (fileName : RandomAccessFile) : void + **abstract** save ( ) : void + **abstract** print ( ) : void + **abstract** find (s : string) : Boolean + delete ( ) : void + add ( ) : void

«control class» **Estimate Funds for Week Class**
+ <<static>> compute ( ) : void

«boundary class» **Estimate Funds Report Class**
+ <<static>> printReport ( ) : void

«entity class» **Investment Class**
– investmentName : string – expectedAnnualReturn : float – expectedAnnualReturnUpdated : string
+ getInvestmentName ( ) : string + setInvestmentName (n : string) : void + getExpectedAnnualReturn ( ) : float + setExpectedAnnualReturn (r : float) : void + getExpectedAnnualReturnUpdated ( ) : string + setExpectedAnnualReturnUpdated (d : string) : void + totalWeeklyReturnOnInvestment ( ) : float + find (findInvestmentID : string) : Boolean + read (fileName : RandomAccessFile) : void + write (fileName : RandomAccessFile) : void + save ( ) : void + print ( ) : void + printAll ( ) : void + obtainNewData ( ) : void + performDeletion ( ) : void + readInvestmentData ( ) : void + updateInvestmentName ( ) : void + updateExpectedReturn ( ) : void

«boundary class» **Investments Report Class**
+ <<static>> printReport ( ) : void

«control class» **Manage an Asset Class**
+ <<static>> manageInvestment ( ) : void + <<static>> manageMortgage ( ) : void

«entity class» **Mortgage Class**
– mortgageeName : string – price : float – dateMortgageIssued : string – currentWeeklyIncome : float – weeklyIncomeUpdated : string – annualPropertyTax : float – annualInsurancePremium : float – mortgageBalance : float + <<static final>> INTEREST_RATE : float + <<static final>> MAX_PER_OF_INCOME : float + <<static final>> NUMBER_OF_MORTGAGE_PAYMENTS : int + <<static final>> WEEKS_IN_YEAR : float + getMortgageeName ( ) : string + setMortgageeName (n : string) : void + getPrice ( ) : float + setPrice (p : float) : void + getDateMortgageIssued ( ) : string + setDateMortgageIssued (w : string) : void + getCurrentWeeklyIncome ( ) : float + setCurrentWeeklyIncome (i : float) : void + getWeeklyIncomeUpdated ( ) : string + setWeeklyIncomeUpdated (w : string) : void + getAnnualPropertyTax ( ) : float + setAnnualPropertyTax (t : float) : void + getAnnualInsurancePremium ( ) : float + setAnnualInsurancePremium (p : float) : void + getMortgageBalance ( ) : float + setMortgageBalance (m : float) : void + totalWeeklyNetPayments ( ) : float + find (findMortgageID : string) : Boolean + read (fileName : RandomAccessFile) : void + write (fileName : RandomAccessFile) : void + obtainNewData ( ) : void + performDeletion ( ) : void + print ( ) : void + <<static>> printAll ( ) : void

+ save ( ) : void + readMortgageData ( ) : void + updateBalance ( ) : void + updateDate ( ) : void + updateInsurancePremium ( ) : void + updateMortgageeName ( ) : void + updatePrice ( ) : void + updatePropertyTax ( ) : void + updateWeeklyIncome ( ) : void

«boundary class» **Mortgages Report Class**
+ <<static>> printReport ( ) : void

«entity class» **MSG Application Class**
– <<static>> estimatedAnnualOperatingExpenses : float – <<static>> estimatedFundsForWeek : float
– <<static>> getAnnualOperatingExpenses ( ) : float – <<static>> setAnnualOperatingExpenses (e : float) : void + <<static>> getEstimatedFundsForWeek ( ) : float + <<static>> setEstimatedFundsForWeek (e : float) : void + <<static>> initializeApplication ( ) : void + <<static>> updateAnnualOperatingExpenses ( ) : void + <<static>> main ( )

«boundary class» **User Interface Class**
+ <<static>> clearScreen ( ) : void + <<static>> pressEnter ( ) : void + <<static>> displayMainMenu ( ) : void + <<static>> displayInvestmentMenu ( ) : void + <<static>> displayMortgageMenu ( ) : void + <<static>> displayReportMenu ( ) : void + <<static>> getChar ( ) : char + <<static>> getString ( ) : string + <<static>> getInt ( ) : int

Appendix H

Implementation Workflow: The MSG Foundation Case Study (C++ Version)

The complete C++ source code for the MSG Foundation product is available on the World Wide Web at www.mhhe.com/schach.

Appendix

Implementation Workflow: The MSG Foundation Case Study (Java Version)

The complete Java source code for the MSG Foundation product is available on the World Wide Web at www.mhhe.com/schach.

Appendix J

Test Workflow: The MSG Foundation Case Study

The test workflow of the MSG Foundation case study is presented in four sections:

Section 11.11 (requirements)
Section 13.17 (analysis)
Section 14.11 (design)
Section 15.23 (implementation)

This page intentionally left blank