



By:

Barry Chung

&

Myron LaCome

Final Project – Part I and II

CIS 405B Summer Quarter 2008

Professor Huarng, PhD

Table of Contents

Part I

Project Description 1
Preliminary Investigation 2
Systems Development Life Cycle 3

Part II – Functional Decomposition Diagrams

Conceptual Diagram..... 1
Process 1 2
Process 2 5
Process 3 8
Process 4 9
Process 5 10
Process 6 11

Part III – Dataflow Diagrams

Conceptual Diagram 1
Zero Diagram 2
Consolidated DBMS..... 6
Diagram 1..... 7
Diagram 2..... 10
Diagram 3..... 13
Diagram 4..... 14
Diagram 5..... 15
Diagram 6..... 17

Part IV – Process Specifications

Process 1 1

Process 2	5
Process 3	10
Process 4	11
Process 5	12
Process 6	14

Part V – Structure Chart

Module 0	1
Module 1	2
Module 1.1	3
Module 1.2	4
Module 2	5
Module 2.2	6
Module 2.4	7
Module 3	8
Module 3.1	9
Module 4	10
Module 5	11
Module 5.2	12
Module 6	13

Entity Relationship Diagram	1
-----------------------------------	---

Part VI – Unified Modeling Language & Entity Relationship Diagram

Use Case Diagram	1
------------------------	---

Class Diagram	1
---------------------	---

State Transition Diagram	1 – 3
--------------------------------	-------

Activity Diagram	1 -2
------------------------	------

Part VII – Prototype

MySQL Database Creation Script	1
MySQL Database Test Data.....	3
Java Application Code	4

Part I

I – Project Description

Clients send listings to real estate agency. Agents at the real estate office accept or deny real estate services to clients based on location of listing. Listing addresses are compared to the address domain. If listings are within the address range an acceptance package containing a contract agreement and fee schedule is sent to the client. Listing information such as: location, value and description are logged into the listing inventory. If the listing is not within the address range, then a rejection letter is sent to the client. Clients are to respond within two business days with payment for services to commence.

Available agents are assigned to the listings. Available agents are found in the agent schedule.

Sales entries are added to the sales log by agents, who receive a commission for their services.

Payment receipts are sent to the accounting department.

Advertisements are generated for accepted listings and sent to the advertisement agency. The advertisement agency sends a receipt, which is added to the listing transactions file.

Management can request reports, which may consist of all logged data. Once the reports are generated, they are sent to management.

Deposits for transactions are sent to banks. Receipts for bank deposits are logged in the business transactions file and a monthly report is sent to the accounting department.

Confirmations for both payment receipts and bank deposit monthly reports are logged in the business transactions file.

All data stores will be combined into one DBMS.

II – Preliminary Investigation

1.)

The purpose of implementing the Real Estate Tracking System is to give managers and owners more accurate sales information in a timely fashion.

The goals of the system are: increase sales information available to managers, decrease the time needed to make sales information available, maintain reliability, maintain security and maintain presence of sales information. Secondary goals are: integrate sales information with client, real estate agent, general ledger, advertising and bank transactions.

In order to develop a Real Estate Sales Tracking system, a prototype system will be created. The prototype system will focus on tracking sales as the primary function and will include the other listed functions as the schedule allows.

The application will have functional groups with specific privileges for each group. The groups will correspond to the users, being; real estate agents, managers and owners. Real estate agents will be allowed to access the client function with read-only privileges. They will also be allowed to access real estate agent functions with read/write privileges for only sales entries. Managers will have read-only privileges for all functions and write privileges for real estate agent functions, excluding sales entries. They will also have write privileges for advertising agency, bank and manager functions. Owners will have read-only privileges for the entire system.

Phase one of the system is due August 2008. Additional functions will be added as time allows. Additional phasing will be allotted a time table after phase one has been implemented. The entire system is not to exceed the allotted sum of funds. Each phase will require full disclosure of expenditures, along with phase deliverable, in order for final payment to be released. All hardware and software will belong to Real Estate Office LLC upon completion of each phase. All hardware and software required to operate system will be housed on or in REO, LLC property or facilities. The system is to be designed for a current capacity of 30 users with additional capacity for a maximum of 65 users. Deliverables are to be received according to the timetable agreed upon in the signed contract.

Assumptions:

- For simplicity, all data will be assumed to be complete and will not contain errors.
- Data stores have been established and contain all necessary data
- Limited redundancy has been implemented

2.)

There are three owners. Owners will request desired information, related to sales, from managers. There are five managers. Managers will submit queries to the system to retrieve reports. The reports will be generated from sales, and other, information from data input by real estate agents or their assistants. The number of real estate agents constantly changes, but averages out to be 20.

3.)

The primary use of this system will be to log sales made by real estate agents and sum up the total value of sales made per agent. This information will be kept in data stores and made available to managers, who can query the data stores for desired sales information. Managers will create their own reports based on the queries to be submitted to the owners.

4.)

All transactions are kept in random and miscellaneous forms. Some real estate agents use spread sheets to track sales, others use note pads. Each agent is responsible for their own sales. All sales logs are filed in file cabinets, held in each agent's cubicle or office. Once a sale has been closed, the agent fills out the appropriate paperwork and processes it manually, using regular mail, facsimile, photocopying and courier services. If a manager wants a report on sales activity, a request is given to an individual agent, who then compiles the request from the file in their file cabinet. Delivery of the report comes in many forms such as; word processed files, printed copies, typed copies and hand written notes. A report may take from one business day to one month in order to be received by management. Owner's updates are generally given at meetings with managers. All other transactions are handled in much the same fashion. Each agent has their preferential method of handling transactions. Each transaction handled by a specific agent is performed according to their preference.

5.)

All client information is collected and filed by a specific real estate agent, who is responsible for the client. The files are kept in file cabinets in each agent's cubicle or office. Closed sales are filed and then a value of the sale is submitted to the office secretary. The office secretary issues an invoice to the client for the amount owed and a request for release of commission to a manager. Sales and business transactions are kept in general ledger or in the secretary's file cabinet. Managers keep their reports in file cabinets in their offices.

III – Systems Development Life Cycle

1.)

The development of this system will be done using a combination of spiral and systems development life cycle. Spiral development will be used for the planning of each system phase. Once the goals for that phase have been determined, SDLC will be used to attain those goals. The process will be iterated until the client has a system that they are satisfied with or a new system is developed.

Determined system goals

The aim of this system is to have all documentation involved with all business transaction for the Real Estate Office to be in one location that is readily available to real estate agents, managers and owners for their various uses.

At this point, phases will include the entities of clients, real estate agents, advertising agency, management and bank. The system is to streamline communications between the entities and the system, while storing all documentation in one location.

Phase one will include a prototype for the client and real estate agent entities. Additional transactions may be included, but are dependent on available time. Additional phases will add listed entities after a review period, which will determine the validity of each entity.

Planning

The initial phase will be to create a prototype of a system that will be capable of logging sales under simplified conditions. The logging of sales will include; receiving a request from a client, processing that request, accepting or denying the request, the client paying for services, assignment of a real estate agent, logging sales transactions and allowing managers to query activities. Additional transactions will be added as time allows. Real estate agents will be surveyed at the beginning of each phase to how they accomplish their tasks. The prototype will then be given to the office for testing and observed use. Real estate agents will be interviewed and asked how the system is functioning. Managers will be interviewed after they have submitted queries to the system, with the same premise. Comments, suggestions and additional transactions will be added to the prototype system. The prototype will again be given to the office. This process will be repeated until the real estate office has a system that they feel comfortable with.

Analysis

Analysis will be done by constructing logical models of the entire system and its interactions with the listed entities. The logical models will include a functional decomposition diagram, a dataflow diagram, process specifications and a structural diagram. The emphasis will focus on the client and real estate agent entities.

Design

An application, with modularity in mind, will be developed using Java. The application will consist of a user interface that will access the data stores for the user and perform entry, retrieval and query functions. Each function will be allocated by group status; meaning that agents, managers and owners will have access to functions specific to their group. The emphasis will be on function, usability and the ability to be improved upon. Structured Query Language will be used for all data stores.

The system will be designed to support five owners, ten managers and 50 real estate agents. It will initially consist of one file server, one laser printer, and 30 workstations all networked via a common hub, router and modem hardware system. The physical network is not part of this design, but will be installed by another organization.

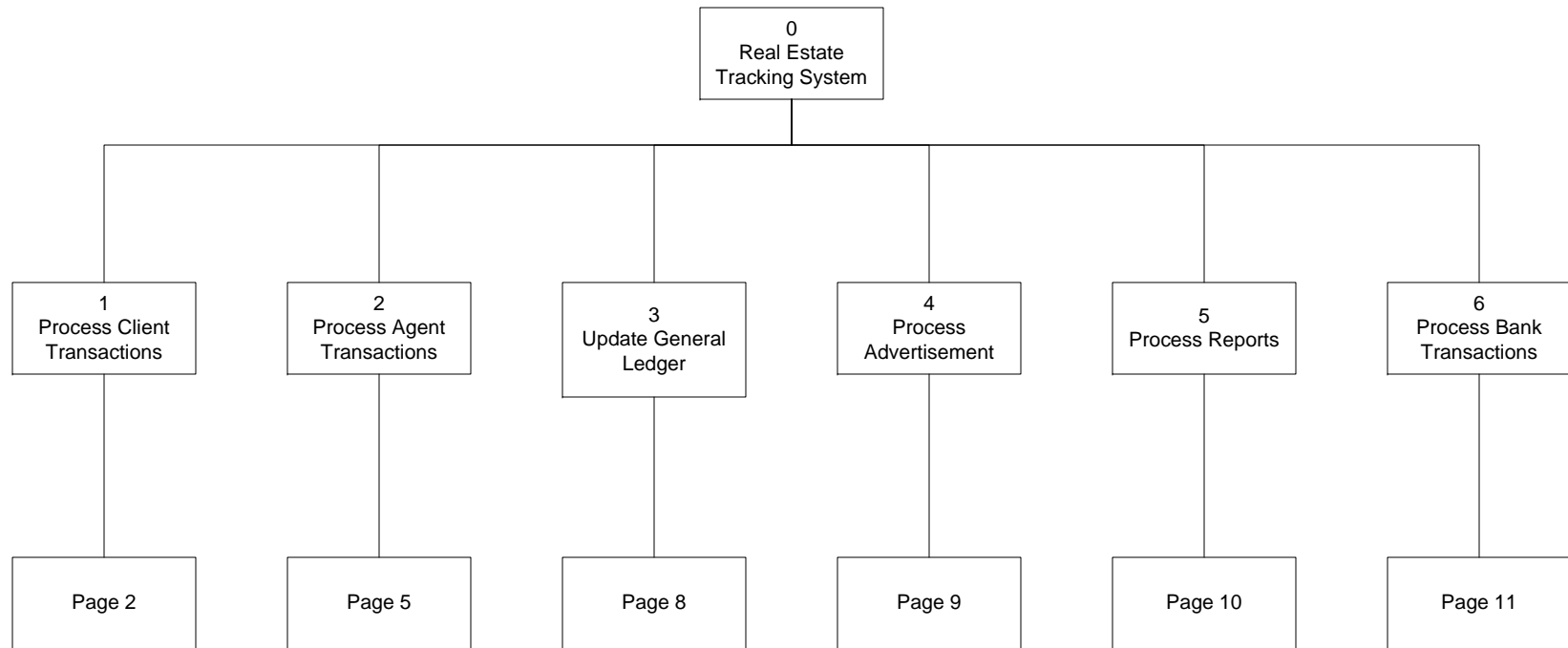
Implementation

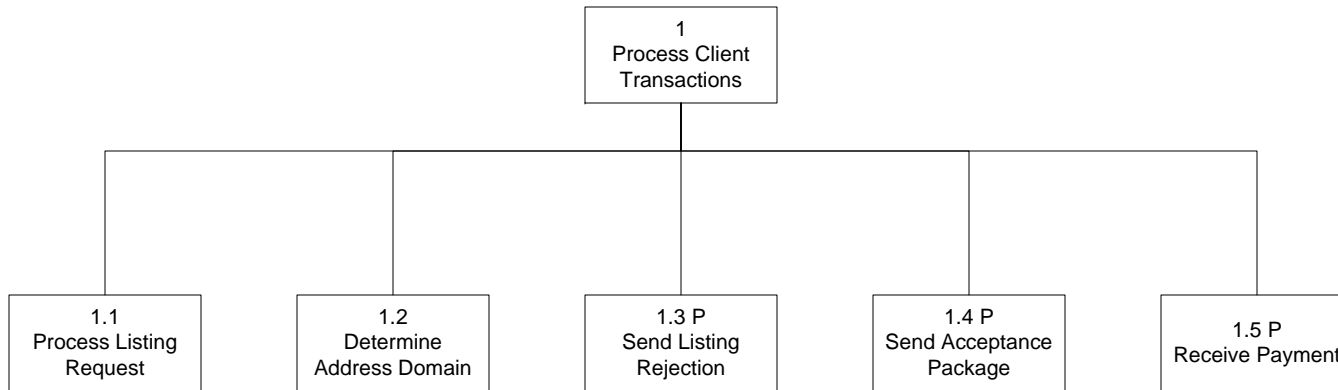
The prototype system will be given to the client office and tested for a period of two weeks. At the end of the testing, the prototype will be put into operation or be debugged until it is operational.

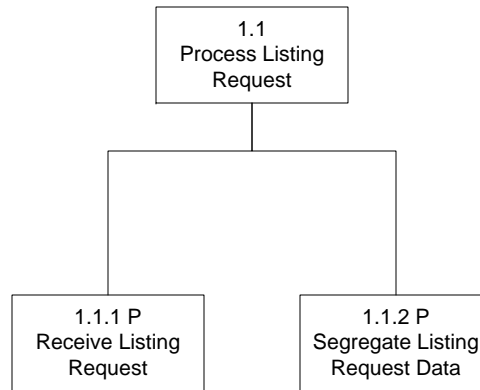
Maintenance

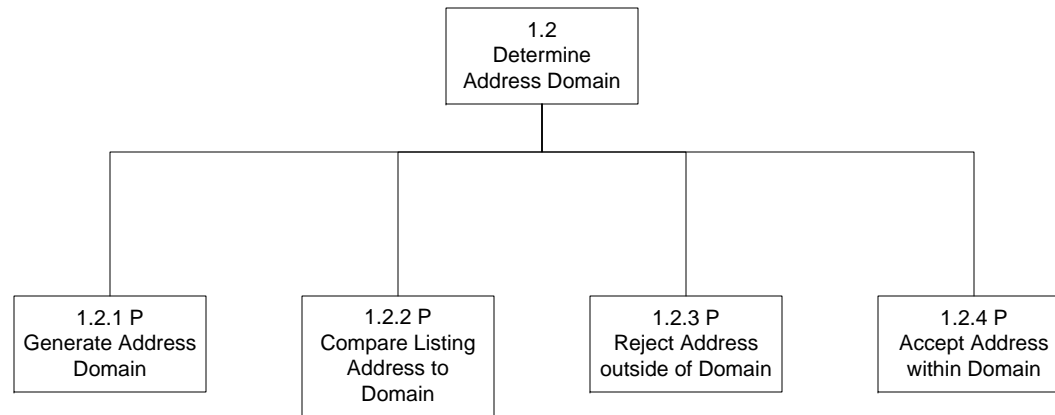
Maintenance will consist of daily, weekly and monthly backups of data. The system will also be checked as to capacity. At 80 percent usage, a new system will be developed.

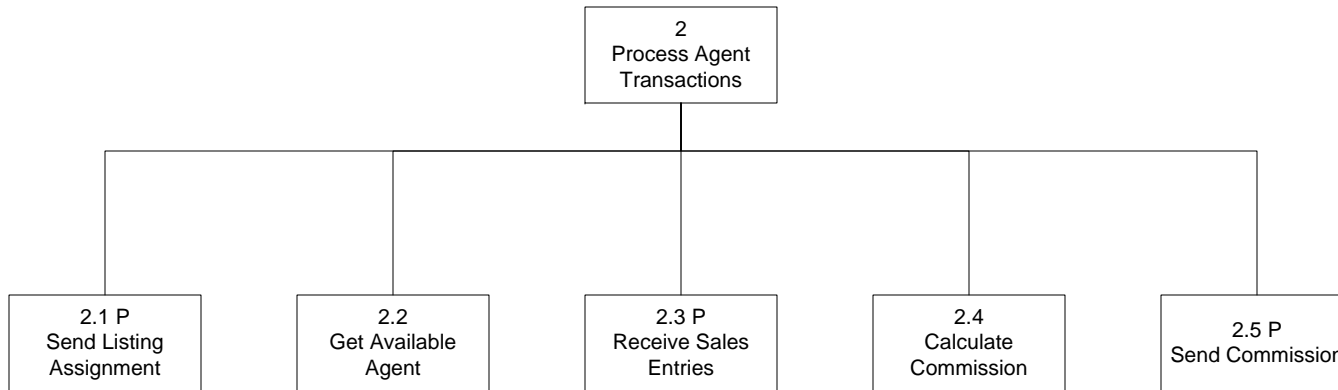
Part II

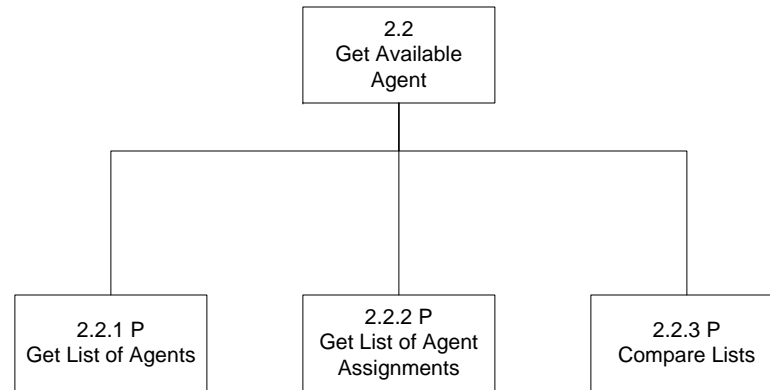


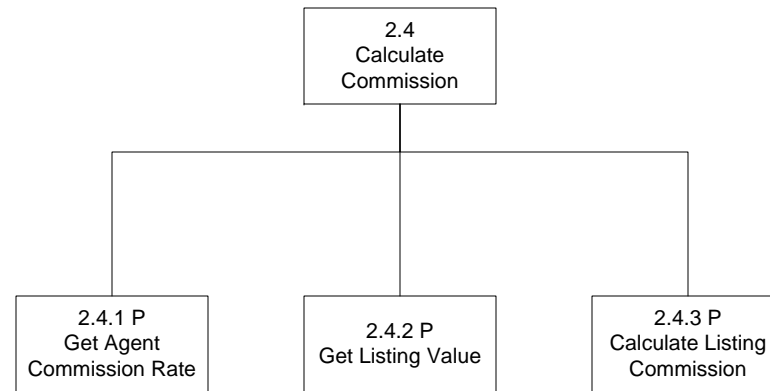


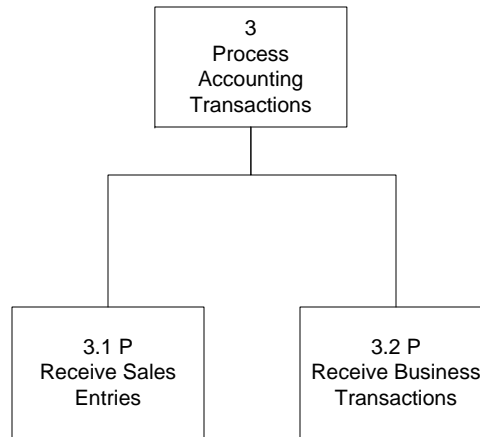


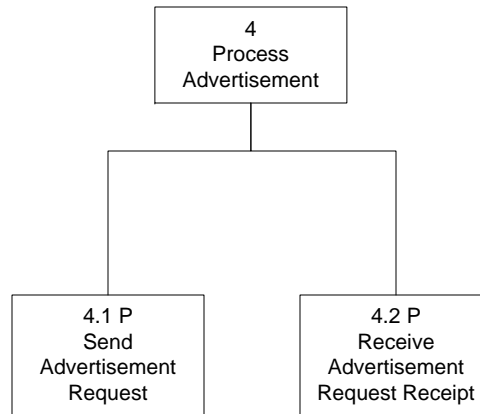


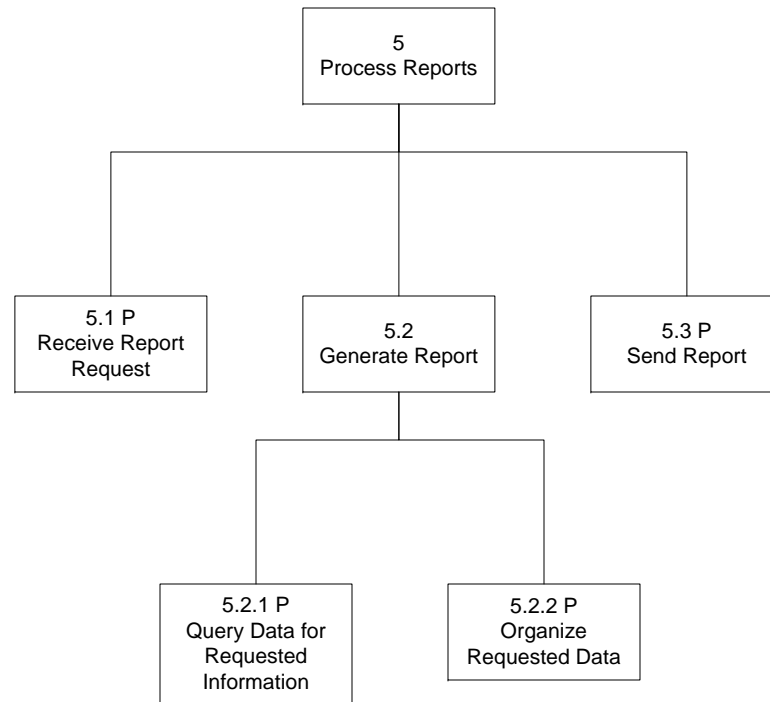


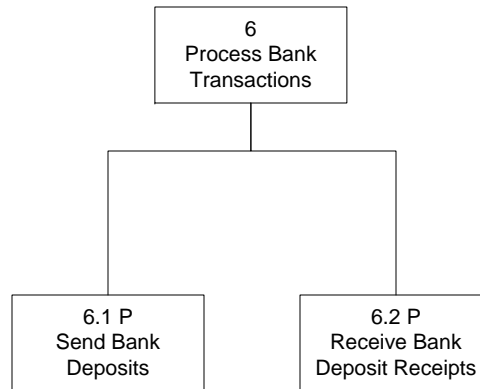




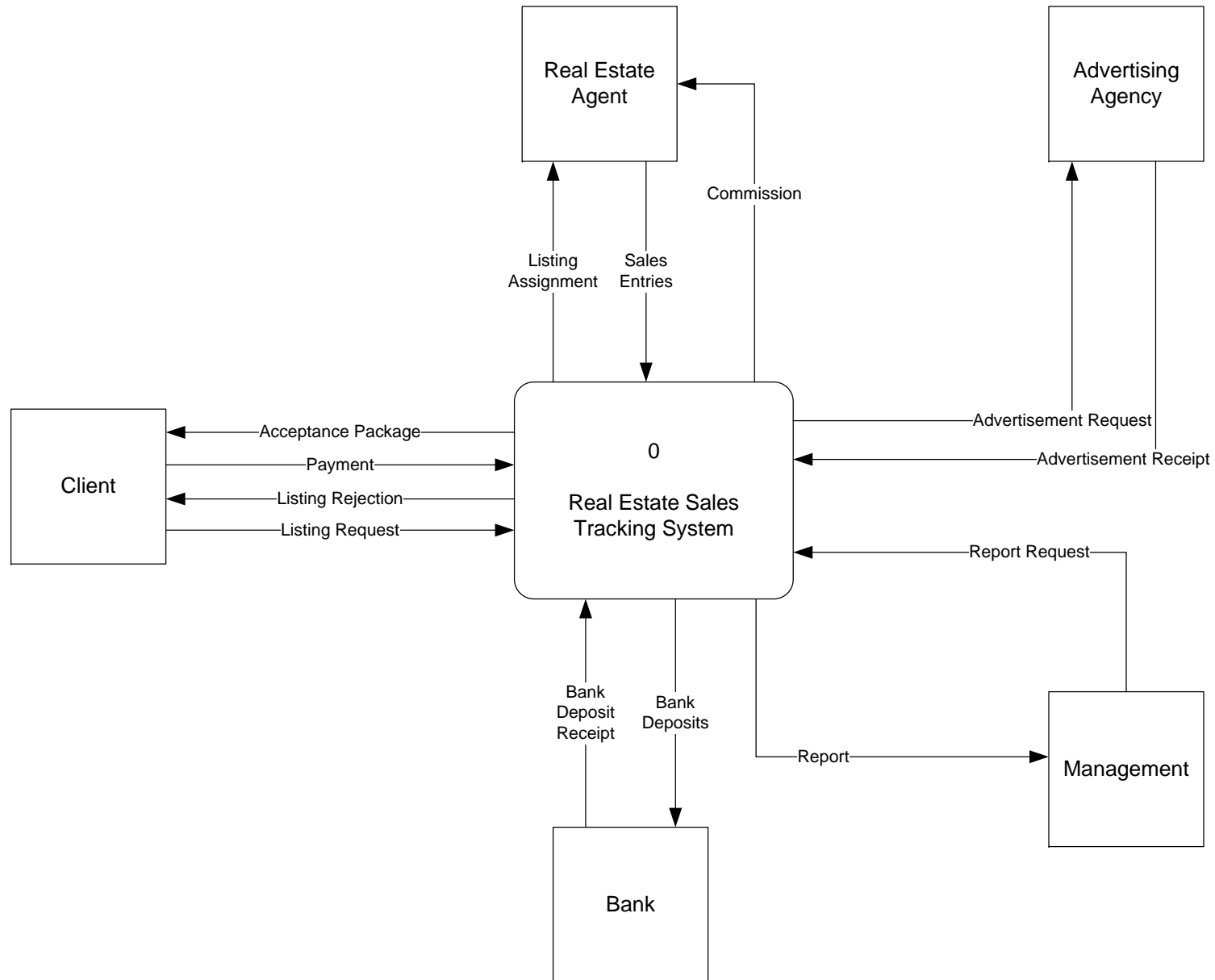


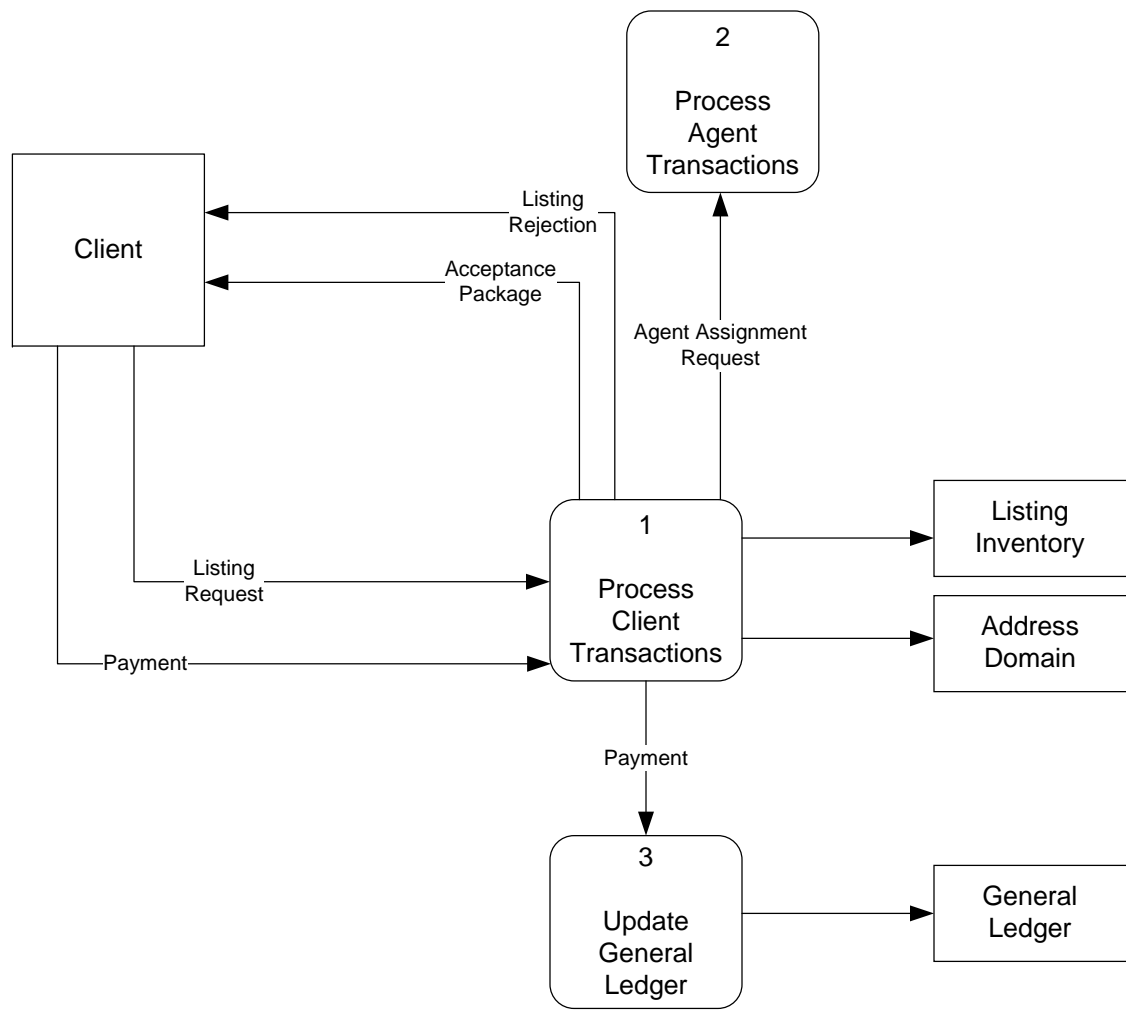


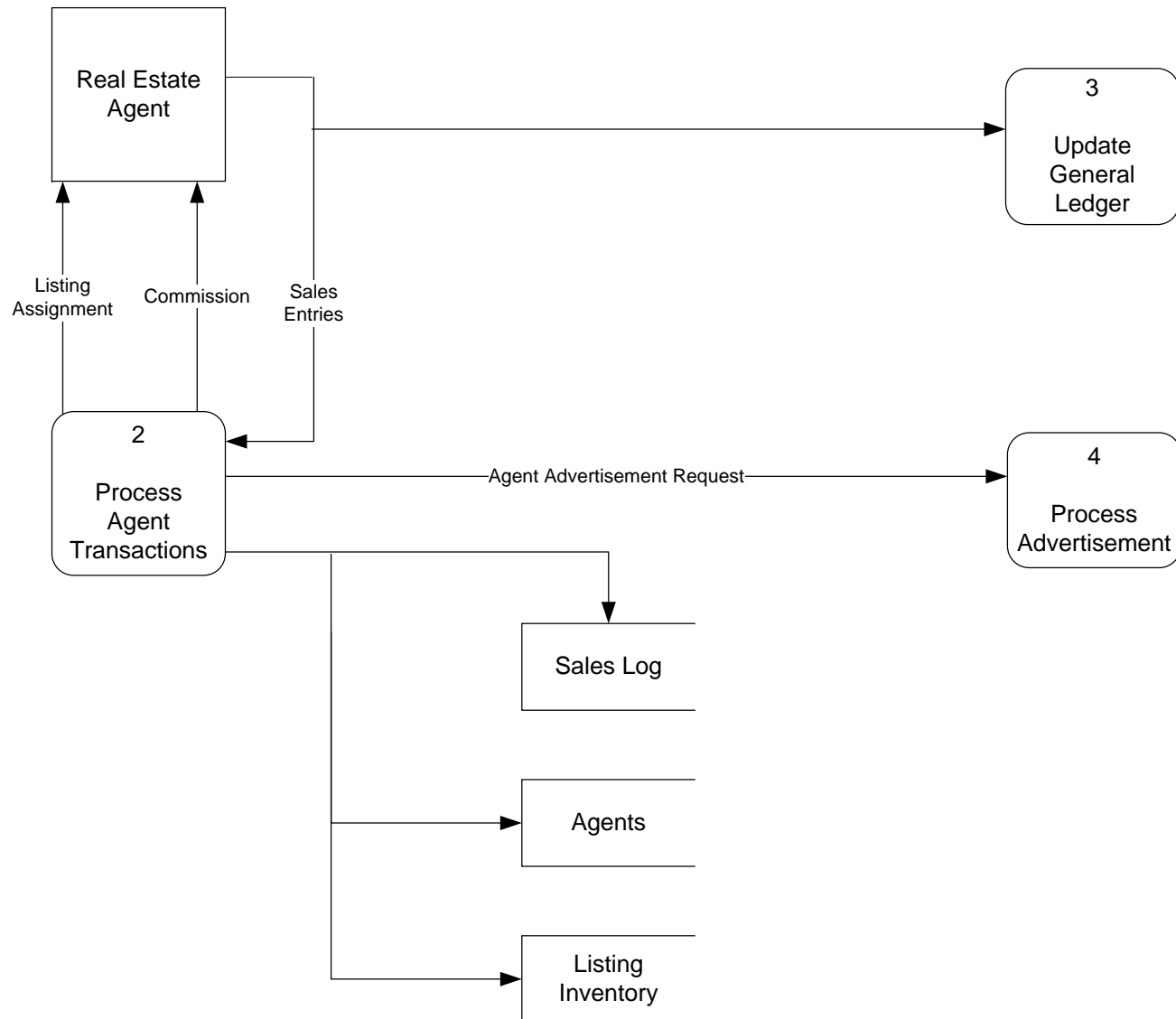


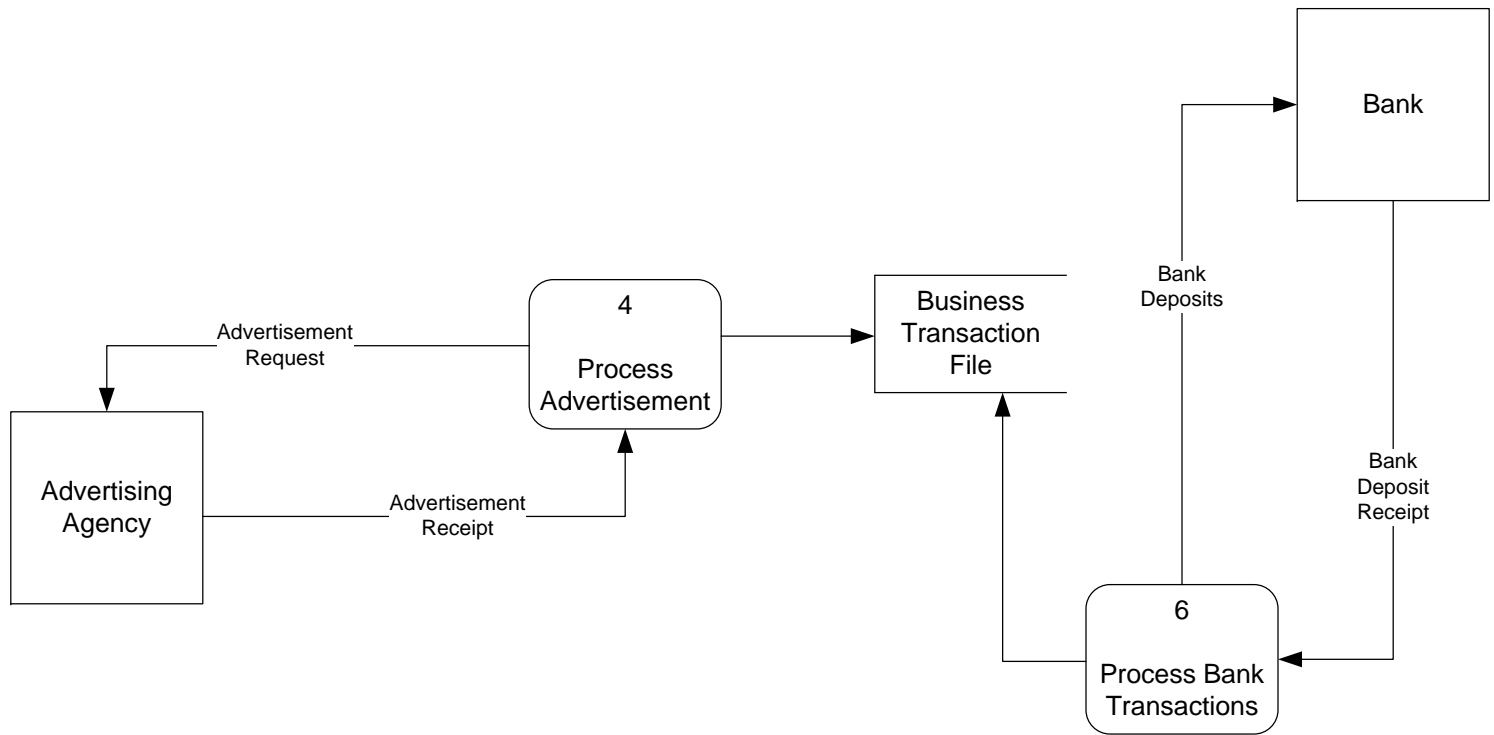


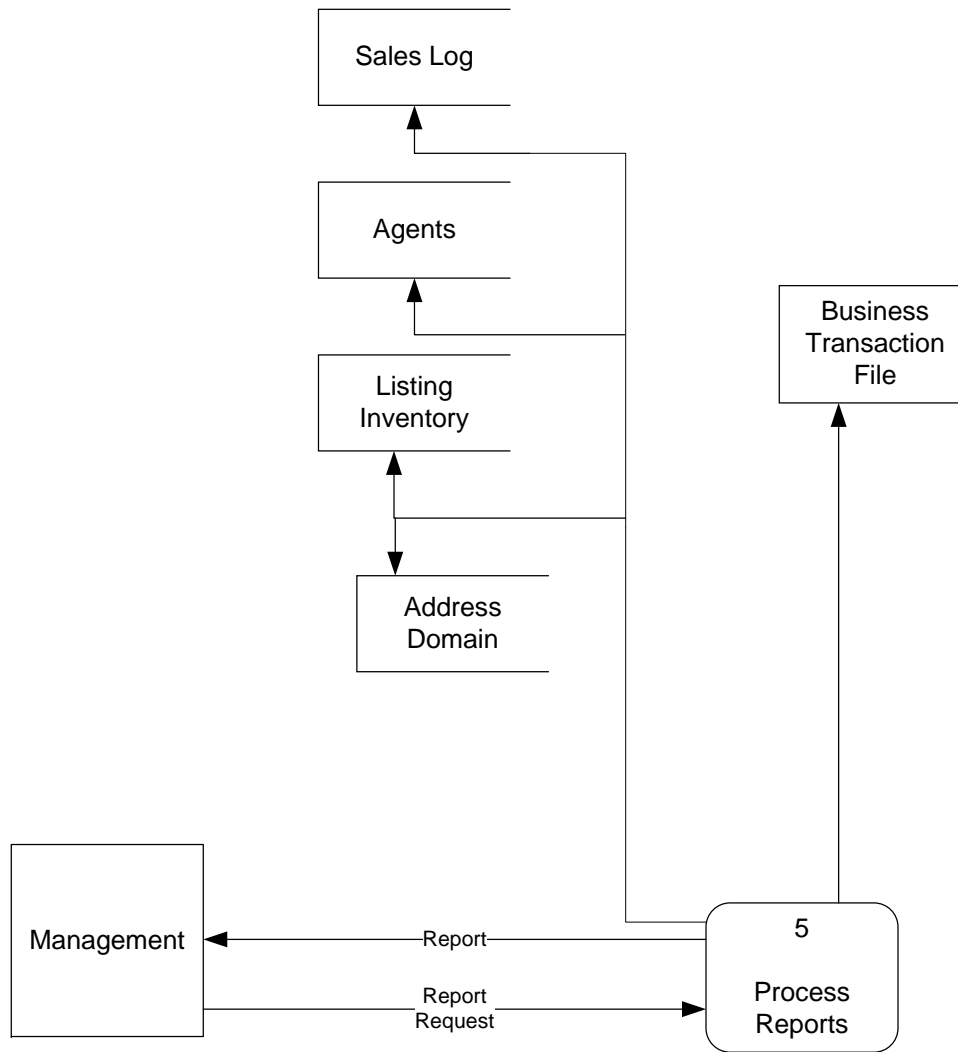
Part III

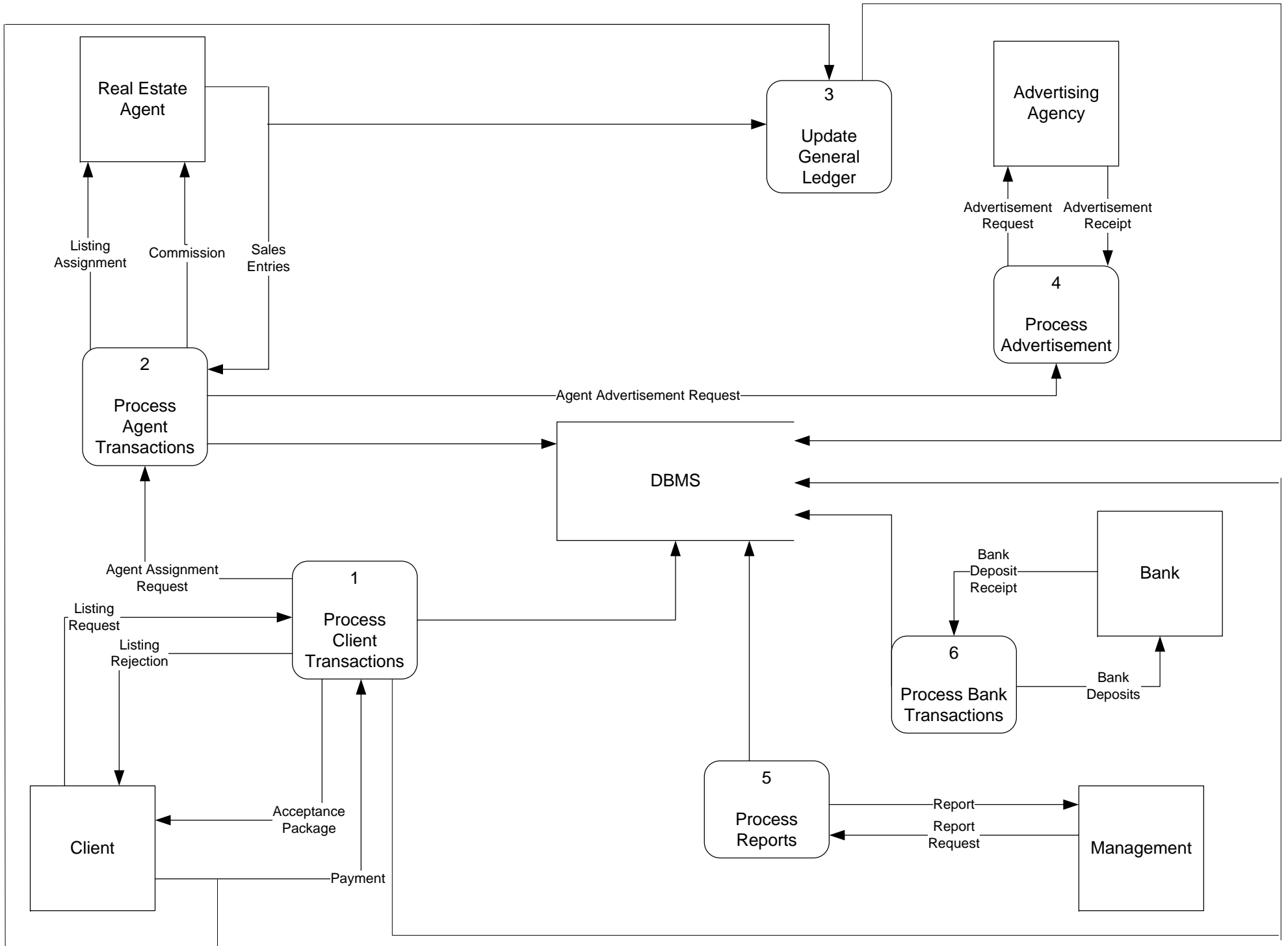


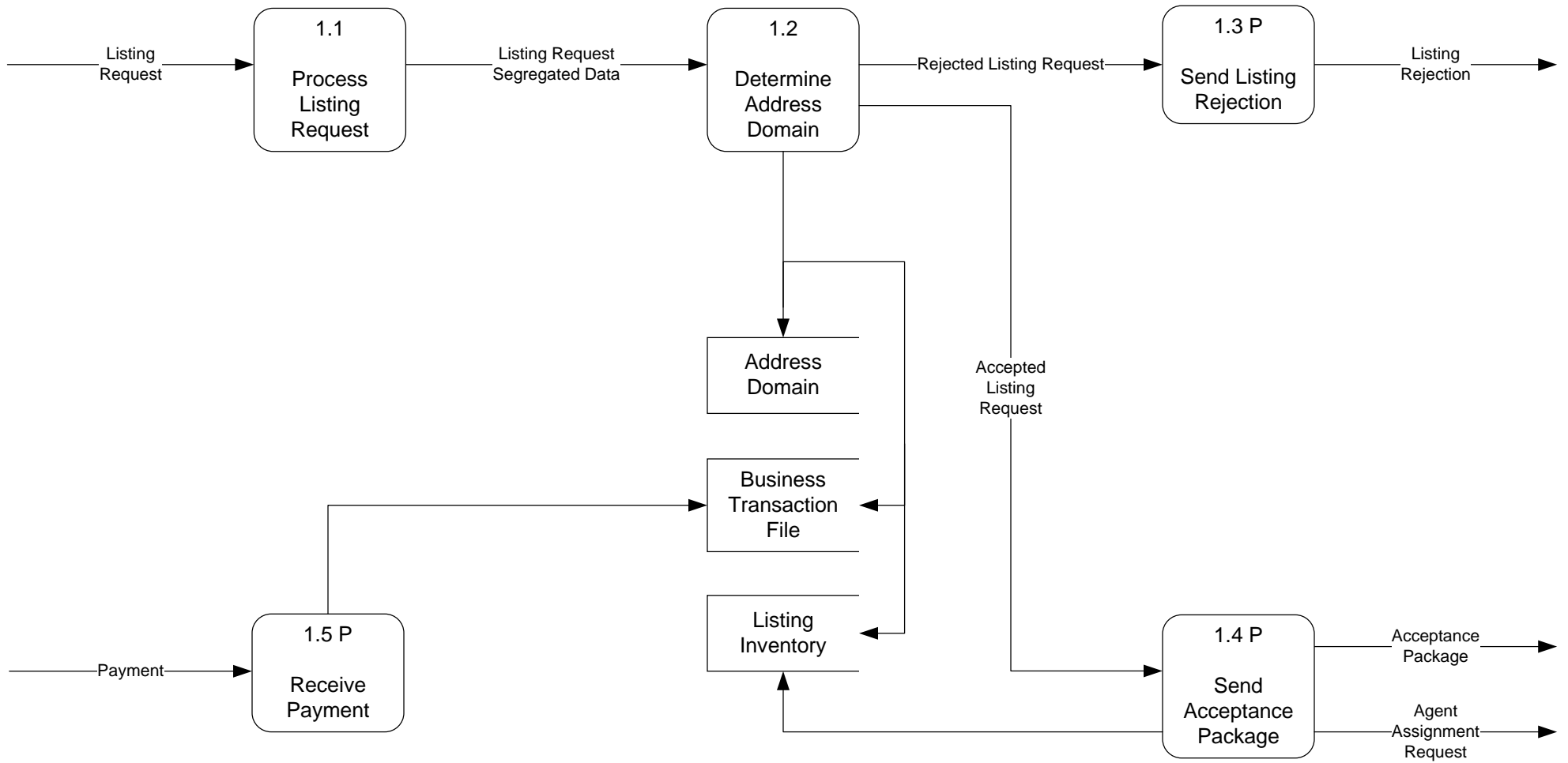




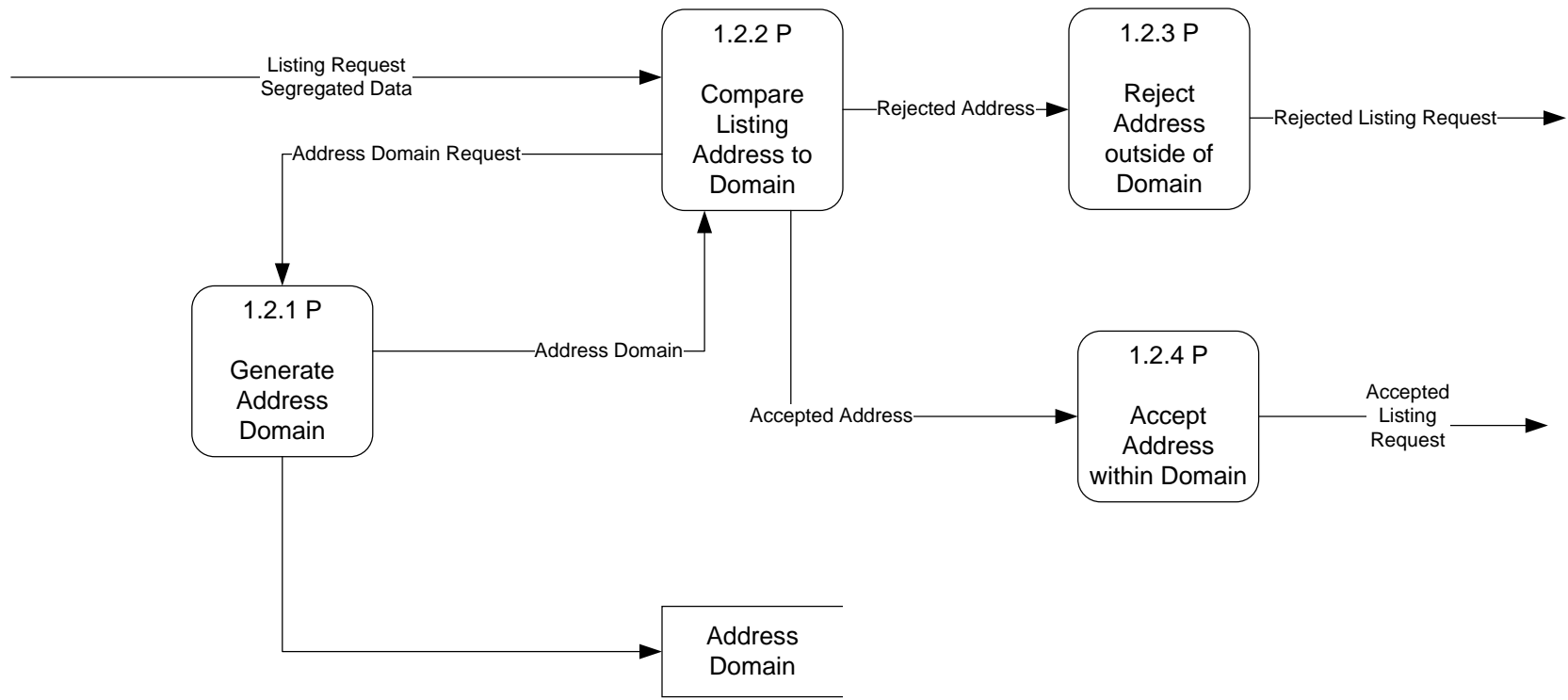


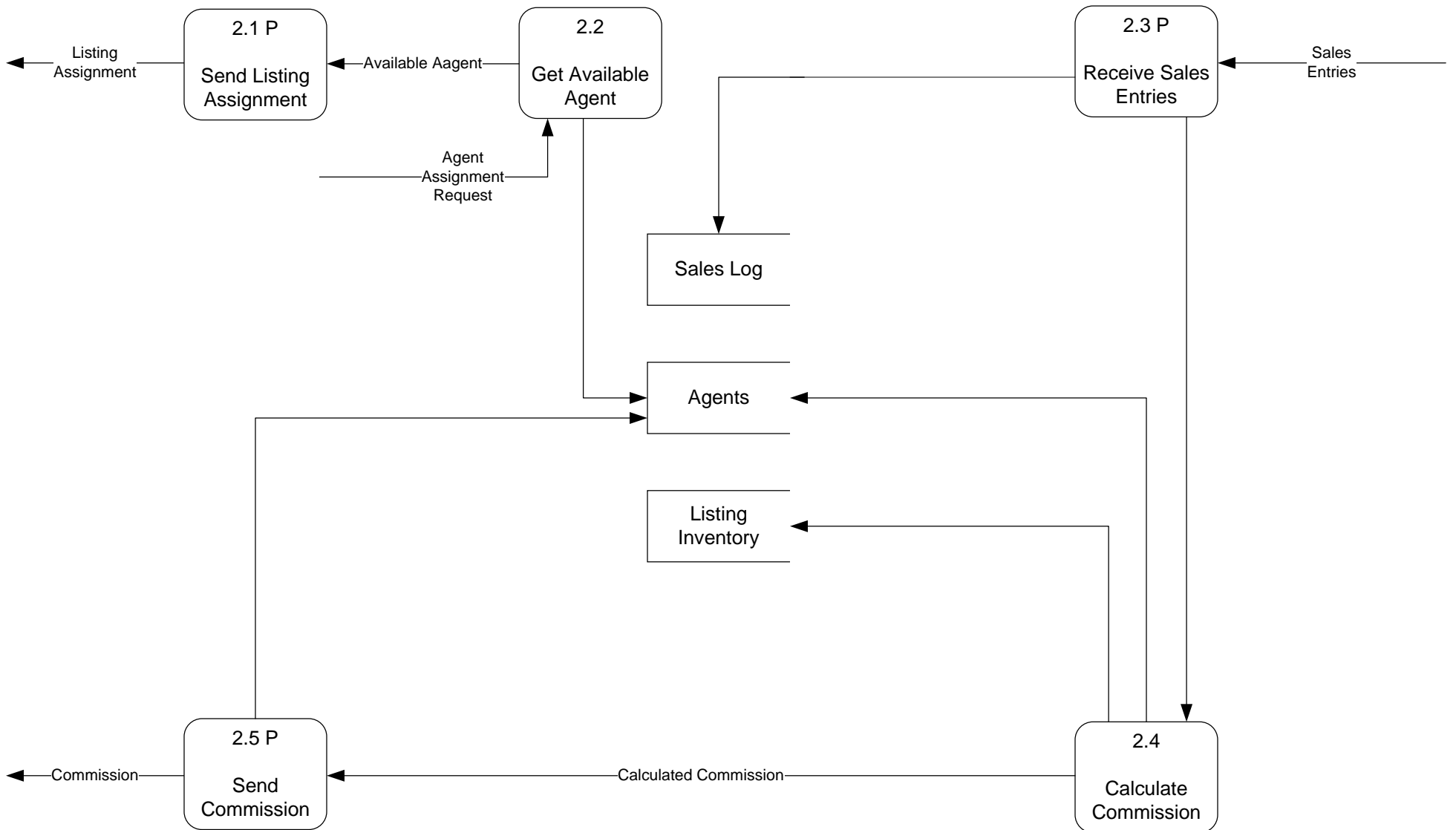


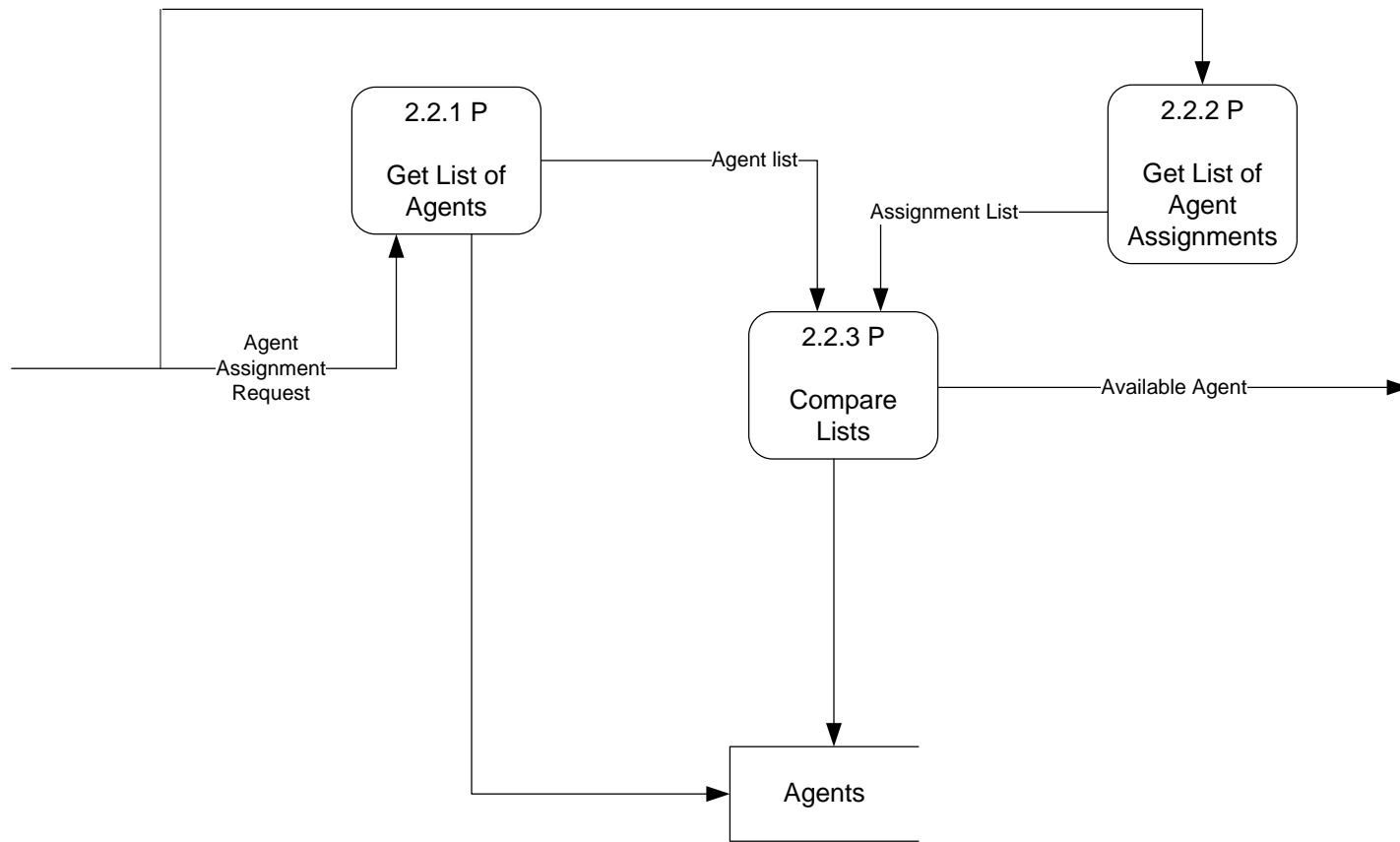


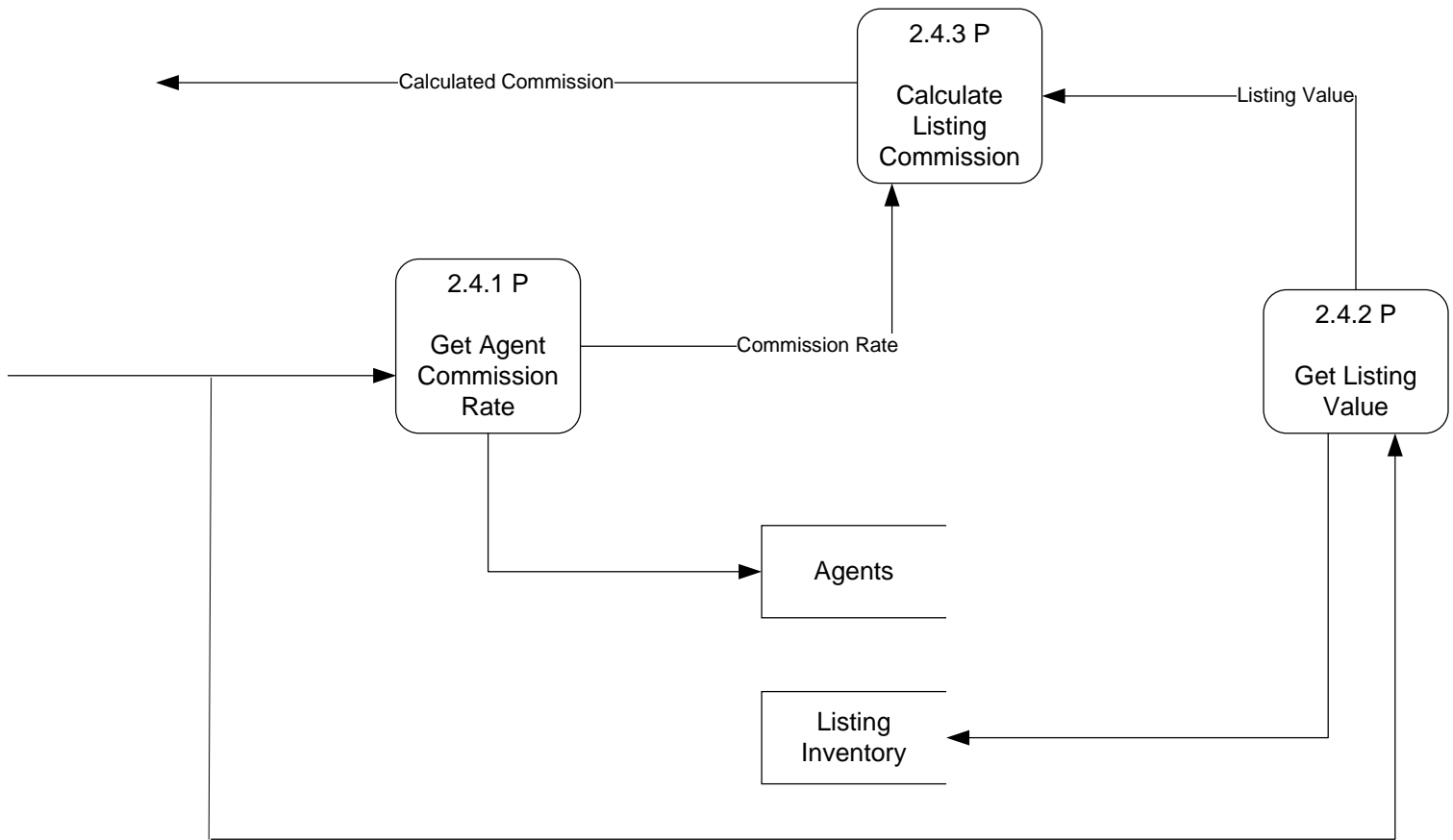


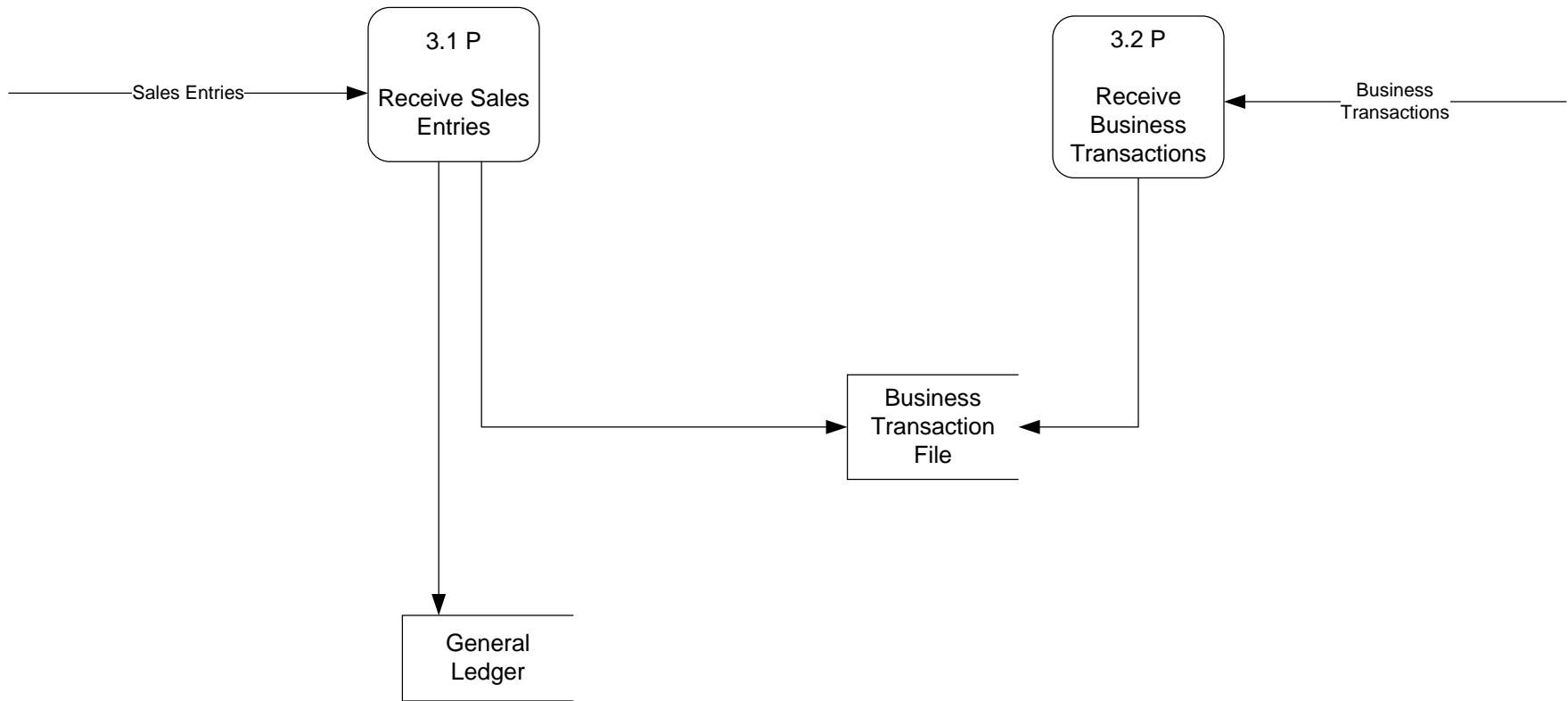


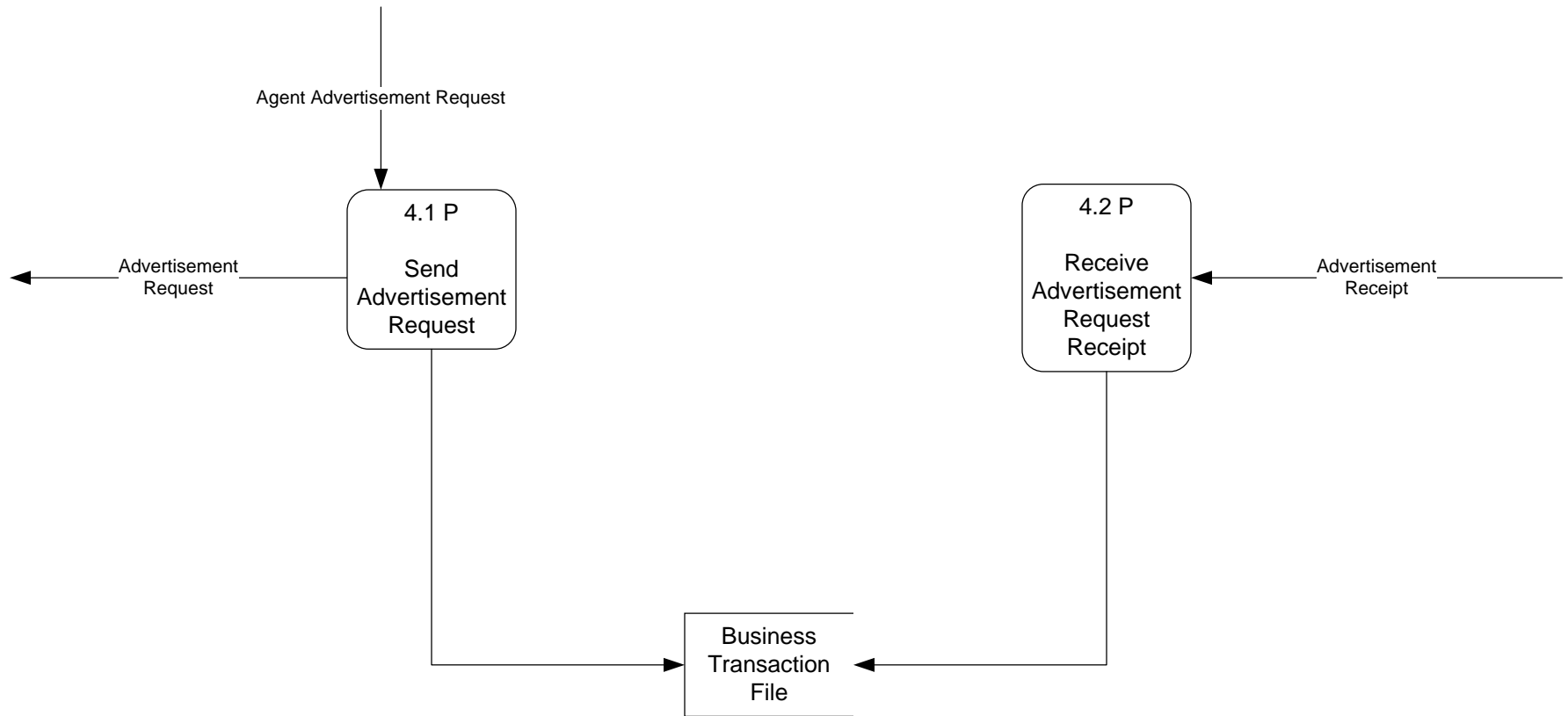


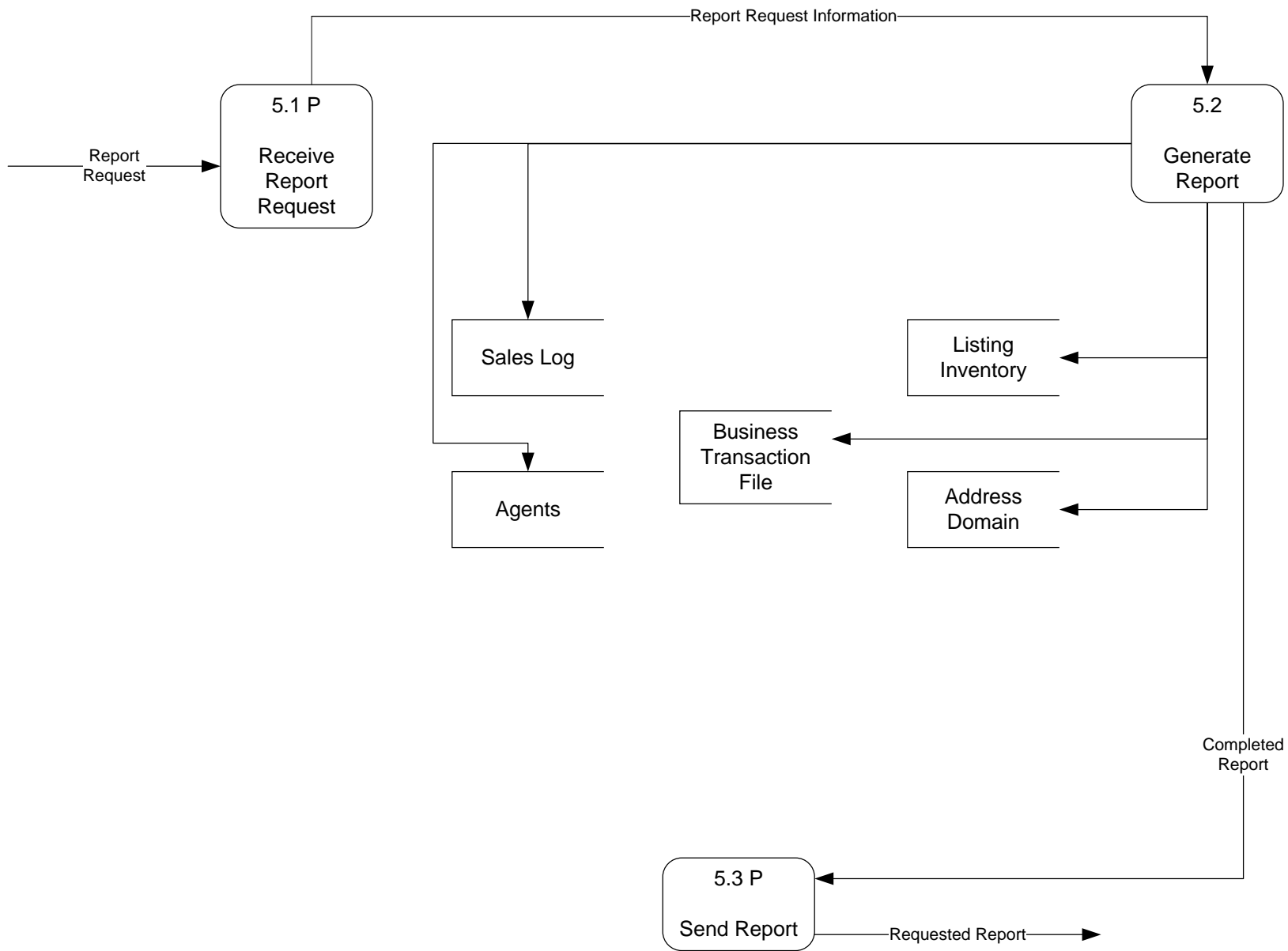


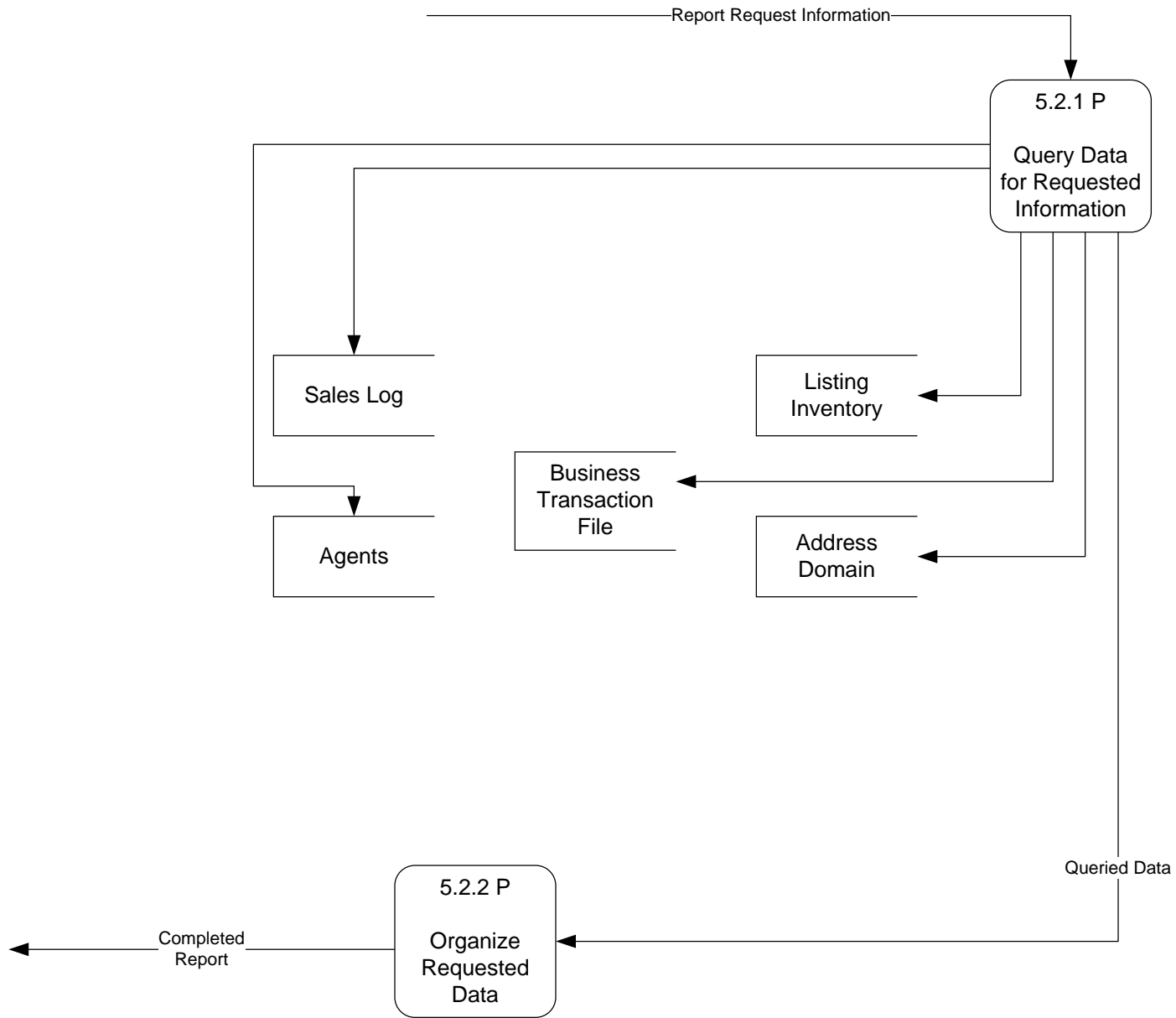


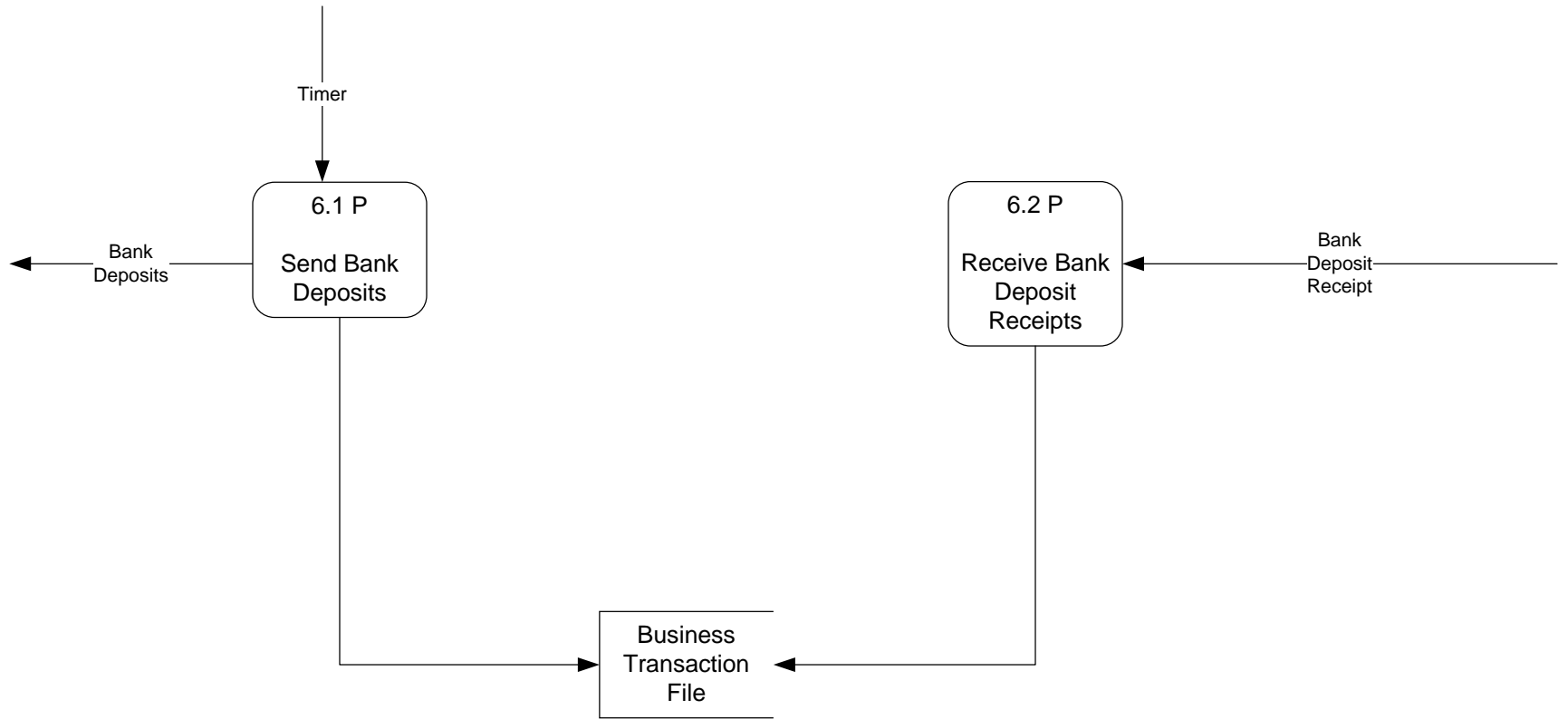












Part IV



Label: RECEIVE LISTING REQUEST

Entry Type: Process

Description: Information from the request is itemized, assigned a variable and forwarded for further processing

Process #: 1.1.1 P

Process Description: Input dataflow: LISTING REQUEST
Output dataflow: LISTING DATA

For each LISTING REQUEST,

LISTING NUMBER = CLIENT L_NAME + LISTING ADDRESS

LISTING VALUE = APPRAISED VALUE

LISTING ADDRESS = ADDRESS

CLIENT NAME = CLIENT L_NAME + CLIENT F_NAME

SEND ADDRESS = CLIENT ADDRESS

Notes:

Label: SEGREGATE LISTING REQUEST DATA

Entry Type: Process

Description: Variables are sorted and categorized

Process #: 1.1.2 P

Process Description: Input dataflow: LISTING DATA
Output dataflow: LISTING REQUEST SEGREGATED DATA

For each LISTING DATA,

LISTING = LISTING NUMBER + LISTING VALUE + LISTING ADDRESS + CLIENT NAME

ADDRESS TO BE VERIFIED = LISTING ADDRESS

Notes:



Label: GENERATE ADDRESS DOMAIN
Entry Type: Process
Description: Create list of addresses that are in area of operation
Process #: 1.2.1 P
Process Description: Input dataflow: ADDRESS DOMAIN REQUEST
Output dataflow: ADDRESS DOMAIN
For every ADDRESS DOMAIN REQUEST,
Get list of addresses

Notes:

Label: COMPARE LISTING ADDRESS TO DOMAIN
Entry Type: Process
Description: Find address in generated list
Process #: 1.2.2 P
Process Description: Input data inflow: LISTING REQUEST SEGREGATED DATA & ADDRESS DOMAIN
Output dataflow: ADDRESS DOMAIN REQUEST, REJECTED ADDRESS & ACCEPTED ADDRESS
For every LISTING REQUEST SEGREGATED DATA,
List array = all addresses
If List array (int) = LISTING ADDRESS
Return LISTING ADDRESS
Else call reject address outside o f domain

Notes:



Label: REJECT ADDRESS OUTSIDE OF DOMAIN

Entry Type: Process

Description: Add all listing input data to rejected address and send rejected listing request

Process #: 1.2.3 P

Process Description: Input dataflow: REJECTED ADDRESS

Output dataflow: REJECTED LISTING REQUEST

For every REJECTED ADDRESS,

REJECTED LISTING REQUEST = REJECTED ADDRESS + LISTING NUMBER +
LISTING VALUE + LISTING ADDRESS + CLIENT NAME

Submit for delivery

Notes:

Label: ACCEPT ADDRESS WITHIN DOMAIN

Entry Type: Process

Description: Add all listing input data to accepted address and send rejected listing request

Process #: 1.2.4 P

Process Description: Input dataflow: ACCEPTED ADDRESS

Output dataflow: ACCEPTED LISTING REQUEST

For every ACCEPTED ADDRESS,

ACCEPTED LISTING REQUEST = ACCEPTED ADDRESS + LISTING NUMBER
+ LISTING VALUE + LISTING ADDRESS + CLIENT NAME

Submit for delivery

Notes:



Label: SEND LISTING REJECTION

Entry Type: Process

Description: Take rejected listing request, apply contact information and submit listing rejection for delivery.

Process #: 1.3 P

Process Description: Input dataflow: REJECTED LISTING REQUEST
Output dataflow: LISTING REJECTION

For each REJECTED LISTING REQUEST,
REJECTED LISTING REQUEST contact information = LISTING REJECTION
delivery location
Submit for delivery

Notes:

Label: SEND ACCEPTANCE PACKAGE

Entry Type: Process

Description: Take accepted listing request, apply contact information and submit acceptance package for delivery.

Process #: 1.4 P

Process Description: Input dataflow: ACCPETED LISTING REQUEST
Output dataflow: ACCEPTANCE PACKAGE

For each ACCEPTED LISTING REQUEST,
ACCEPTED LISTING REQUEST client information = ACCEPTANCE
PACKAGE delivery location
Submit for delivery

Notes:



Label: RECEIVE PAYMENT

Entry Type: Process

Description: Apply payment from clients to business transaction file

Process #: 1.5 P

Process Description: Input dataflow: PAYMENT

For each PAYMENT,

new BALANCE = BALANCE – PAYMENT

Update BUSINESS TRANSACTION FILE

Notes: The output dataflow, update, is linked directly to the data store, BUSINESS TRANSACTION FILE. Being inherent of a dataflow linked to a data store, it is not directly called out.

Label: SEND LISTING ASSIGNMENT

Entry Type: Process

Description: Add available agent to accepted listing

Process #: 2.1 P

Process Description: Input dataflow: AVAILABLE AGENT

Output dataflow: LISTING ASSIGNMENT

For every AVAILABLE AGENT,

LISTING ASSIGNMENT = ACCEPTED LISTING + AVAILABLE AGENT

Notes:



Label: GET LIST OF AGENTS

Entry Type: Process

Description: Generate a list of real estate agents with available time to add additional work hours

Process #: 2.2.1 P

Process Description: Input dataflow: AGENT ASSIGNMENT REQUEST

Output dataflow: AGENT LIST

For each AGENT ASSIGNMENT REQUEST,

List array = Agents < 35 hours per week

Print

Return agents

Notes:

Label: GET LIST OF AGENT ASSIGNMENTS

Entry Type: Process

Description: Generate a list of real estate agents' assigned listings

Process #: 2.2.2 P

Process Description: Input dataflow: AGENT ASSIGNMENT REQUEST

Output dataflow: ASSIGNMENT LIST

For every AGENT ASSIGNMENT REQUEST,

List array = All Assignments

Return Assignments

Notes:



Label: COMPARE LISTS

Entry Type: Process

Description: Filter available agents and assignments for each agent

Process #: 2.2.3 P

Process Description: Input dataflow: AGENT LIST & ASSIGNMENT LIST
Output dataflow: AVAILABLE AGENT

i = 10

While i < AGENT HOURS & ASSIGNMENT LIST <= i

AVAILABLE AGENT = AGENT HOURS + ASSIGNMENT LIST

Return ASSIGNMENT LIST

Else i = i + 10

Notes:

Label: RECEIVE SALES ENTRIES

Entry Type: Process

Description: Log sales at closing

Process #: 2.3 P

Process Description: Input dataflow: SALES ENTRIES

For every SALES ENTRY,

ACCEPTED LISTING = CLOSED LISTING

Notes: Output dataflow: is to update sales log data store.



Label: GET AGENT COMMISSION RATE

Entry Type: Process

Description: Query agent data store for commission rate of real estate agent who closed a listing

Process #: 2.4.1 P

Process Description: Input dataflow: REQUEST AGENT COMMISSION RATE
Output dataflow: COMMISSION RATE
For every REQUEST AGENT COMMISSION RATE,
Agent pay rate for List array (int)
Return pay rate

Notes:

Label: GET LISTING VALUE

Entry Type: Process

Description: For closed listing get value of sale

Process #: 2.4.2 P

Process Description: Input dataflow: REQUEST AGENT COMMISSION RATE
Output dataflow: LISTING VALUE
For every REQUEST AGENT COMMISSION RATE,
Listing Value for List array (int)
Return Listing Value

Notes:



Label: CALCULATE LISTING COMMISSION

Entry Type: Process

Description: Multiply commission rate by listing value to determine calculated commission

Process #: 2.4.3 P

Process Description: Input dataflow: COMMISSION RATE & LISTING VALUE

Output dataflow: CALCULATED COMMISSION

$\text{CALCULATED COMMISSION} = \text{COMMISSION RATE} \times \text{LISTING VALUE}$

Return CALCULATED COMMISSION

Notes:

Label: SEND COMMISSION

Entry Type: Process

Description: Take calculated commission and submit it for delivery

Process #: 2.5 P

Process Description: Input dataflow: CALCULATED COMMISSION

Output dataflow: COMMISSION

For every CALCULATED COMMISSION

- Get delivery method for Agent for List array (int)
- Submit for delivery

Notes:



Label: RECEIVE SALES ENTRIES
Entry Type: Process
Description: Generate entry to be logged in the general ledger
Process #: 3.1 P
Process Description: Input dataflow: SALES ENTRIES
For every SALES ENTRY,
 new SALES ENTRY
 Return SALES ENTRY

Notes:

Label: RECEIVE BUSINESS TRANSACTIONS
Entry Type: Process
Description: Generate entries for business transactions to be logged in the business transactions file
Process #: 3.2 P
Process Description: Input dataflow: BUSINESS TRANSACTIONS
For every BUSINESS TRANSACTION,
 new BUSINESS TRANSACTION
 Return BUSINESS TRANSACTION

Notes:



Label: SEND ADVERTISEMENT REQUEST

Entry Type: Process

Description: Add payment to real estate agent advertisement request and send to advertising agency

Process #: 4.1 P

Process Description: Input dataflow: AGENT ADVERTISEMENT REQUEST
Out dataflow: ADVERTISEMENT REQUEST
For each AGENT ADVERTISEMENT REQUEST,
 Get payment amount
 Add payment to ADVERTISEMENT REQUEST
 Submit for delivery

Notes:

Label: RECEIVE ADVERTISEMENT REQUEST RECEIPT

Entry Type: Process

Description: Update business transaction file data store

Process #: 4.2 P

Process Description: Input dataflow: ADVERTISEMENT RECEIPT
For every ADVERTISEMENT RECEIPT,
 Update BUSINESS TRANSACTION FILE

Notes:



Label: RECEIVE REPORT REQUEST
Entry Type: Process
Description: Segregate data in report request
Process #: 5.1 P
Process Description: Input dataflow: REPORT REQUEST
Output dataflow: REPORT REQUEST INFORMATION
For each REPORT RERQUEST,
REPORT REQUEST = DATA QUERY
REPORT REQUEST INFORMATION = DATA QUERY
Return REPORT REQUEST INFORMATION

Notes:

Label: QUERY DATA FOR REQUESTED INFORMATION
Entry Type: Process
Description: Query data stores for requested data
Process #: 5.2.1 P
Process Description: Input dataflow: REPORT REQUEST INFORMATION
Output dataflow: QUERIED DATA
For each REPORT REQUEST INFORMATION,
Run Query
Return Query

Notes:



Label: SEND BANK DEPOSITS

Entry Type: Process

Description: On a daily basis, bank deposits are sent

Process #: 6.1 P

Process Description: Input dataflow: TIME SETTING

Output dataflow: BANK DEPOSITS

For each end-of-business day,

Sum BANK DEPOSITS

Submit for delivery

Return BANK DEPOSITS

Notes:

Label: RECEIVE BANK DEPOSITS RECEIPT

Entry Type: Process

Description: Update business transaction file data store

Process #: 6.2 P

Process Description: Input dataflow: BANK DEPOSIT RECEIPT

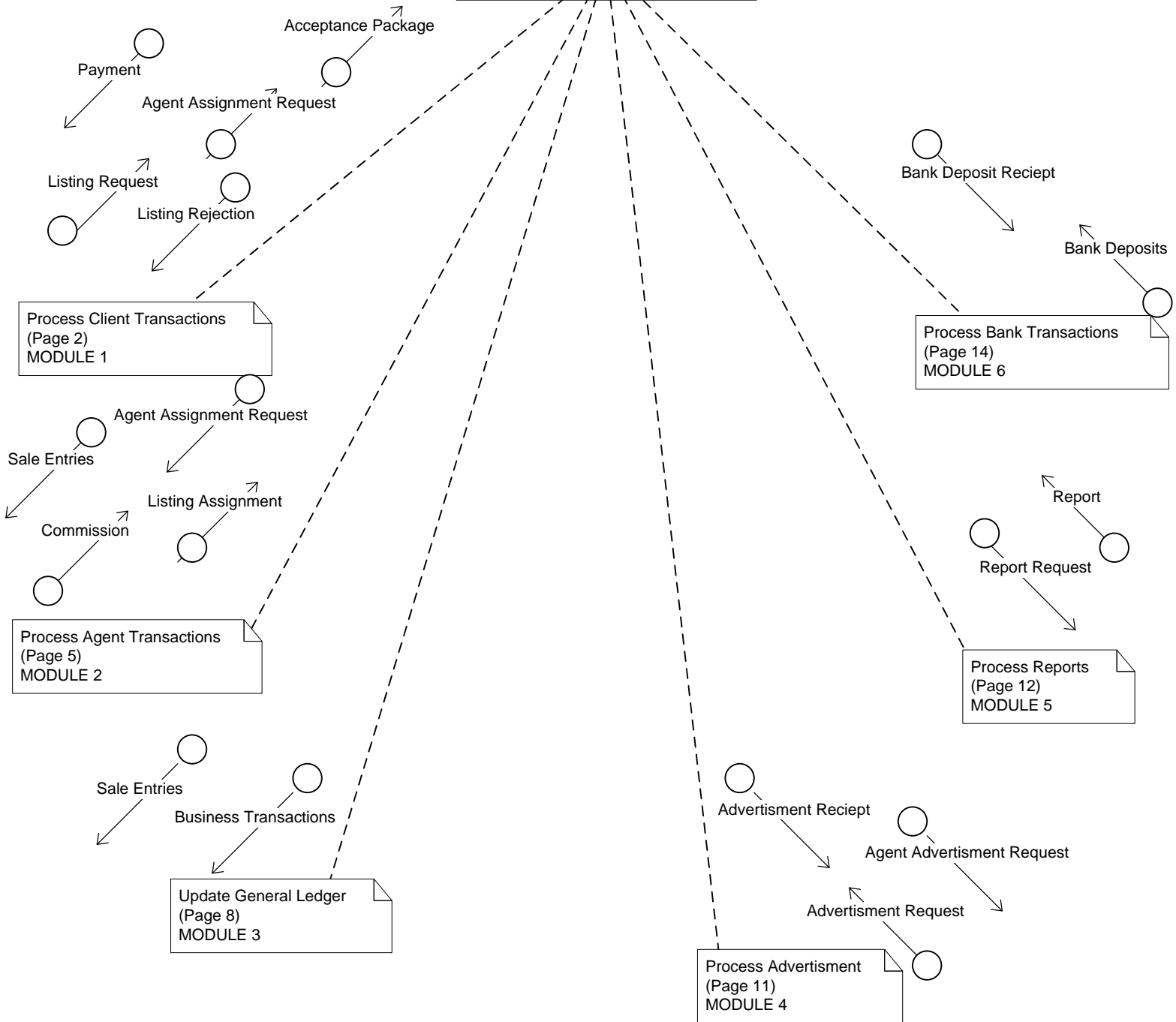
For each BANK DEPOSIT RECEIPT,

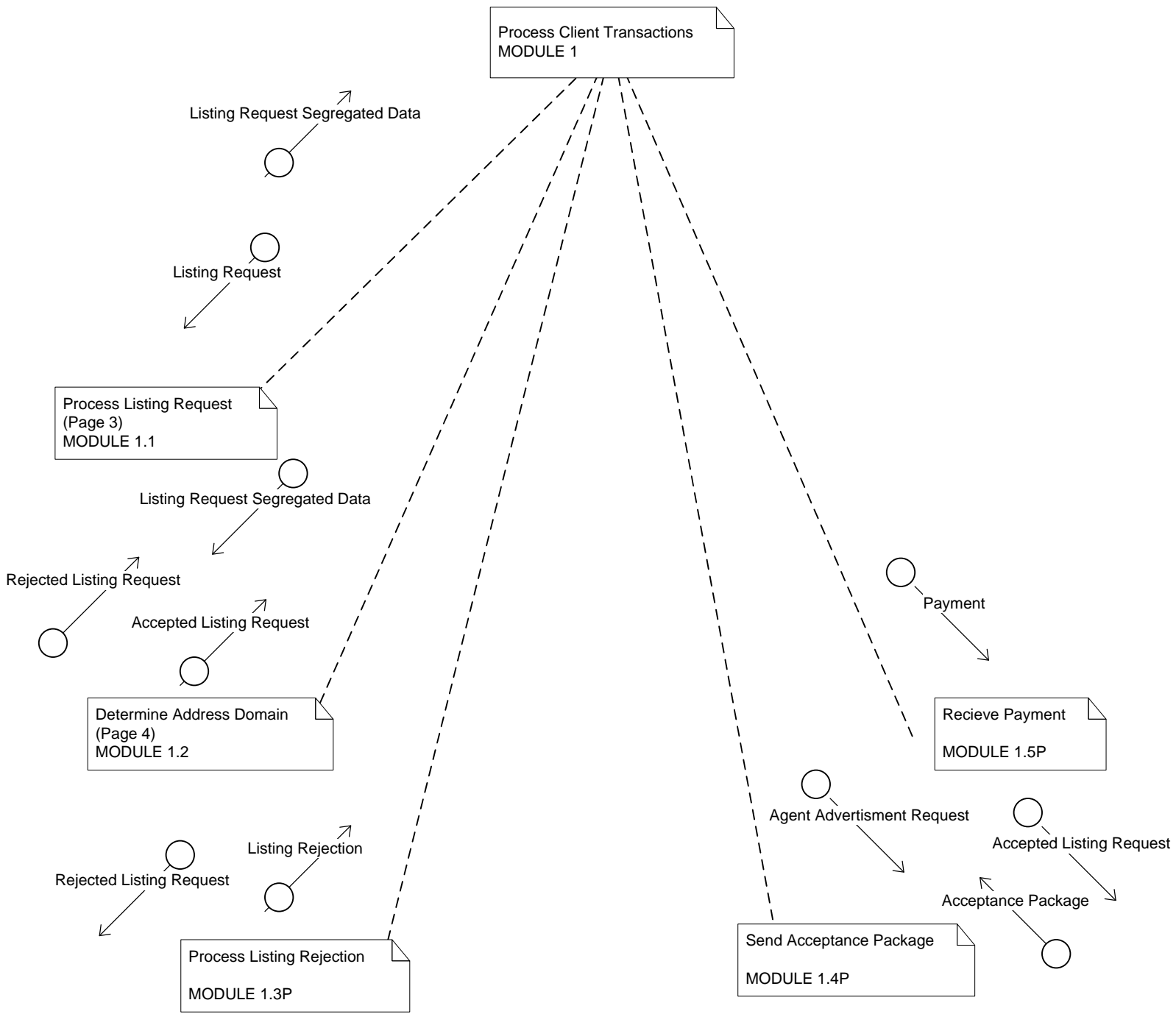
Update BUSINESS TRANSACTION FILE data store

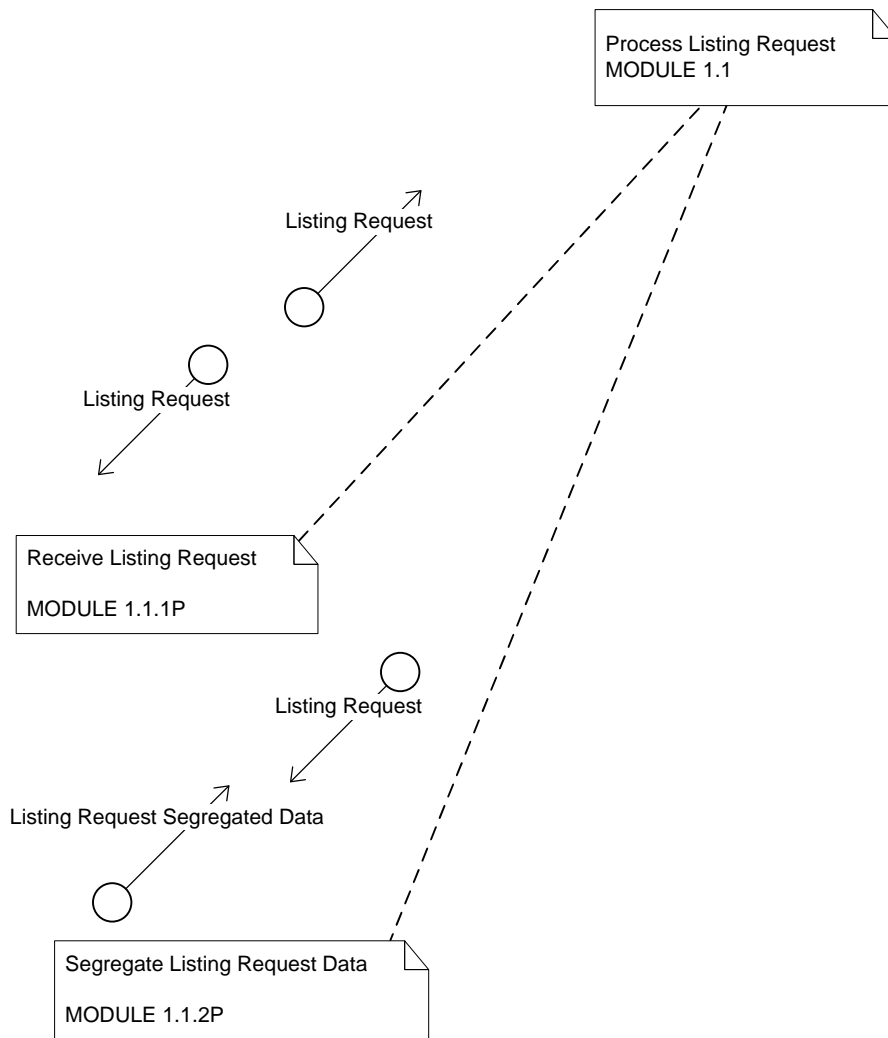
Notes:

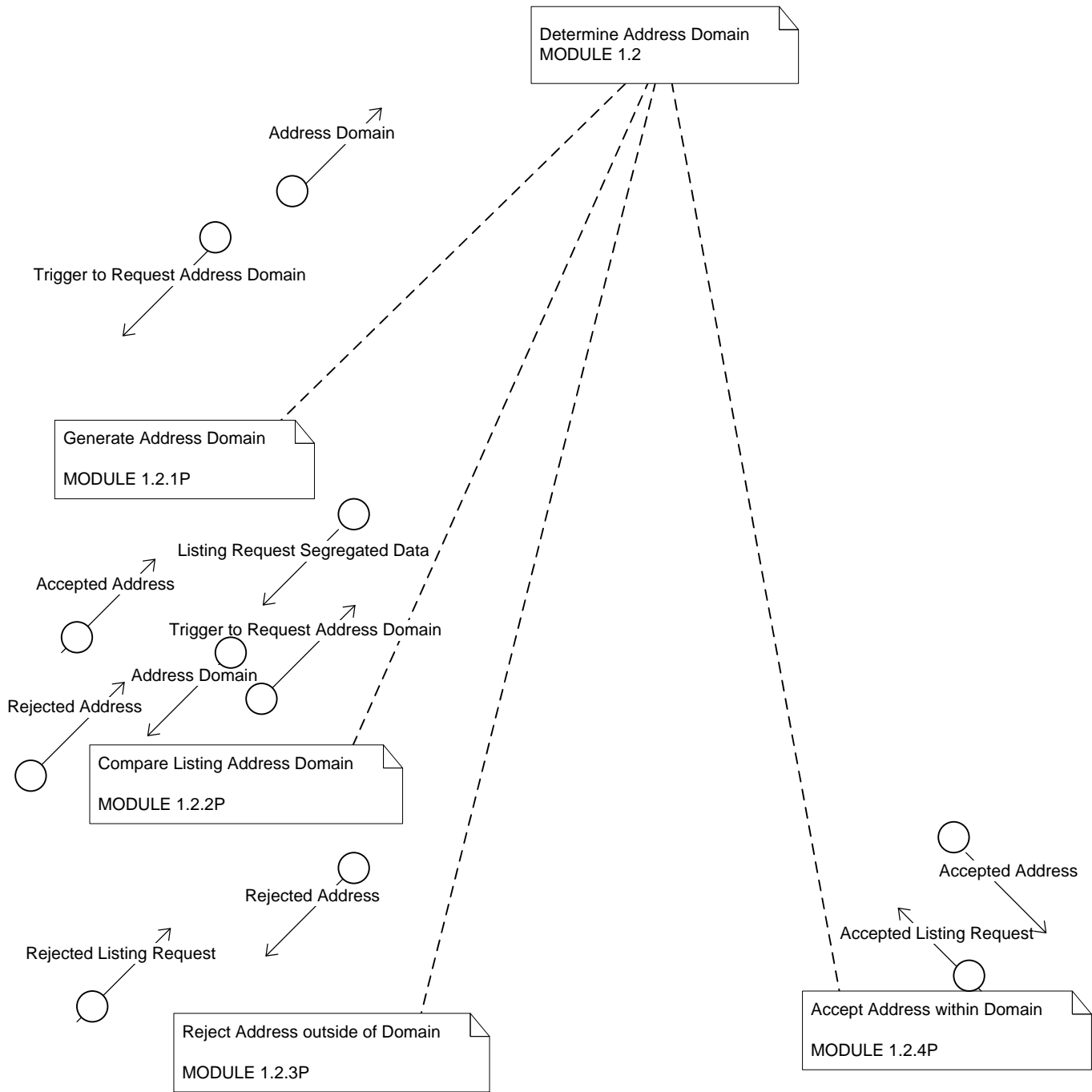
Part V

Real Estate Sales Tracking System
MODULE 0

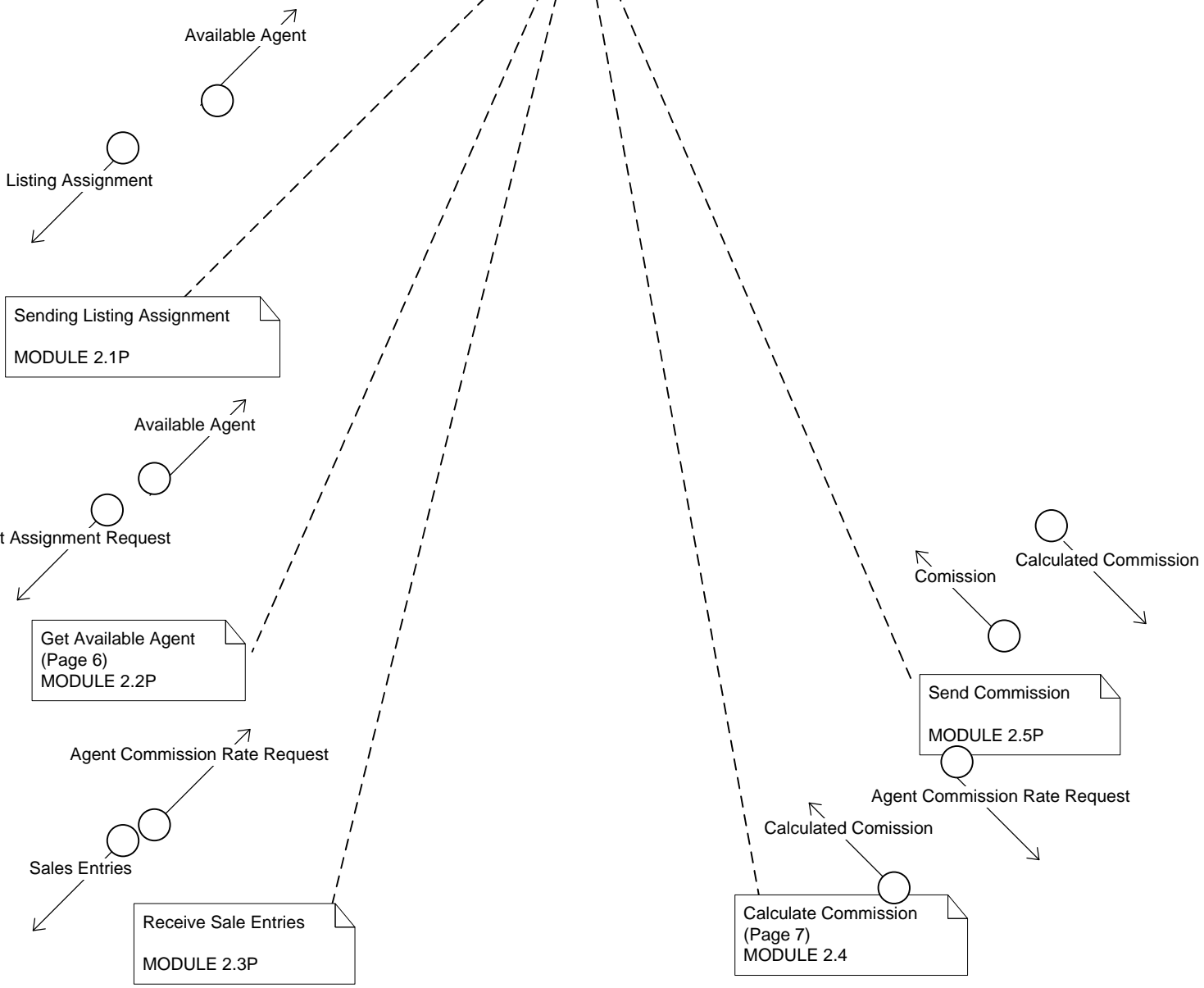


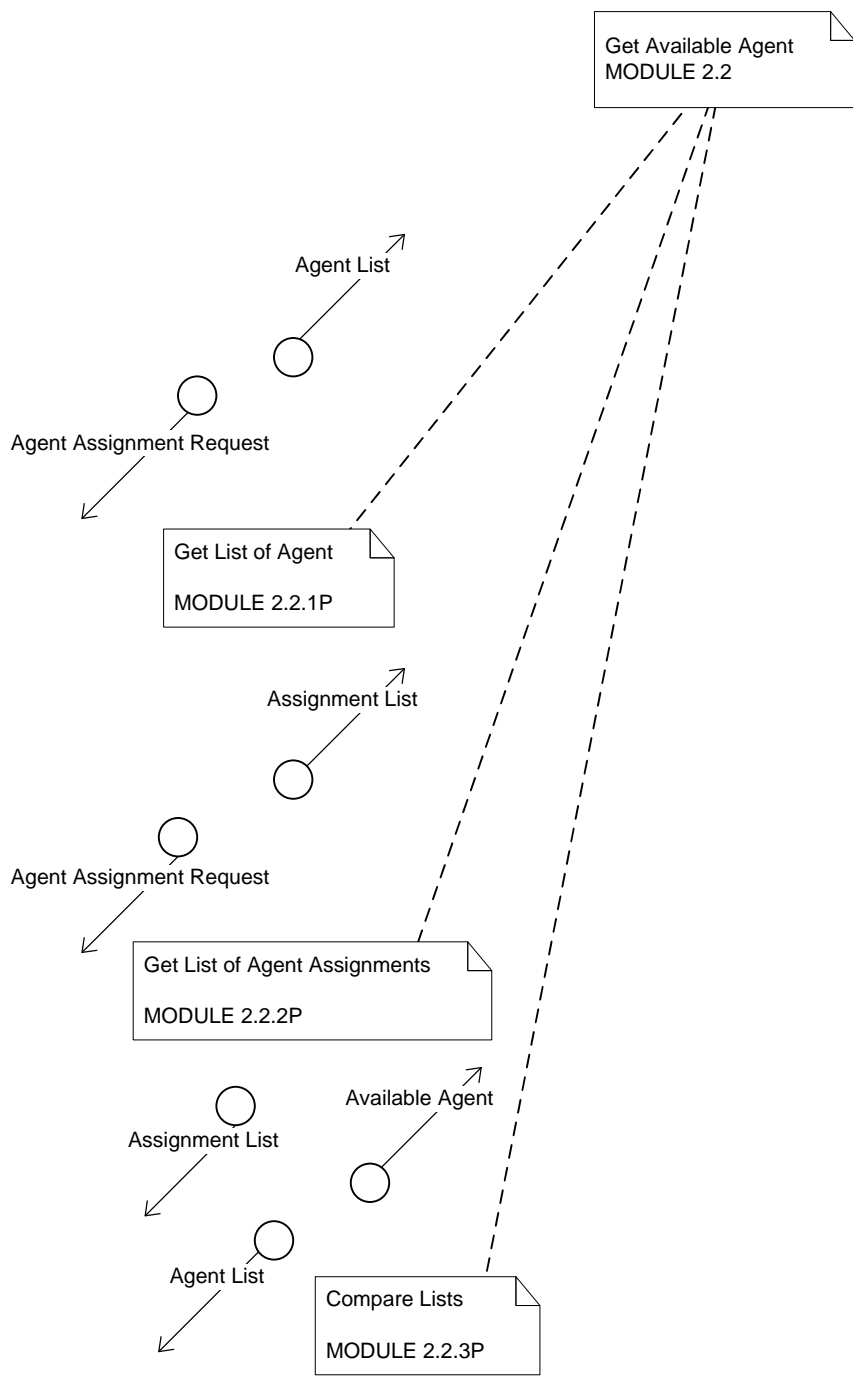






Process Agent Transaction
MODULE 2





Calculate Comission
MODULE 2.4

Commission Rate



Agent Commission Rate Request

Get Agent Commission Rate
MODULE 2.4.1P

Listing Value



Agent Commission Rate Request

Get Listing Value
MODULE 2.4.2P

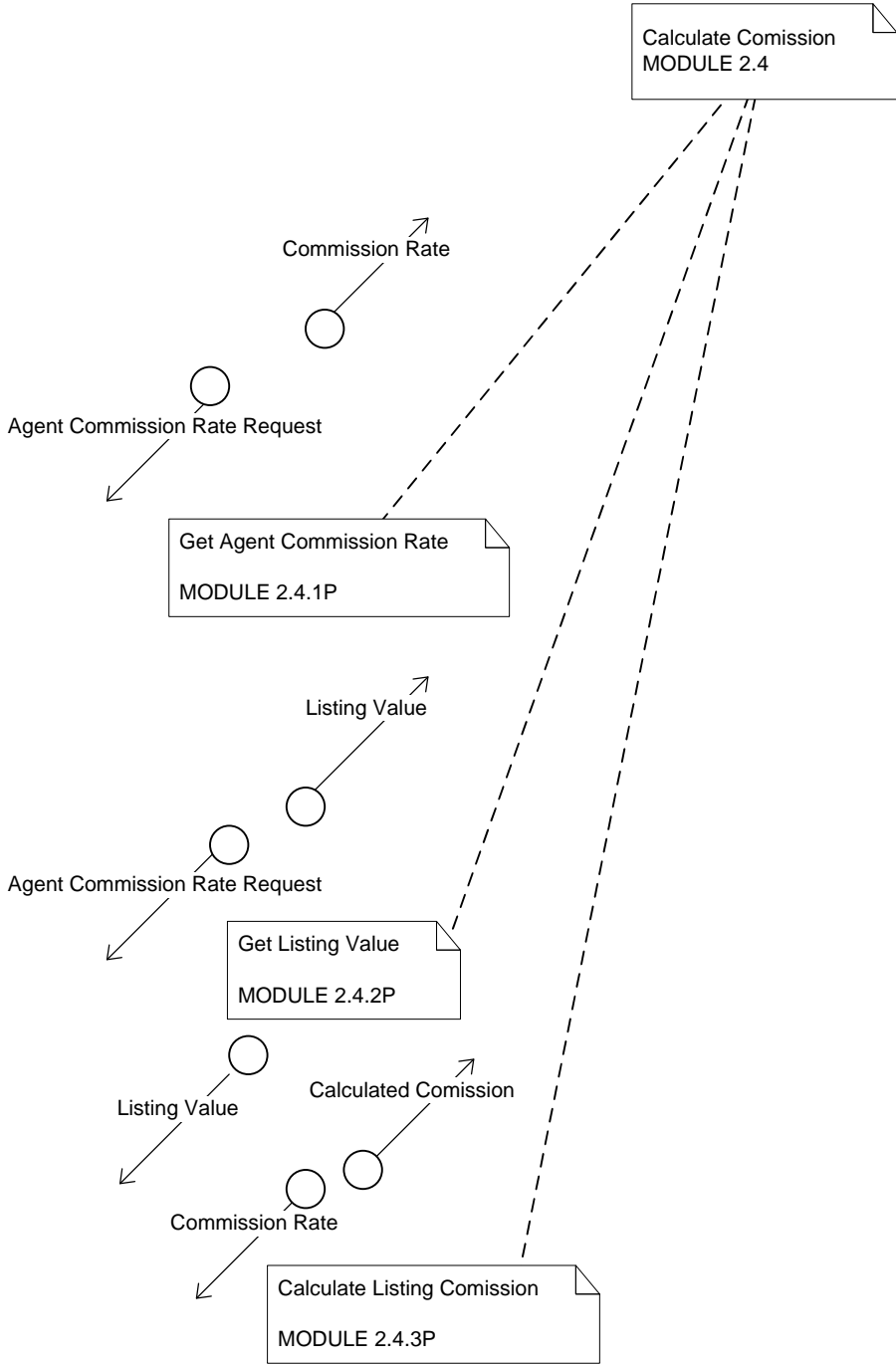
Listing Value



Calculated Comission

Commission Rate

Calculate Listing Comission
MODULE 2.4.3P



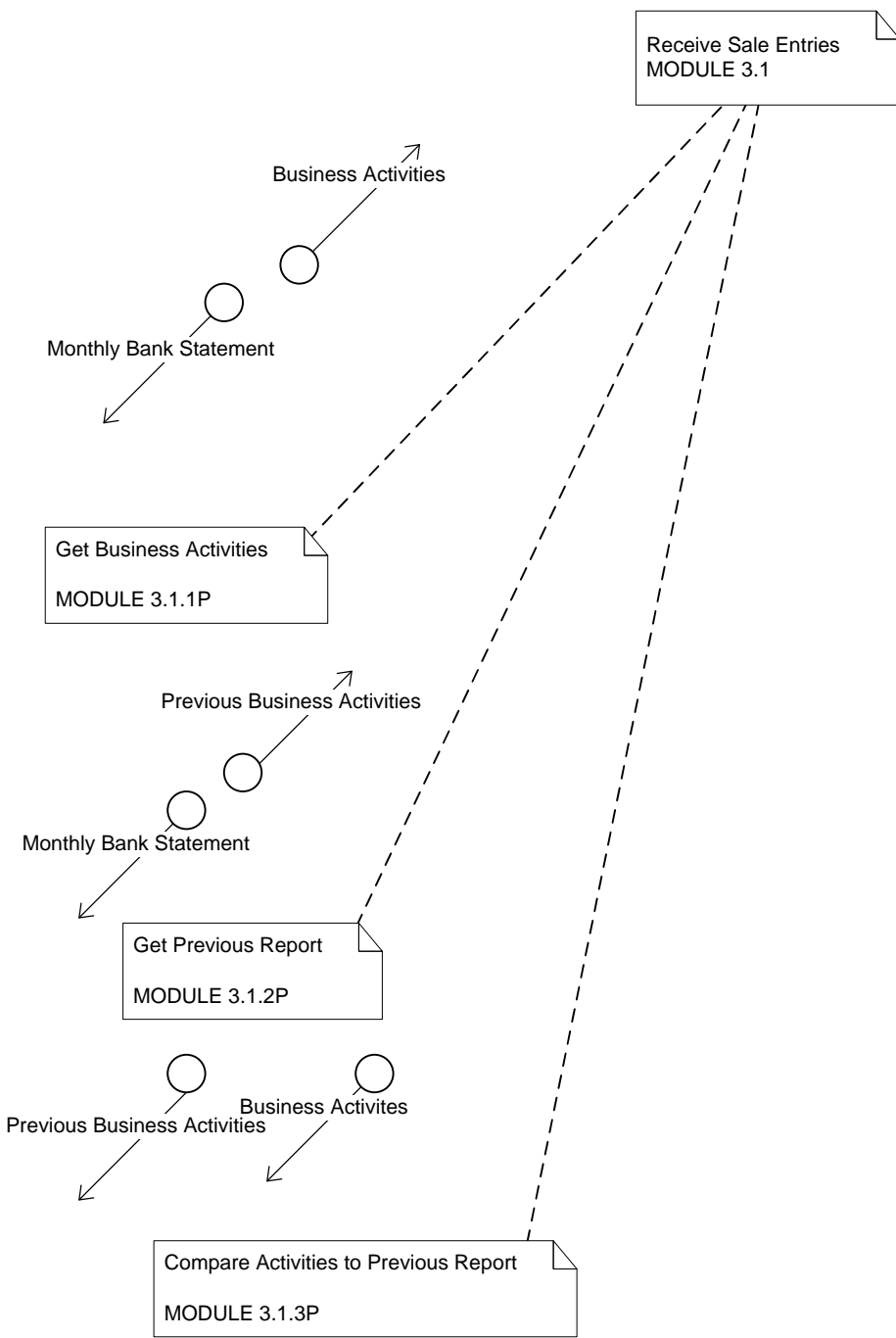
Update General Ledger
MODULE 3

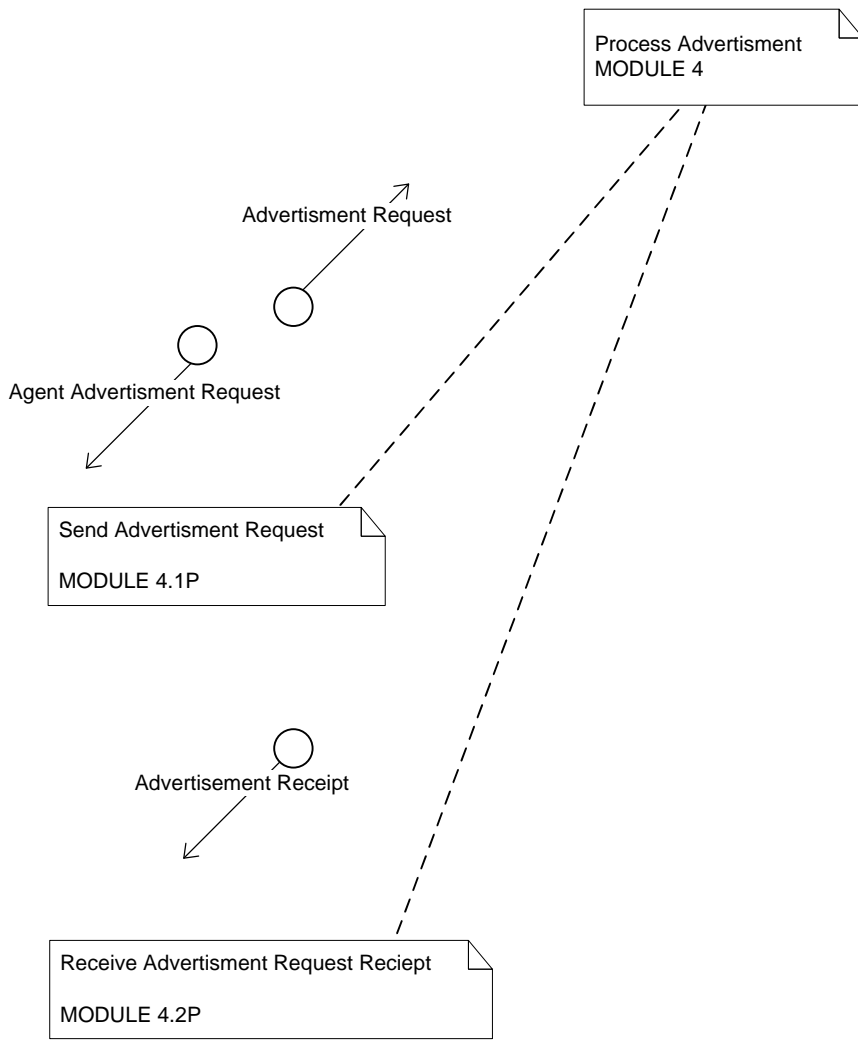
Business Transactions

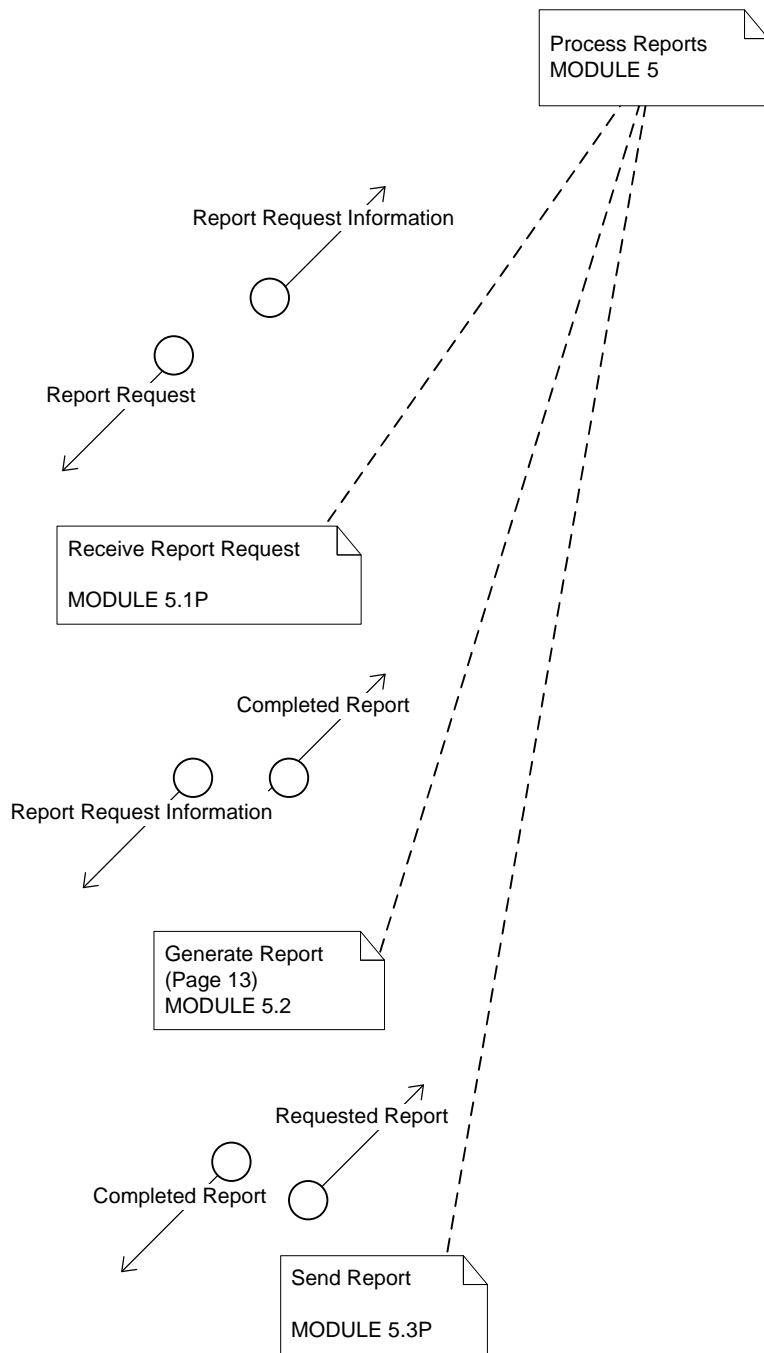
Sale Entries

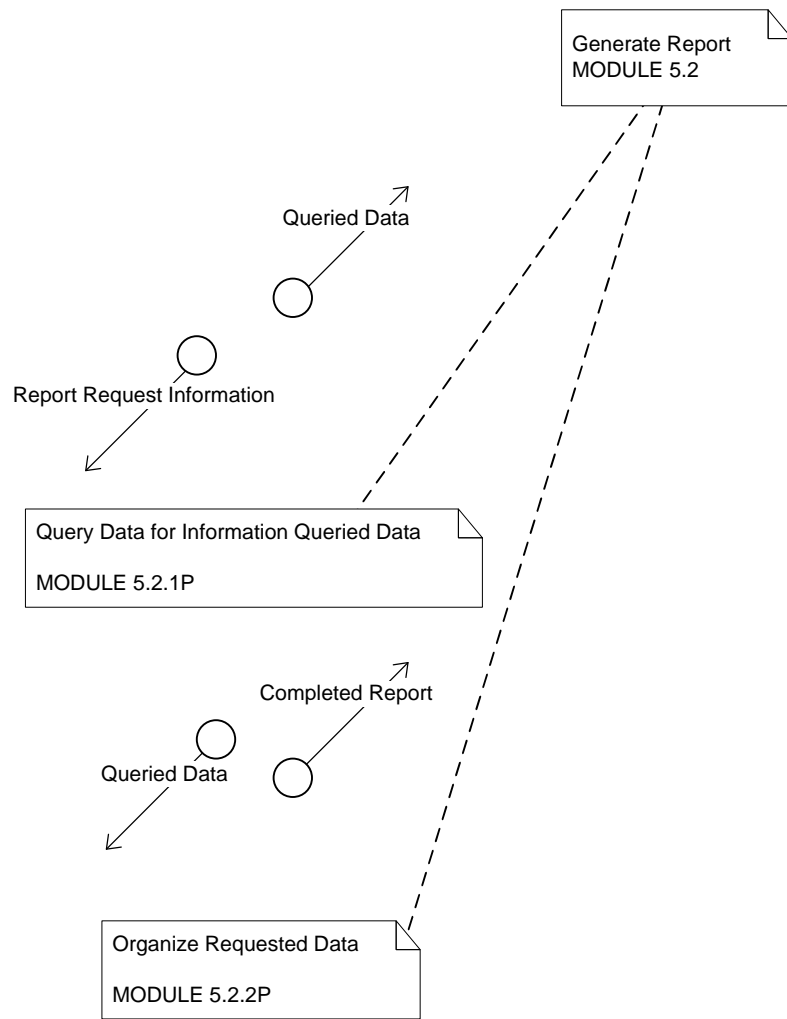
Receive Sale Entries
(Page 9)
MODULE 3.1

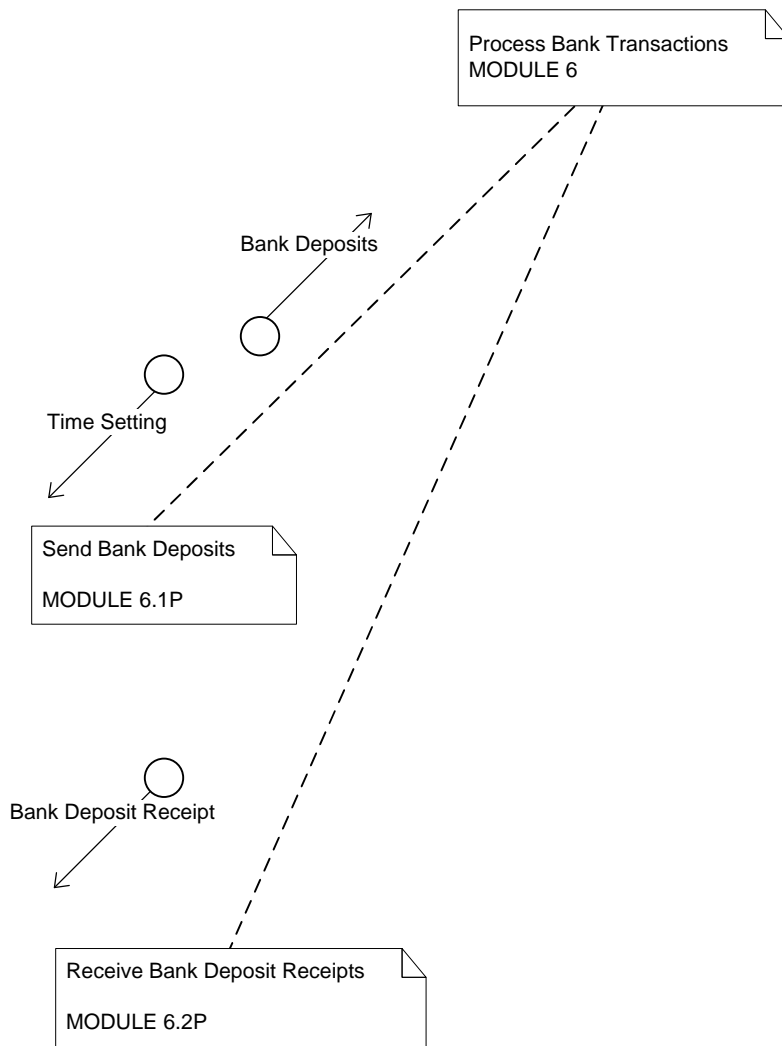
Receive Monthly Bank Deposit Report
MODULE 3.2.P



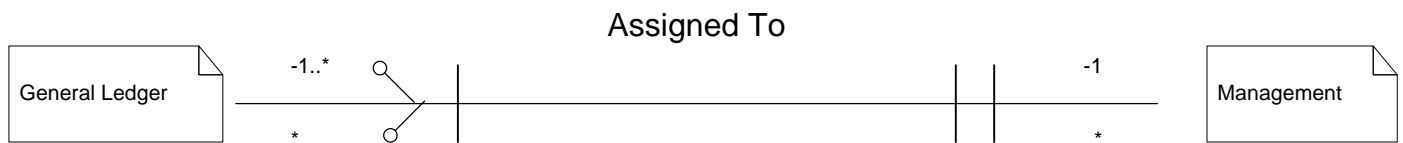
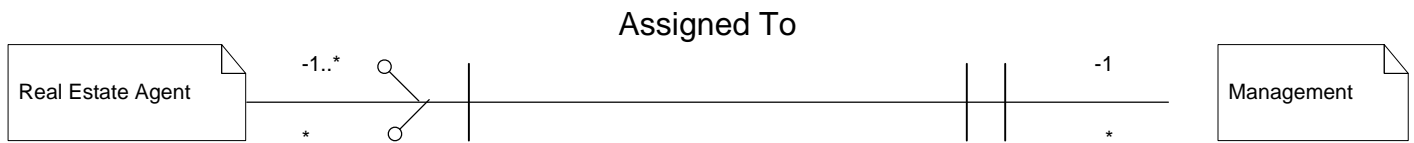
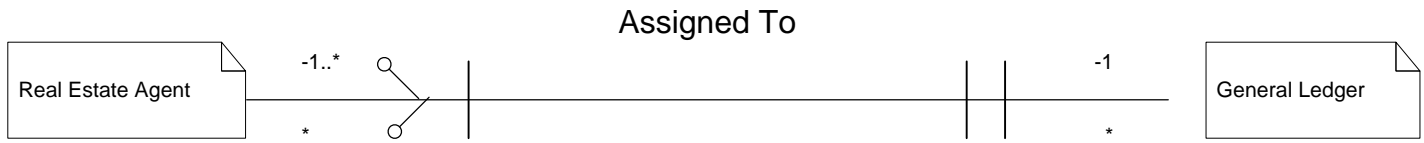
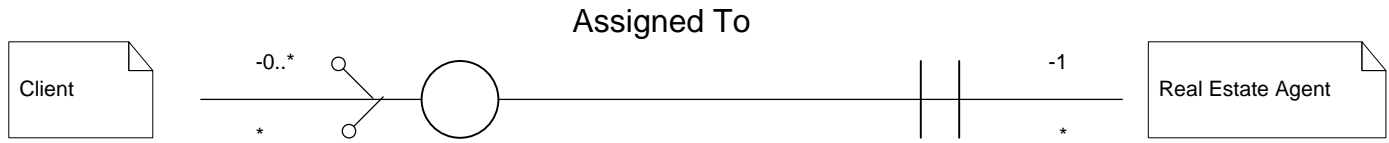
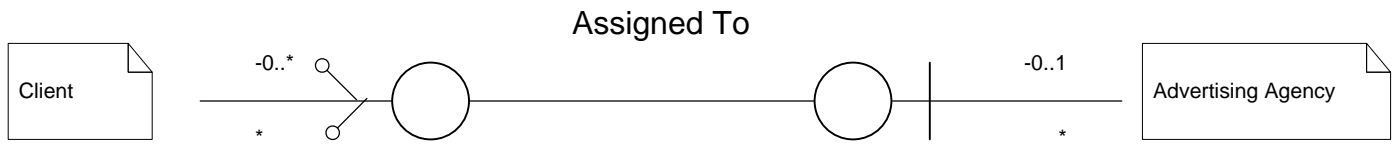






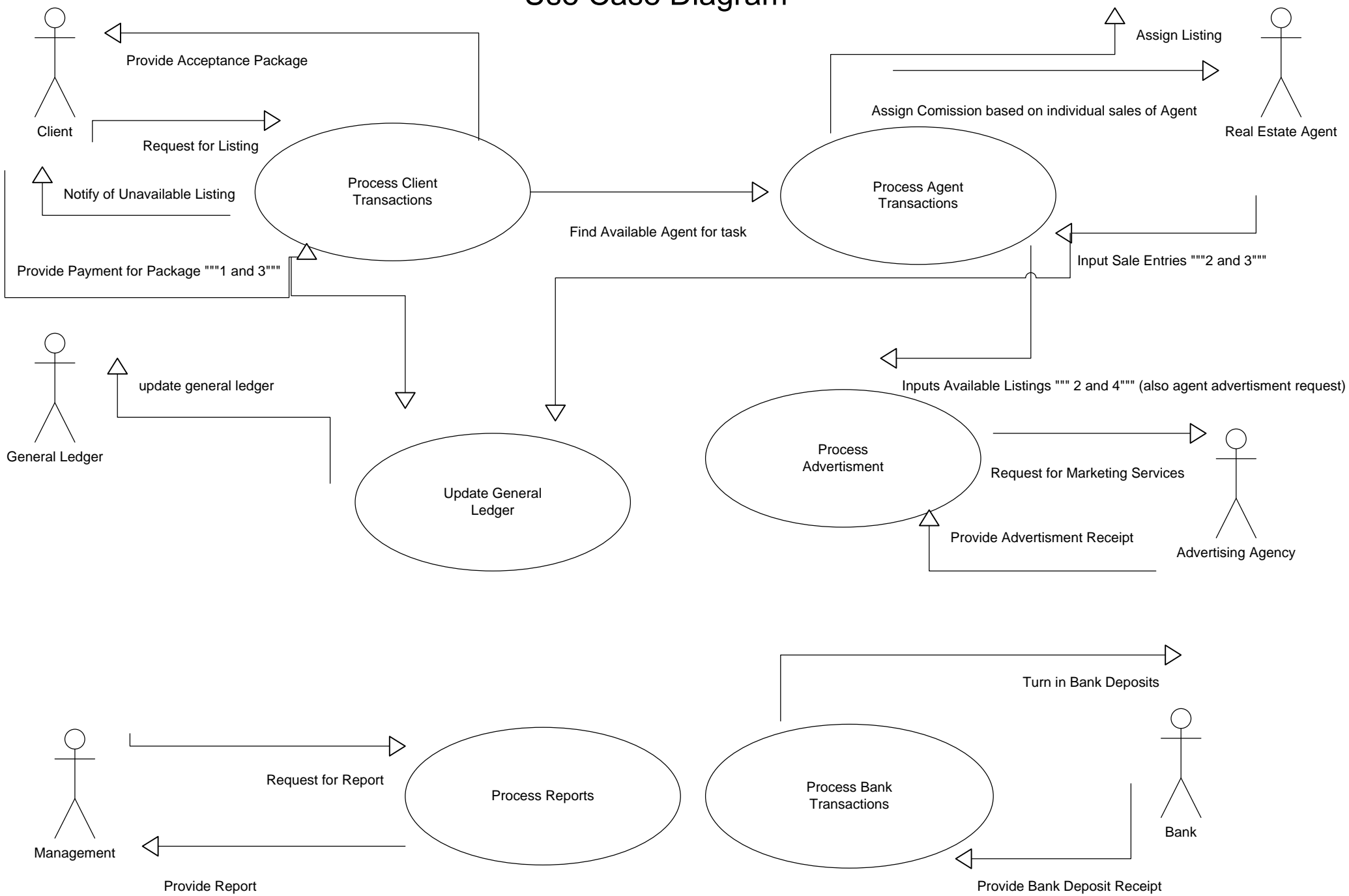


Entity Relationship Diagram (Database)

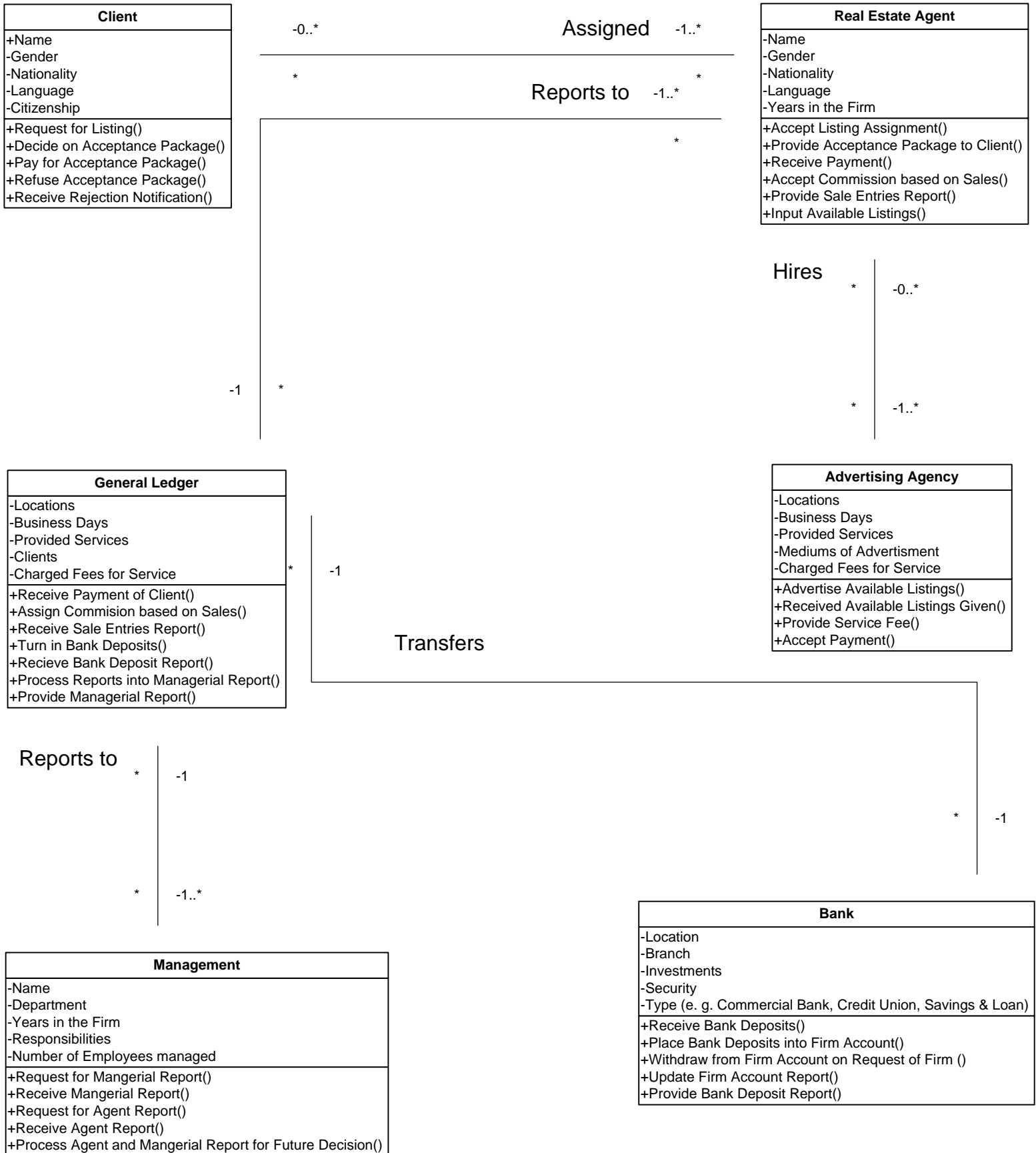


Part VI

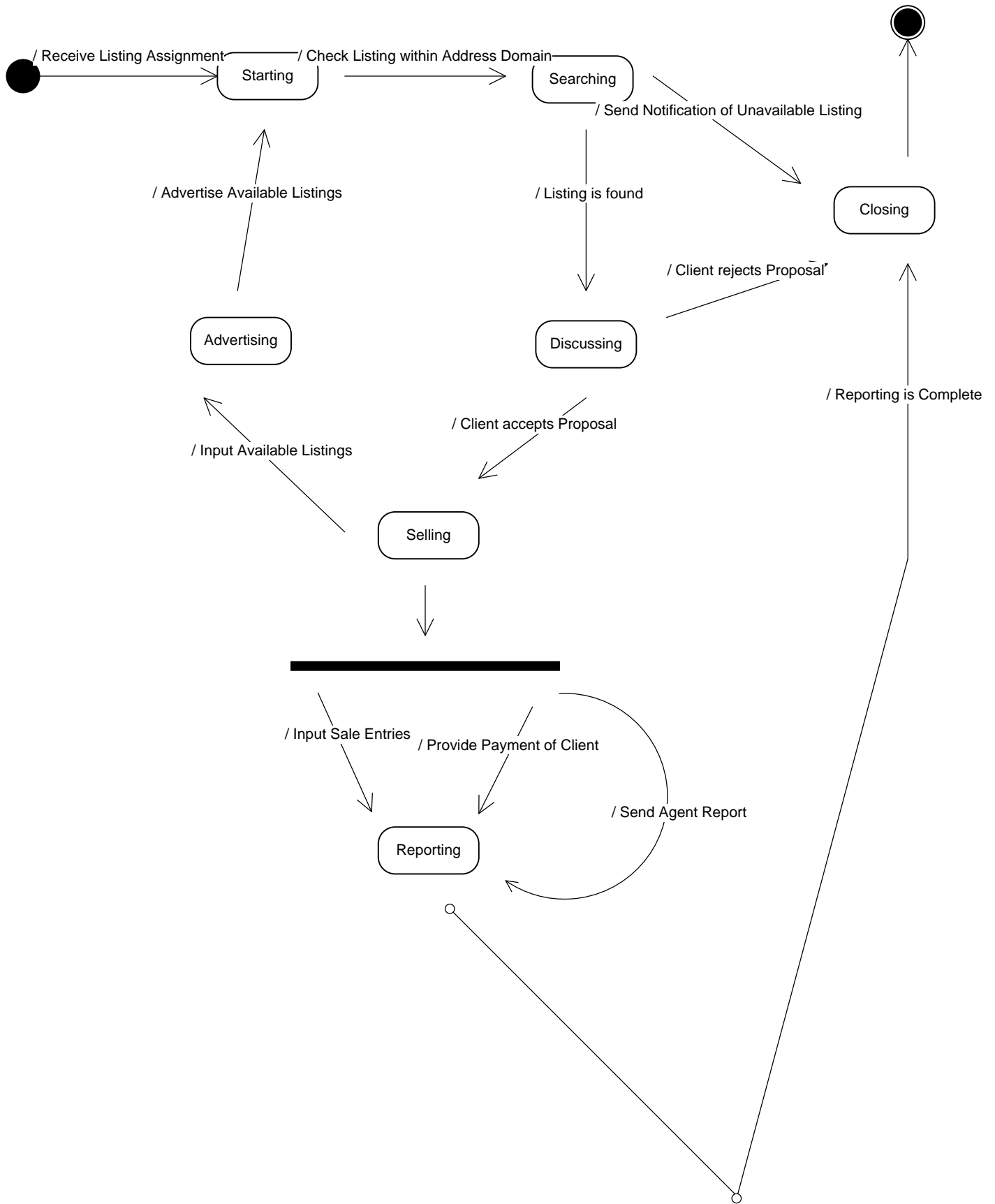
Use Case Diagram



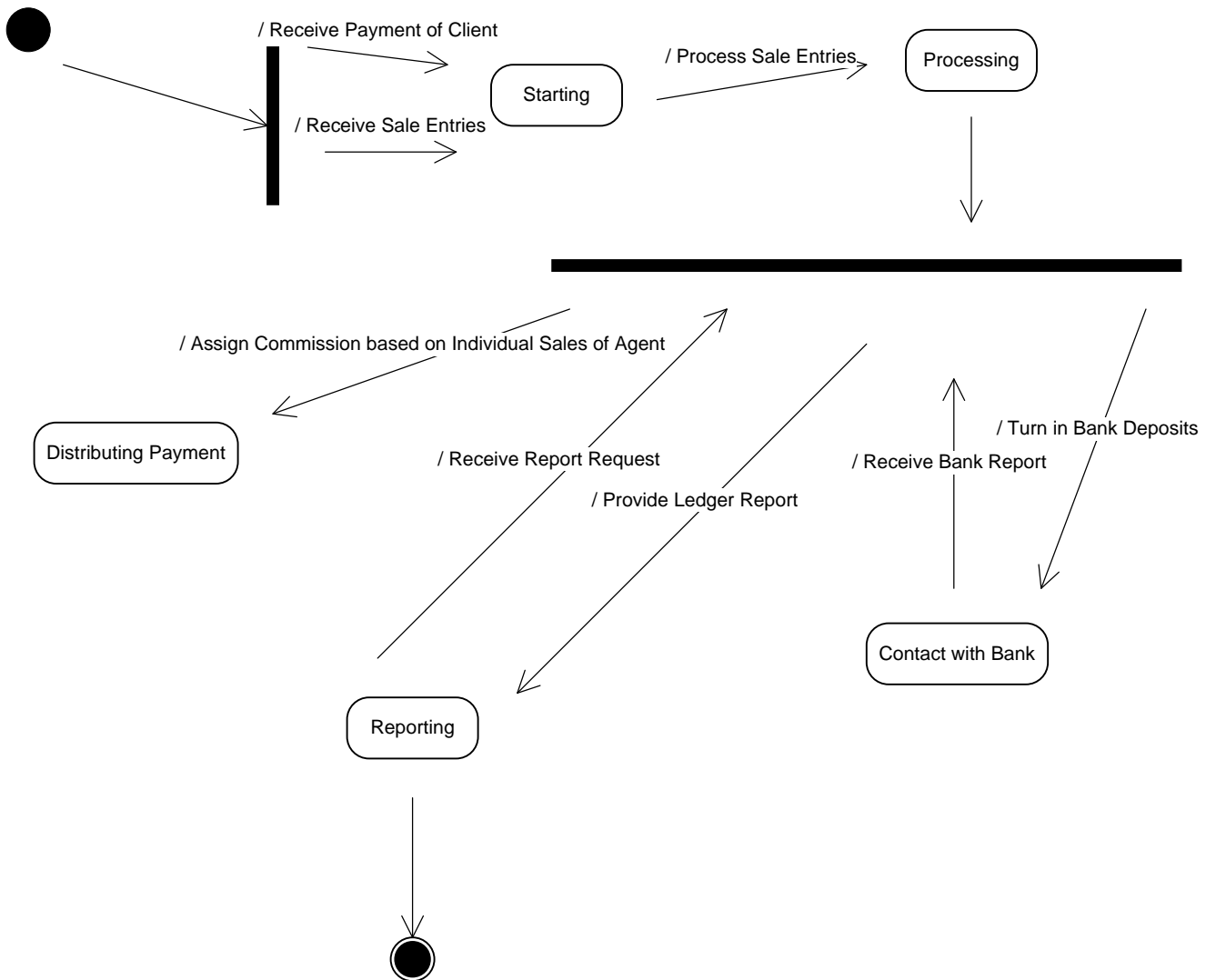
Class Diagram



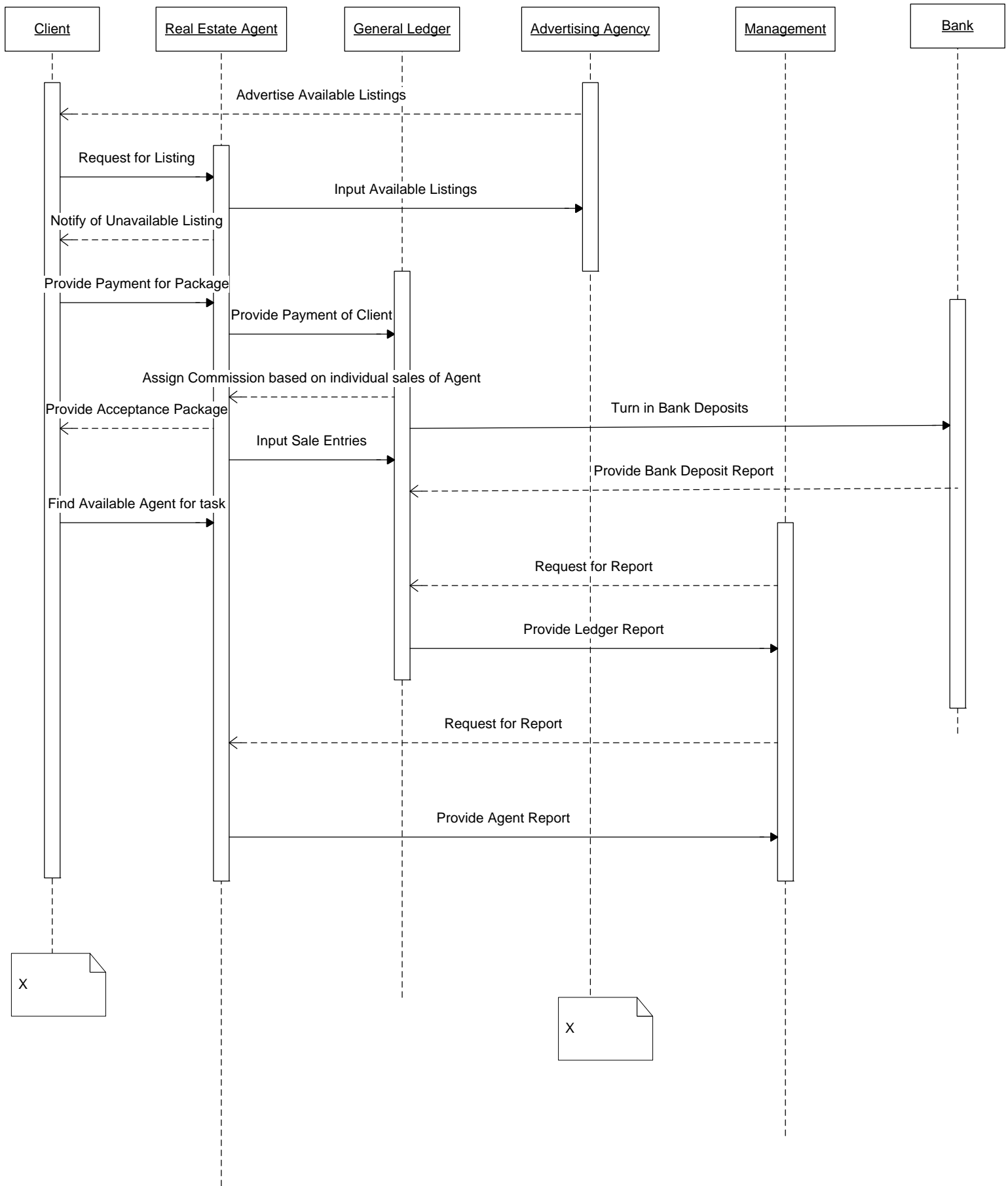
State Transition Diagram for Real Estate Agent



State Transition Diagram for General Ledger



THIS IS THE SEQUENCE DIAGRAM DISPLAYING THE DFD CONTEXT DIAGRAM.
It's based on the entities.



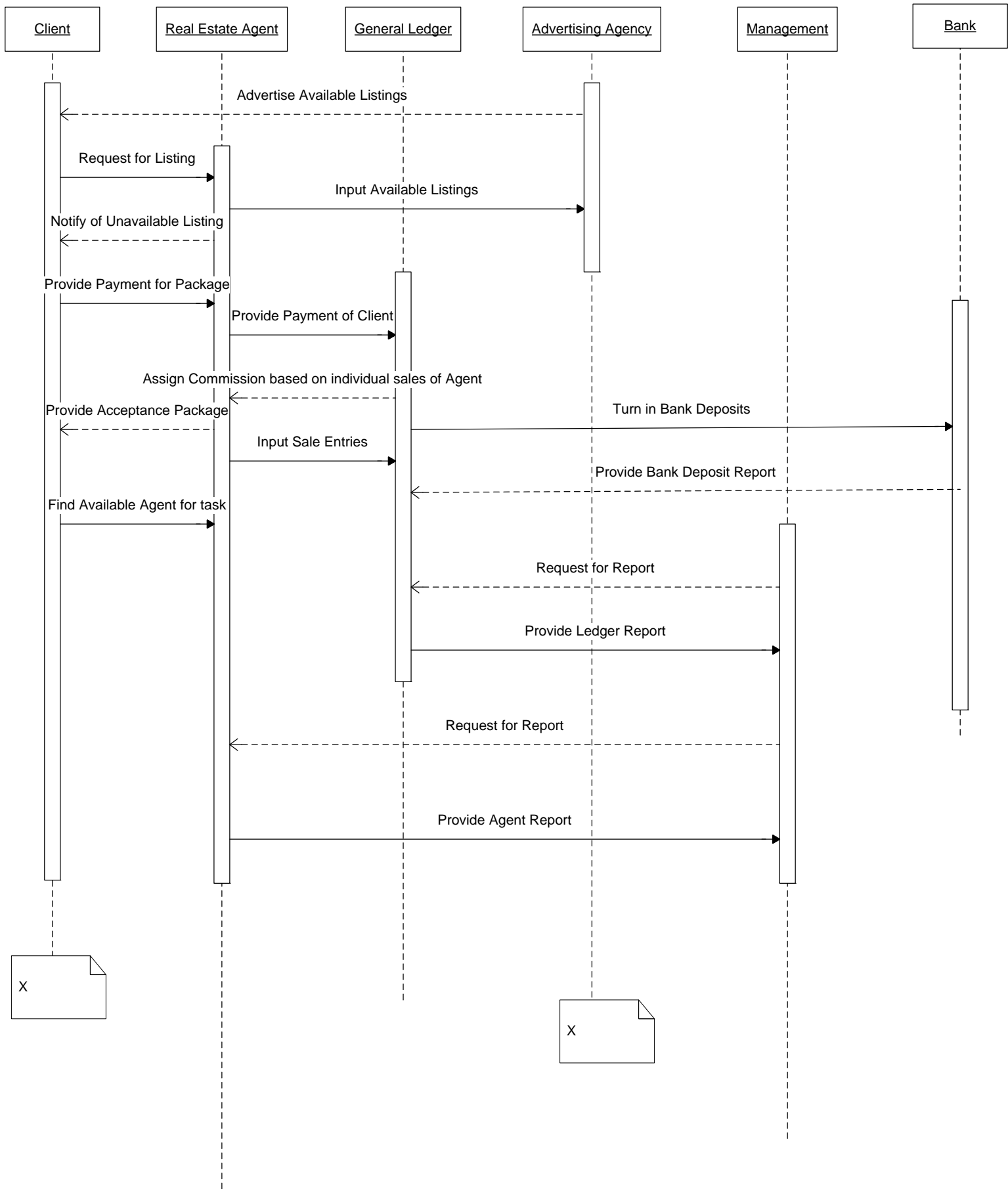
X

X

Activity Diagram for Client



THIS IS THE SEQUENCE DIAGRAM DISPLAYING THE DFD CONTEXT DIAGRAM.
It's based on the entities.



X

X

Part VII

real.txt

-- Project Title: Real Estate Business Tracking System Prototype

-- Name: Chung, Barry and LaCome, Myron

-- Course: CIS 405B

-- Assignment: Project

-- Date: August 21, 2008

-- The project was created using MySQL 5.0

-- Drop tables to re-run script

```
DROP TABLE IF EXISTS CLIENT CASCADE;
DROP TABLE IF EXISTS AGENT CASCADE;
```

-- CREATE TABLES

CREATE TABLE CLIENT

```
(
  clientId          INT(4)          NOT NULL          auto_increment,
  fname             VARCHAR(50),
  lname            VARCHAR(50),
  low              VARCHAR(50),
  high             VARCHAR(50),
  neighborhood     VARCHAR(50),
  comments         VARCHAR(50),
  primary key(clientId)
);
```

CREATE TABLE AGENT

```
(
  agentId          INT(4)          NOT NULL          auto_increment,
  fname            VARCHAR(50),
  lname            VARCHAR(50),
  cprice           VARCHAR(50),
  primary key(agentId)
);
```

-- INSERT TEST DATA INTO CLIENT TABLE

INSERT INTO CLIENT VALUES

```
(
1000, 'Jim', 'Jones', 50000, 100000, 'Highland Park',
'All is well'
);
```

-- INSERT TEST DATA INTO AGENT TABLE

INSERT INTO AGENT VALUES

```
(
2000, 'Mary', 'Smith', 250000
);
```

real.txt

-- Create log

tee c:\Real_File_Test_Data.txt

-- TEST CLIENT

SELECT *
FROM CLIENT;

-- TEST AGENT

SELECT *
FROM AGENT;

-- End log

notee

----- END - OF - PROJECT-----

Real_File_Test_Data.txt

clientId	fname	lname	low	high	neighborhood	comments
1000	Jim	Jones	50000	100000	Highland Park	All is well

1 row in set (0.00 sec)

agentId	fname	lname	price
2000	Mary	Smith	250000

1 row in set (0.00 sec)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.text.NumberFormat;

public class Frame
{
    private JFrame frame, clientFrame, agentFrame, managementFrame;
    //    private JFrame clientFrame;
    //    private JFrame agentFrame;
    //    private JFrame managementFrame;

    private Container buttonPane, textFieldPane;
    //    private Container textFieldPane;

    private JButton buttonClient, buttonAgent, buttonManagement,
        clientButtonOK, agentButtonOK, managementButtonOK, buttonCLOSE,
        buttonClientReport, buttonAgentReport, buttonFullReport;

    private JTextField textClientId, textClientFirst, textClientLast,
        textClientHigh, textClientLow, textClientNeighborhood,
        textClientComments, textAgentId, textAgentFirst, textAgentLast,
        textAgentCprice, textMselect, textMfrom, textMwhere;

    private String field1, field2, field3, field4, field5, field6,
        field7, field8, field9, field10, field11, field12;

    public Frame(){
        makeFrame();
        frame.setLocationRelativeTo(null);
    }

    private void makeFrame(){
        buttonListener bl = new buttonListener();

        frame = new JFrame("Real Estate Business Tracking");

        buttonPane = frame.getContentPane();

        buttonClient = new JButton("Client");
        buttonAgent = new JButton("Agent");
        buttonManagement = new JButton("Management");
        buttonCLOSE = new JButton("Close");
    }
}
```

```
        buttonPane.add(buttonClient);
        buttonPane.add(buttonAgent);
        buttonPane.add(buttonManagement);
        buttonPane.add(buttonCLOSE);

        buttonClient.addActionListener(bl);
        buttonAgent.addActionListener(bl);
        buttonManagement.addActionListener(bl);
        buttonCLOSE.addActionListener(bl);

        buttonPane.setLayout(new FlowLayout(FlowLayout.CENTER, 0,500));
        frame.setSize(750,600);
        frame.setVisible(true);

    }

    private void callClientTextField(){

        clientFrame = new JFrame();
        clientFrame.setLocationRelativeTo(null);

        clientFrame.setTitle("Client Information");
        //        clientFrame.setDefaultCloseOperation(
        //            JFrame.EXIT_ON_CLOSE);

        JPanel panell = new JPanel();
        panell.setLayout(new GridBagLayout());

        addItem(panell, new JLabel("Client ID:"),
            0, 0, 1, 1, GridBagConstraints.EAST);
        addItem(panell, new JLabel("First Name:"),
            0, 1, 1, 1, GridBagConstraints.EAST);
        addItem(panell, new JLabel("Last Name:"),
            0, 2, 1, 1, GridBagConstraints.EAST);
        addItem(panell, new JLabel("Min Property Value:"),
            0, 3, 1, 1, GridBagConstraints.EAST);
        addItem(panell, new JLabel("Max Property Value:"),
            0, 4, 1, 1, GridBagConstraints.EAST);
        addItem(panell, new JLabel("Neighborhood:"),
            0, 5, 1, 1, GridBagConstraints.EAST);
        addItem(panell, new JLabel("Comments:"),
            0, 6, 1, 1, GridBagConstraints.EAST);

        textClientId = new JTextField(5);
        textClientFirst = new JTextField(12);
        textClientLast = new JTextField(12);
        textClientLow = new JTextField(8);
        textClientHigh = new JTextField(8);
```

```
textClientNeighborhood = new JTextField(30);
textClientComments = new JTextField(30);

addItem(panell, textClientId, 1, 0, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textClientFirst, 1, 1, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textClientLast, 1, 2, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textClientLow, 1, 3, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textClientHigh, 1, 4, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textClientNeighborhood, 1, 5, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textClientComments, 1, 6, 2, 1,
        GridBagConstraints.WEST);

clientButtonOK = new JButton("OK");

addItem(panell, clientButtonOK, 0, 10, 0, 0,
        GridBagConstraints.SOUTH);

clientFrame.add(panell);
clientFrame.pack();
clientFrame.setVisible(true);

cfieldListener fl = new cfieldListener();

clientButtonOK.addActionListener(fl);
}

private void callAgentTextField(){

agentFrame = new JFrame();
agentFrame.setLocationRelativeTo(null);

agentFrame.setTitle("Agent Information");
// agentFrame.setDefaultCloseOperation(
// JFrame.EXIT_ON_CLOSE);

JPanel panell = new JPanel();
panell.setLayout(new GridBagLayout());

addItem(panell, new JLabel("Agent ID:"),
        0, 0, 1, 1, GridBagConstraints.EAST);
addItem(panell, new JLabel("First Name:"),
        0, 1, 1, 1, GridBagConstraints.EAST);
```

```
addItem(panell, new JLabel("Last Name:"),
        0, 2, 1, 1, GridBagConstraints.EAST);
addItem(panell, new JLabel("Closing Property Value:"),
        0, 3, 1, 1, GridBagConstraints.EAST);

textAgentId = new JTextField(5);
textAgentFirst = new JTextField(12);
textAgentLast = new JTextField(12);
textAgentCprice = new JTextField(8);

addItem(panell, textAgentId, 1, 0, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textAgentFirst, 1, 1, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textAgentLast, 1, 2, 2, 1,
        GridBagConstraints.WEST);
addItem(panell, textAgentCprice, 1, 3, 2, 1,
        GridBagConstraints.WEST);

agentButtonOK = new JButton("OK");

addItem(panell, agentButtonOK, 0, 10, 0, 0,
        GridBagConstraints.SOUTH);

agentFrame.add(panell);
agentFrame.pack();
agentFrame.setVisible(true);

afieldListener fl = new afieldListener();

agentButtonOK.addActionListener(fl);
}

private void callManagementTextField(){

    mgButtonListener bl = new mgButtonListener();

    managementFrame = new JFrame();
    managementFrame.setLocationRelativeTo(null);

    managementFrame.setTitle("Management Report Funtions");

//    buttonListener bl = new buttonListener();
//
//    frame = new JFrame("Real Estate Business Tracking");
//
    buttonPane = managementFrame.getContentPane();
```

```
        buttonClientReport = new JButton("Client Report");
        buttonAgentReport = new JButton("Agent Report");
        buttonFullReport = new JButton("Full Report");

        buttonPane.add(buttonClientReport);
        buttonPane.add(buttonAgentReport);
        buttonPane.add(buttonFullReport);

        buttonClientReport.addActionListener(bl);
        buttonAgentReport.addActionListener(bl);
        buttonFullReport.addActionListener(bl);

        buttonPane.setLayout(new FlowLayout(FlowLayout.CENTER, 0,100));
        managementFrame.setSize(500,250);
        managementFrame.setVisible(true);

//
//      JPanel panell = new JPanel();
//      panell.setLayout(new GridBagLayout());
//
//      addItem(panell, new JLabel("Select Fields:"),
//              0, 0, 1, 1, GridBagConstraints.EAST);
//      addItem(panell, new JLabel("From Tables:"),
//              0, 1, 1, 1, GridBagConstraints.EAST);
//      addItem(panell, new JLabel("Where Condition:"),
//              0, 2, 1, 1, GridBagConstraints.EAST);
//
//      textMselect = new JTextField(30);
//      textMfrom = new JTextField(30);
//      textMwhere = new JTextField(30);
//
//      addItem(panell, textMselect, 1, 0, 2, 1,
//              GridBagConstraints.WEST);
//      addItem(panell, textMfrom, 1, 1, 2, 1,
//              GridBagConstraints.WEST);
//      addItem(panell, textMwhere, 1, 2, 2, 1,
//              GridBagConstraints.WEST);
//
//      managementButtonOK = new JButton("OK");
//
//      addItem(panell, managementButtonOK, 0, 10, 0, 0,
//              GridBagConstraints.SOUTH);
//
//      managementFrame.add(panell);
//      managementFrame.pack();
//      managementFrame.setVisible(true);
//
//      mgFieldListener fl = new mgFieldListener();
//
```

```
//      managementButtonOK.addActionListener(fl);

}

private void addItem(JPanel p, JComponent c,
    int x, int y, int width, int height,
    int align){

    GridBagConstraints gc =
        new GridBagConstraints();
    gc.gridx = x;
    gc.gridy = y;
    gc.gridwidth = width;
    gc.gridheight = height;
    gc.weightx = 100.0;
    gc.weighty = 100.0;
    gc.insets = new Insets(5, 5, 5, 5);
    gc.anchor = align;
    gc.fill = GridBagConstraints.NONE;
    p.add(c, gc);

}

private class buttonListener implements ActionListener{
    public void actionPerformed(ActionEvent e){

        if (e.getSource() == buttonClient){

            callClientTextField();

        }

        if (e.getSource() == buttonAgent){

            callAgentTextField();

        }

        if (e.getSource() == buttonManagement){

            callManagementTextField();

        }

        if (e.getSource() == buttonCLOSE){

            System.exit(0);

        }

    }

}
```

```
    }
}

private class cfieldListener implements ActionListener{
    public void actionPerformed(ActionEvent e){

        if (e.getSource() == clientButtonOK){

            field1 = textClientFirst.getText();
            field2 = textClientLast.getText();
            field3 = textClientLow.getText();
            field4 = textClientHigh.getText();
            field5 = textClientNeighborhood.getText();
            field6 = textClientComments.getText();

            Connection con = getConnection();
            try{
                Statement s = con.createStatement();
                String insert = "insert into client (fname, lname, low,
high, neighborhood, comments) value ('" + field1 + "', '" + field2 + "', '" +
field3 + "', '" + field4 + "', '" + field5 + "', '" + field6 + "')";
                s.executeUpdate(insert);
            }
            catch (SQLException exceptn) {
                System.out.println(exceptn.getMessage());
            }

            clientFrame.dispose();

            JOptionPane.showMessageDialog(null, "Client has been added"
);
//            System.exit();

        }

    }

private class afieldListener implements ActionListener{
    public void actionPerformed(ActionEvent e){

        if (e.getSource() == agentButtonOK){

            field7 = textAgentFirst.getText();
            field8 = textAgentLast.getText();
            field9 = textAgentCprice.getText();
```

```
        Connection con = getConnection();
        try{
            Statement s = con.createStatement();
            String insert = "insert into agent (fname, lname, cprice
) value ('" + field7 + "','', '" + field8 + "','', '" + field9 + "','')";
            s.executeUpdate(insert);
        }
        catch (SQLException exceptn) {
            System.out.println(exceptn.getMessage());
        }

        agentFrame.dispose();

        JOptionPane.showMessageDialog(null, "Agent has been added" )
;

    }

}

private class mgButtonListener implements ActionListener{
    public void actionPerformed(ActionEvent e){

        if (e.getSource() == buttonClientReport){

            Connection con = getConnection();
            try{

                Statement s = con.createStatement();
                System.out.println(s);

            }
            catch (SQLException exceptn) {
                System.out.println(exceptn.getMessage());
            }

            managementFrame.dispose();

            JOptionPane.showMessageDialog(null, "Report has been created
" );

        }

    }
}
```

```
    }

    private static Connection getConnection(){

        Connection con = null;
        try{

            Class.forName ("com.mysql.jdbc.Driver").newInstance();
            String url = "jdbc:mysql://localhost/realestate";
            String user = "root";
            String pw = "mdl06090";
            con = DriverManager.getConnection(url, user, pw);
        }
        catch (Exception exceptn) {
            System.out.println(exceptn.getMessage() + "Database Connecti
on Error");
            System.exit(0);
        }

        return con;

    }
}
```