SOFTWARE DESIGN (SWD)

Instructor: Dr. Hany H. Ammar

Dept. of Computer Science and Electrical Engineering, WVU

OUTLINE

- Introduction TO Software Design (SWD) and the SW Design Description (SDD) document
- Software Design Criteria
- Software Design Methodologies
- Structured Design for (SD) Software Using ICASE

Software Design Methodologies Structured Design

- Following the structured analysis and object-oriented analysis methodologies used in the requirements phase, Design methodologies consist of

  - Structured Design

    a) Produces a design that can be implemented in structured programming languages such as C

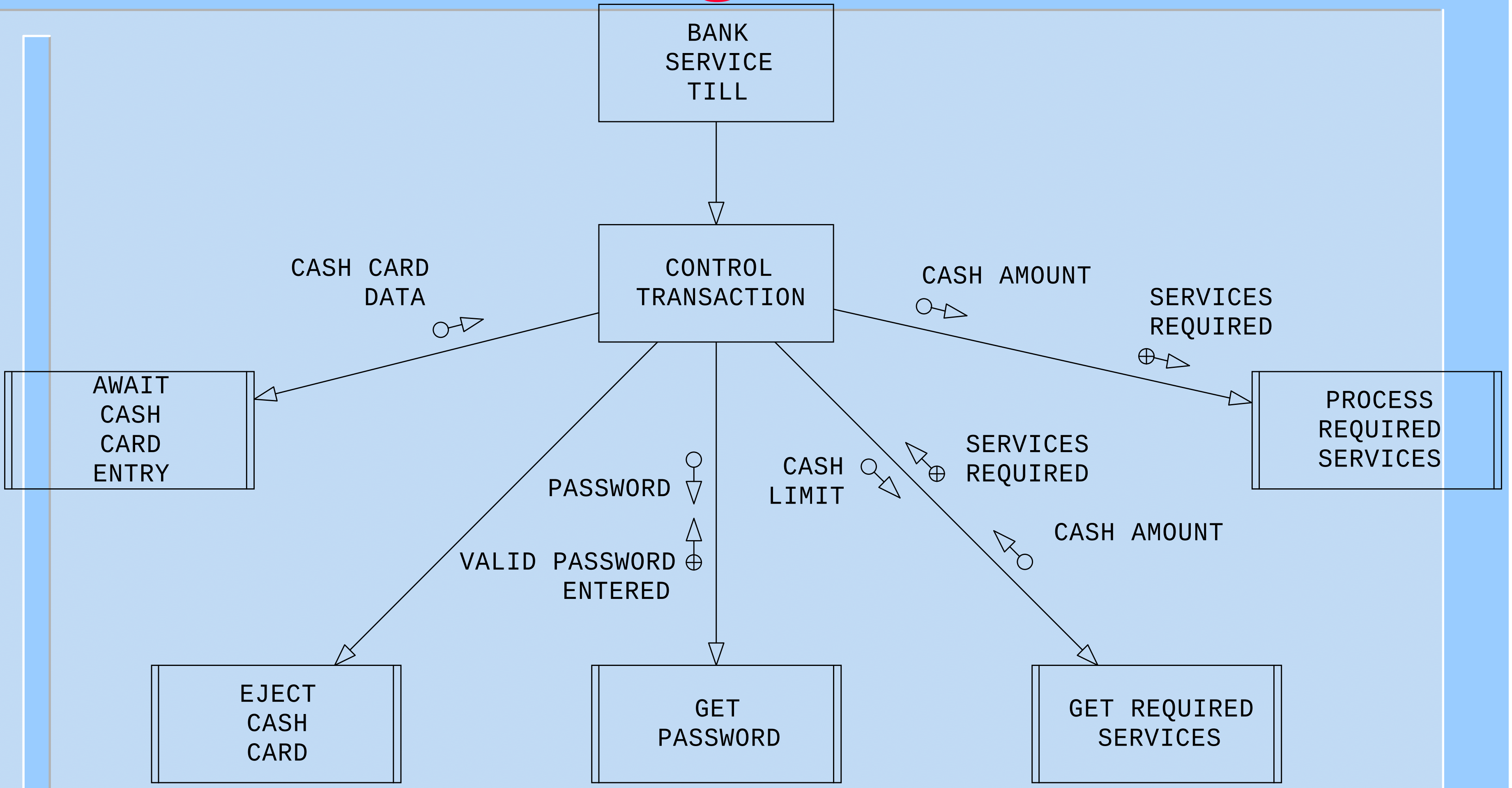
    b) characterized by the development of structured hierarchy of modules

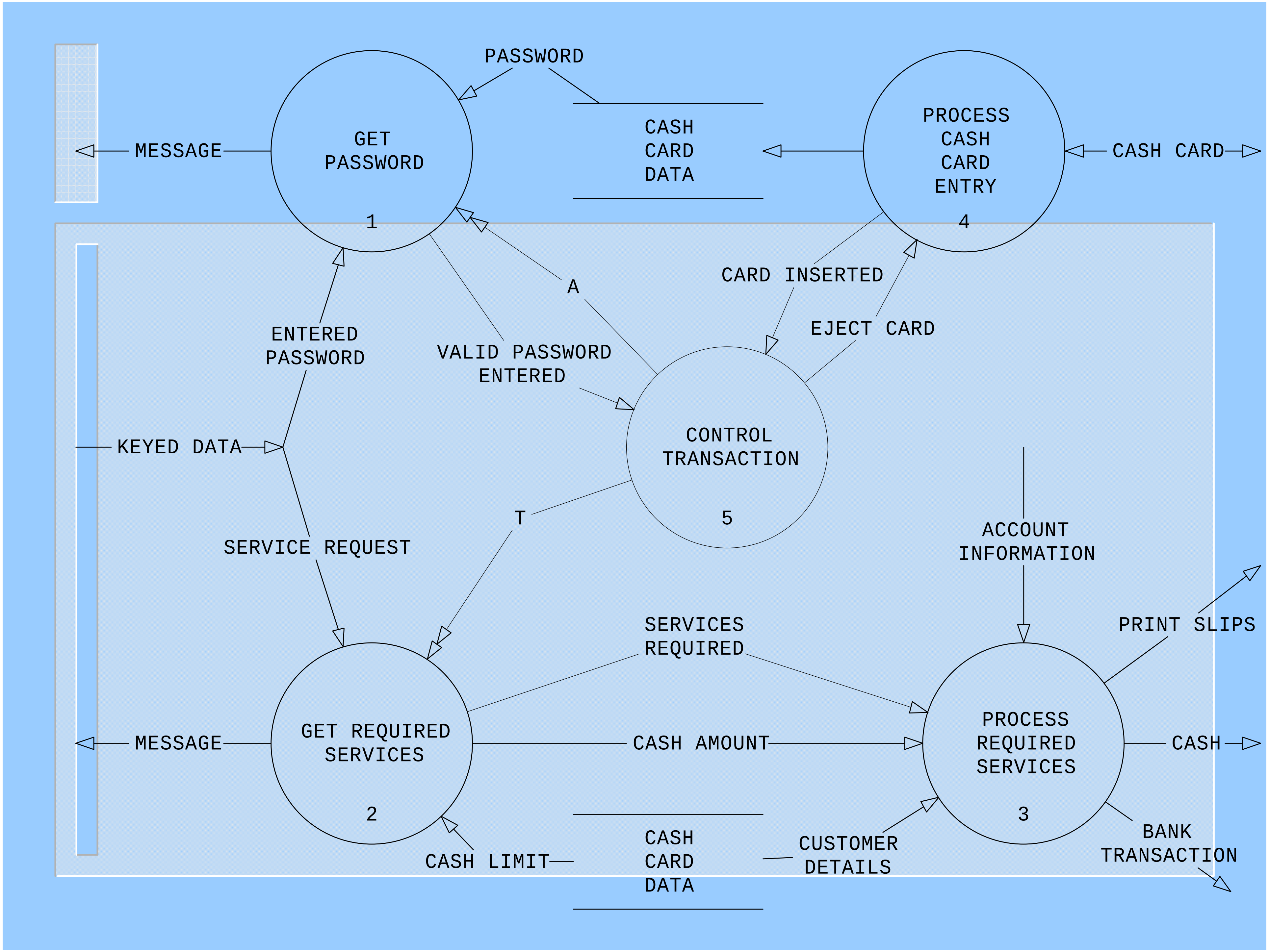
    specified using Structure Charts (SCs)

Software Design Methodologies Structured Design

- SCs are used for modeling the partitioning/grouping of data/control functions defined in the specifications into modules, using the software design criteria
- The hierarchical organization of these modules, and the data/control interfaces between them are defined in the SC
- Each module declared in the SC must be accompanied by a module specification (M-specs). The pre/post conditions, and algorithms specified using a design description language or a flow chart

Software Design Methodologies
Structured Design




PASSWORD
MESSAGE
GET PASSWORD
1
CASH CARD DATA
PROCESS CASH CARD ENTRY
4
CASH CARD
A
CARD INSERTED
ENTERED PASSWORD
VALID PASSWORD ENTERED
EJECT CARD
KEYED DATA
CONTROL TRANSACTION
5
T
SERVICE REQUEST
ACCOUNT INFORMATION
SERVICES REQUIRED
PRINT SLIPS
MESSAGE
GET REQUIRED SERVICES
2
CASH AMOUNT
PROCESS REQUIRED SERVICES
3
CASH
CASH LIMIT
CASH CARD DATA
CUSTOMER DETAILS
BANK TRANSACTION

Software Design Methodologies Structured Design

- SCs are developed from specifications represented by structured analysis graphs (DFDs/CFDs, C-specs, etc.)
- C-Specs is mapped to in the upper-level modules in the SC, since they are responsible for controlling the decisions and the activities in the lower levels modules,
- The controller Control_Transaction in the previous slide is mapped to the upper-level module controlling the execution of the SC shown

Software Design Methodologies Structured Design

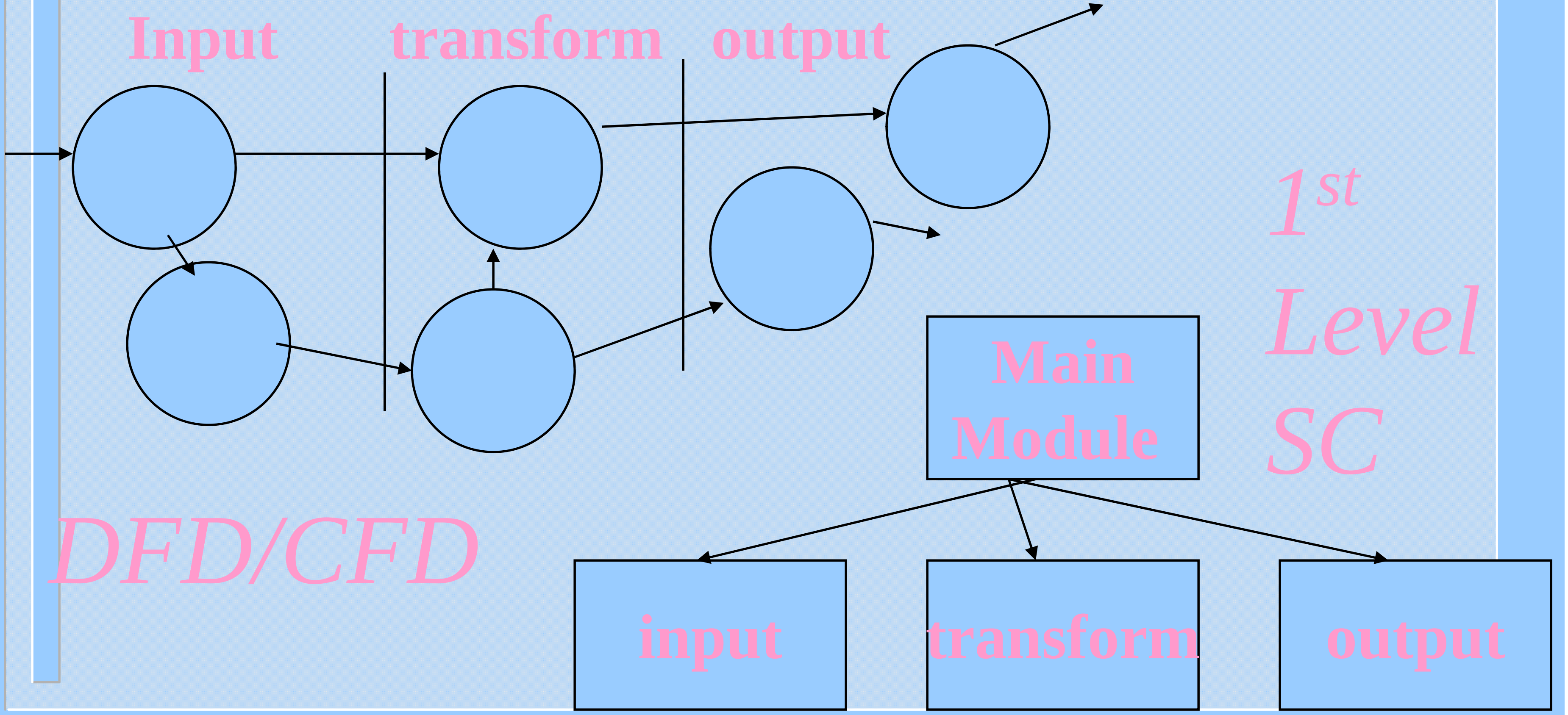
- Data processes specified in Data Flow Diagrams (DFDs) are allocated to modules using two techniques discussed as follows

  - transform-oriented design, processes are divided into *input and data preprocessing* functions, *data processing functions*, and output related functions

  - transaction-oriented design, in this case the design consists of an *input module*, a *dispatcher module*, *transaction processing modules* one module for each type of transaction/command/or request

Software Design Methodologies
Structured Design

Transform-Oriented

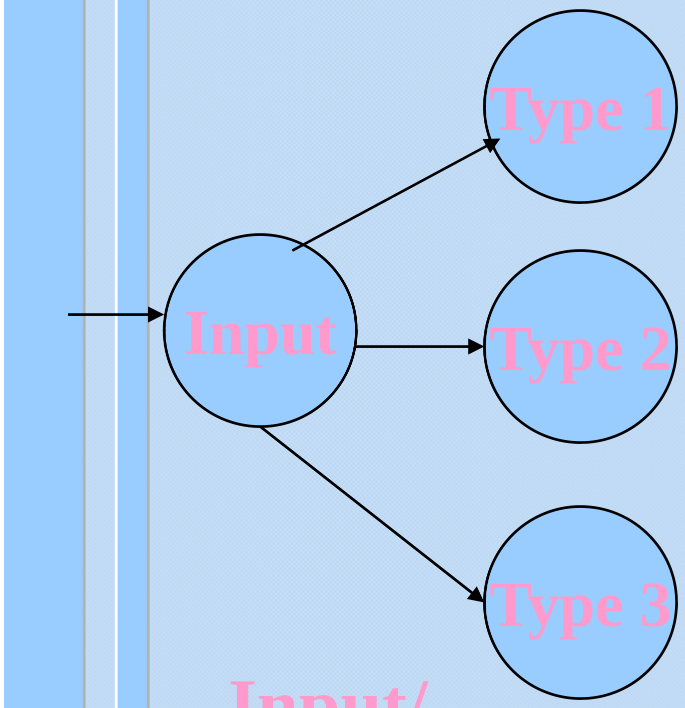


1st Level SC

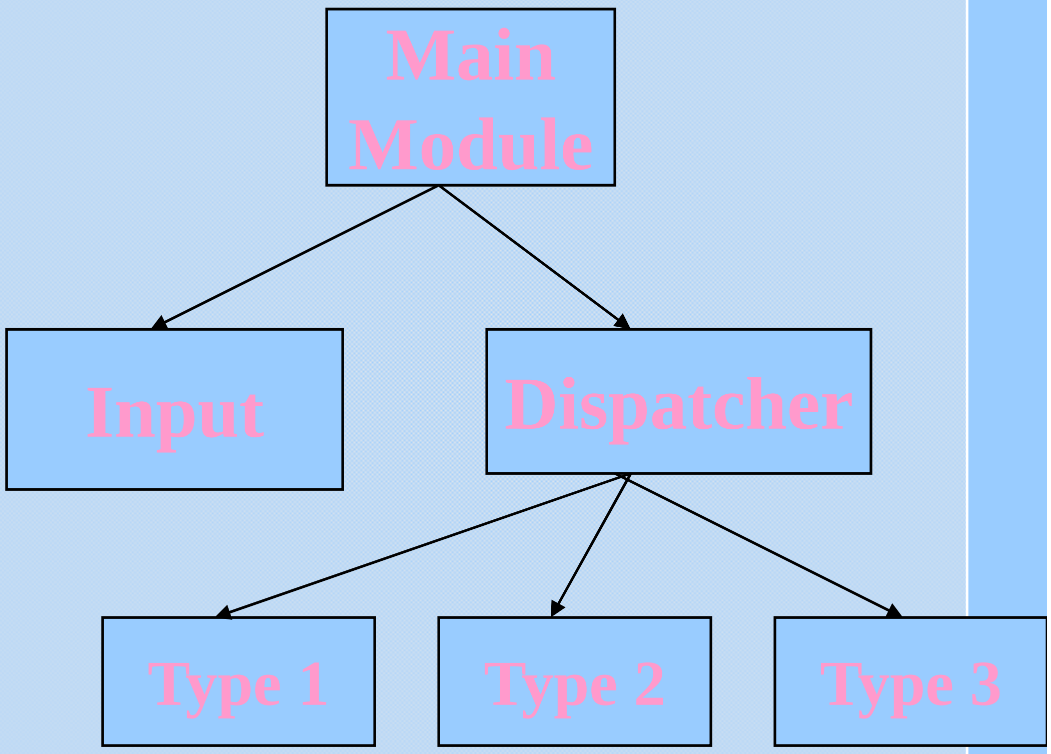
DFD/CFD

Software Design Methodologies
Structured Design

- Transaction driven

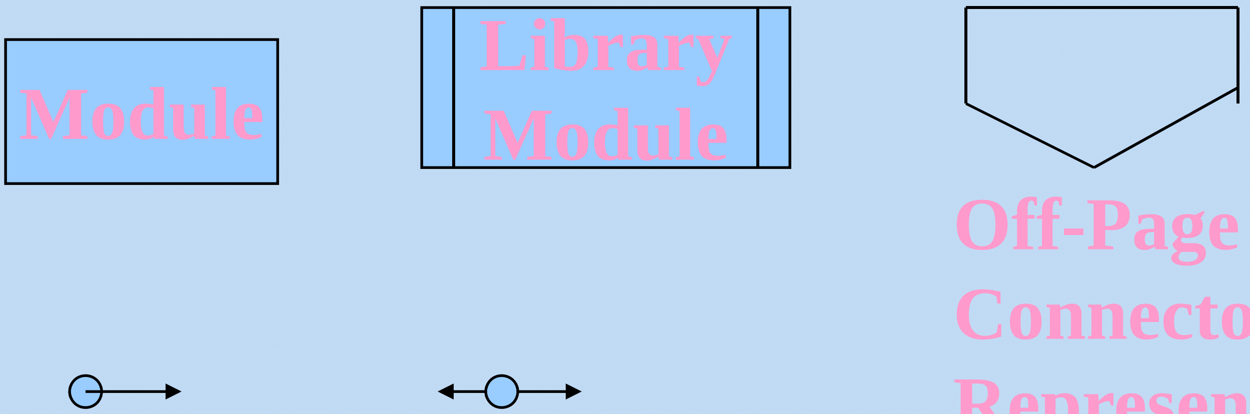


Input/
process transactions



Structured Design (SD) Using ICASE

- StP/SE Structure chart Editor symbols (used in the Notes of Chapter 4)



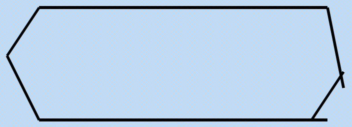
Module

Library Module

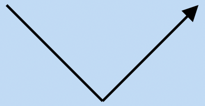
Off-Page Connector Representing A subsystem

Unidirectional/
bi-directional data couple,
A control couple is distinguished
by a black circle

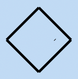
Structured Design (SD) Using ICASE



Global data



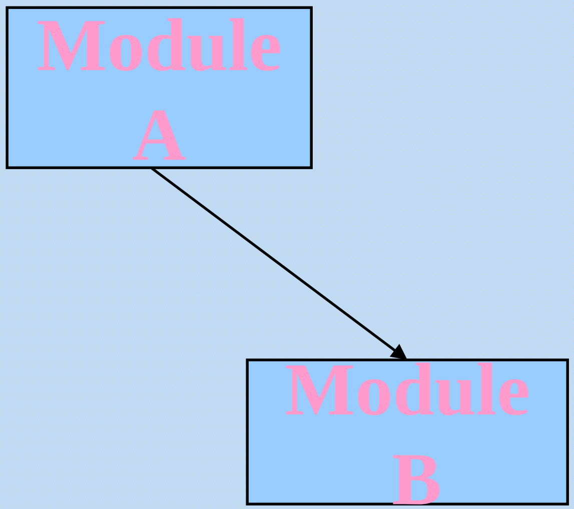
Iteration symbol
Used at the
invocation lines
Out of a module



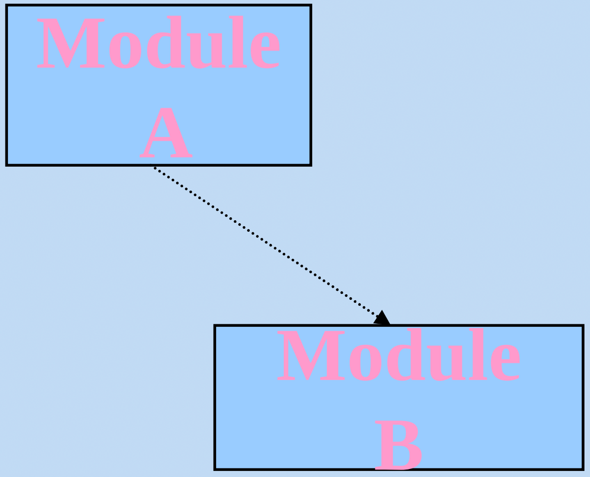
Selection
Between
Invocation
Lines out
Of a module
(Conditional
invocation

Anchors and comments
Can also be used in the
Structure chart

Structured Design (SD) Using ICASE

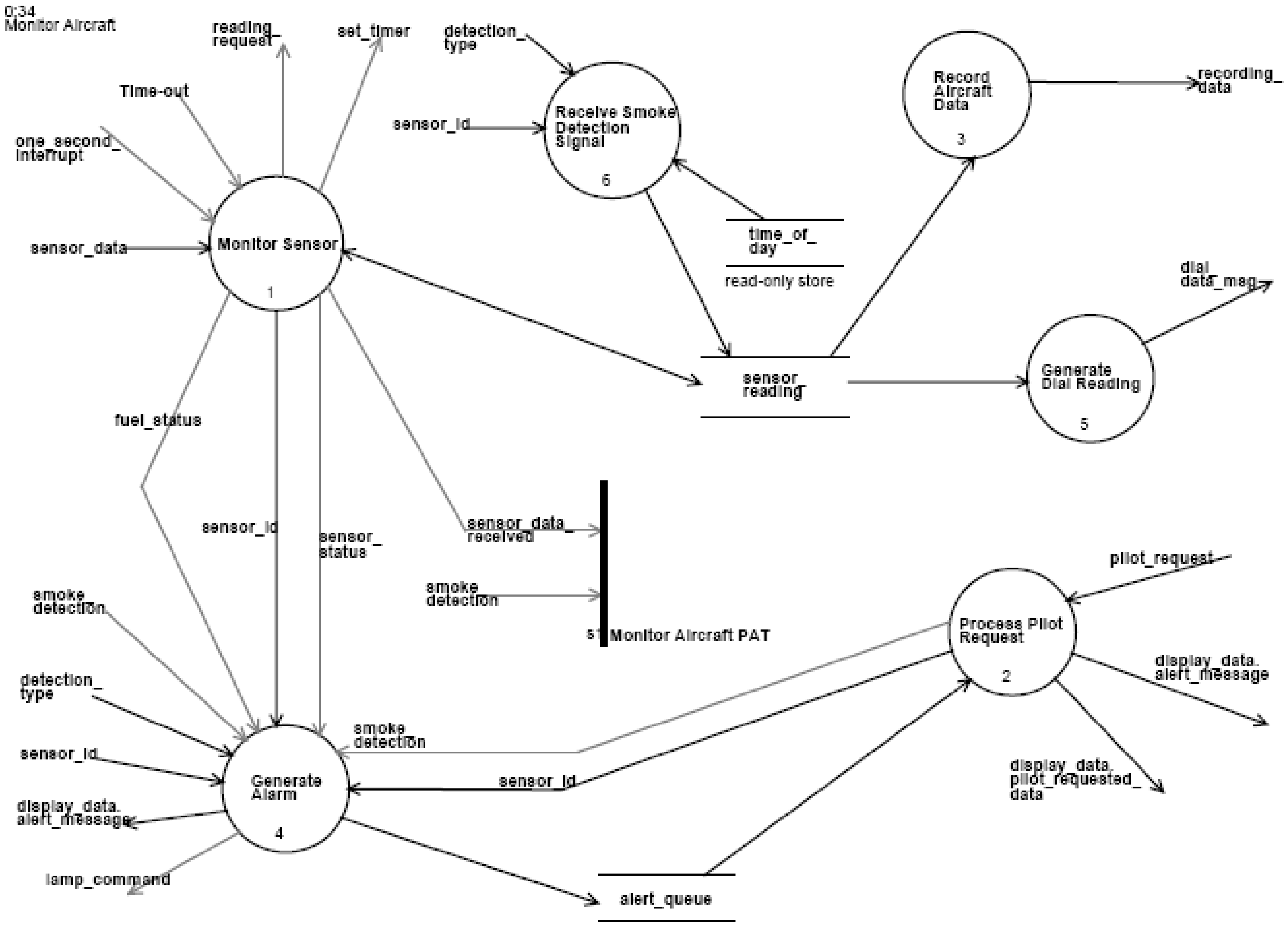


Synchronous
Invocation,
Module A invokes B
And waits until B returns



Asynchronous
Invocation,
Module a invokes B,
Then continues (not
In StP/SE SCE)

Figure 3.23 DFD 0 Monitor Aircraft



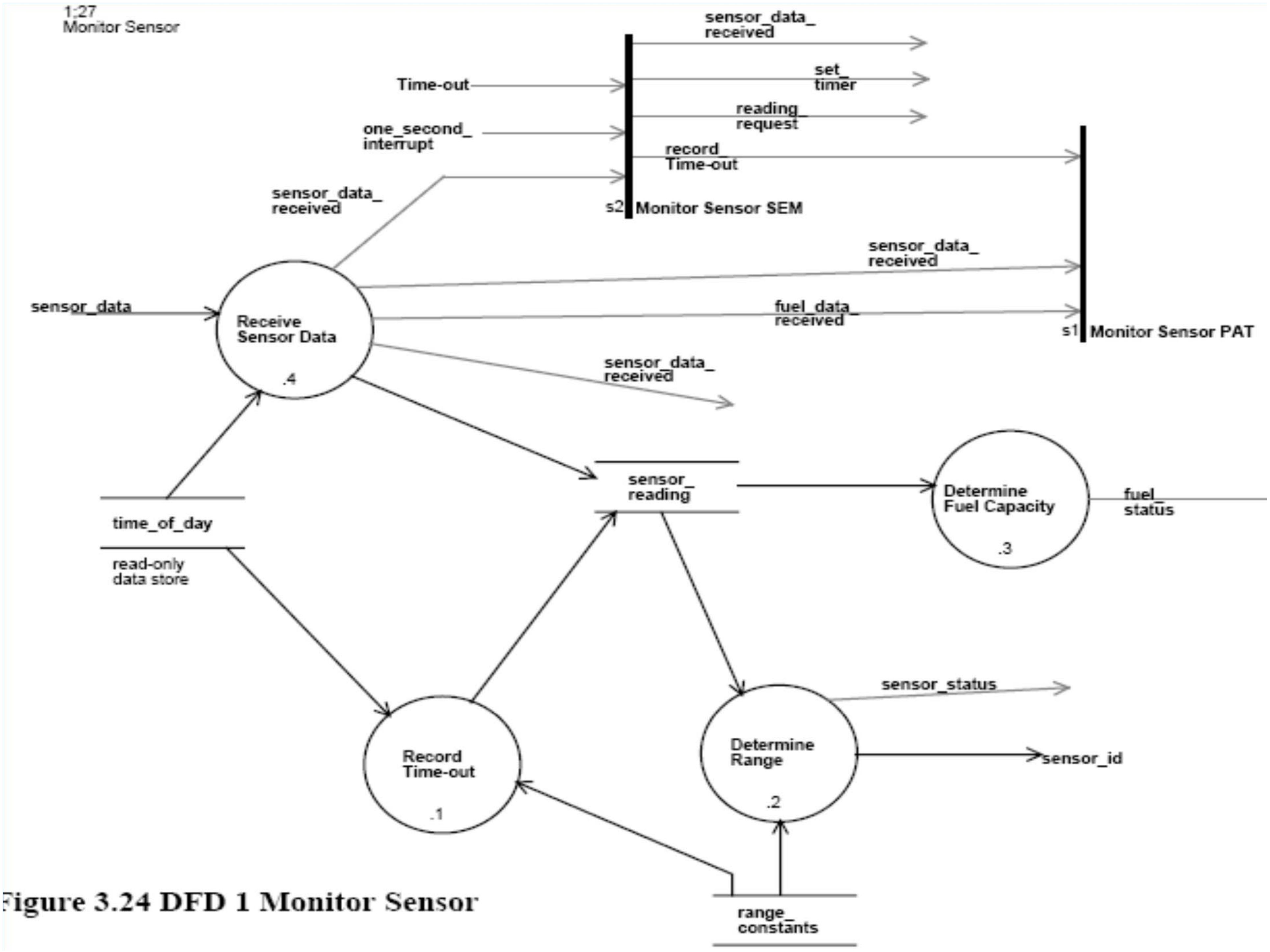


Figure 3.24 DFD 1 Monitor Sensor

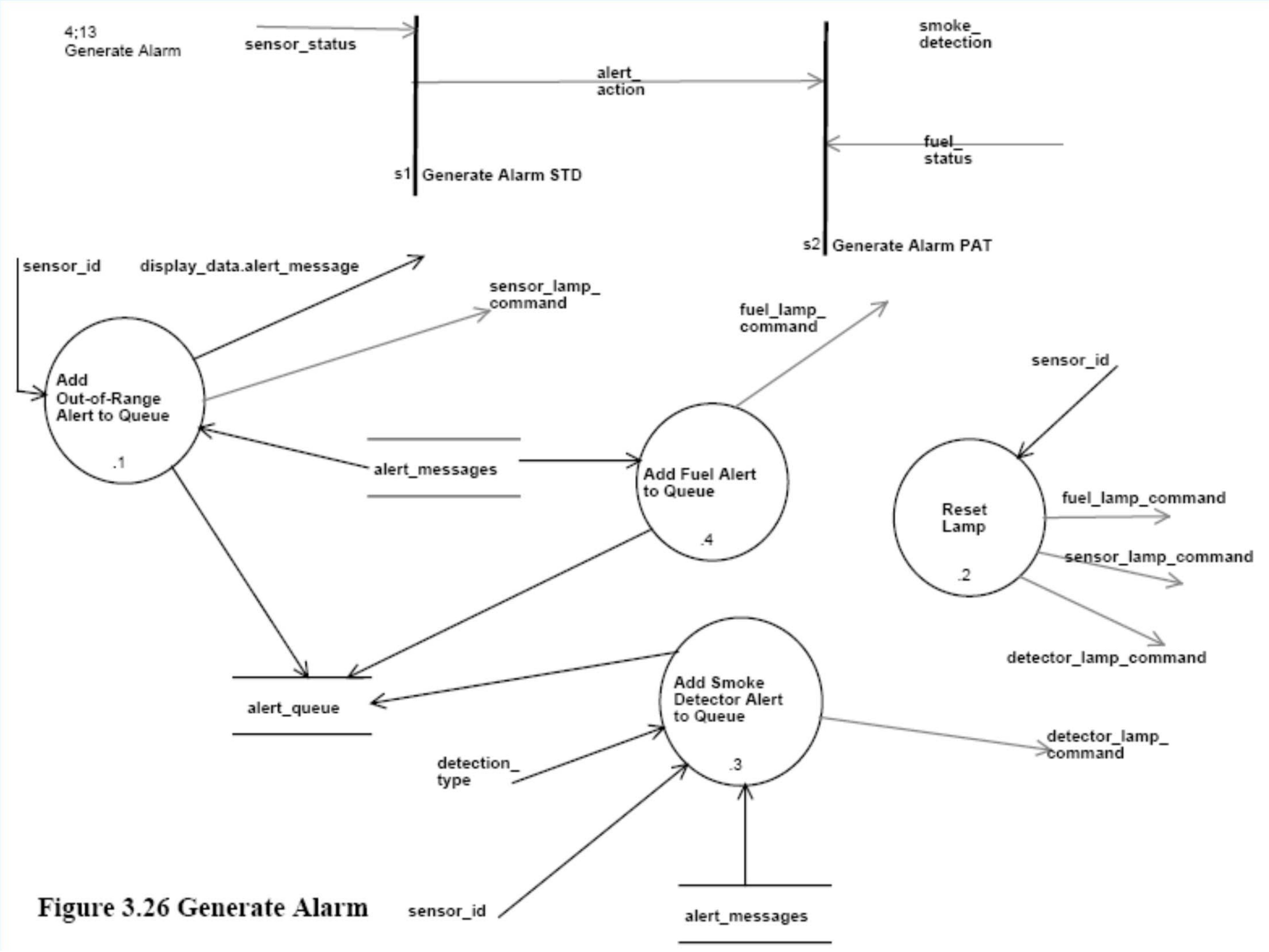
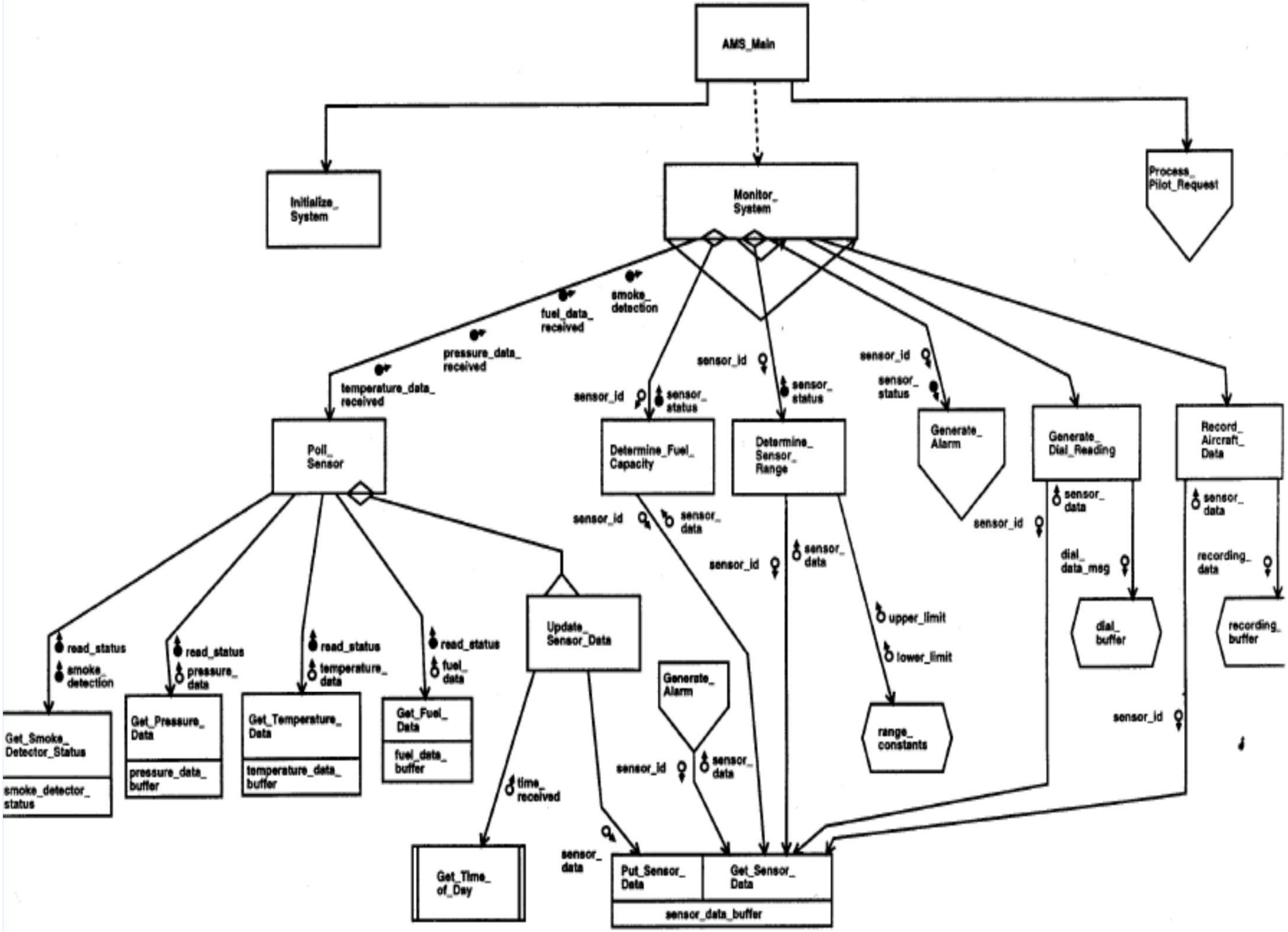
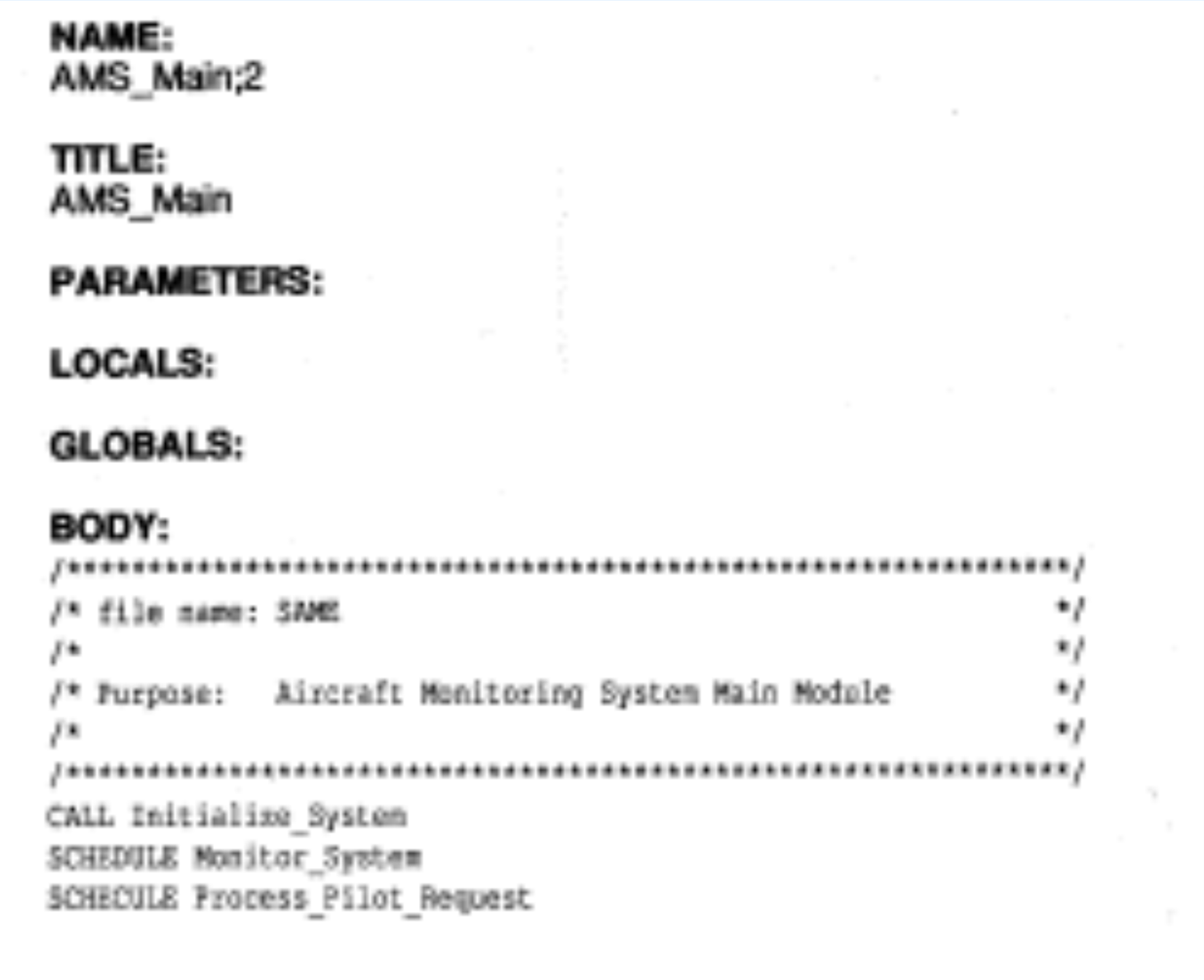


Figure 3.26 Generate Alarm

AMS_Main;9
AMS Main Structure Chart





NAME:
AMS_Main;2

TITLE:
AMS_Main

PARAMETERS:

LOCALS:

GLOBALS:

BODY:

```
/****************************************************************/
/* file name: SAME                                              */
/*                                                              */
/* Purpose:   Aircraft Monitoring System Main Module            */
/*                                                              */
/****************************************************************/
CALL Initialise_System
SCHEDULE Monitor_System
SCHECULE Process_Pilot_Request
```

NAME:
Monitor_System;5

TITLE:
Monitor_System

PARAMETERS:

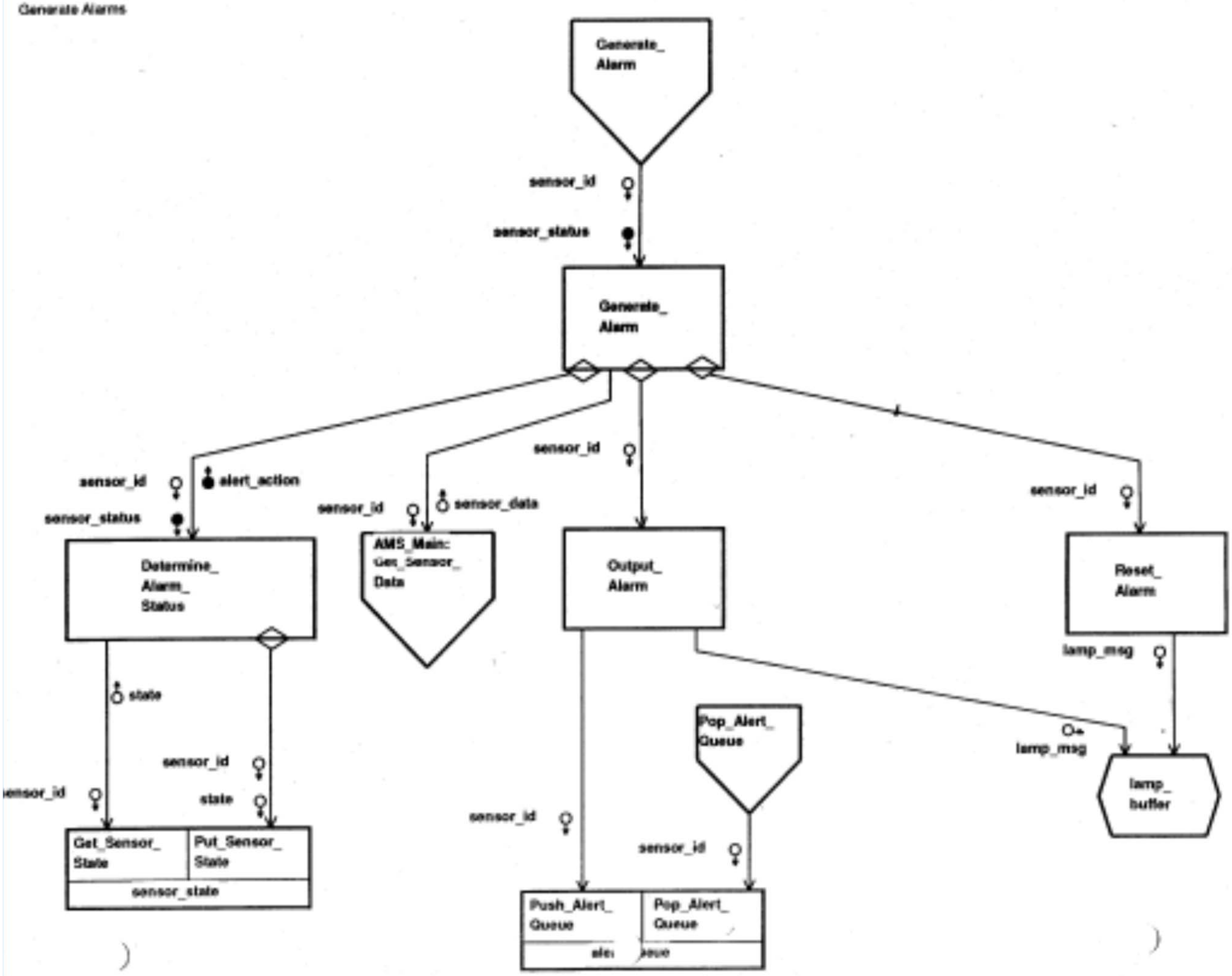
LOCALS:
sensor_id
pressure_data_received
sensor_status
smoke_detection
fuel_data_received
temperature_data_received

GLOBALS:

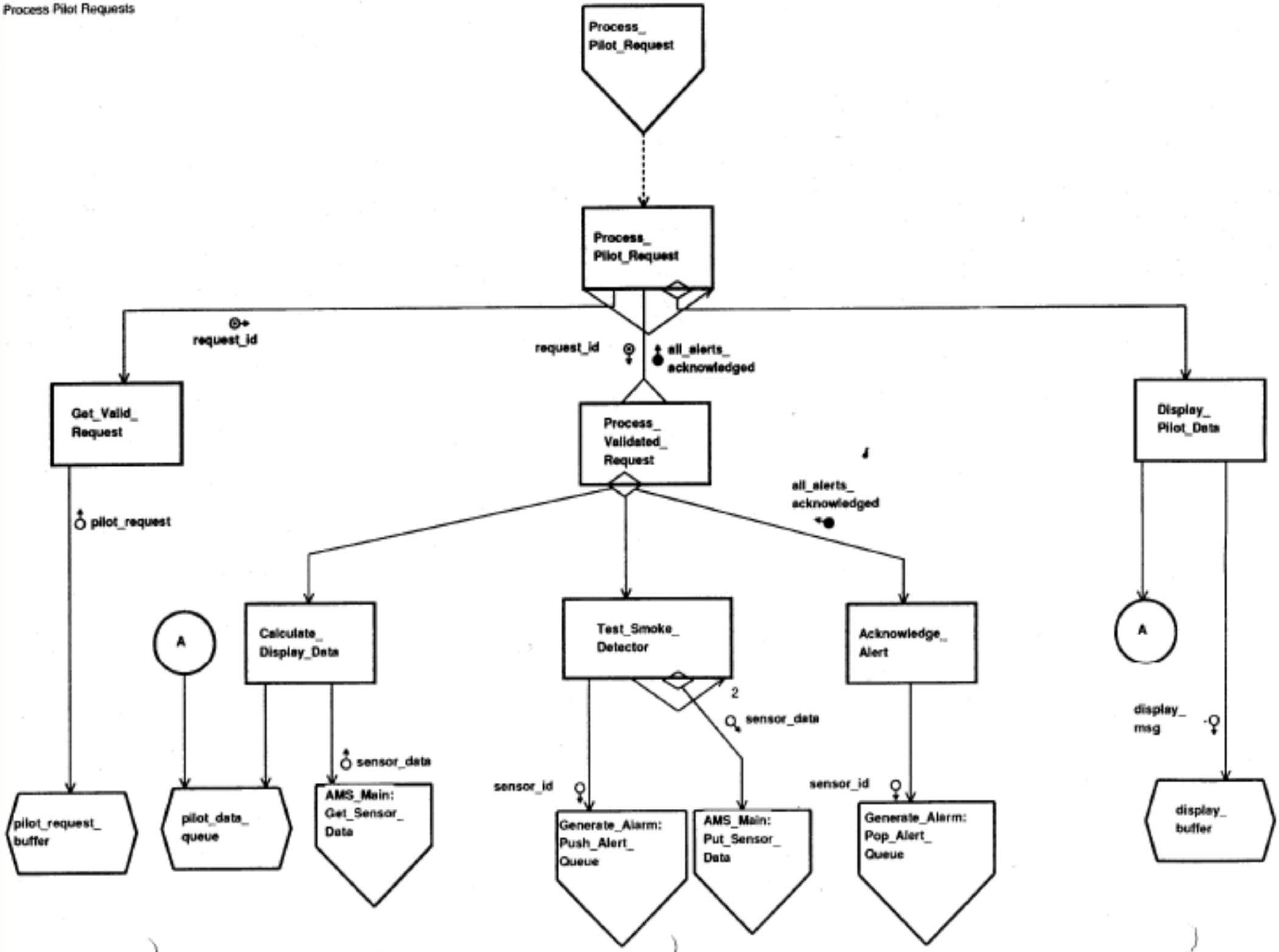
BODY:

```
/*************************************************/
/* file name:   SAME                              */
/*                                                */
/* Purpose:     Monitor system sensors and        */
/*              generate necessary alarms.        */
/*                                                */
/*************************************************/

CYCLE 1-second
 CALL Poll_Sensor(sensor_data_received,fuel_data_received)

 /*  Check fuel  */
 sensor_id = "F01"
 IF fuel_data_received = "TRUE" THEN
  CALL Determine_Sensor_Range(sensor_id,sensor_status)
  IF sensor_status = OK THEN
   CALL Determine_Fuel_Capacity(sensor_id,sensor_status)
  ENDIF
 ELSE
  sensor_status = "TIMED OUT"
 ENDIF
 IF sensor_status <> "OK" THEN
   CALL Generate Alarms
```

Generate_Alarm;8
Generate Alarms



Process_Pilot_Request;3
Process Pilot Requests



Structured Design for (SD) Software Using ICASE

Steps for developing structure charts

The following set of steps are described to guide the designer in developing an architectural design which conforms to the design criteria

- Step 1- Review and refine the diagrams developed in the analysis phase. The analysis diagrams contained in the Software Requirements Specification document are reviewed and refined for the design phase to include greater detail

  - A refined specification contains a more flattened view of the logical model of the system, by bringing lower level functions to upper level DFDs

Structured Design for (SD) Software Using ICASE

- Step 2- Identify and label the necessary concurrent modules from the refined analysis diagrams

  - The phrase necessary concurrent modules here means that these modules have to be running concurrently for the correct real-time operation of this system

  - If the identified functions in the various modules can be invoked sequentially and still satisfy the timing specifications for the output events then there is no need for concurrency.

Structured Design for (SD) Software Using ICASE

- Step 3- Implement, using asynchronous/synchronous invocations or clear comments, in a structure chart the invocation of concurrent and sequential modules from the main or the root module (this root module usually carries the name of the software under development).

Structured Design for (SD) Software Using ICASE

- Step 4- For each concurrent module, Determine whether the refined DFD/CFD diagrams have transform or transaction flow

  - Determine the first level factoring of these modules, and document their specifications using M-specs.

  - Specify the couples using data dictionary entries (or data structure diagrams and comments)

Structured Design for (SD) Software Using ICASE

- Step 5- Refine the first-cut design obtained above to reflect design criteria such as coupling, cohesion, information hiding, and complexity
- Step 6- The complex modules specified in the previous steps should be factored out using steps 1 through 5 above and the process should continue until all lower level modules are simple enough to specify using simple M-specs
- Step 7 Complete the descriptions of all module interfaces and global data structures

Example of Step 4 of Design Procedure

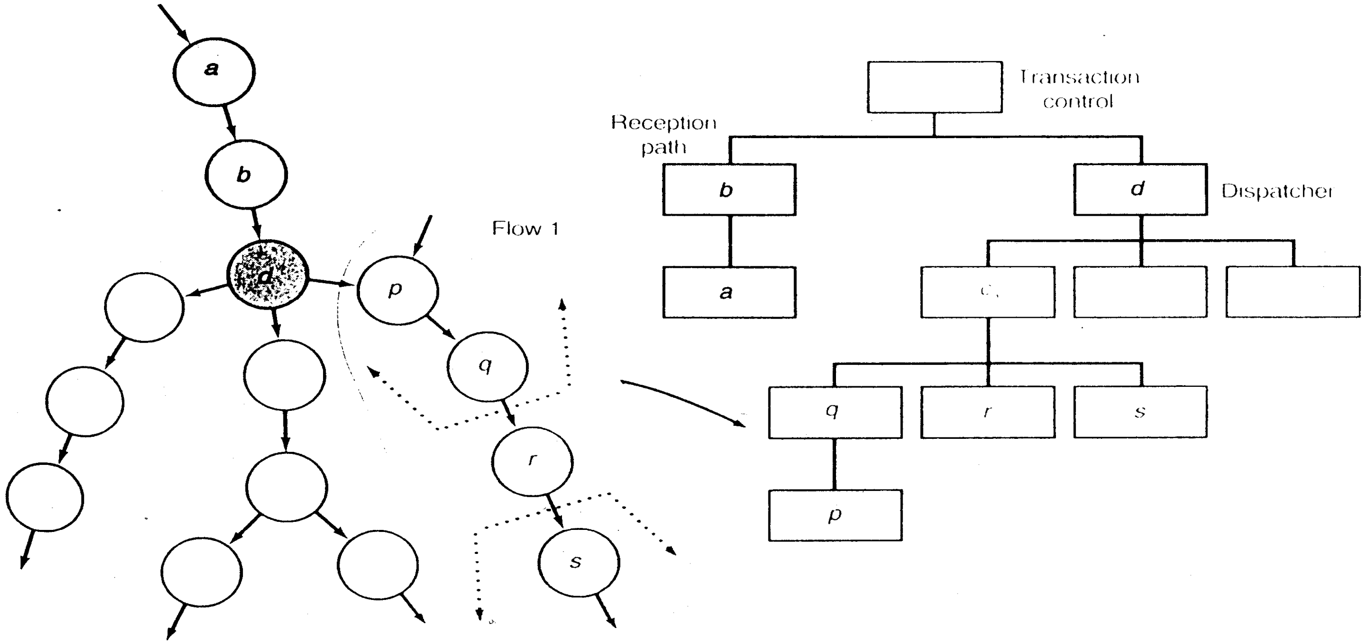


FIGURE 11.17 Transaction mapping

3

Example of Step 5 of Design Procedure

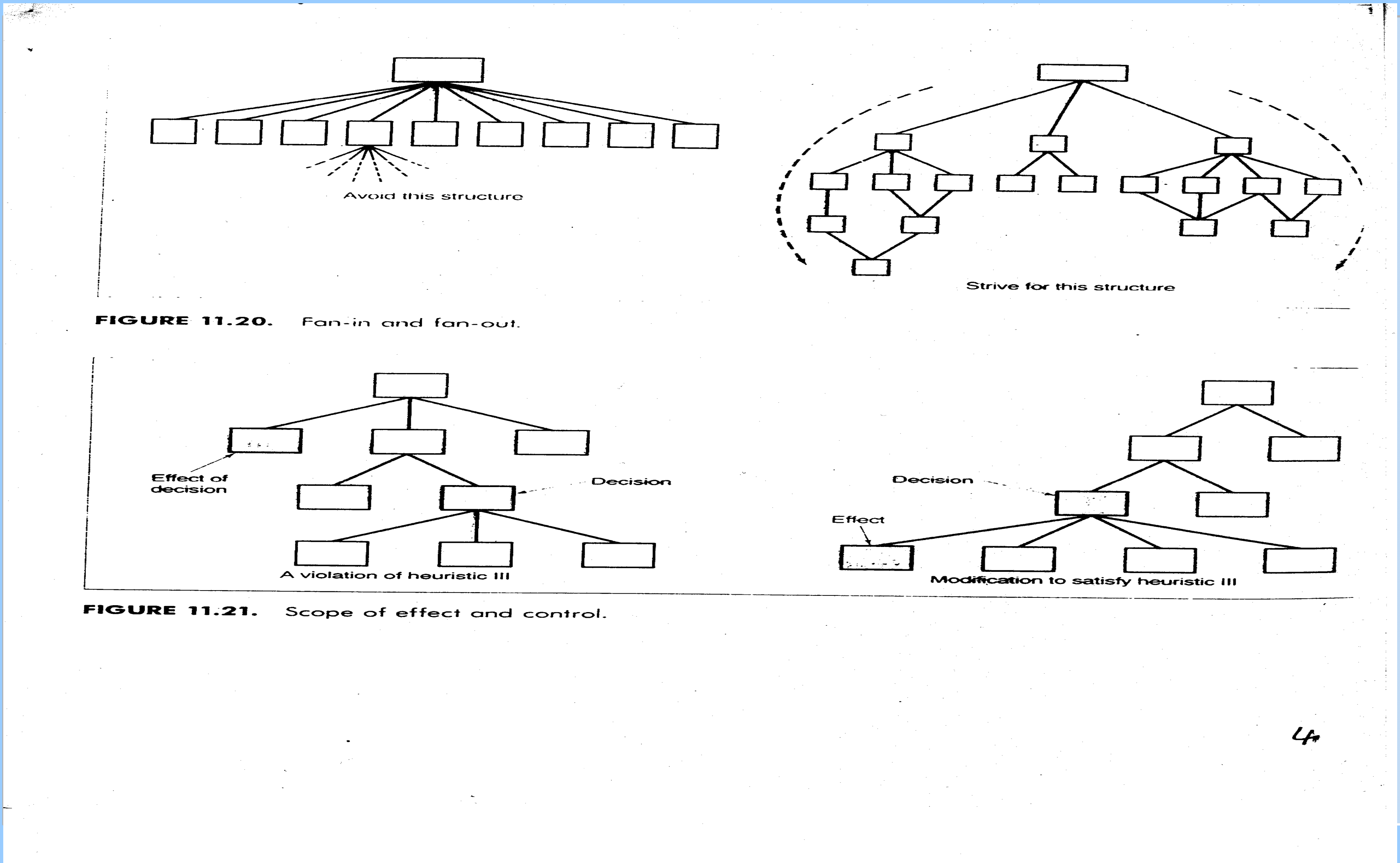

Avoid this structure
Strive for this structure

FIGURE 11.20. Fan-in and fan-out.

Effect of decision
Decision
A violation of heuristic III
Decision
Effect
Modification to satisfy heuristic III

FIGURE 11.21. Scope of effect and control.

4

A Simple Example: A Home Security System

Example:

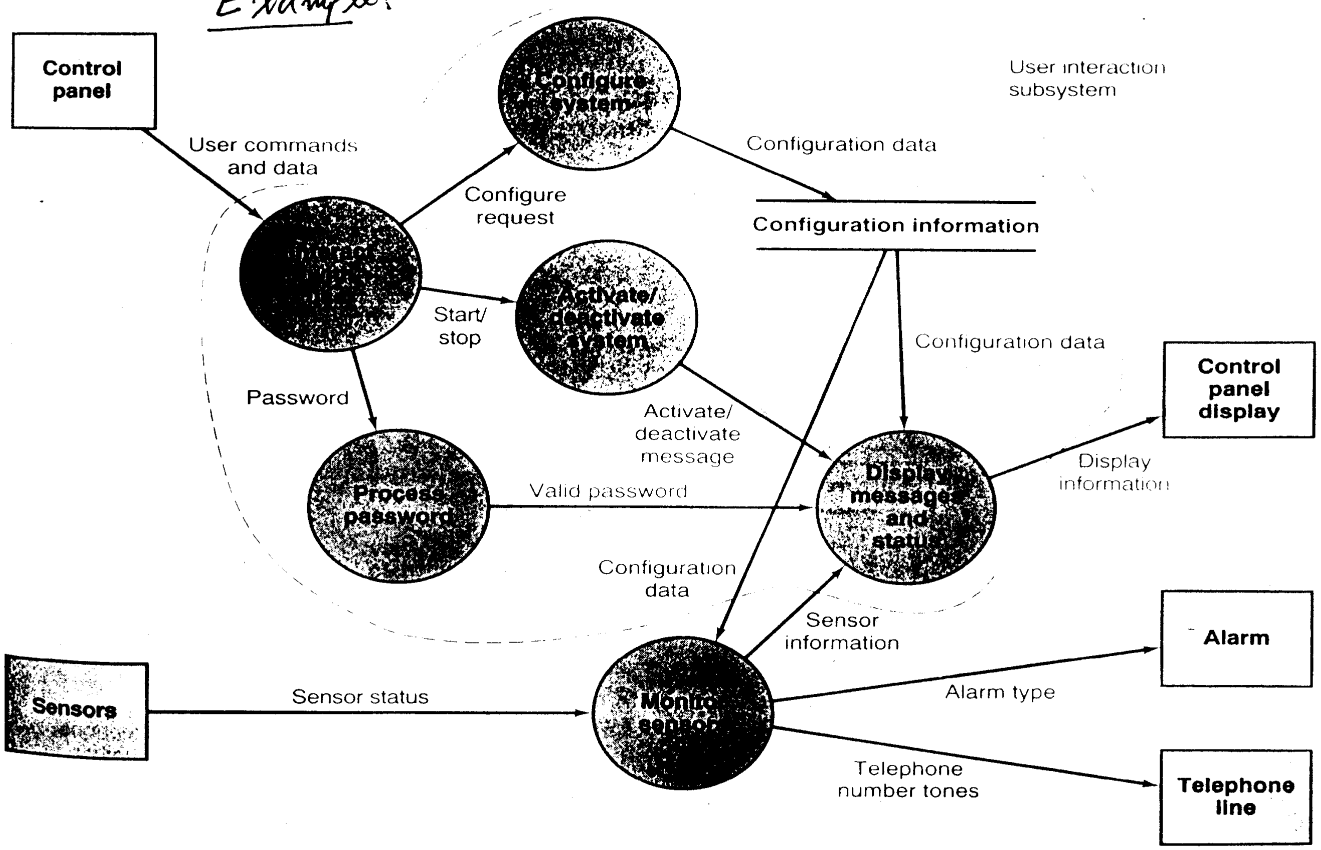
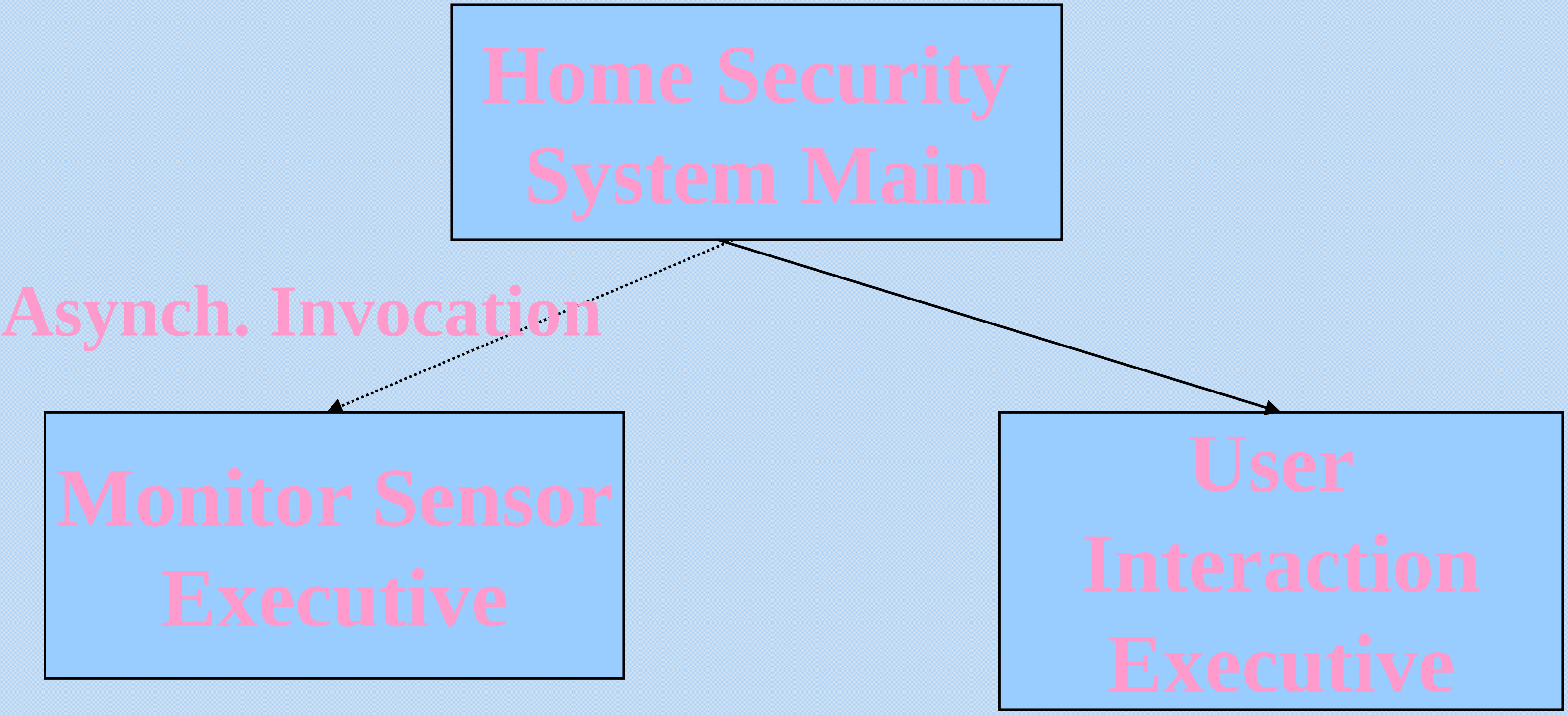


FIGURE 11.5. Level 1 DFD for *SafeHome*.

5

Design Procedure: Step 2


Home Security System Main

Asynch. Invocation

Monitor Sensor Executive

User Interaction Executive

Monitor Sensor DFD from specs

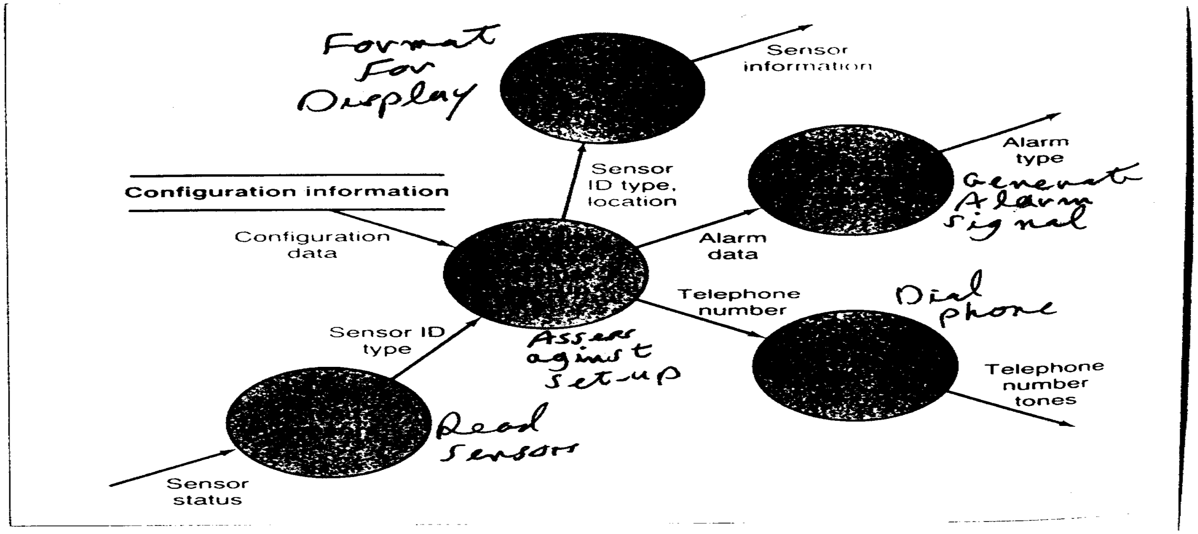
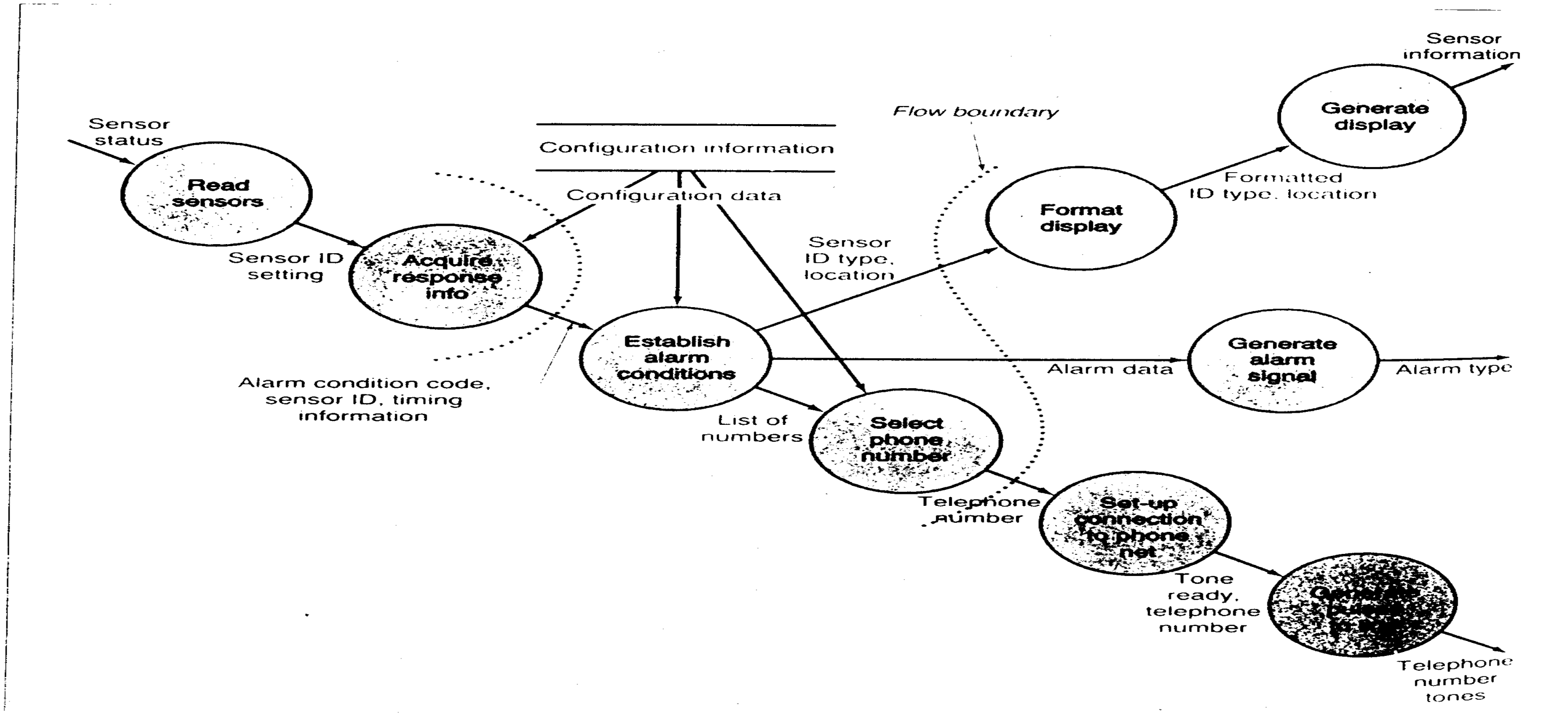
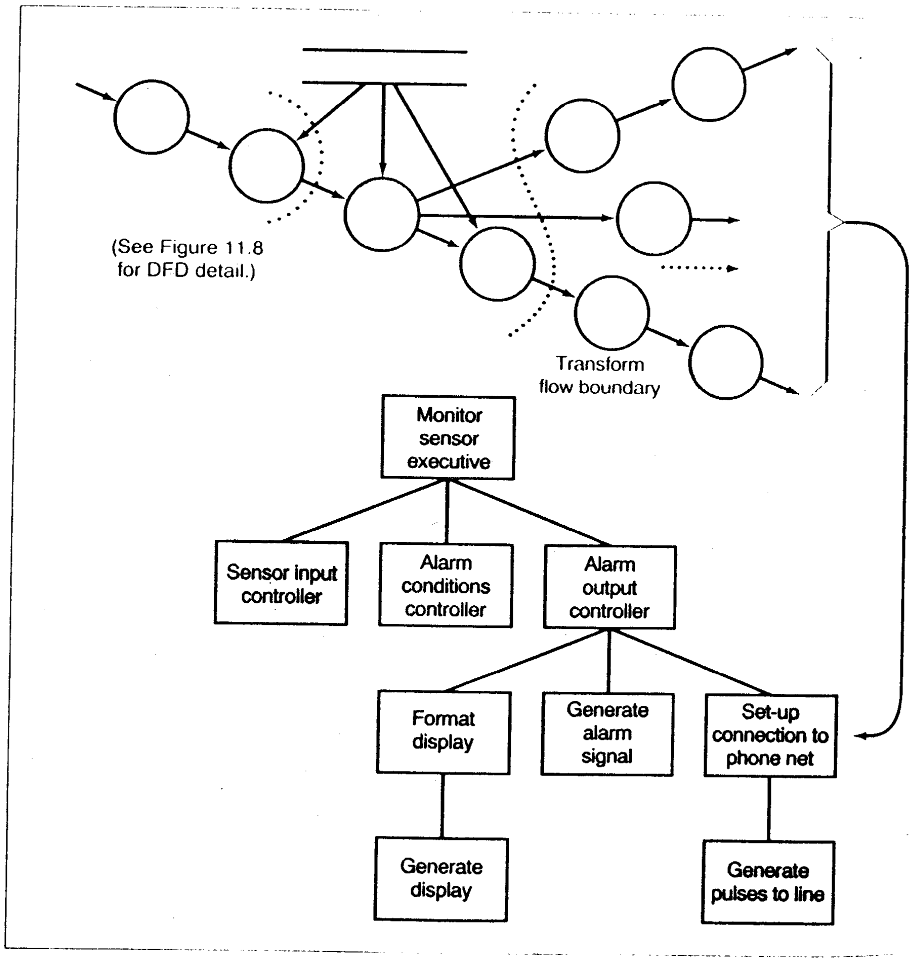


FIGURE 11.6. Level 2 DFD that refines the **Monitor Sensors** process.




(See Figure 11.8 for DFD detail.)

Transform flow boundary

Monitor sensor executive

Sensor input controller

Alarm conditions controller

Alarm output controller

Format display

Generate alarm signal

Set-up connection to phone net

Generate display

Generate pulses to line

FIGURE 11.12. First-level factoring for **Monitor Sensors.**

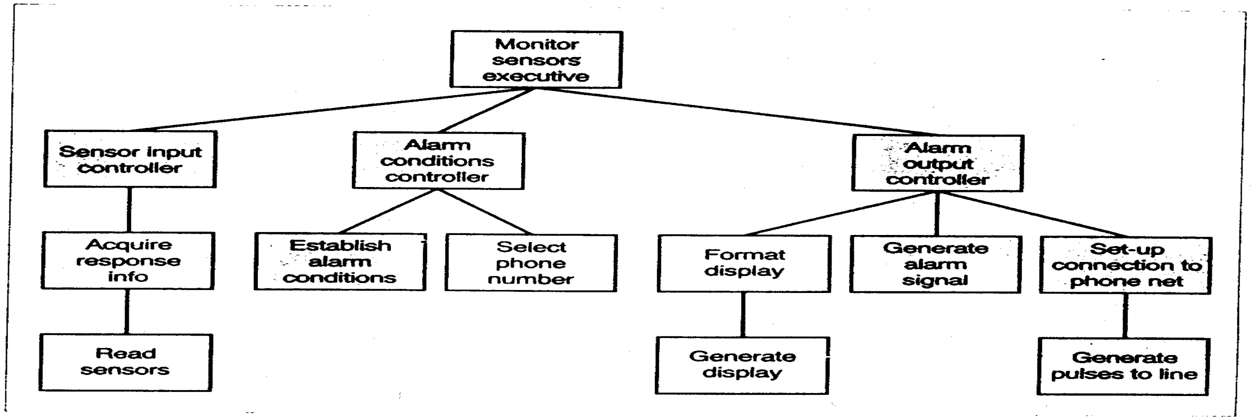


FIGURE 11.13. "First-cut" program structure (structure chart) for **Monitor Sensors.**

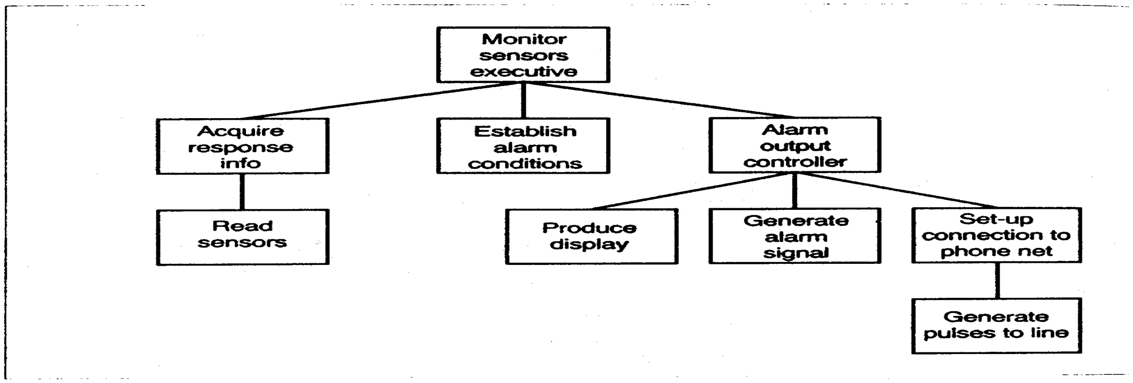


FIGURE 11.14. Refined program structure for **Monitor Sensors.**

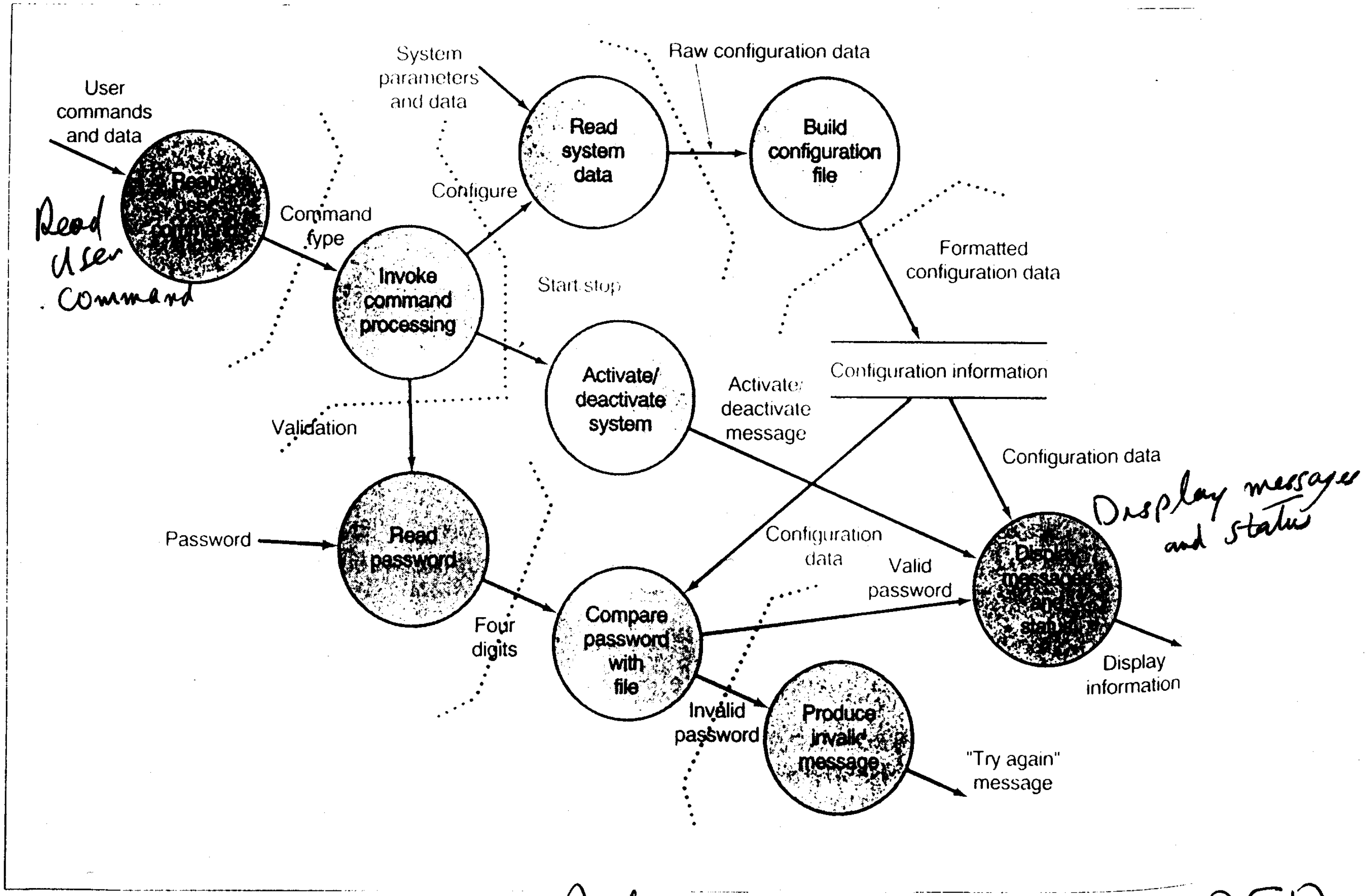


FIGURE 11.16. Establishing flow boundaries.

Refined User Interation DFD

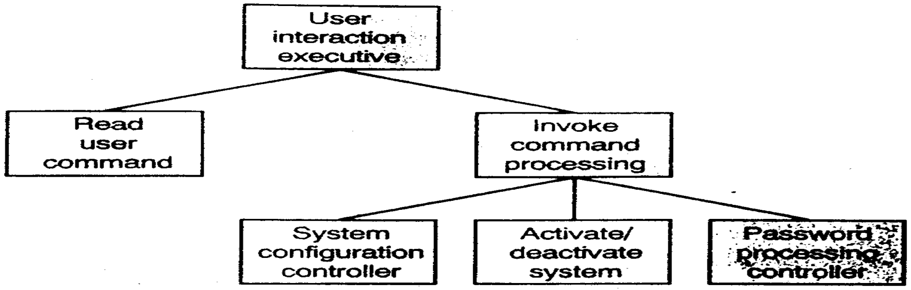

User interaction executive
Read user command
Invoke command processing
System configuration controller
Activate/ deactivate system
Password processing controller

FIGURE 11.18. First-level factoring for user interaction subsystem.

information flow characteristics. We have already noted that transform or transaction flow may be encountered. The action path-related "substructure" is developed using the design steps discussed in this and the preceding section.

As an example, consider the "password processing" information flow shown (inside shaded area) in Figure 11.16. The flow exhibits classic trans-

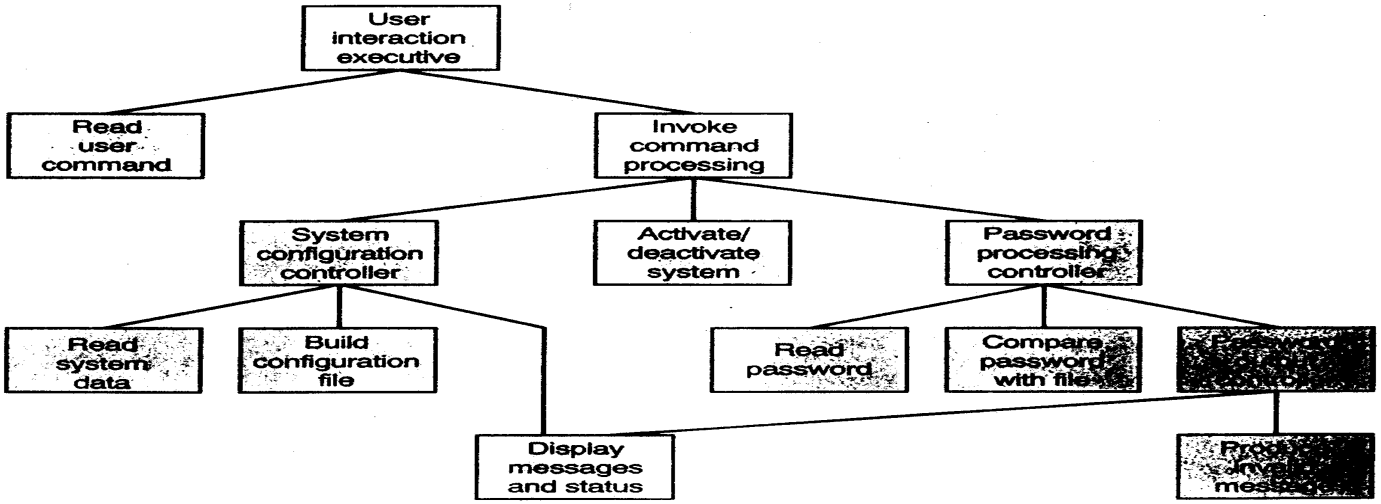

User interaction executive
Read user command
Invoke command processing
System configuration controller
Activate/ deactivate system
Password processing controller
Read system data
Build configuration file
Read password
Compare password with file
Display messages and status

FIGURE 11.19. "First-cut" program structure for user interaction subsystem

ATM Design Example

Recall first the ATM analysis, and use it to develop a design

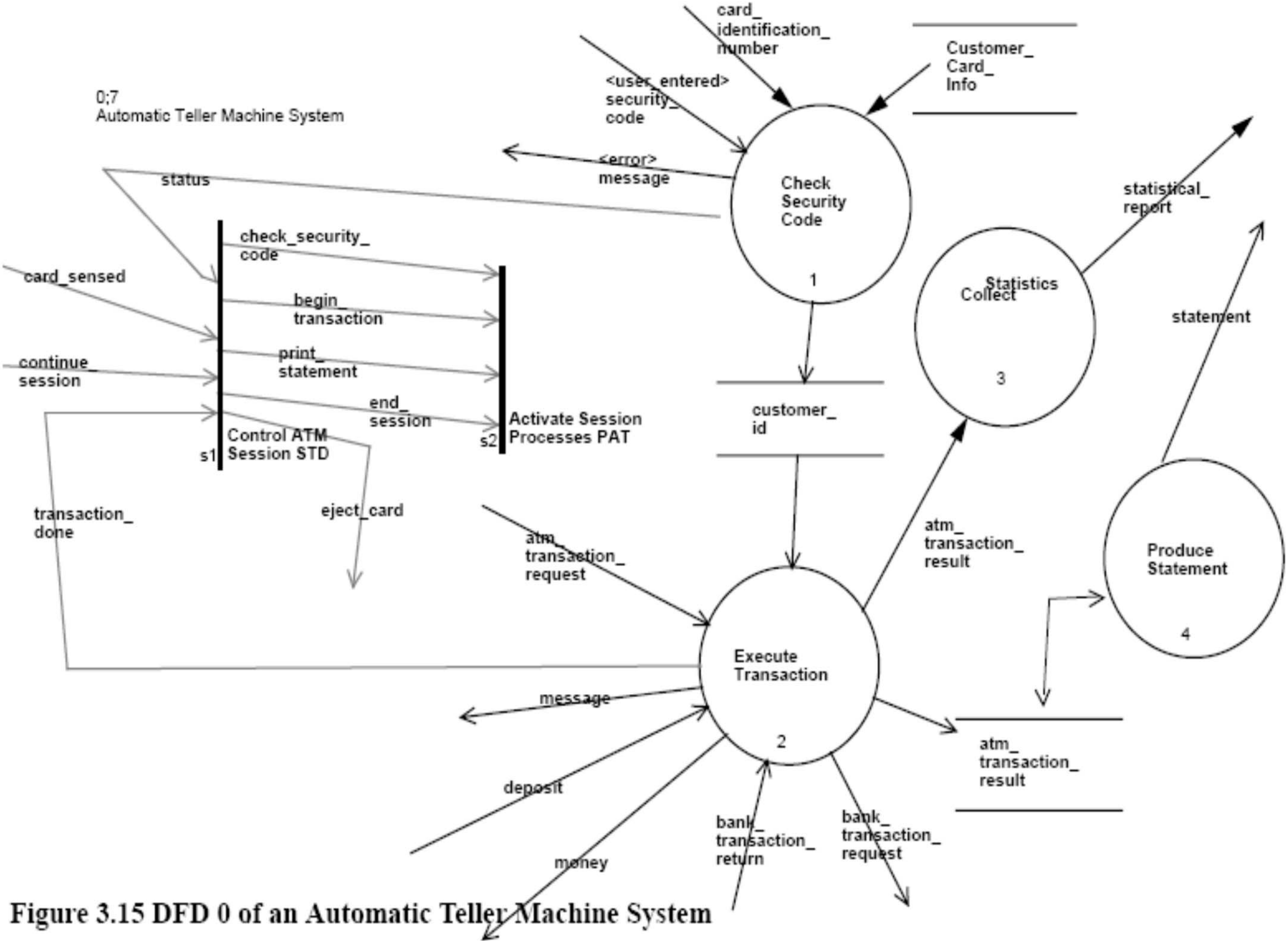
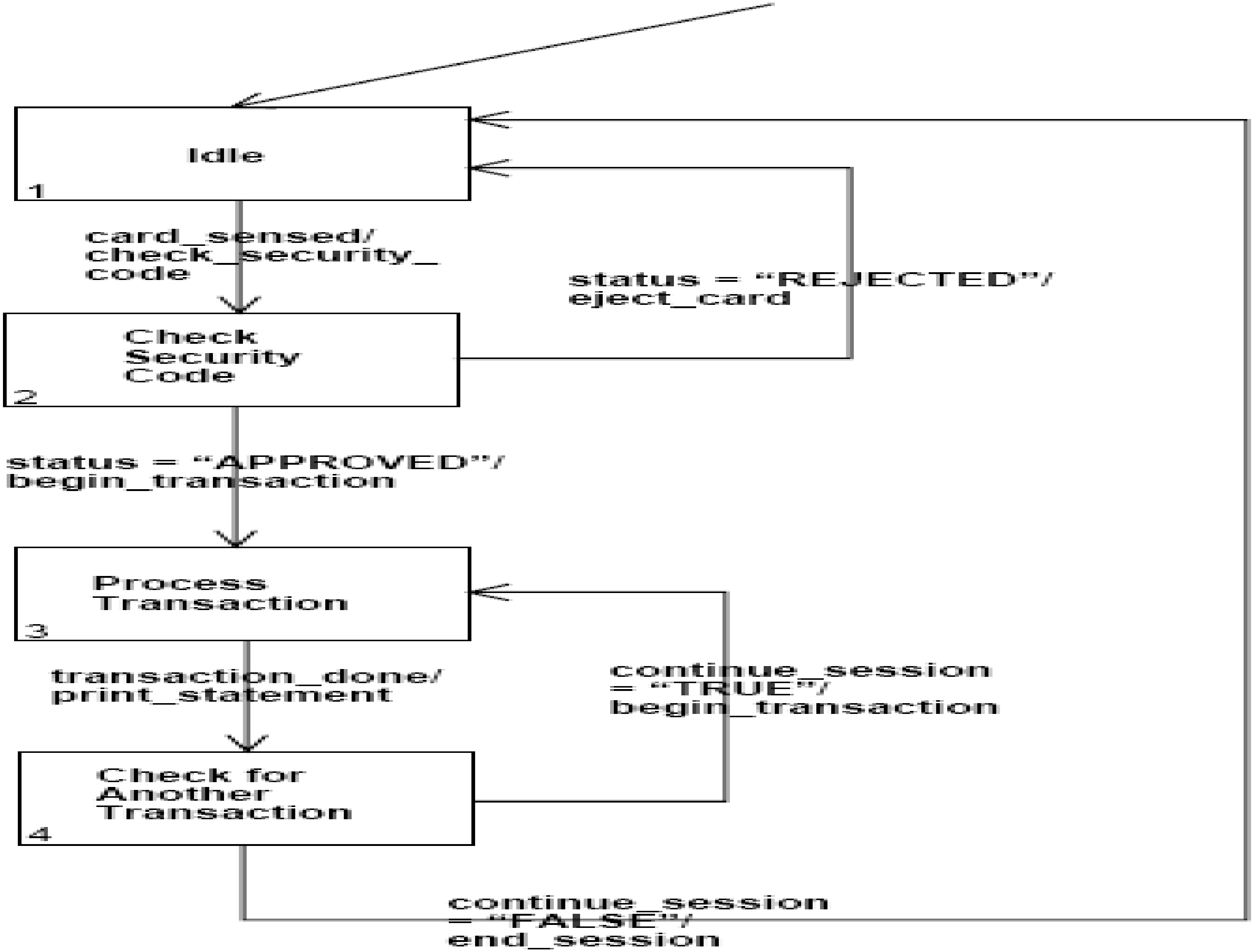


Figure 3.15 DFD 0 of an Automatic Teller Machine System

0-s1;5
Control ATM Session STD



gure 3.16 STD of a Control ATM Session

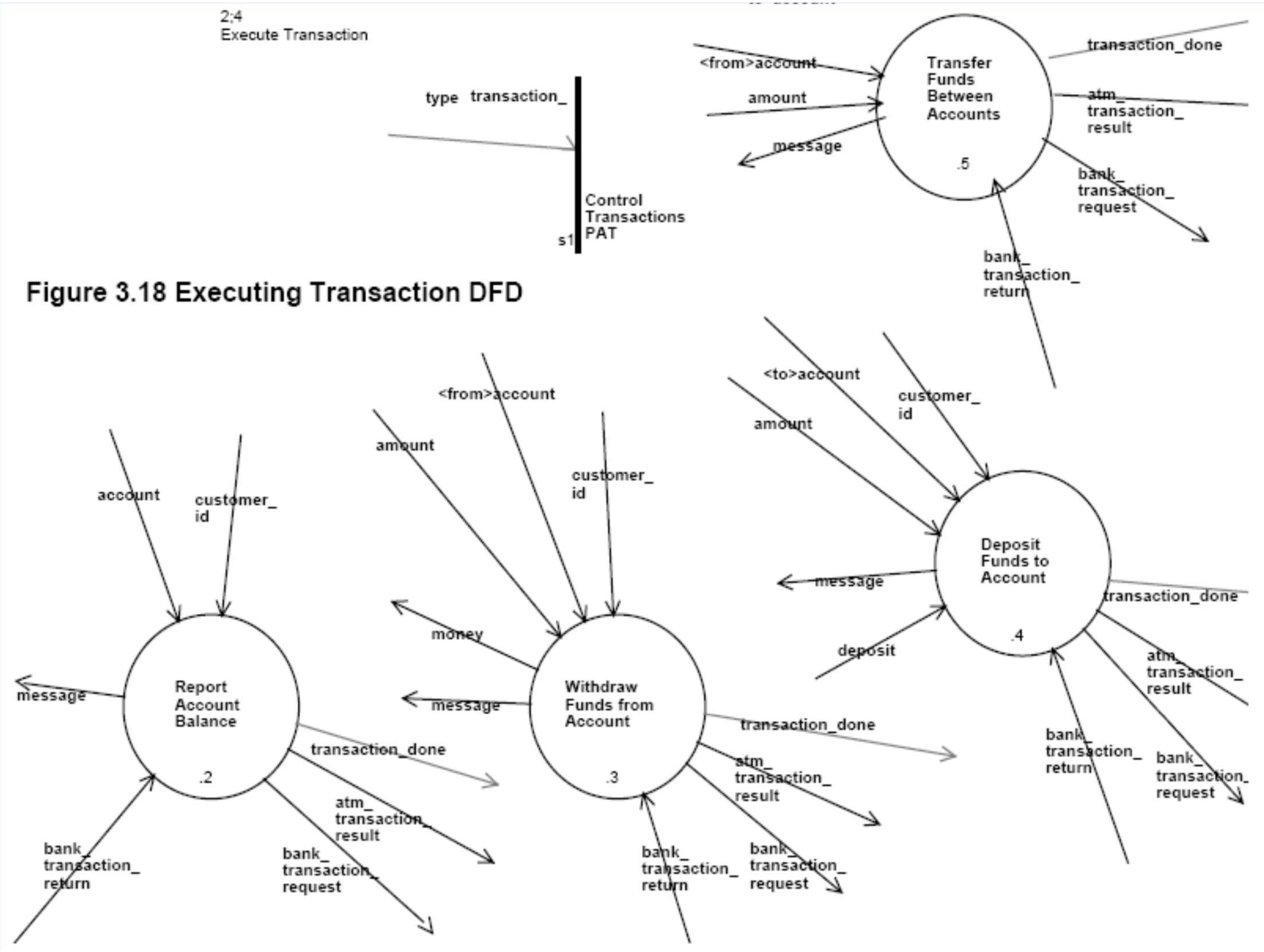


Figure 3.18 Executing Transaction DFD