ARCHITECTURAL DESIGN

Software Architecture

As shown in Figure 5.1, the *structured analysis* activity transforms the SRS document into a graphic model called the DFD model. On the other hand, the aim of *structured design* is to transform the results of the structured analysis (that is, the DFD model) into a structure chart. A structure chart represents the **software architecture.**

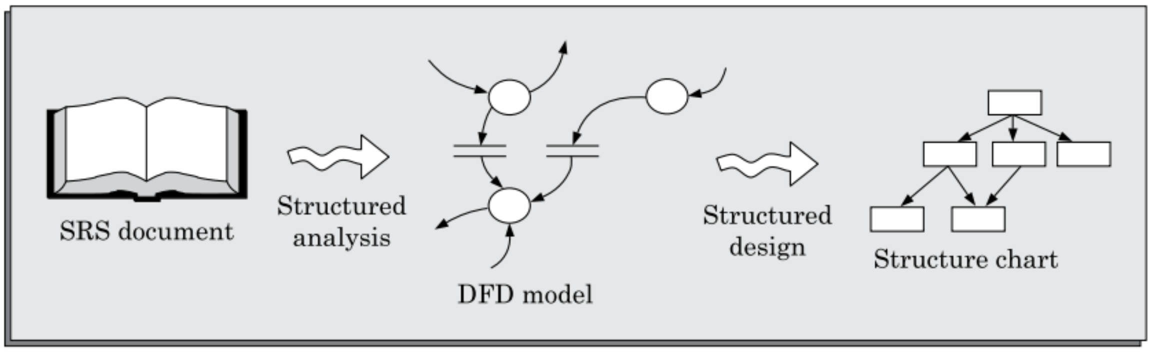


Figure 5.1: Structured analysis and structured design methodology.

Who does it?

Although a software engineer can design both data and architecture, the job is often allocated to specialists when large, complex systems are to be built.

Mapping Requirements into A Software Architecture:

The transition from information flow (represented as a DFD) to program structure, to do this there is a need to know the type of information flow, where the type of information flow is the driver for the mapping approach. In the following sections, we examine two flow types:

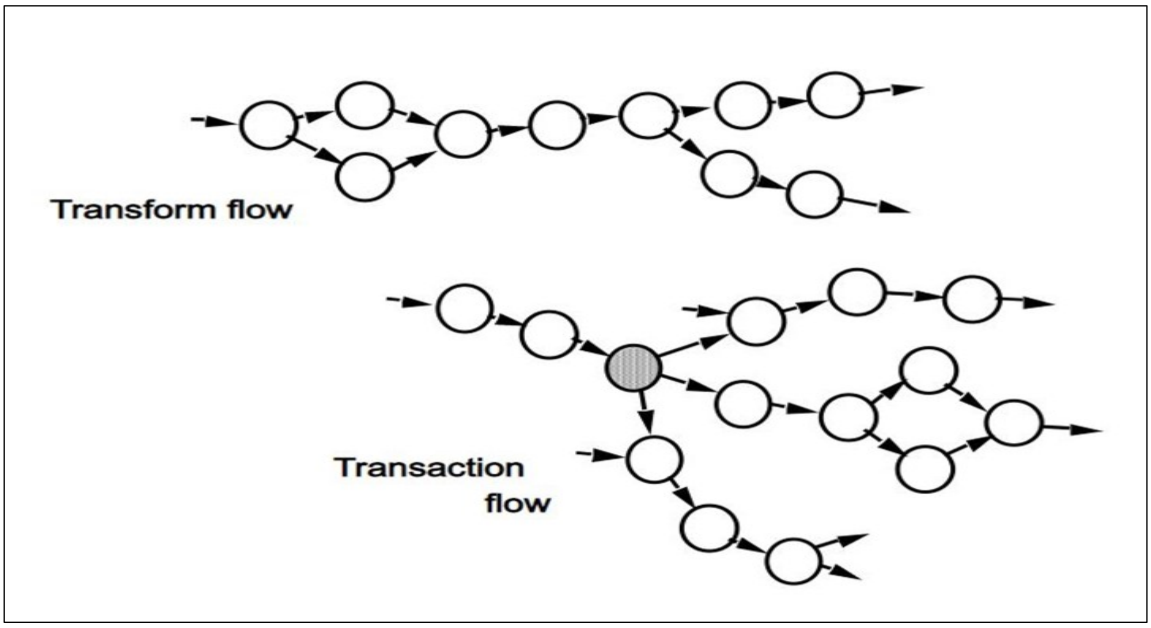
1- **Transform Flow**
2- **Transaction Flow**

At each level of transformation, it is important to first determine whether the transform or the transaction analysis is applicable to a particular DFD.

- **Whether to apply transform or transaction processing?**

Given a specific DFD of a model, how does one decide whether to apply transform analysis or transaction analysis?

For this, one would have to examine the data input to the diagram. The data input to the diagram can be easily observed because they are represented by dangling arrows. If all the data flow into the diagram are processed in similar ways (i.e., if all the input data flow arrows are incident on the same bubble in the DFD) then transform analysis is applicable. Otherwise, transaction analysis is applicable.



An Example

transform mapping and transaction mapping are described by applying design steps to an example system—the *SafeHome* security software.

The product monitors the real world and reacts to changes that it encounters. It also interacts with a user through a series of typed inputs and alphanumeric displays.

The home security function would protect against and recognize a variety of undesirable "situations" such as illegal entry, fire, flooding, and others. It'll use wireless sensors to detect each situation. It can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

- The SafeHome security function enables the homeowner to configure the security system when it is installed, and interacts with the homeowner through a **control panel.**
- During installation Each **sensor** is assigned a number and type, The system enables the homeowner to monitor all sensors connected to the security system.
- A master password is programmed for arming and disarming the system.
- When a sensor event is recognized, the software invokes an audible **alarm** attached to the system.
- After a delay time that is specified by the homeowner during system configuration activities, the software dials a telephone number of a monitoring service, provides information about the location, reporting the nature of the event that has been detected.
- The homeowner receives **security information via a control panel**, the PC, or a browser, collectively called an interface.

TRANSFORM MAPPING

Transform mapping is a set of design steps that allows a DFD with transform flow characteristics to be mapped into a specific architectural style. It identifies the primary functional components (modules) and the input and output data for these components.

(We consider only the portion of the SafeHome security function that uses the *monitor sensors*).

Design Steps:

To map data flow diagrams into a software architecture, you would initiate the following design steps:

- **Step 1.** Review the fundamental system model.

The fundamental system model or context diagram depicts the security function as a single transformation. Figure 5.2 illustrate a level 0 DFD (context level).

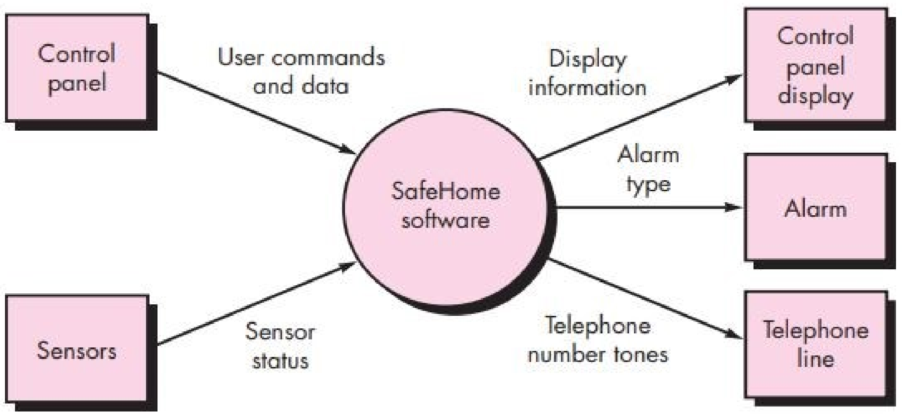


Figure 5.2: Context-level DFD for the SafeHome security function

➢ **Step 2.** Review and refine data flow diagrams for the software.

Information obtained from the requirements model is refined to produce greater detail as shown in level 2 and level 3 DFD.

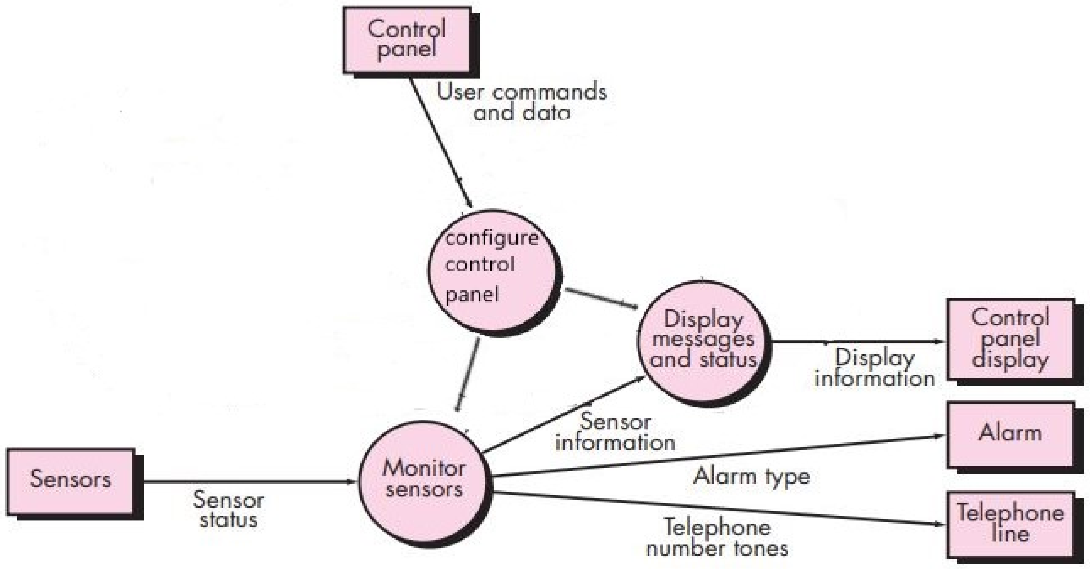


Figure 5.3: Level 1 DFD for the SafeHome security function

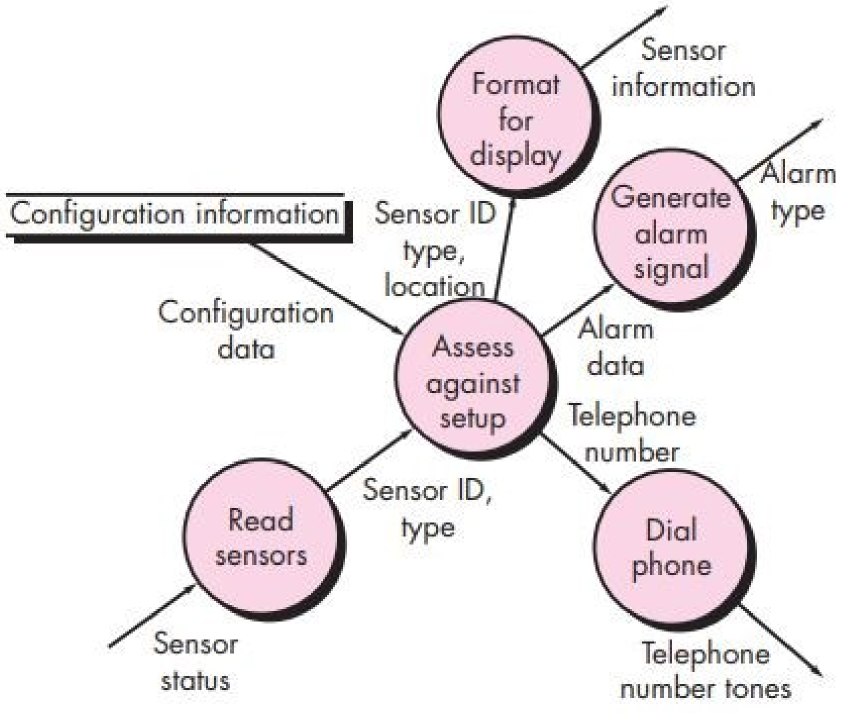


Figure 5.4: Level 2 DFD that refines the monitor sensors transform

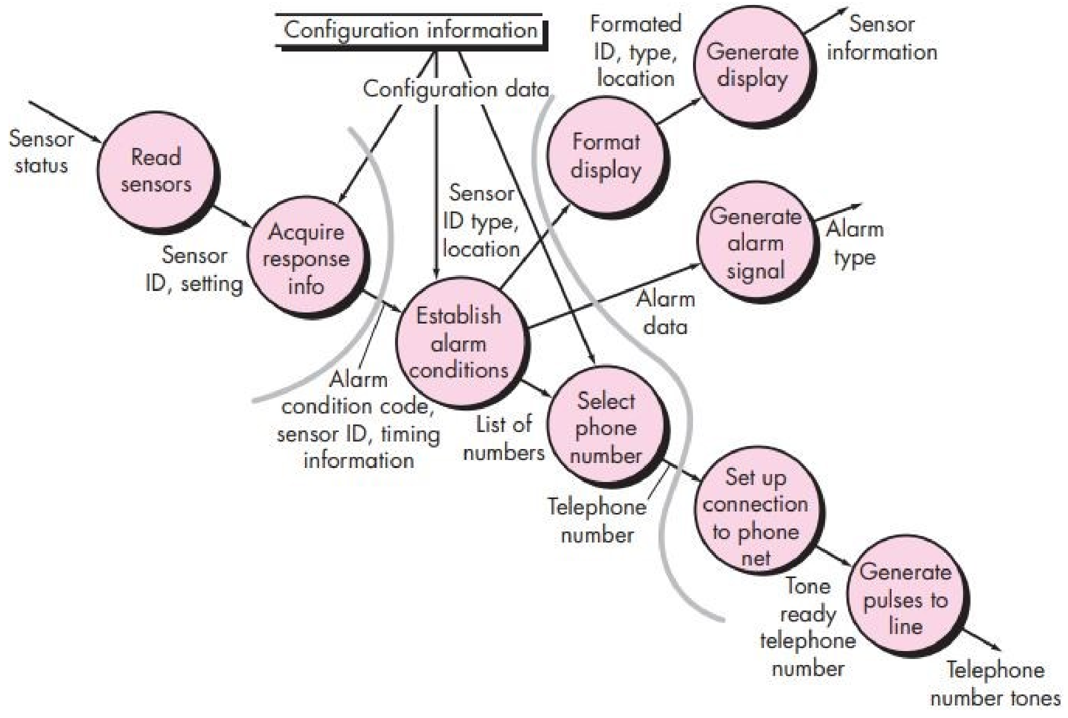


Figure 5.5: Level 3 DFD for monitor sensors with flow boundaries

➢ **Step 3.** Determine whether the DFD has transform or transaction flow characteristics.

Evaluating the DFD (Figure 5.5), we see data entering the software along one incoming path and exiting along three outgoing paths. Therefore, an overall transform characteristic will be assumed for information flow.

➢ **Step 4.** Isolate the transforming center by specifying incoming and outgoing flow boundaries.

- Information enters the system along paths that transform external data into an internal form. These paths are identified as incoming flow.
- At the kernel of the software, a transition occurs.

- Incoming data are passed through a transform center and begin to move along paths that now lead "out" of the software. Data moving along these paths are called outgoing flow.

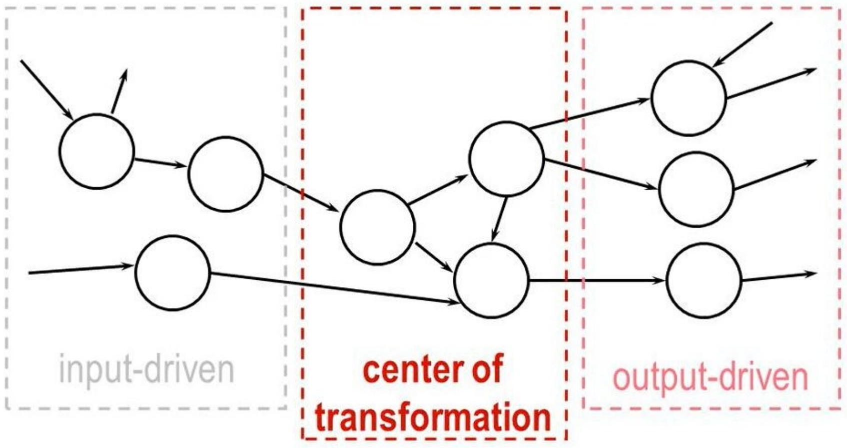


Figure 5.6: parts of transformation analysis.

➢ **Step 5.** Perform “first-level factoring.”

Factoring leads to a program structure in which top-level components perform decision-making and low-level components perform most input, computation, and output work. Middle-level components perform some control and do moderate amounts of work

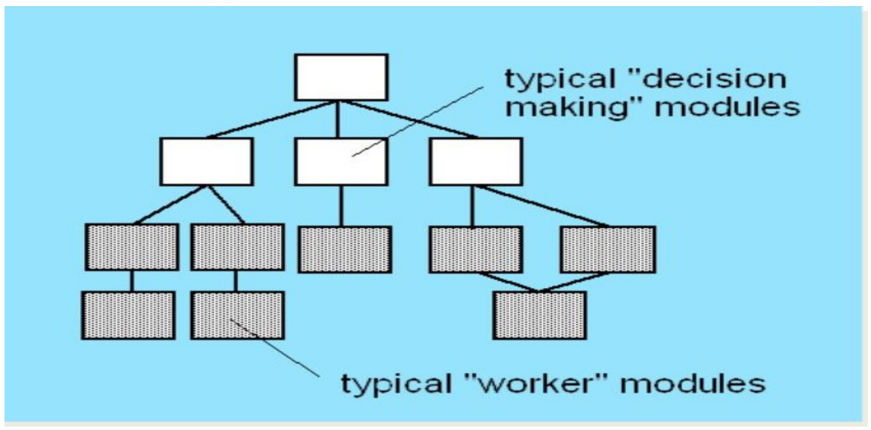


Figure 5.7: factoring.

This first-level factoring for the monitor sensors subsystem is illustrated in Figure 5.8. A main controller (called monitor sensors executive) resides at the top of the program structure and coordinates the following subordinate control functions:

• An incoming information processing controller, called sensor input controller, coordinates the receipt of all incoming data.

• A transform flow controller, called an alarm conditions controller, supervises all operations on data in internalized form.

• An outgoing information processing controller, called alarm output controller, coordinates the production of output information.

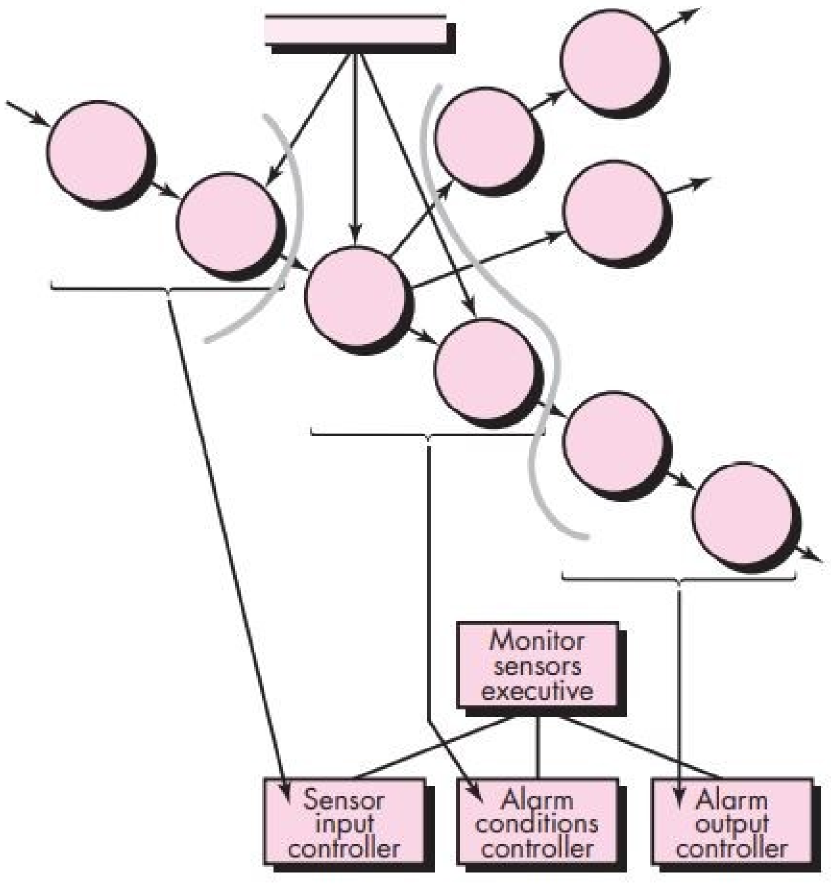


Figure 5.8: First-level factoring for monitor sensors

➢ **Step 6.** Perform "second-level factoring."

- Beginning at the transform center boundary and moving outward along incoming and then outgoing paths, transforms are mapped into subordinate levels of the software structure.
- The transform center of monitor sensors subsystem software is mapped somewhat differently. Each of the data conversion or calculation transforms of the transform portion of the DFD is mapped into a module subordinate to the transforming controller.
- The general approach to second-level factoring is illustrated in Figure 5.9.
- A completed first-iteration architecture is shown in Figure 5.10.

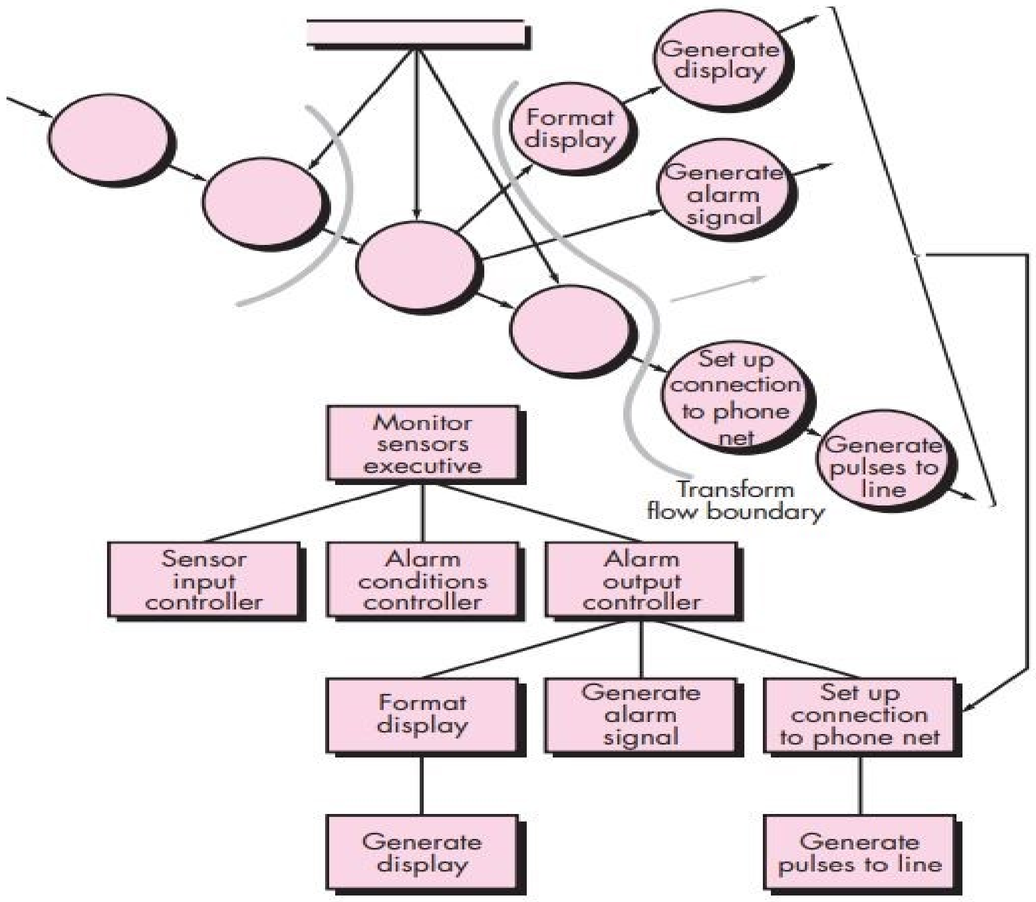


Figure 5.9: Second-level factoring for monitor sensors

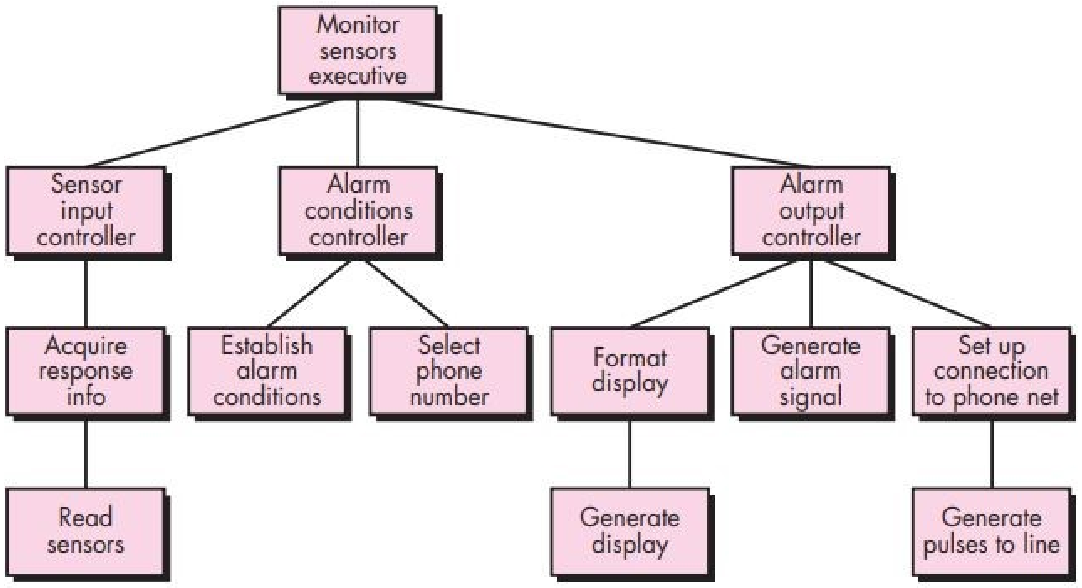
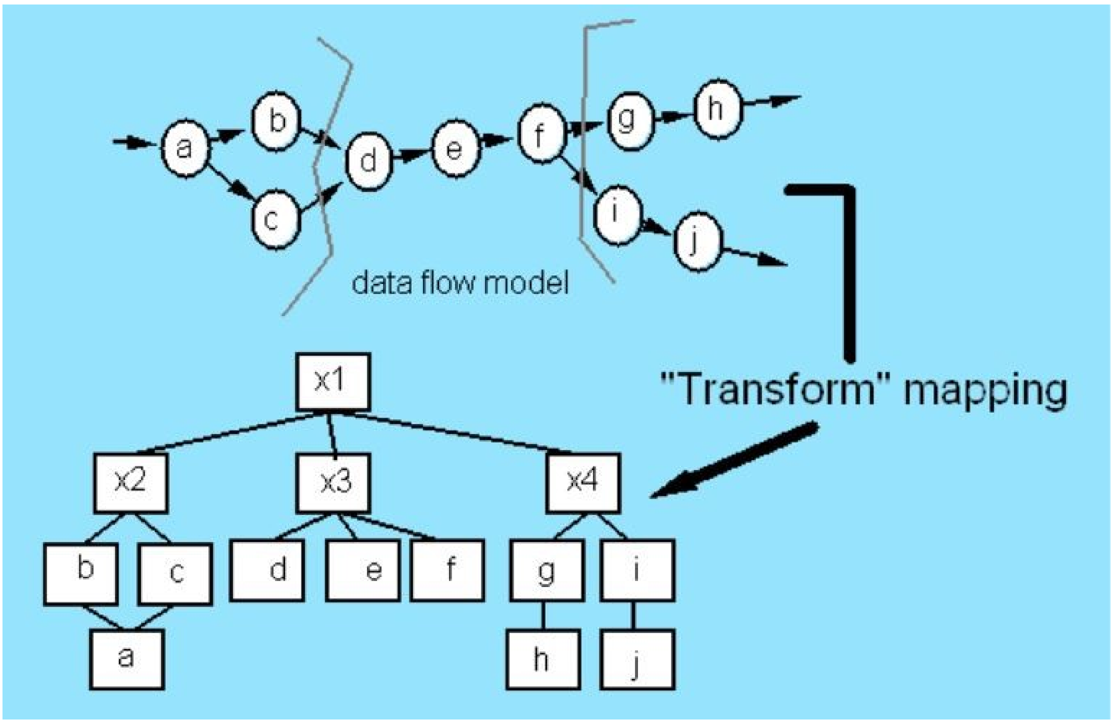


Figure 5.10: First-iteration structure for monitor sensors

➢ **Step 7.** Refine the first-iteration architecture using design heuristics for improved software quality.

Further Example:



Transaction Mapping

Transaction flow is characterized by data moving along an incoming path that converts external world information into a transaction. The transaction flows along one of many action paths is initiated. The hub of information flow from which many action paths is called a transaction center.

(Transaction mapping will be illustrated by considering the *user interaction* subsystem of the SafeHome software).

<u>Design Steps:</u>

The design steps for transaction mapping are similar to the steps for transform mapping. A major difference lies in the mapping of DFD to software structure.

➢ **Step 1.** Review the fundamental system model.
As shown in Figure 5.2.

➢ **Step 2.** Review and refine data flow diagrams for the software.
As shown in Figures 5.3, 5.11, and 5.12.

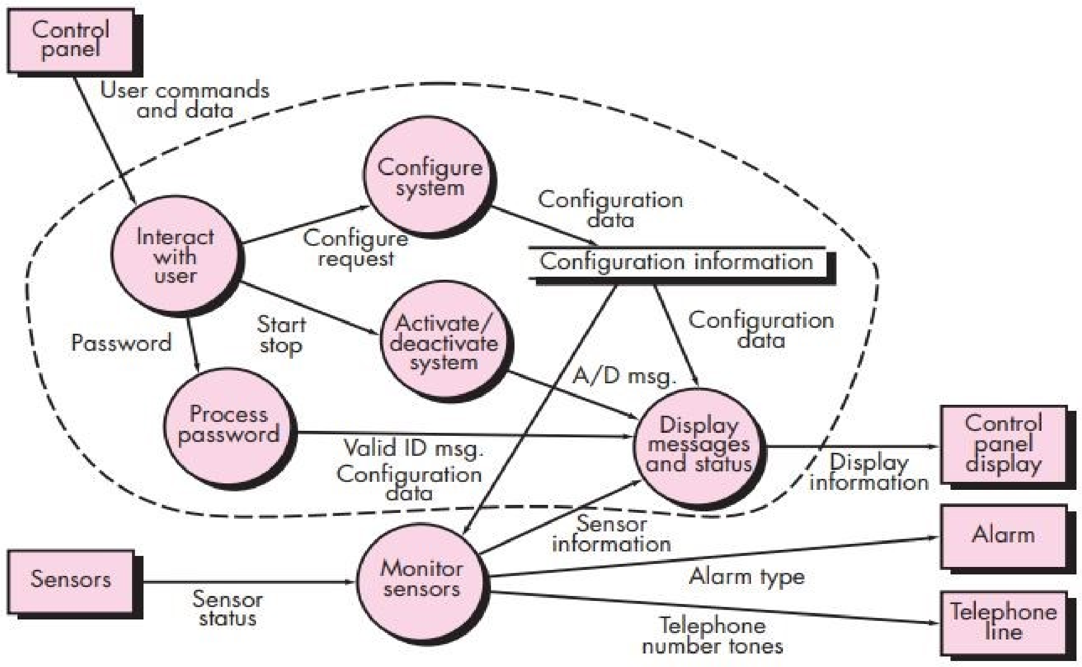


Figure 5.11: Level 2 DFD that refines the user interaction subsystem.

➢ **Step 3.** Determine whether the DFD has transform or transaction flow characteristics.

As shown in figure 5.12, user commands flows into the system and result in additional information flow along one of three action paths. A single data item, *command type*, causes the data flow to fan outward from a hub. Therefore, the overall data flow characteristic is transaction oriented.

➢ **Step 4.** Identify the transaction center and the flow characteristics along each of the action paths.

The transaction center is at the beginning of a number of workflows flowing from it as shown in Figure 5.13.

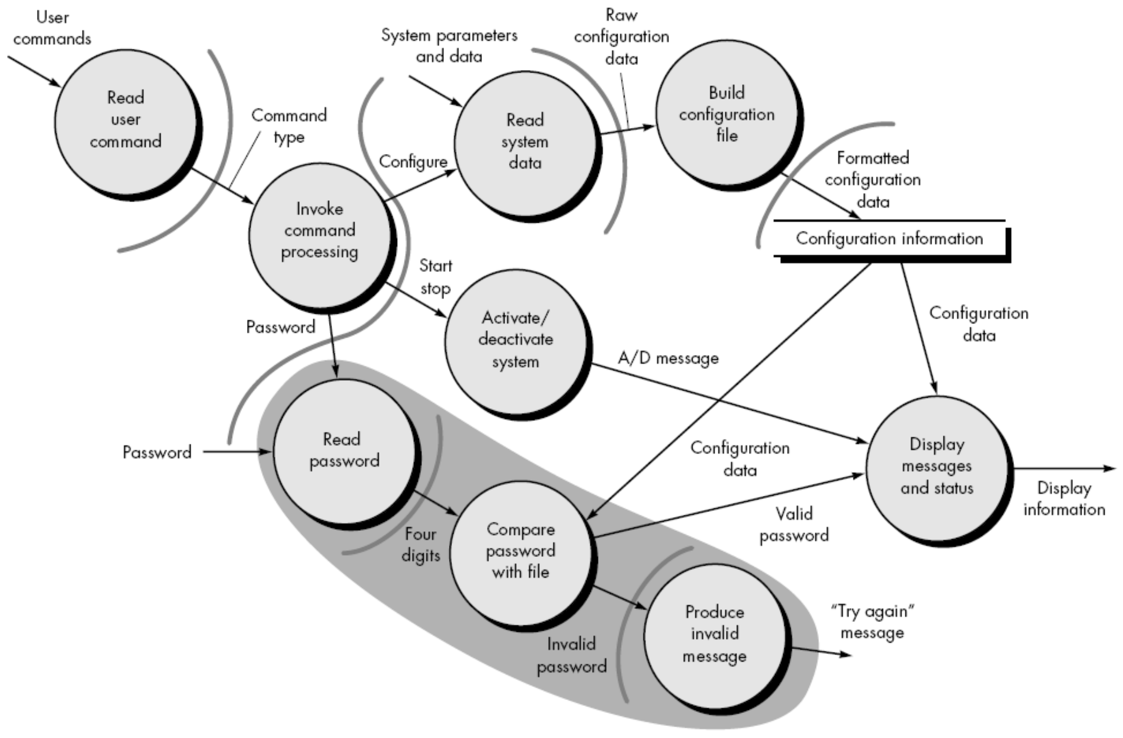


Figure 5.12: Level 3 DFD that refines the user interaction subsystem.

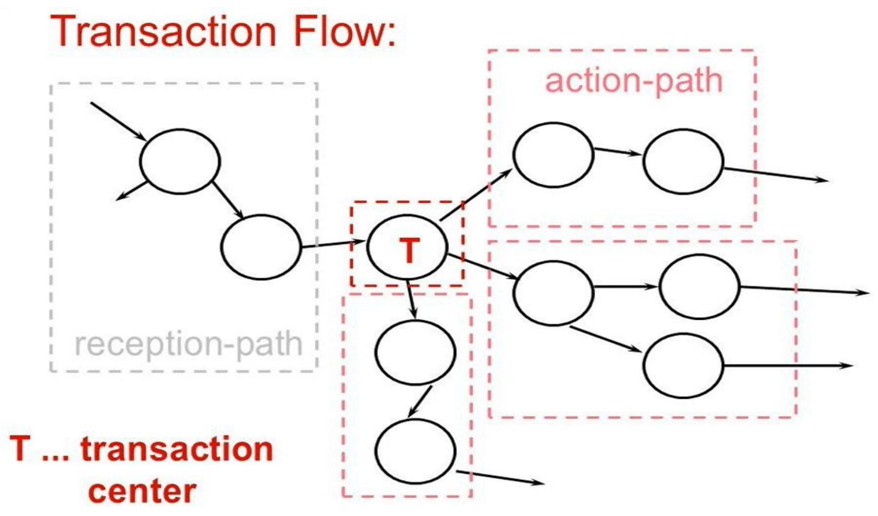


Figure 5.13: parts of transaction analysis.

For the flow shown in figure 5.12, the *invoke command processing* bubble is the transaction center. The incoming path and all action paths must also be isolated. For example, the "password" path has transformed characteristics. Incoming, transform, and outgoing flow are indicated with boundaries.

- **Step 5.** Map the DFD in a program structure amenable to transaction processing.
  Transaction flow is mapped into an architecture that contains an incoming branch and a dispatch branch. The structure of the incoming branch is developed in much the same way as transform mapping. Starting at the transaction center, bubbles along the incoming path are mapped into modules. The structure of the dispatch branch contains a dispatcher module that controls all subordinate action modules. Each action flow path of the DFD is mapped to a structure that corresponds to its specific flow characteristics. This process is illustrated in figure 5.14. Considering the user interaction subsystem data flow, first-level factoring for step 5 is shown in figure 5.15.

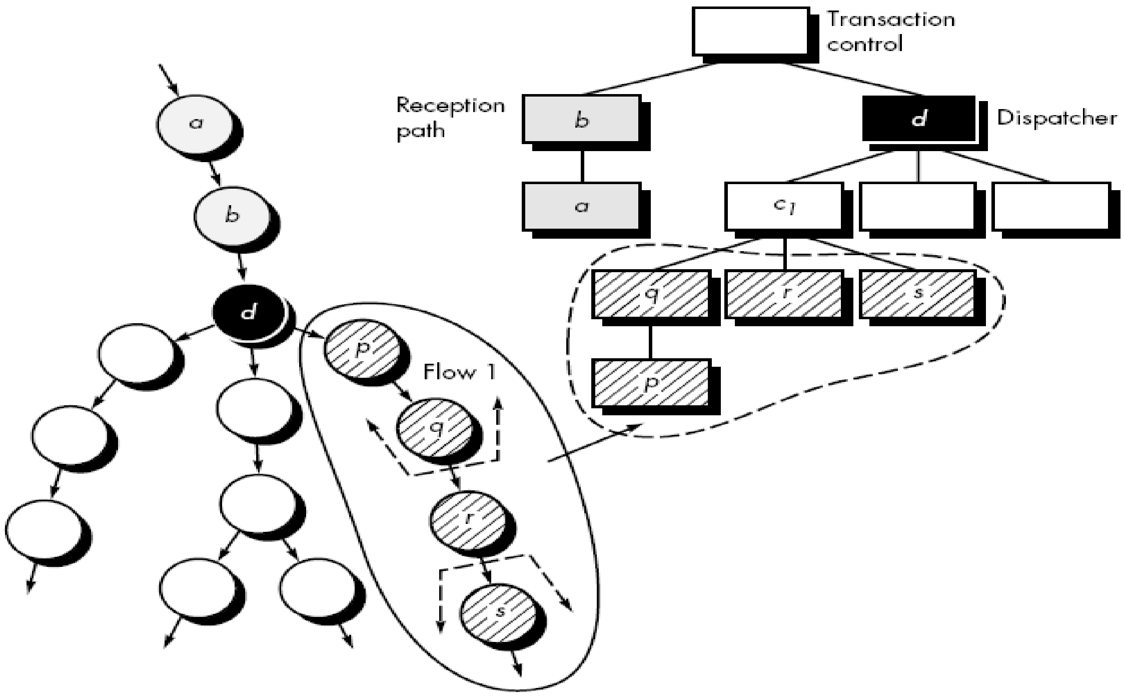


Figure 5.14: Transaction mapping

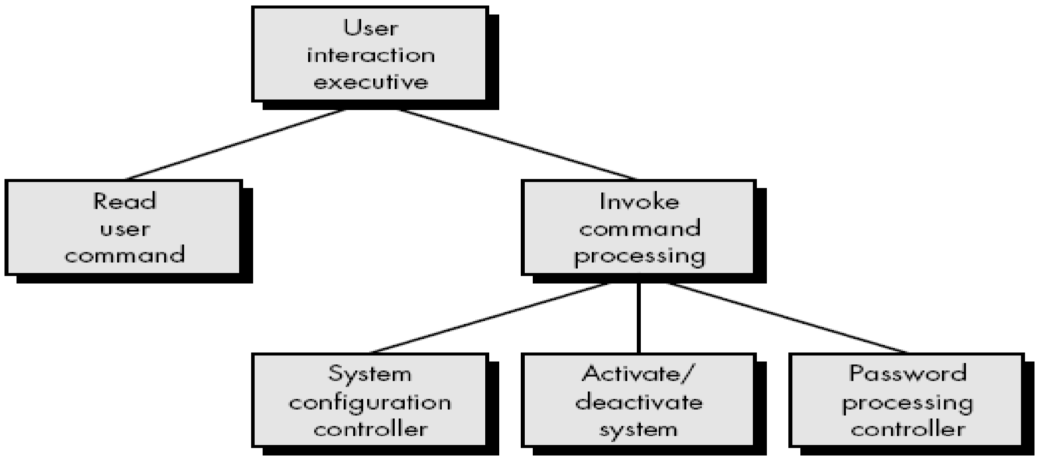

User interaction executive
Read user command
Invoke command processing
System configuration controller
Activate/ deactivate system
Password processing controller

Figure 5.15: First-level factoring for user interaction subsystem.

- **Step 6**. Factor and refine the transaction structure and the structure of each action path. As shown in Figure 5.16.
- **Step 7.** Refine the first-iteration architecture using design heuristics for improved software quality.

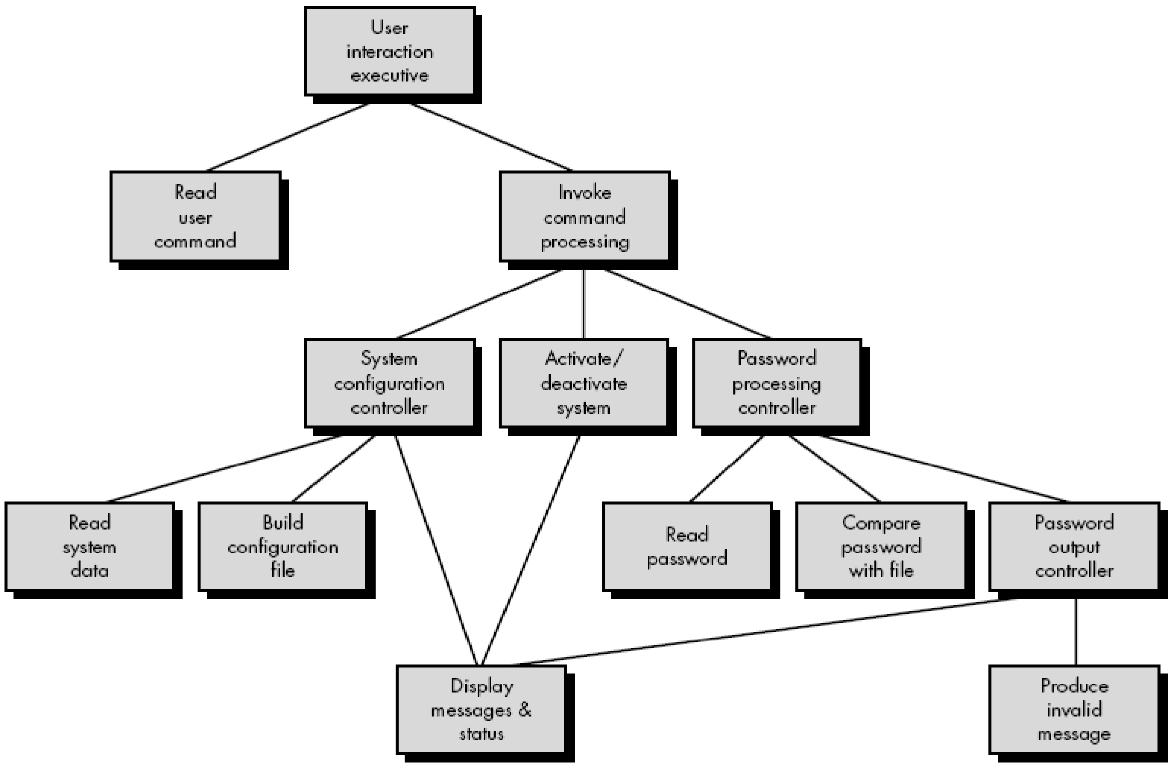
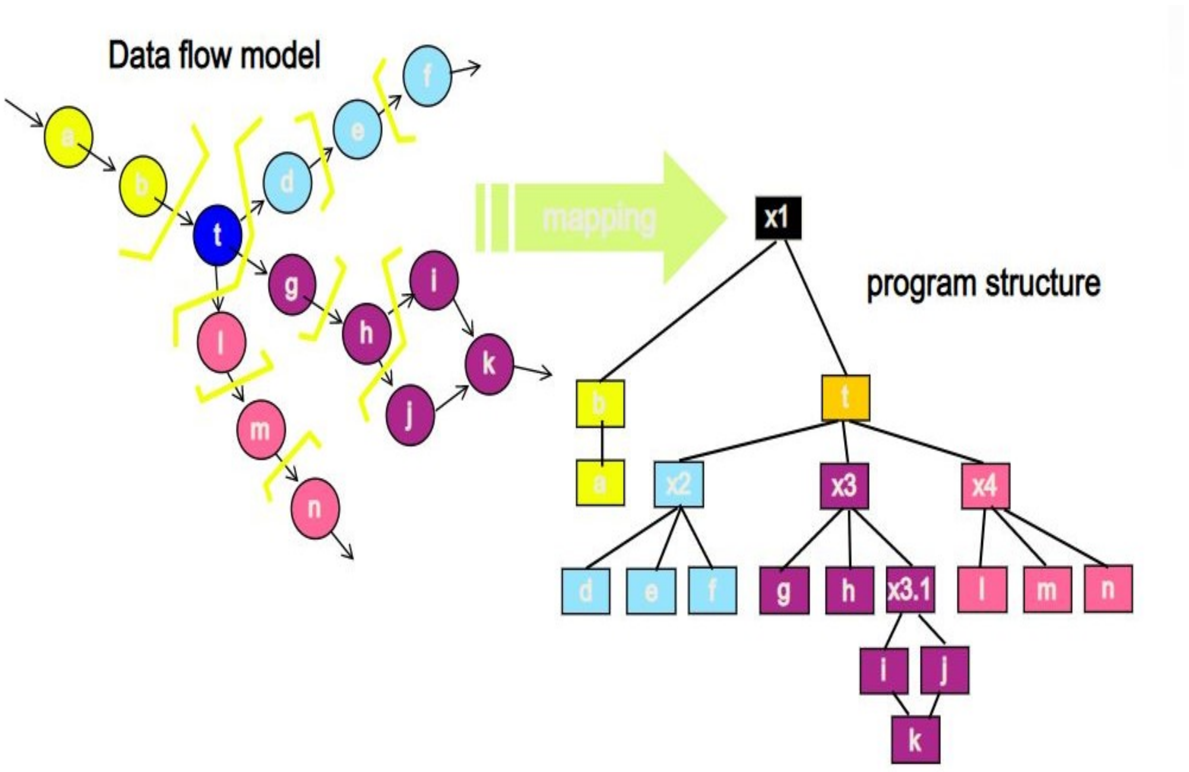

User interaction executive
Read user command
Invoke command processing
System configuration controller
Activate/ deactivate system
Password processing controller
Read system data
Build configuration file
Read password
Compare password with file
Password output controller
Display messages & status
Produce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

Figure 5.16: First-iteration architecture for user interaction subsystem.

Further Example:


Data flow model
a
b
t
d
e
f
g
h
i
j
k
l
m
n
mapping
program structure
x1
b
a
t
x2
x3
x4
d
e
f
g
h
x3.1
l
m
n
i
j
k