DESIGN METHOD
---DATA AND ARCHITECTURAL DESIGN

DATA DESIGN

- What is data design?
  - Transform the information domain model created during analysis into data structure required to implement the software
  - Well-designed data lead to better program structure and modularity, reduced procedural complexity

DATA DESIGN PROCESS

- Define data structures identified during the requirements and specification phase.
  - Often base decision on algorithm to be used.
- Identify all program modules that must operate directly upon the data structure
  - Constrain the scope of effect of data design decisions
- Or, from OO perspective, define all operations performed on the data structure

PRINCIPLES OF DATA DESIGN

- A data dictionary should be established and used for both data and program design
- Low-level data design decisions should be deferred until late in the design process
- A library of useful data structures and the operations that may be applied to them should be developed. -- Reuse
  - E.g., stacks, lists, arrays, queues

PRINCIPLES OF DATA DESIGN (CONT.)

- The representation of data structures should be known only to those modules that must make direct use of the data contained within the structure.
  - Information hiding
- The software design and programming languages should support the specification and realization of abstract data types.

ARCHITECTURAL DESIGN

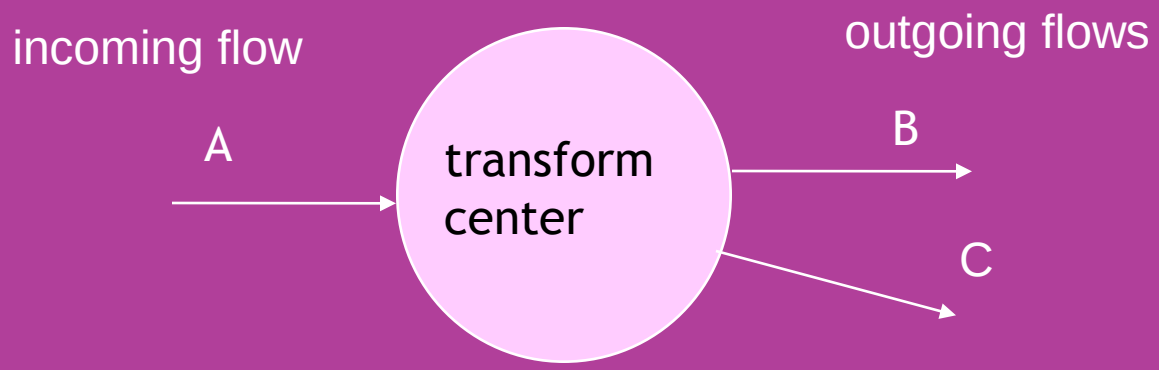
- Objective
  - develop a modular program structure and represent control relationships between modules
- Data flow-oriented design
  - amenable to a broad range of applications
  - very useful when information is processed sequentially, such as microprocessor control application; complex, numerical analysis procedure; etc.
  - two approaches (transform and transaction mapping)

ARCHITECTURAL DESIGN PROCESS

- Five-step Process
  - the type of information flow is established
  - flow boundary are indicated
  - data flow diagram is mapped into program structure
  - control hierarchy is defined by factoring
  - resultant structure is refined using design measures heuristics

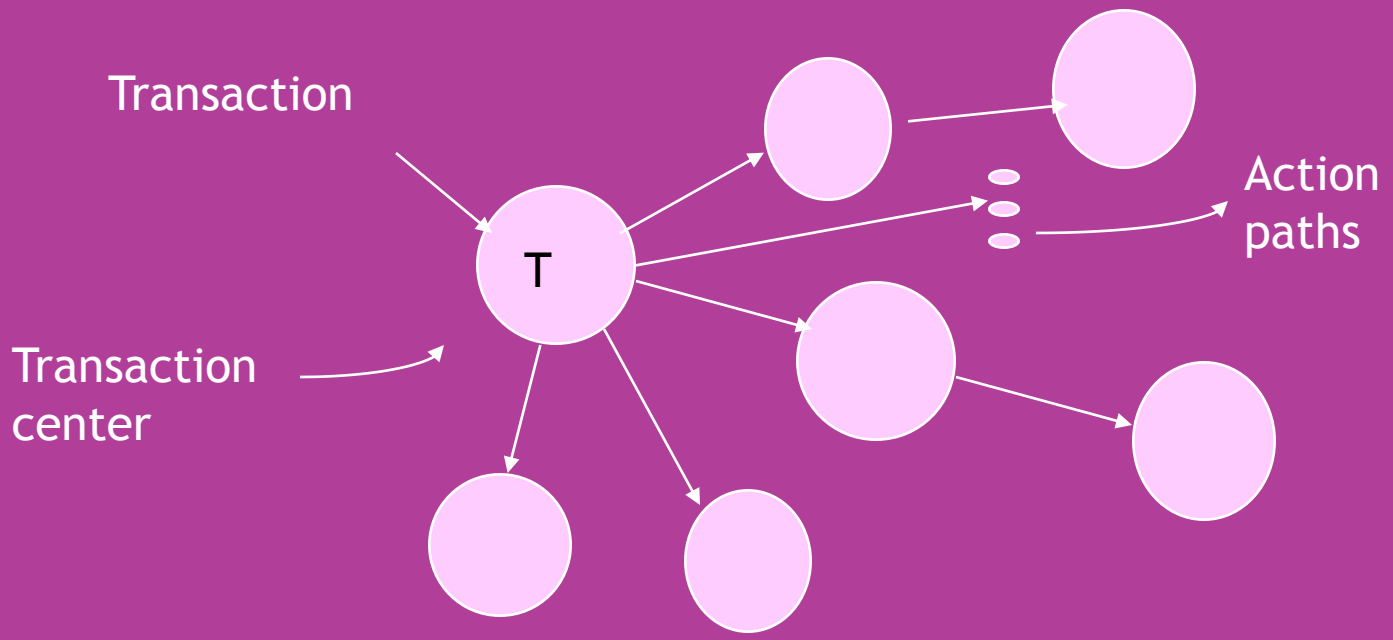
ARCHITECTURAL DESIGN PROCESS (CONT.)

- Transform Flow



ARCHITECTURAL DESIGN PROCESS (CONT.)

- Transaction Flow



TRANSFORM MAPPING

- Allow data flow diagram(DFD) with transform flow characteristics to be mapped into a predefined template for program structure

LEVEL 0 SAFEHOME DFD

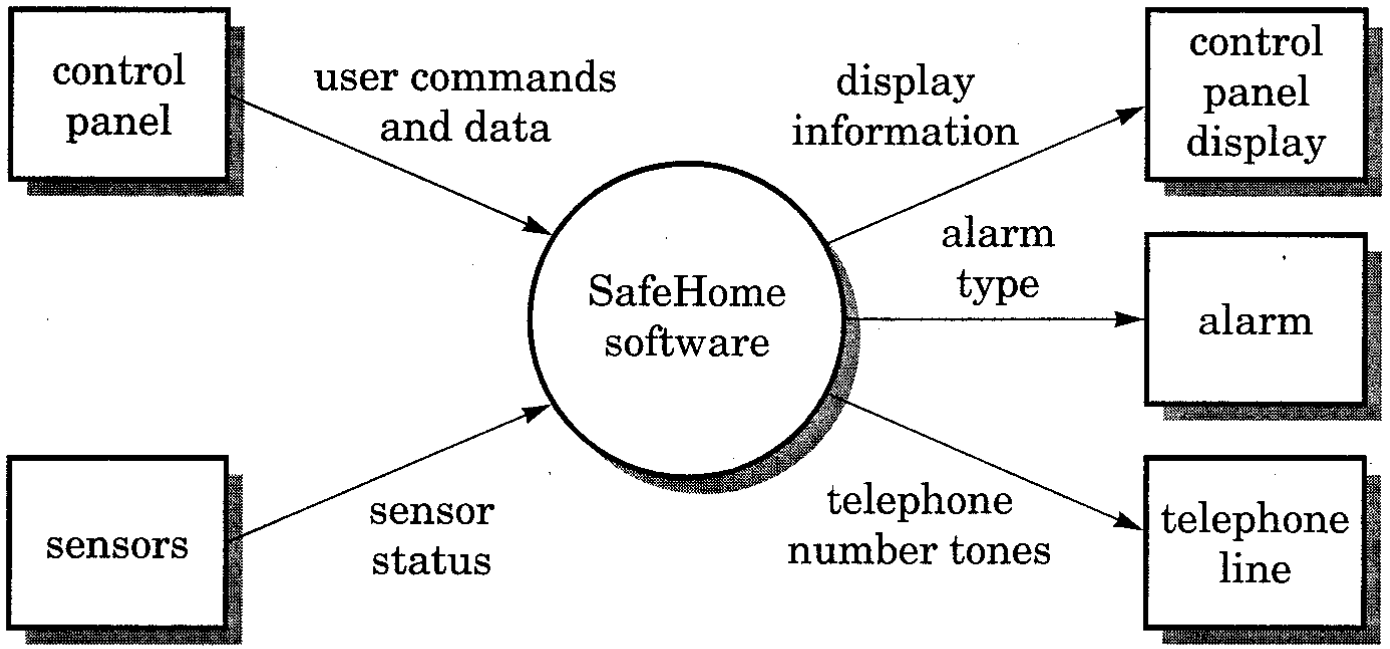


FIGURE 14.3.
Context-level DFD for SafeHome

LEVEL 1 SAFEHOME DFD

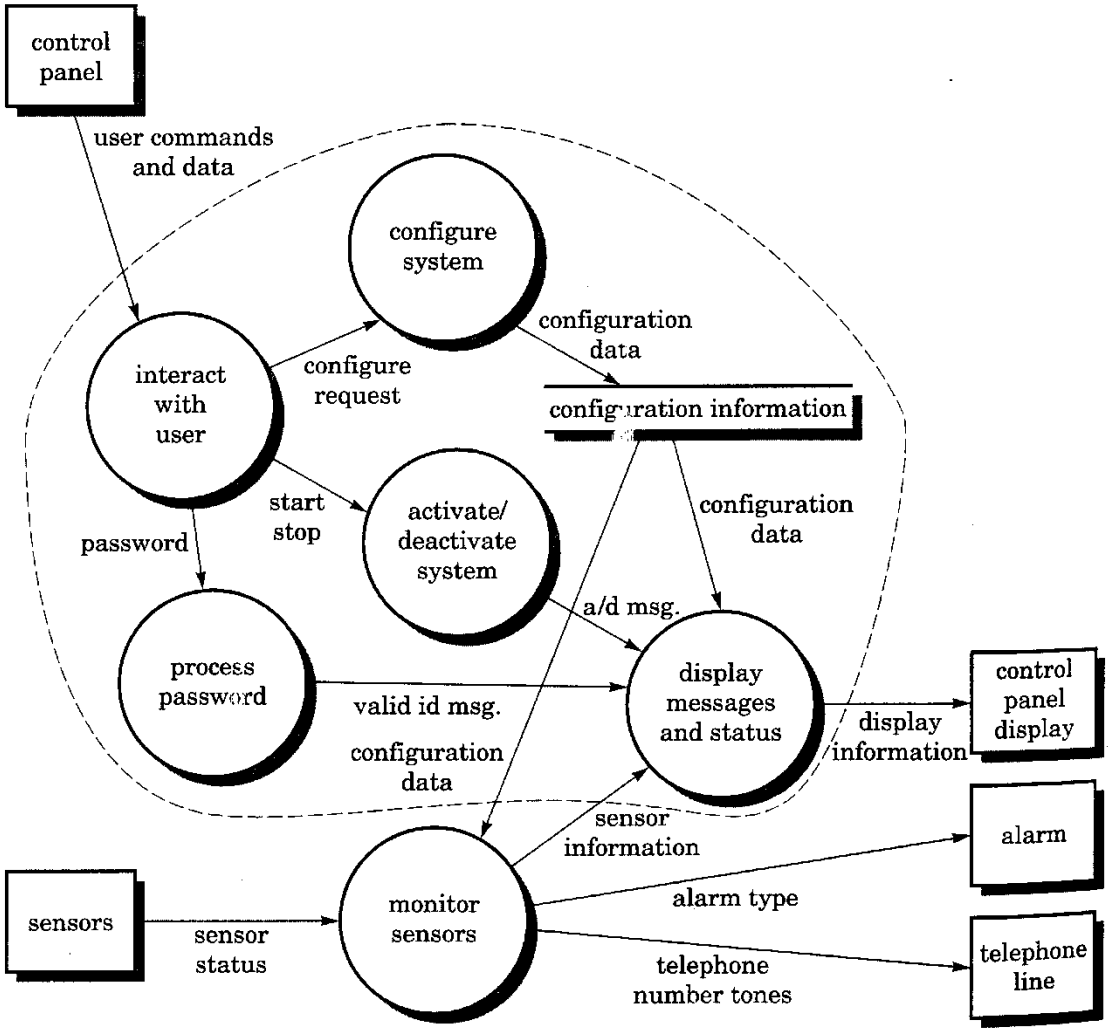


FIGURE 14.4. Level 1 DFD for SafeHome

LEVEL 2 SAFEHOME DFD - MONITOR

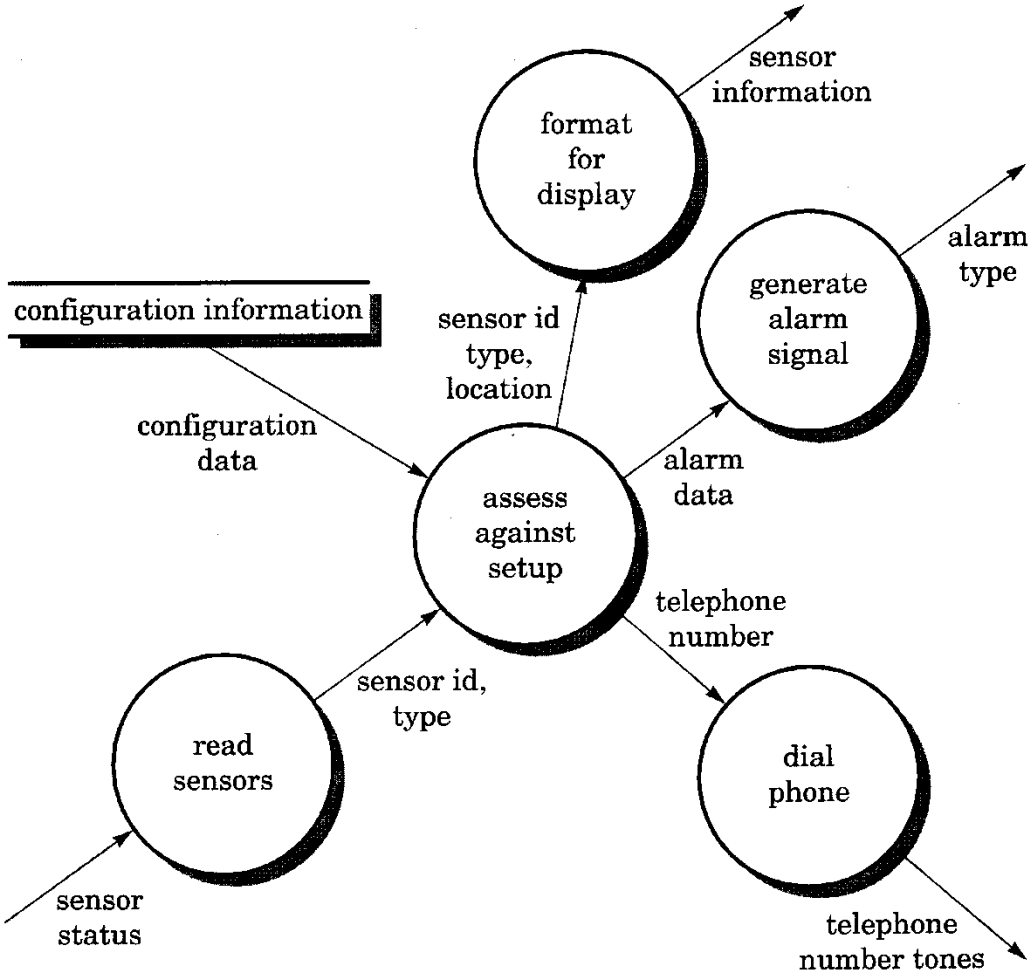


FIGURE 14.5.
Level 2 DFD that refines the *monitor sensors* process

TRANSFORM MAPPING (CONT)

- Design steps
  - Step 1. Review the fundamental system model.
  - Step 2. Review and refine data flow diagrams for the software.
  - Step 3. Determine whether DFD has transform or transaction flow characteristics.

    in general---transform flow

    special case---transaction flow

LEVEL 3 DFD FOR MONITOR SENSORS

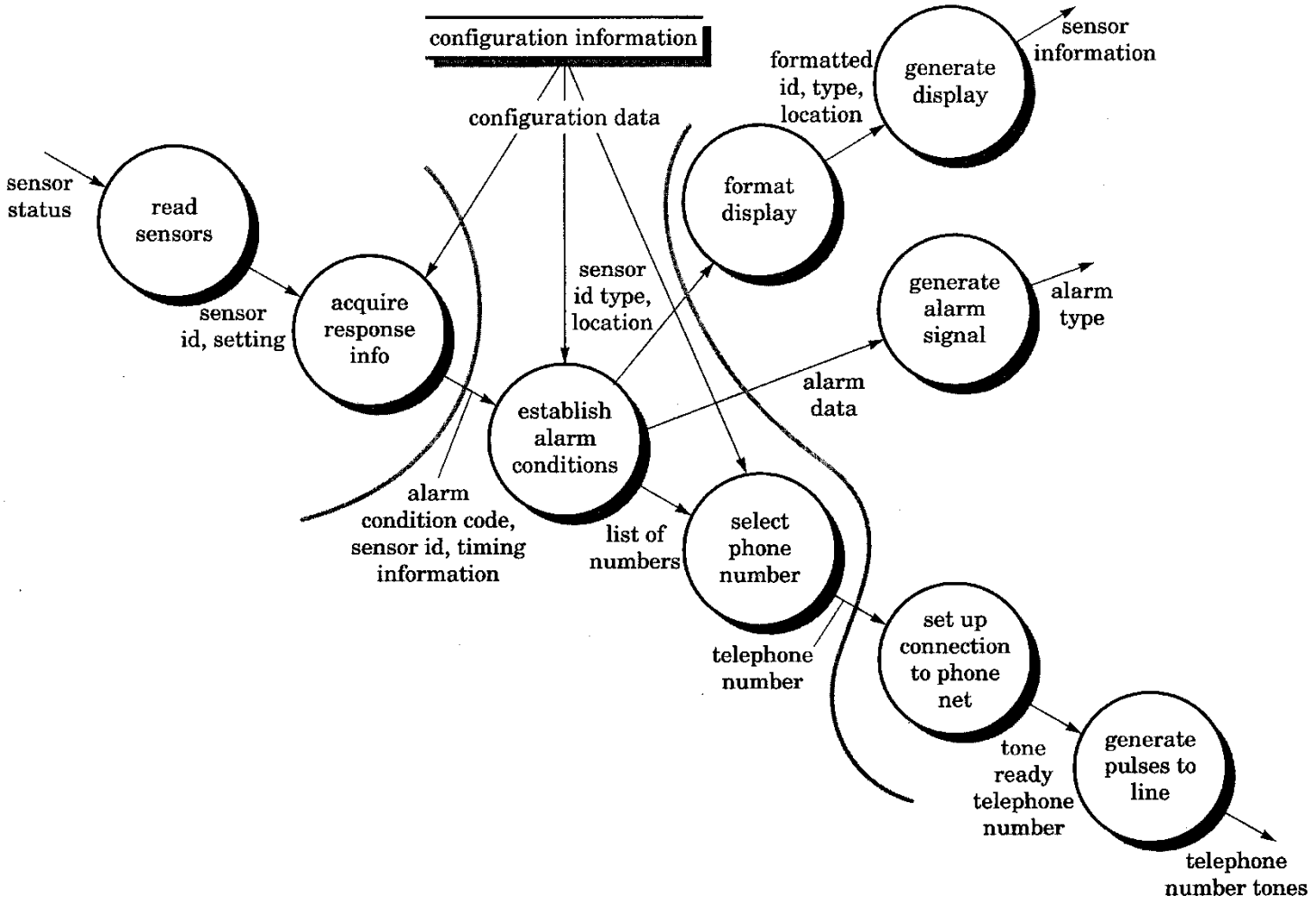


FIGURE 14.6. Level 3 DFD for *monitor sensors* with flow boundaries

TRANSFORM MAPPING (CONT)

- step 4. Isolate the transform center by specifying incoming and outgoing flow boundaries
  - different designers may select slightly differently
  - transform center can contain more than one bubble.
- step 5. Perform “first-level factoring”
  - program structure represent a top-down distribution control.
  - factoring results in a program structure(top-level, middle-level, low-level)
  - number of modules limited to minimum.

FIRST LEVEL FACTORING

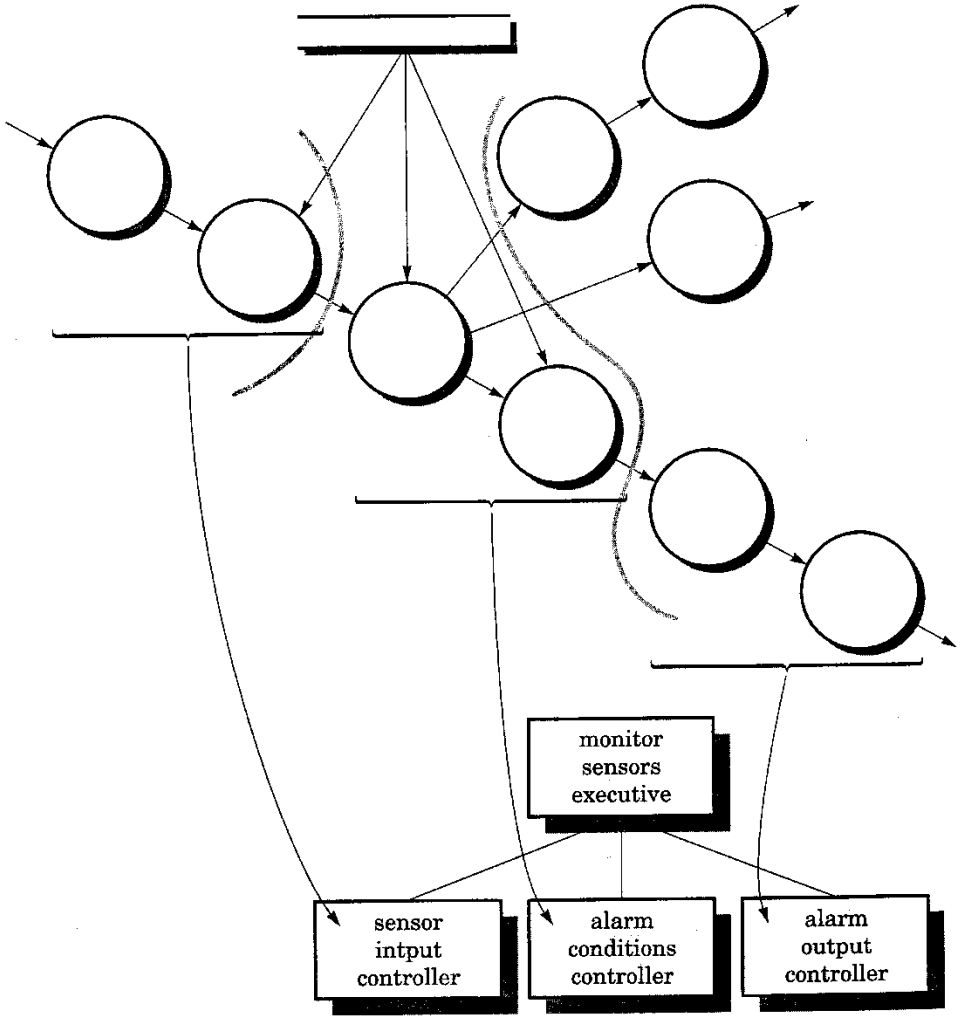


FIGURE 14.7. First-level factoring for *monitor sensors*

TRANSFORM MAPPING (CONT)

- step 6. Perform “second-level factoring”

  mapping individual transforms(bubbles) to appropriate modules.

  factoring accomplished by moving outwards from transform center boundary.
- step 7. Refine the first iteration program structure using design heuristics for improved software quality.

SECOND LEVEL FACTORING

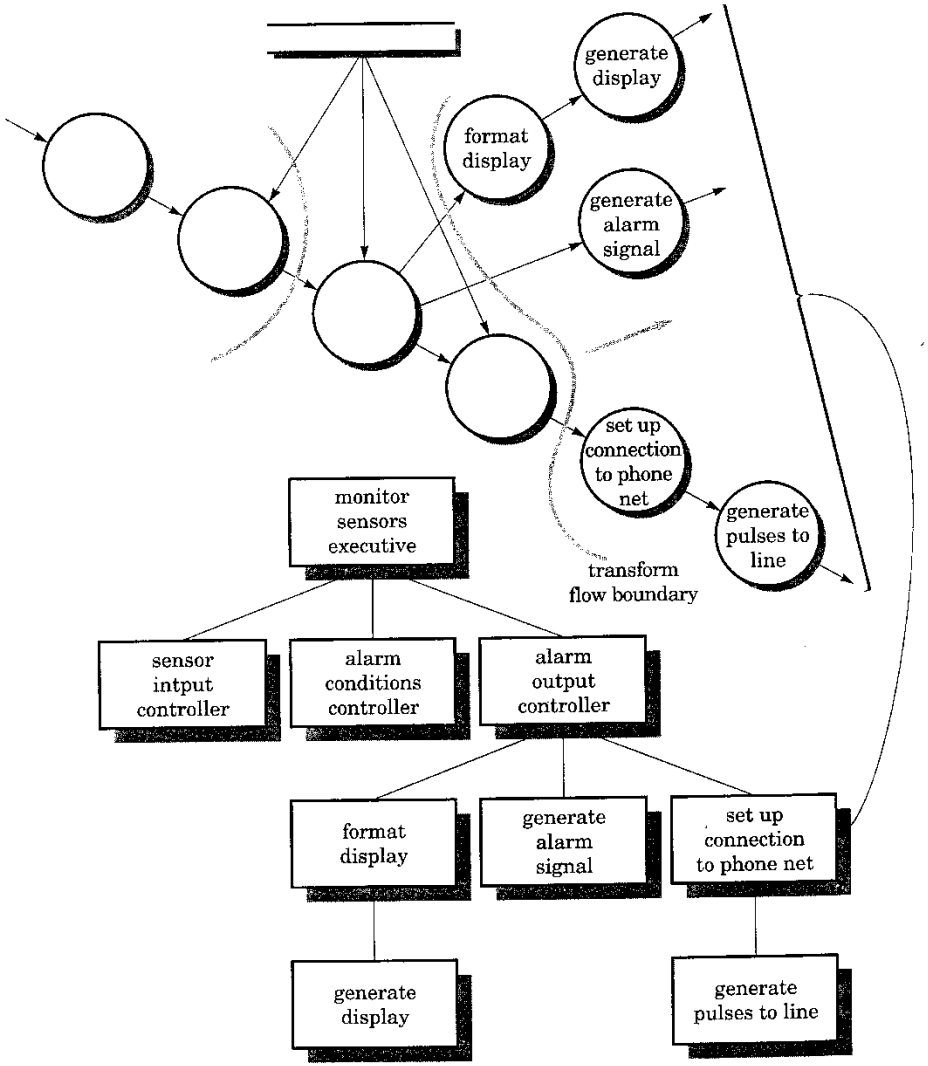


FIGURE 14.8. First level factoring for *monitor sensors*

FIRST-CUT PROGRAM STRUCTURE

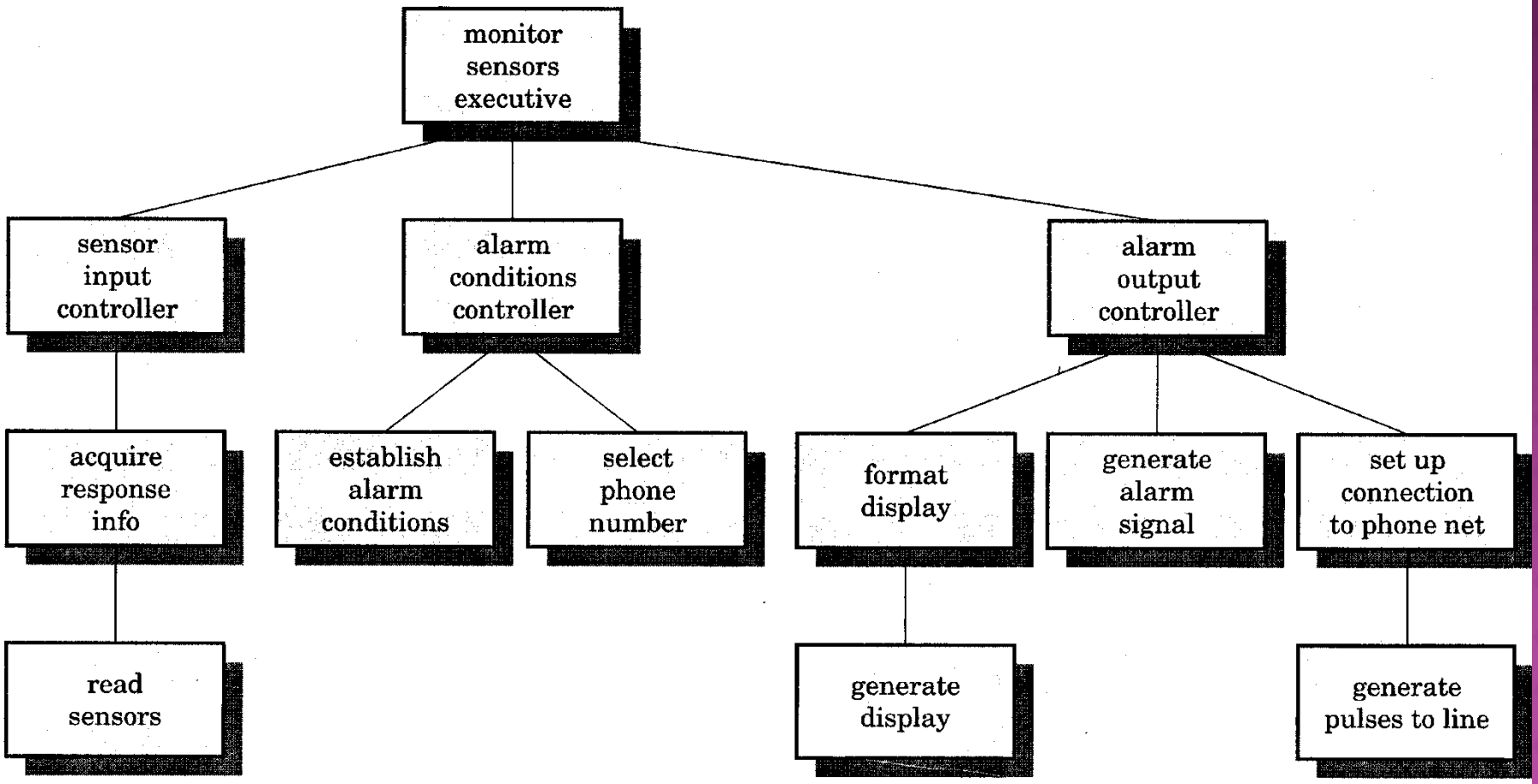


FIGURE 14.9. "First-cut" program structure for *monitor sensors*

REFINED PROGRAM STRUCTURE

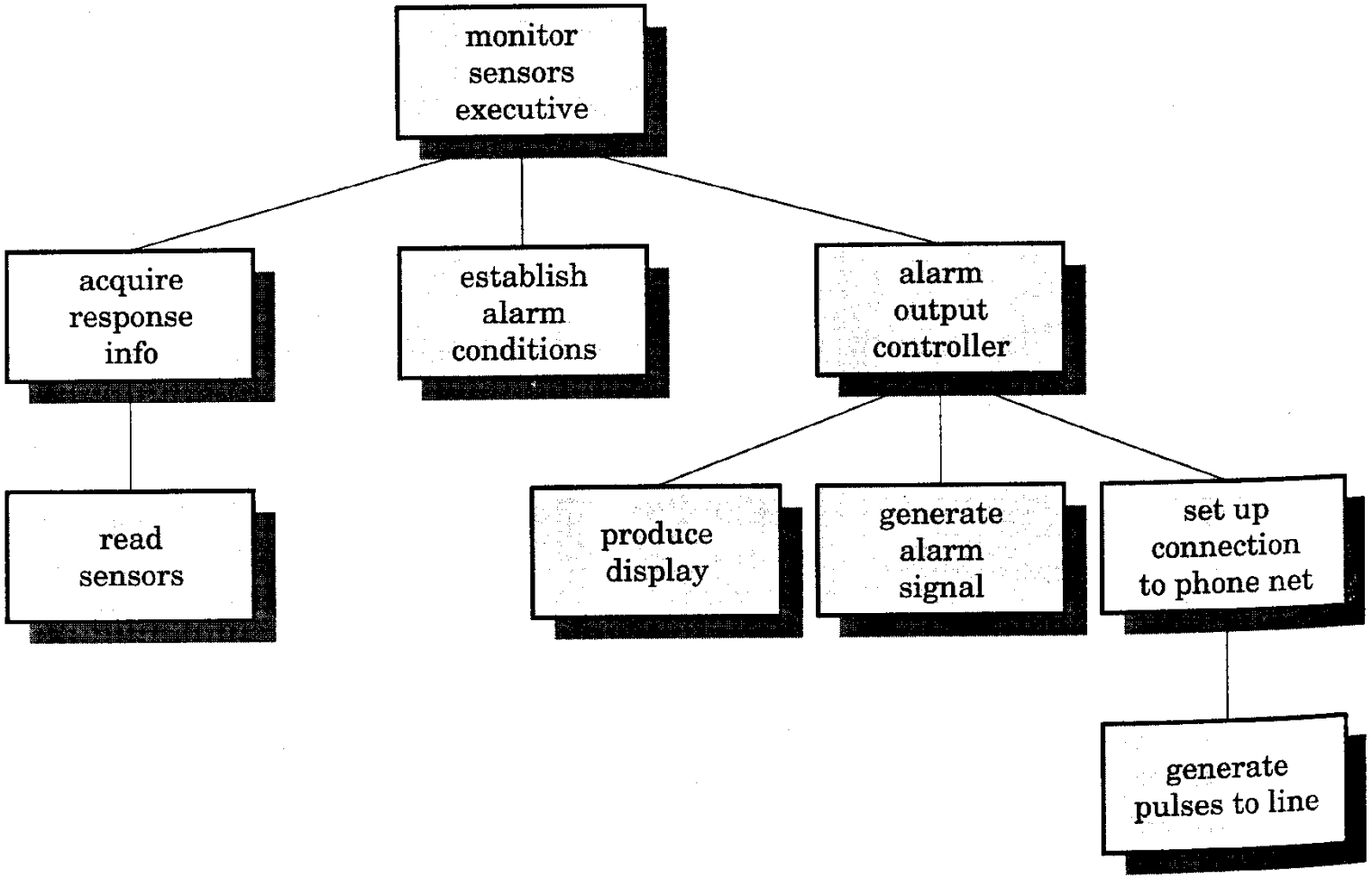


FIGURE 14.10. Refined program structure for *monitor sensors*

TRANSACTION MAPPING

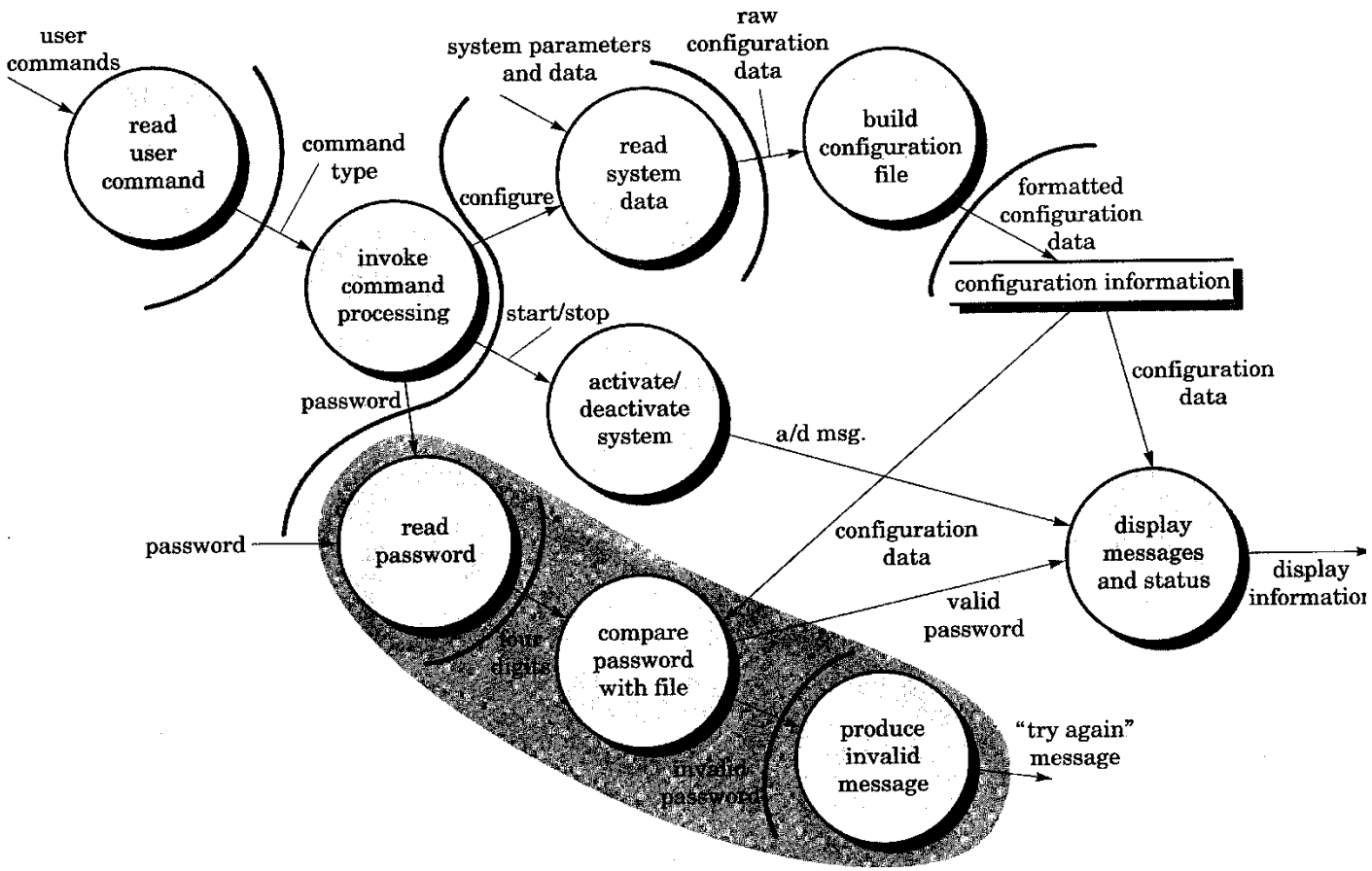


FIGURE 14.11. Level 2 DFD for *user interaction* subsystem with flow boundaries

TRANSACTION MAPPING DESIGN

- Step 1.Review the fundamental system model.
- Step 2.Review and refine DFD for the software
- Step 3.Determine whether the DFD has transform or transaction flow characteristics
- Step 4. Identify the transaction center and flow characteristics along each of the action paths

  isolate incoming path and all action paths

  each action path evaluated for its flow characteristic.

TRANSACTION MAPPING (CONT)

- **step 5. Map the DFD in a program structure amenable to transaction processing**
  - incoming branch

    bubbles along this path map to modules
  - dispatch branch

    dispatcher module controls all subordinate action modules

    each action path mapped to corresponding structure

TRANSACTION MAPPING

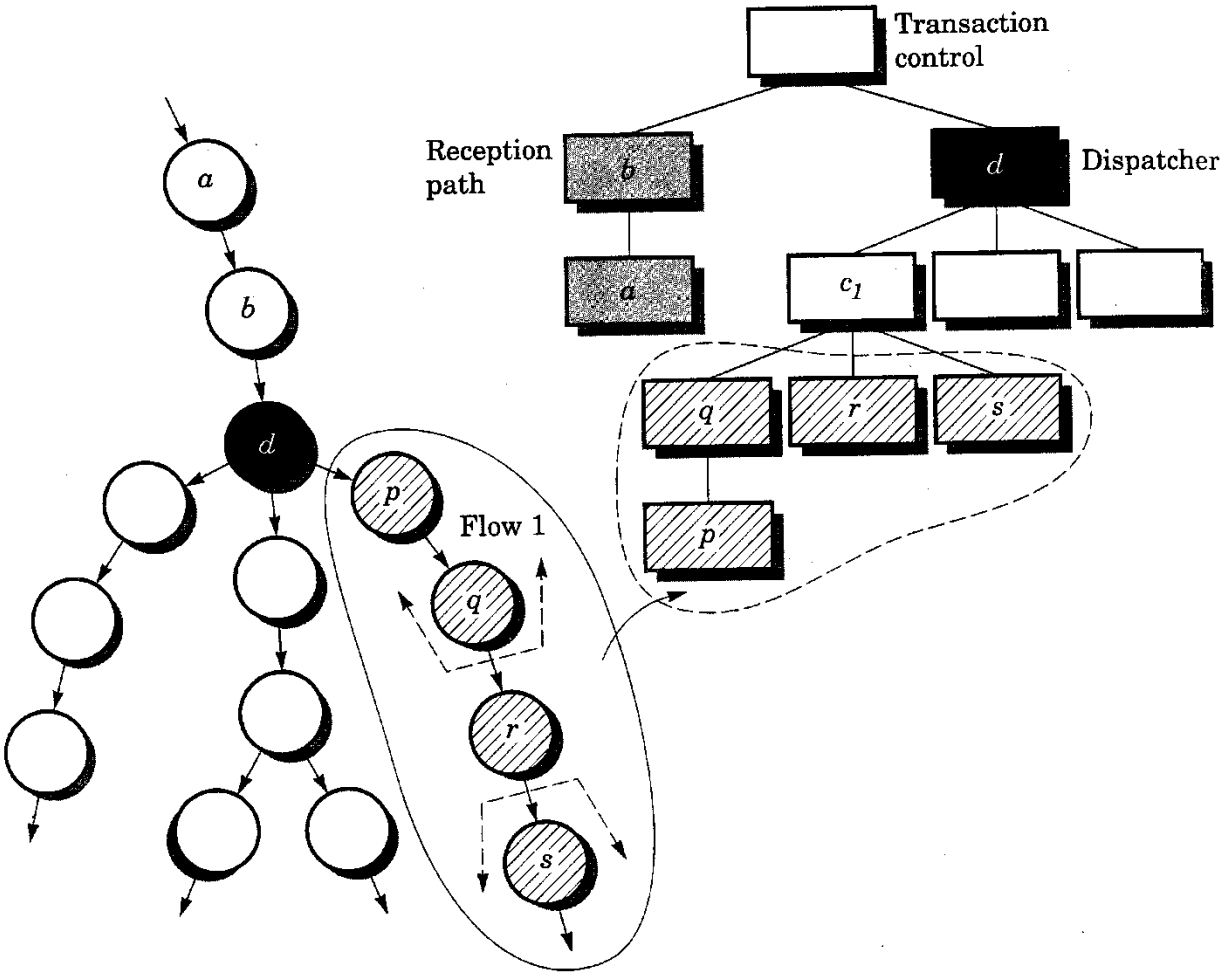


FIGURE 14.12. Transaction mapping

FIRST LEVEL FACTORING

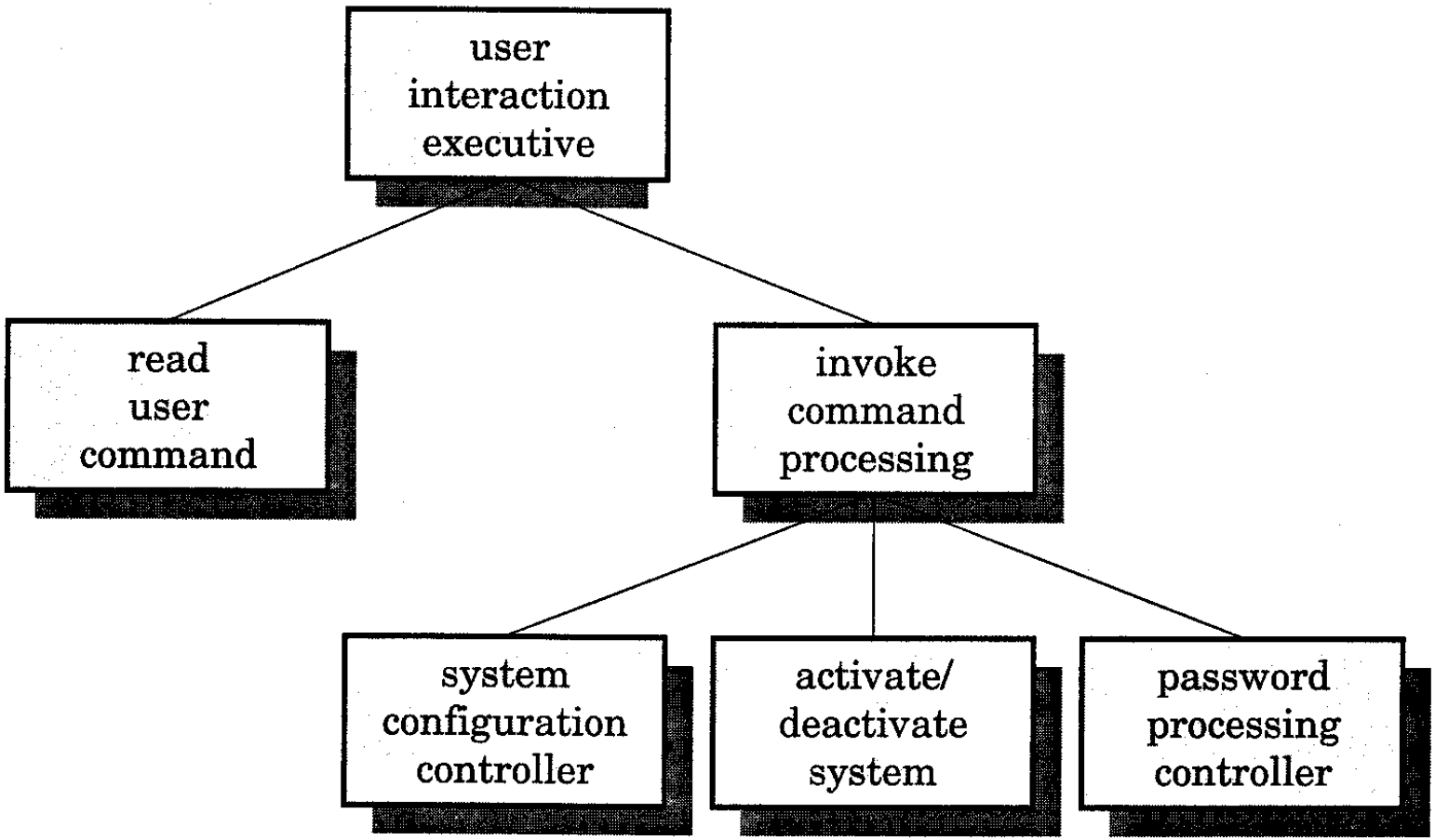


FIGURE 14.13.
First-level factoring for *user interaction* subsystem

FIRST-CUT PROGRAM STRUCTURE

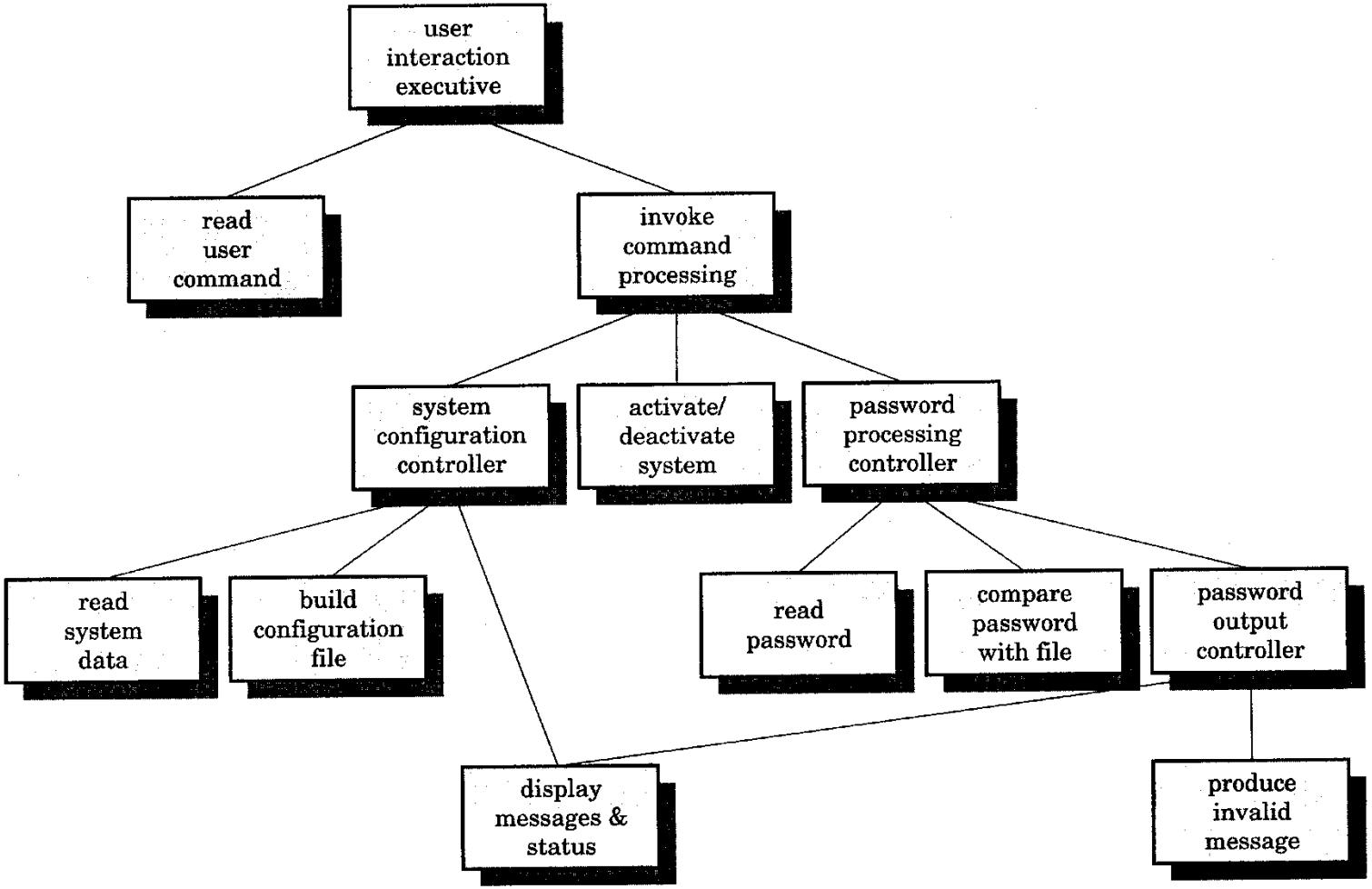


FIGURE 14.14. First-cut program structure for *user interaction* subsystem

TRANSACTION MAPPING (CONT)

- step 6. Factor and refine the transaction structure and the structure of each action path
- step 7. Refine the first iteration program structure using design heuristics for improved software quality

DESIGN POSTPROCESSING

- A processing narrative must be developed for each module
- An interface description is provided for each module
- Local and global data structures are defined
- All design restrictions/limitations are noted
- A design review is conducted
- “Optimization” is considered (if required and justified)