Architecture Design

Ruchita Shah

S/w Architecture

- A representation that enable to

1. Analyze effectiveness of design in meeting its requirements
2. Consider archi alternatives
3. Reducing risk associated with s/w development

Why is S/W Architecture Important

- Enable for communication b/w all interested in development
- Highlights design decisions having impact on all SE works & success of system
- Constitute small, graspable model of how system is structured & components work together
- S/w archi includes data design & architectural design

Data Design

- Creates a model of data & info at a high level of abstraction
- Then refined to more implementation specific representation
- Have influence on archi & s/w
- At component level design of DS & associated algo affects quality
- At application level data model translated into database
- At business level collection of info stored in database & reorganized into data warehouse, data mining

Data Design

- Data Modeling, Data structure, Databases & Data Warehouse :
- Data Modeling : data objects of requirements analysis modeled using ER diagram & data dictionary
- Data Structure : Design translate data model into DS at component level

Data Design

- Data base : database archi at application level
  - previously data archi was DS at program level & database at application level
  - Today moderate size business have dozens of databases
  - Extract useful info from this data
  - Developed data mining techniques
  - Navigate through databases to extract appropriate info
  - data mining difficult because of multiple databases, different structure
  - alternate solution is data warehouse

Data Design

- Data Warehouse : adds additional layer to data archi
  - is a separate env not directly integrated with regular applications
  - encompass all data of business
  - a large independent database
- characteristics of data warehouse are :

1. Subject orientation : organized by major business subjects, excludes data not necessary for data mining

Data Design

2. Integration : regardless of source, data have consistent naming convention, units, measures, encoding structure, attributes
3. Time variancy : for transaction-oriented appli – data accurate at the access & for short time span, for data warehouse, data accessed at a specific time, time span is 5-10 years
4. Nonvolatibility : business application data undergo continuous change, warehouse data do not change

Data Design at Component Level

- Focus on representation of DS directly accessed by programs, a set of principles to specify & design data

1. The systematic analysis principles applied to function & behavior should also be applied to data : Much effort in deriving, specifying functional requirement & preliminary design, data flow & content developed & reviewed, data objects identified, alternates considered, impact on s/w design evaluated ex linked list
2. All data structure & operation to be performed on each should be identified : operation to be performed must taken into account

Data Design at Component Level

3. Data dictionary should be established & used to define both data & program structure : data dictionary represents relationship b/w data objects & constraints on elements
4. Low-level data design should be deferred until late in the design process : overall data organization during requirement analysis, refined during preliminary design & specified in detail during component level design – top-down approach
5. The representation of data structure should be known only to those modules that must make direct use of the data contained within the structure : concept of info hiding & coupling

Data Design at Component Level

6. A library of useful data structures & the operations that may be applied to them should be developed : reusability of DS
7. A s/w design & programming language should support the specification & realization of abstract data types : implementation of DS difficult if language chosen for development do not support

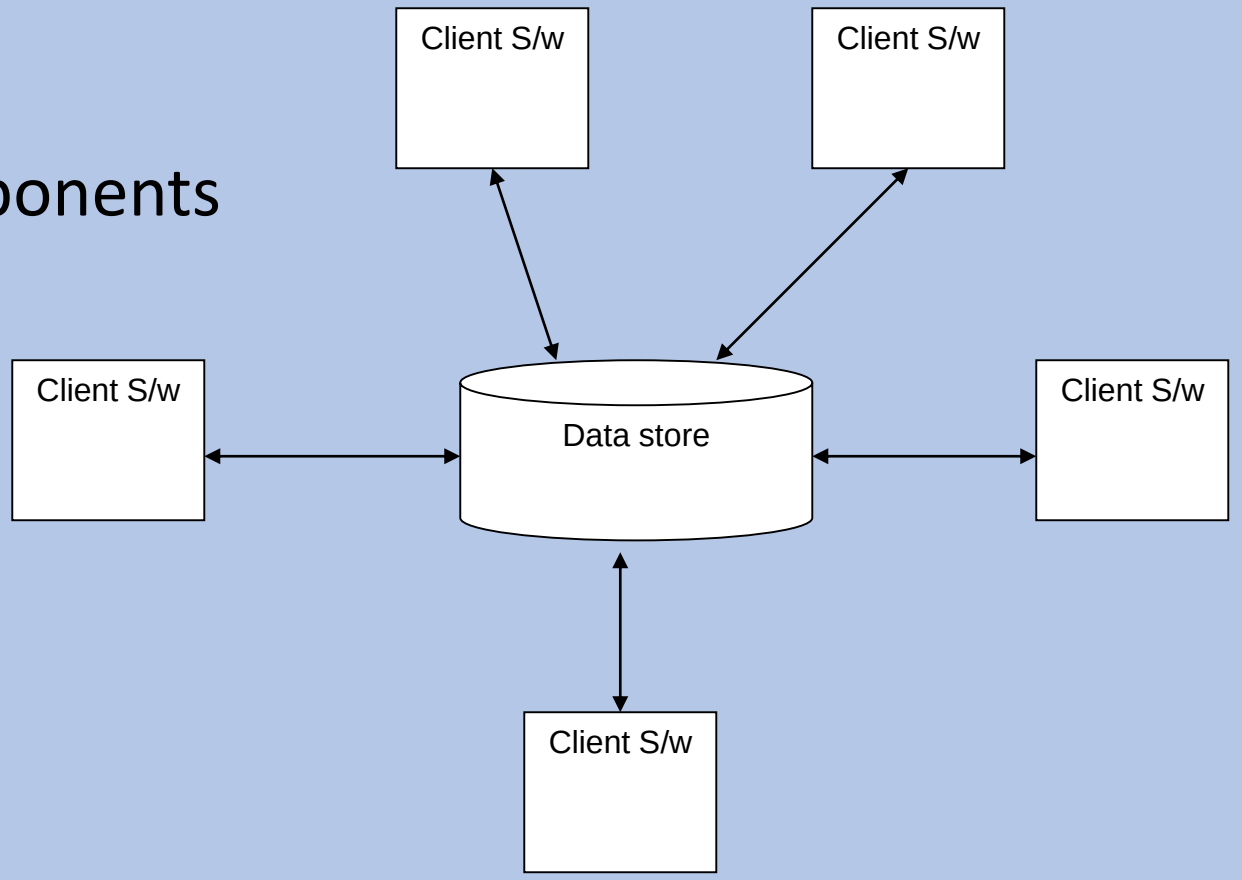
ARCHITECTURAL STYLES

- S/w exhibits one of many archi styles, each has

1. Set of components that perform function
2. Set of connectors to communication, coordinate & cooperation of components
3. Constraints defining how components integrated to form a system
4. Semantic properties that describe overall properties of system

Data Centred Architecture

- Data store at the center
- Accessed & modified by components
- Client-server technology

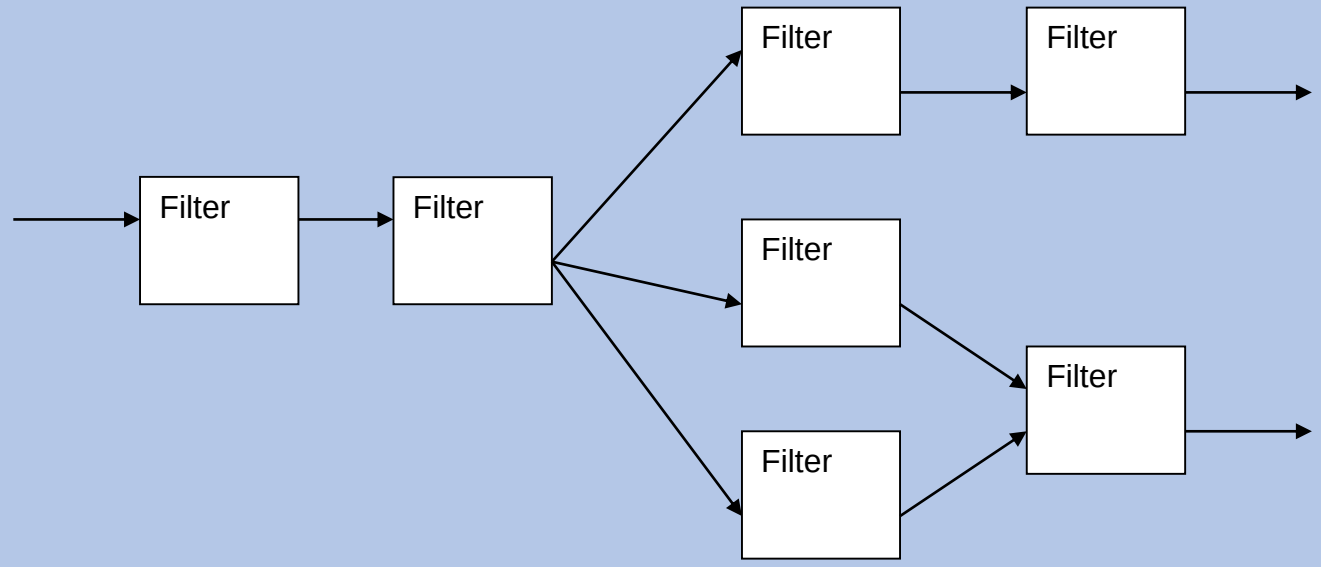


Data Centred Architecture

- Client s/w access central repository
- Sometimes passive i.e. client access data independent of changes
- A notification is sent to client when data of interest change
- Promote integrality – existing component can be changed, new component added without affecting other clients

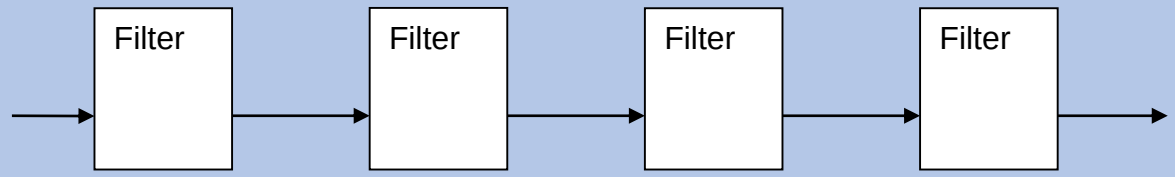
Data-Flow Architecture

- When i/p data transformed thro’ series of processing into o/p
- Pipe & filter pattern
- pipes transmit data
- Filters work independently
- Expect i/p of certain form
- Produce specific o/p



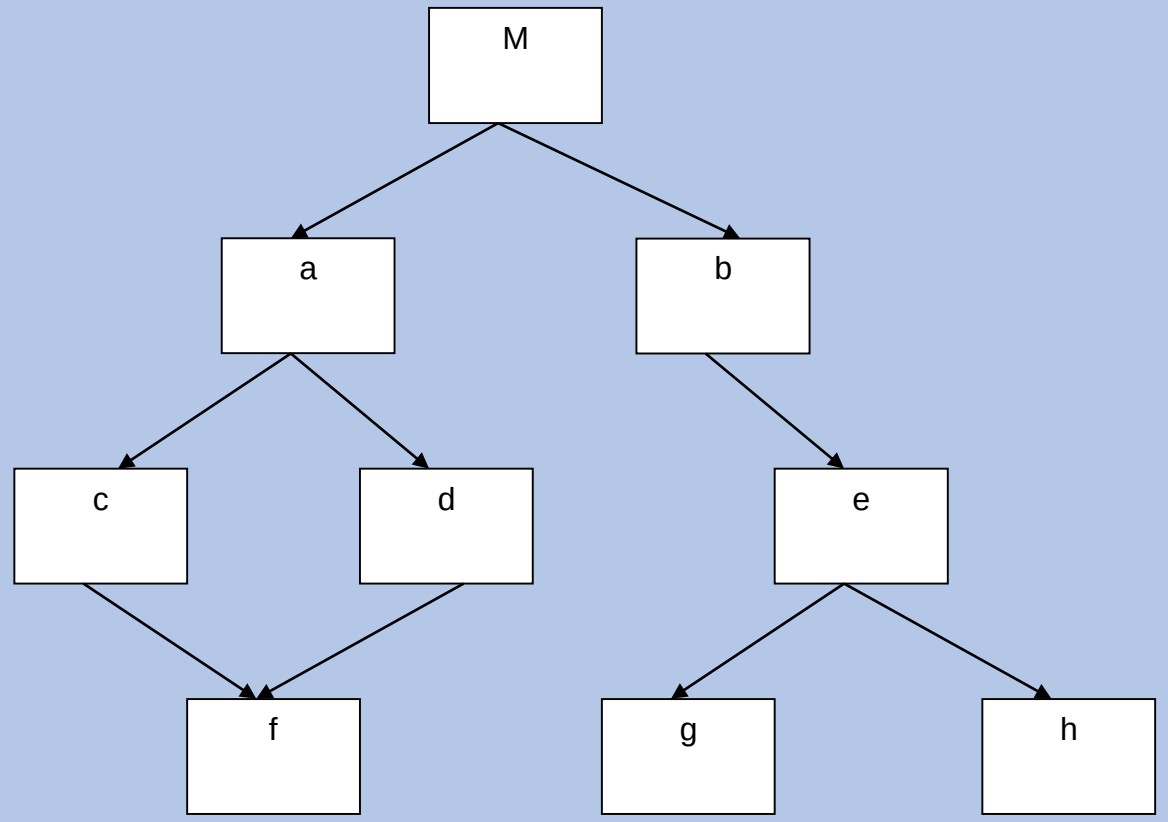
Data-Flow Architecture

- If single line of transform : batch sequential
- Accepts batch data & apply series of processing



Call & Return Architecture

- Main Program/Subprogram Architecture : decomposed function in control hierarchy, main program invokes no of subprogram which again invoke other programs



Call & Return Architecture

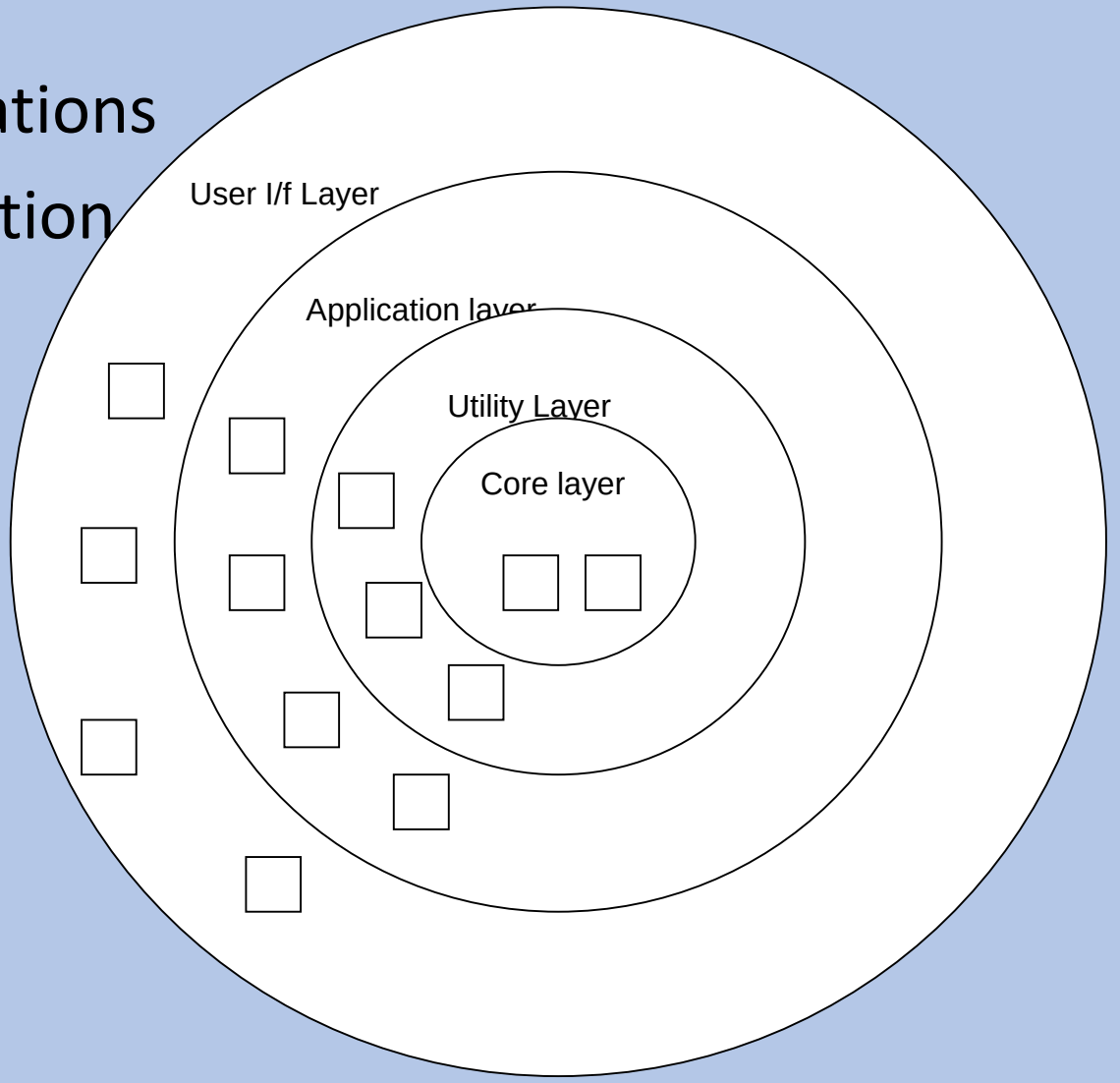
- Remote Procedure Call Architecture : components distributed across multiple computers on n/w

Object-Oriented Architecture

- Encapsulation of data & operation on them
- Communication & coordination via message passing

Layered Architecture

- Number of diff layers performing operations
- progressively closer to machine instruction
- outer layer user i/f operations
- inner layer perform OS i/f
- intermediate layer provide utility & application s/w functions



Refinement of Architecture

- Once requirements identifies characteristics & constraint of the system, archi style or combination best fit is chosen
- No of alternatives for archi design
- A set of design criteria to assess designs
- Control : how control managed, control hierarchy, role of components in it, control topology, synchronization etc must be looked
- Data : data communication b/w components, flow of data – continuous or sporadic, mode of data transfer, how functions interact with data etc must be looked

Analysing Alternate Architectural Design

- An Architecture Trade-off Analysis Method

1. Collect scenario : set of use-cases developed to represent the system from user's view point
2. Elicit requirements, constraints & environment description : part of SE, make certain that all customer, user concerns addressed
3. Describe the archi style/pattern that have been chosen to address the scenario & requirements : should describe archi views such as

   Module view : to analyze components & info hiding achieved

   Process view : for system performance

   Data flow view : degree to which archi meets functional requirements

Analysing Alternate Architectural Design

4. Evaluate quality attributes by considering each attribute in isolation : include reliability, performance, security, maintainability, flexibility, testability, portability, reusability & interoperability
5. Identify the sensitivity of quality attributes to various archi attributes for a specific archi style : by making a small change in archi & determine how sensitive quality attribute to the change, attributes significantly affected by variation are sensitivity points
6. Critique candidate archi using sensitivity analysis : identify elements to which multiple attributes are sensitive

Mapping Requirements into a s/w Architecture

- S/w requirements mapped into design model
- Mapping technique allows to derive complex archi from DFD
- Technique known as structured design
- Stress on modularity, top-down design & structured programming
- Provides convenient transition from DFD to s/w archi

Mapping Requirements into a s/w Architecture

- The transition is six-step process

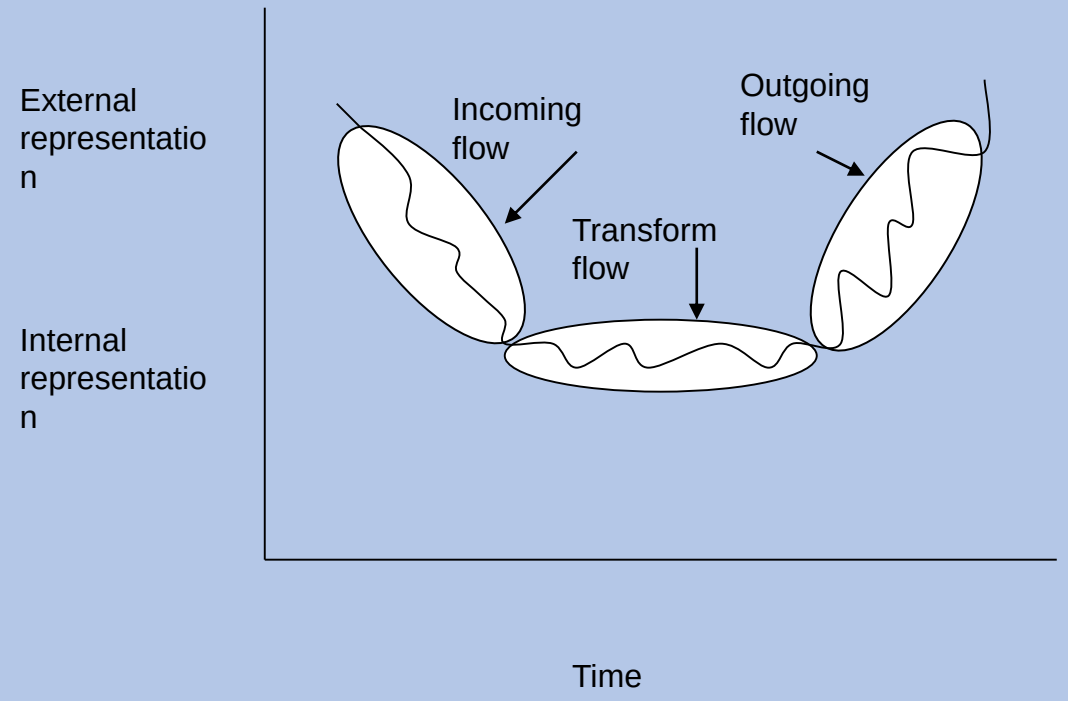
1. The type of info flow is established
2. flow boundaries are indicated
3. The DFD is mapped into program structure
4. Control hierarchy is defined
5. Resultant structure refined using design measures & heuristics
6. Archi refined & elaborated

Transform Flow

- There are two types of info flow
- Transform Flow : info enters & exits s/w in external world form ex. Data typed on keyboard, pictures, tone on telephone line etc
- Entered data are converted into internal form for processing
- Paths that convert external data to internal data is incoming flow
- at a kernel transition occurs and incoming data pass thro transform center
- Than move towards outgoing flow
- Overall flow of data occurs in sequential manner

Transform Flow

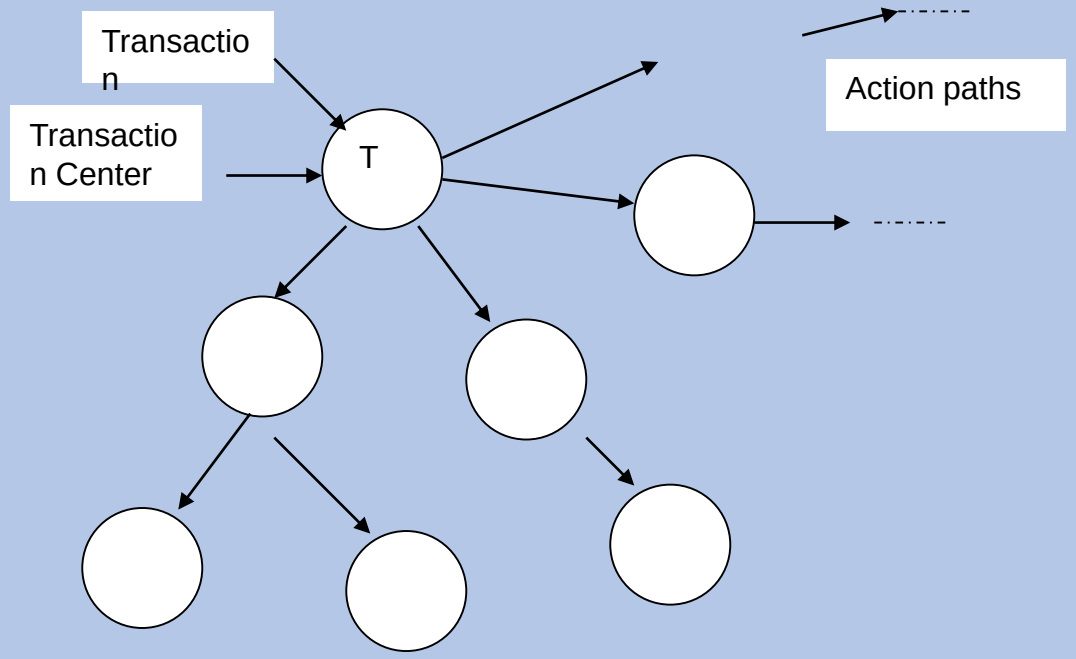
- Transform Flow :



Transaction Flow

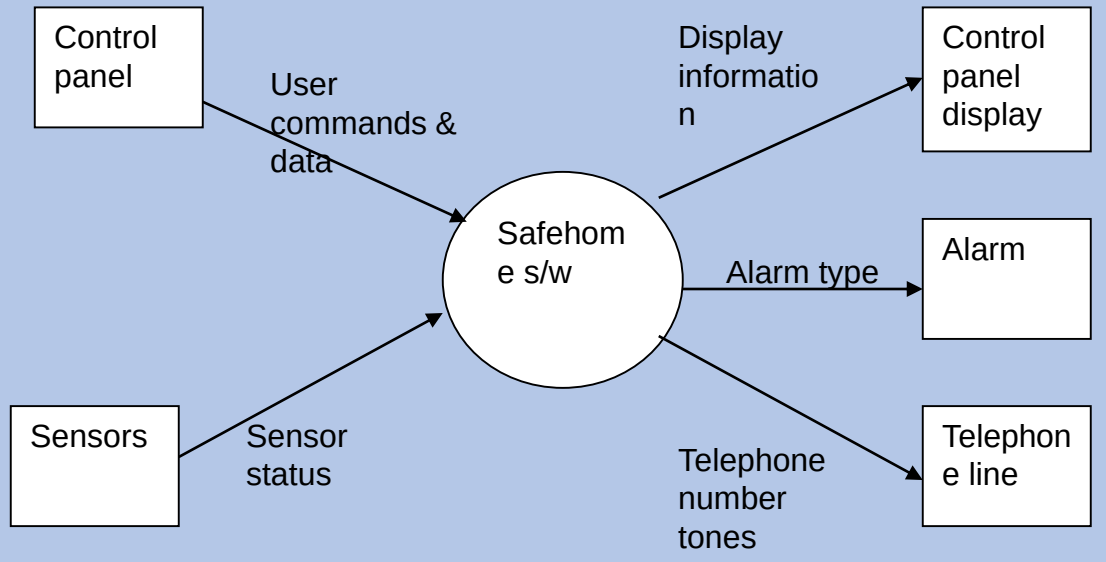
- Transaction Flow : transaction triggers data flow along one of many paths
- Data comes along incoming path that converts external world info into transaction
- Based on its value flow along one of many action paths is initiated the hub is called transaction center.

Transaction Flow



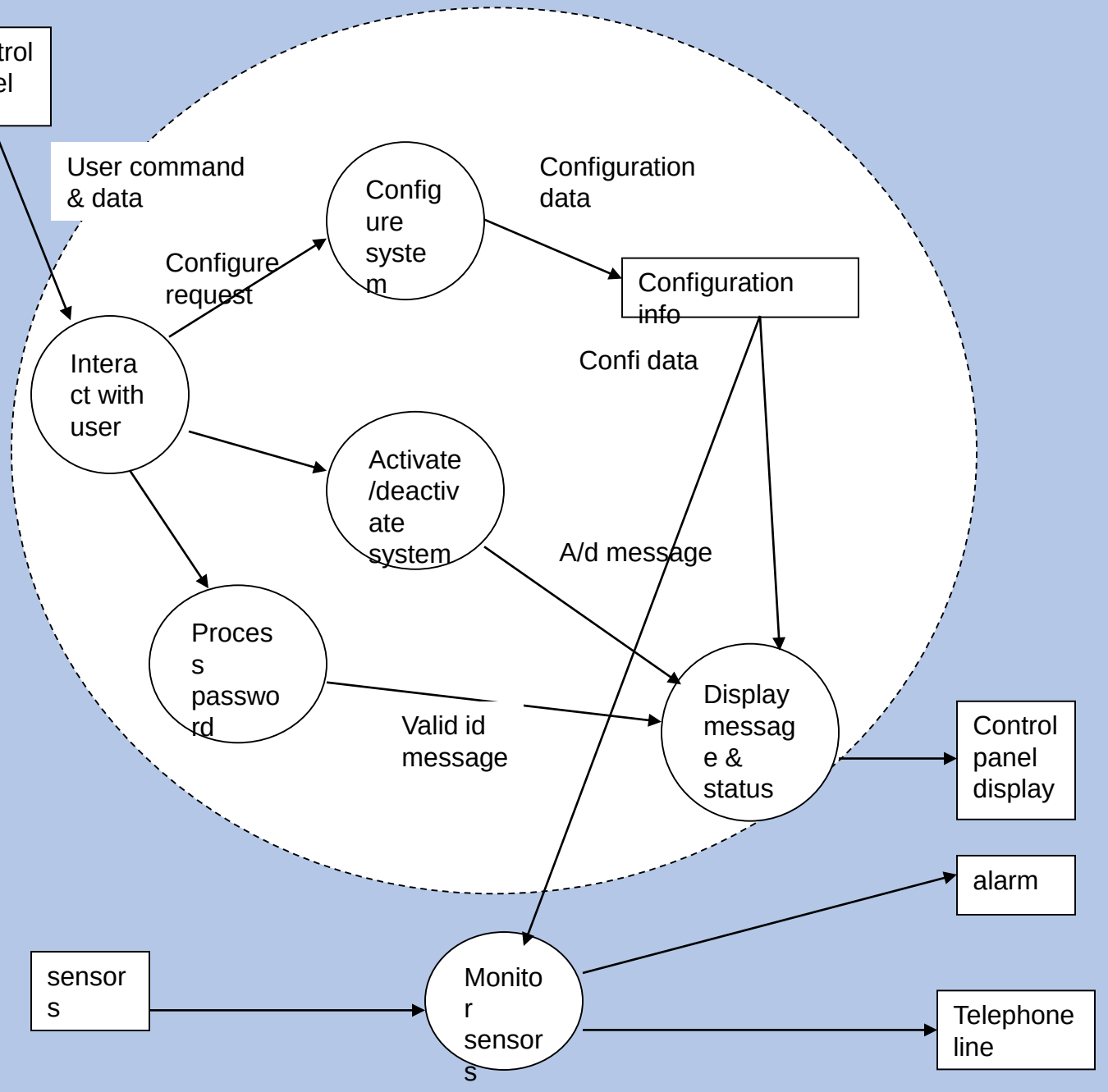
Transform Mapping

- s/w monitors sensors interacts with user through typed commands –
  0 level DFD as above



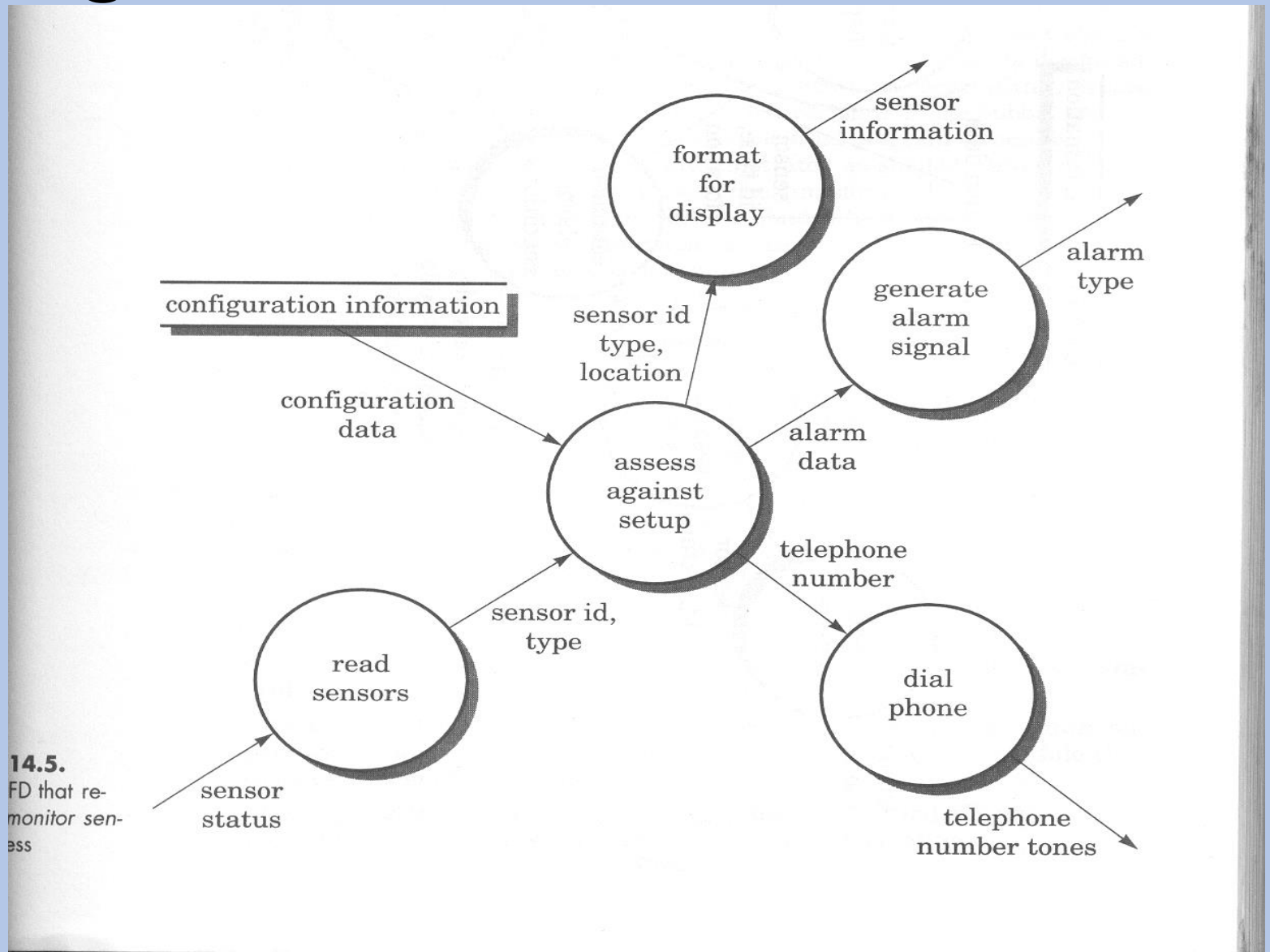
Transaction Flow

1. Review the fundamental system model :



Transform Mapping

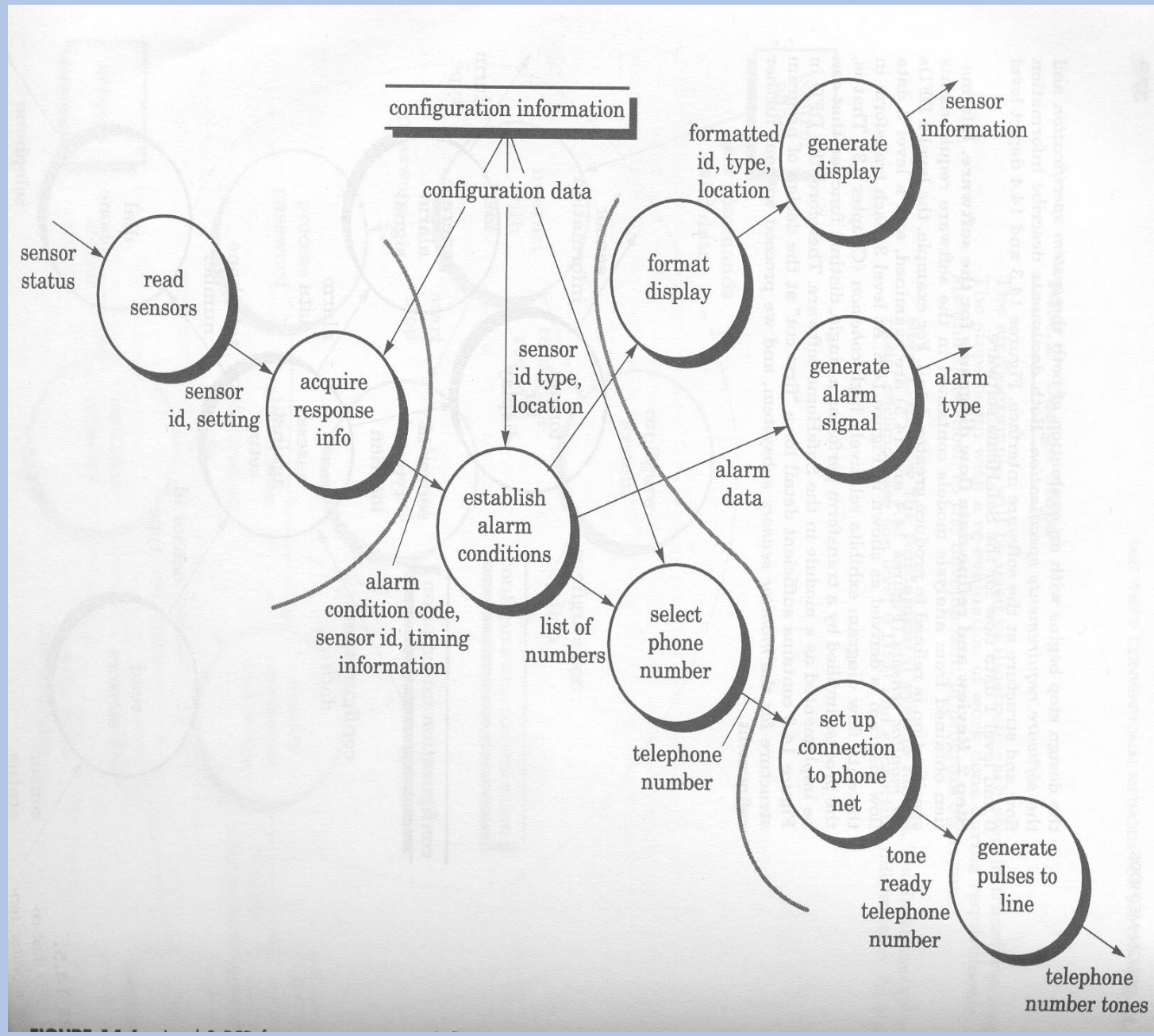
2. Review & refine data flow diagrams for the s/w : Level 2 DFD that refines the monitor sensors process



14.5.
FD that re-
monitor sen-
ess

Transform Mapping

Level 3 monitor sensors with flow boundaries



Transform Mapping

3. Determine whether DFD has transform or transaction flow characteristics : data enters on one incoming path & exit along 3 outgoing path – no distinct transaction center – assumed transform characteristic

Transform Mapping

4. Isolate the transform center by specifying incoming & outgoing flow boundaries : incoming flow is path where info converted from external to internal form & outgoing flow where internal to external form,
   - boundries according to interpretation
   - different flow boundaries – ex curved boundaries in above figure
   - transform bubbles lie within the boundaries
   - emphasis on selecting reasonable boundaries

Transform Mapping

5. Perform first-level factoring : top-down distribution of control

- factoring divides program structure in three
  - top level modules perform decision making
  - middle level performs some control & moderate work
  - low level performs most i/p, computation & o/p
- in transform mapping, DFD is mapped to specific structure that provide control for incoming, transform & outgoing processes

Transform Mapping

- main controller at top, serve to coordinate, subordinate control functions are:
  - Incoming info processor controller – sensor i/p controller, coordinates receipt of incoming data
  - Transform flow controller – alarm conditions controller, supervise operations on data
  - Outgoing info processing controller – alarm o/p controller, coordinates o/p info
- No of modules at first level limited to minimum

Transform Mapping

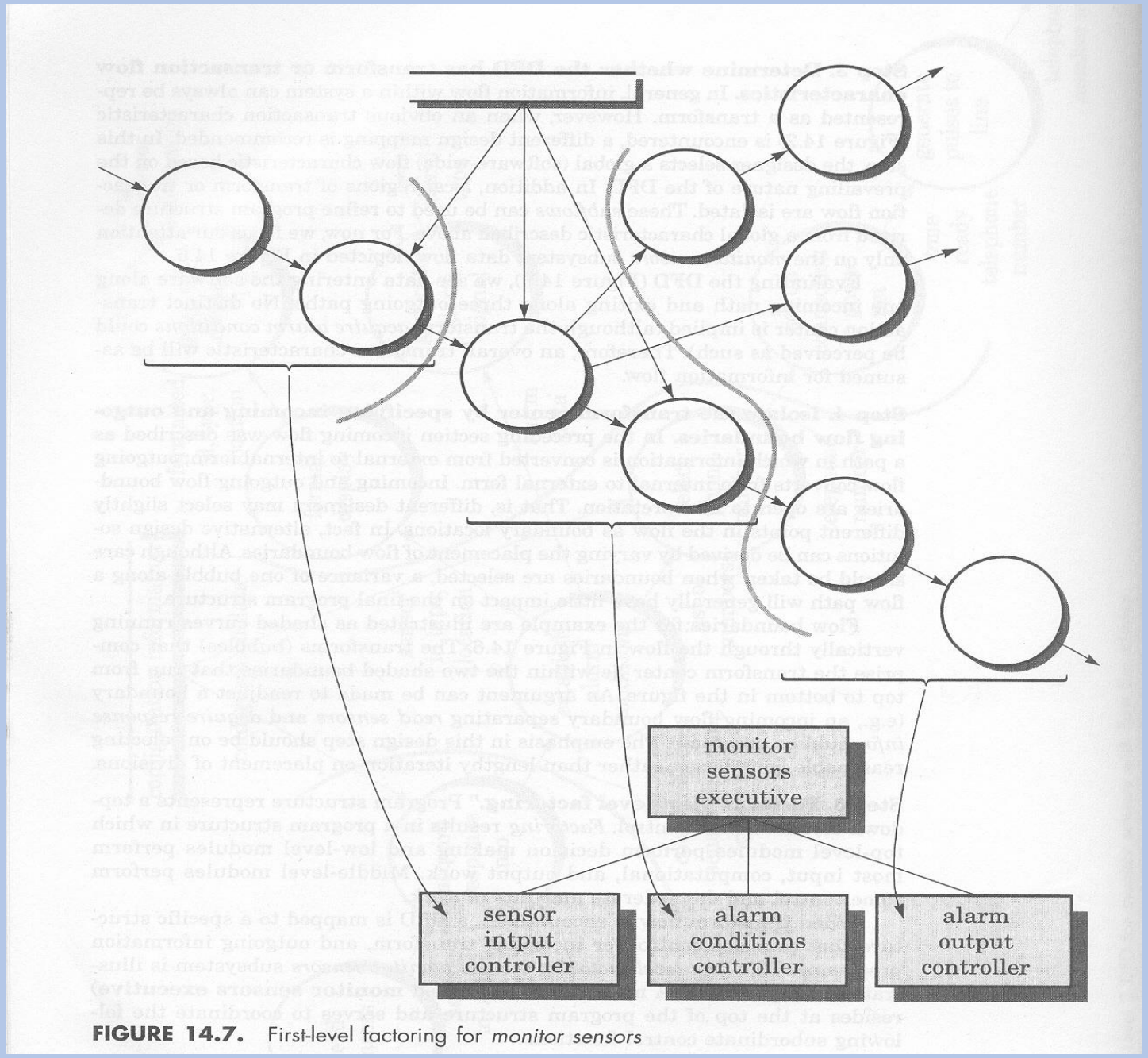


FIGURE 14.7. First-level factoring for *monitor sensors*

Transform Mapping

6. Perform second level factoring : map individual transforms into appropriate module

   begin at transform boundary move outward on incoming & outgoing paths

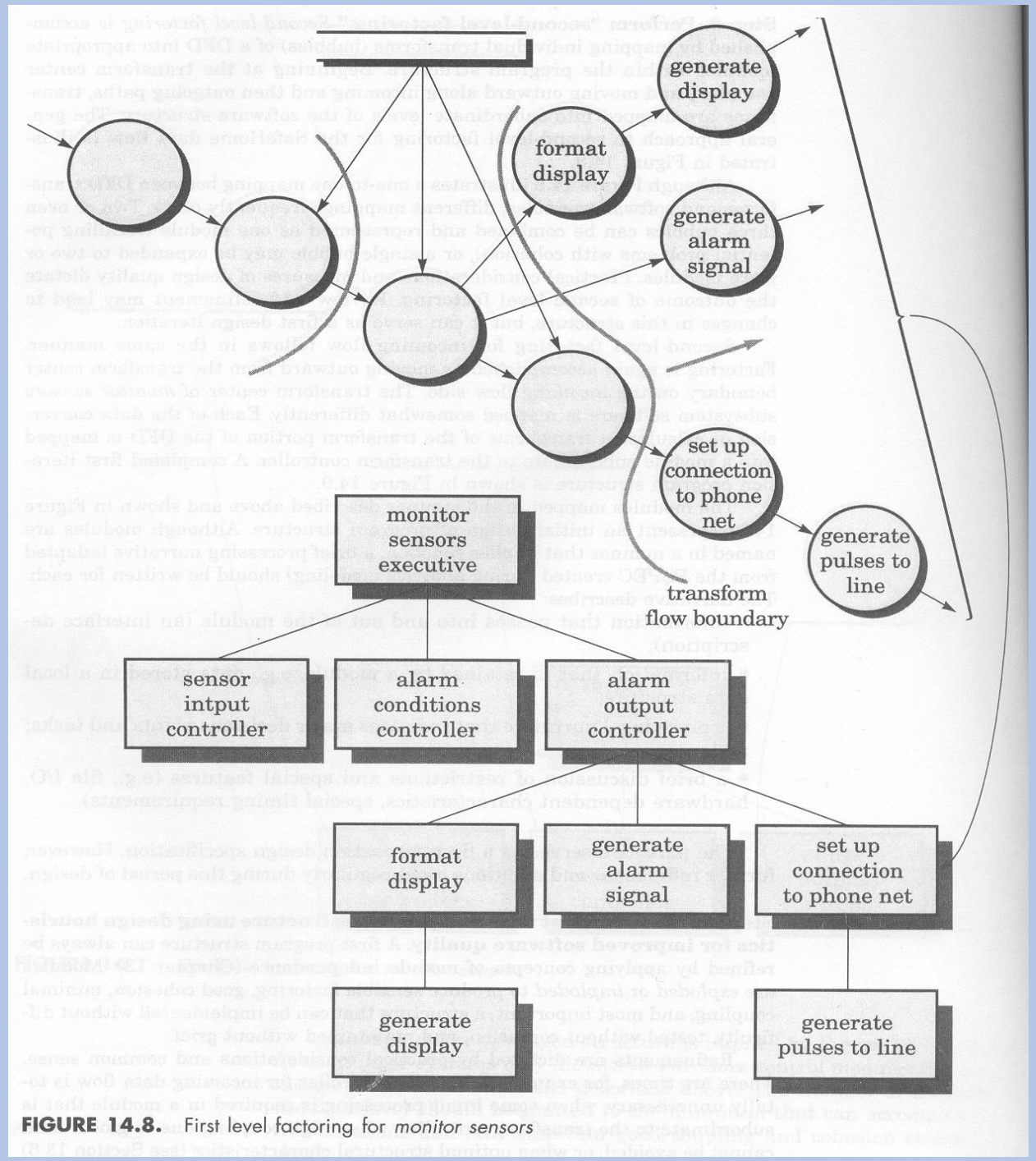


FIGURE 14.8. First level factoring for *monitor sensors*

Transform Mapping

6. Perform second level factoring : map individual transforms into appropriate module

   begin at transform boundary move outward on incoming & outgoing paths

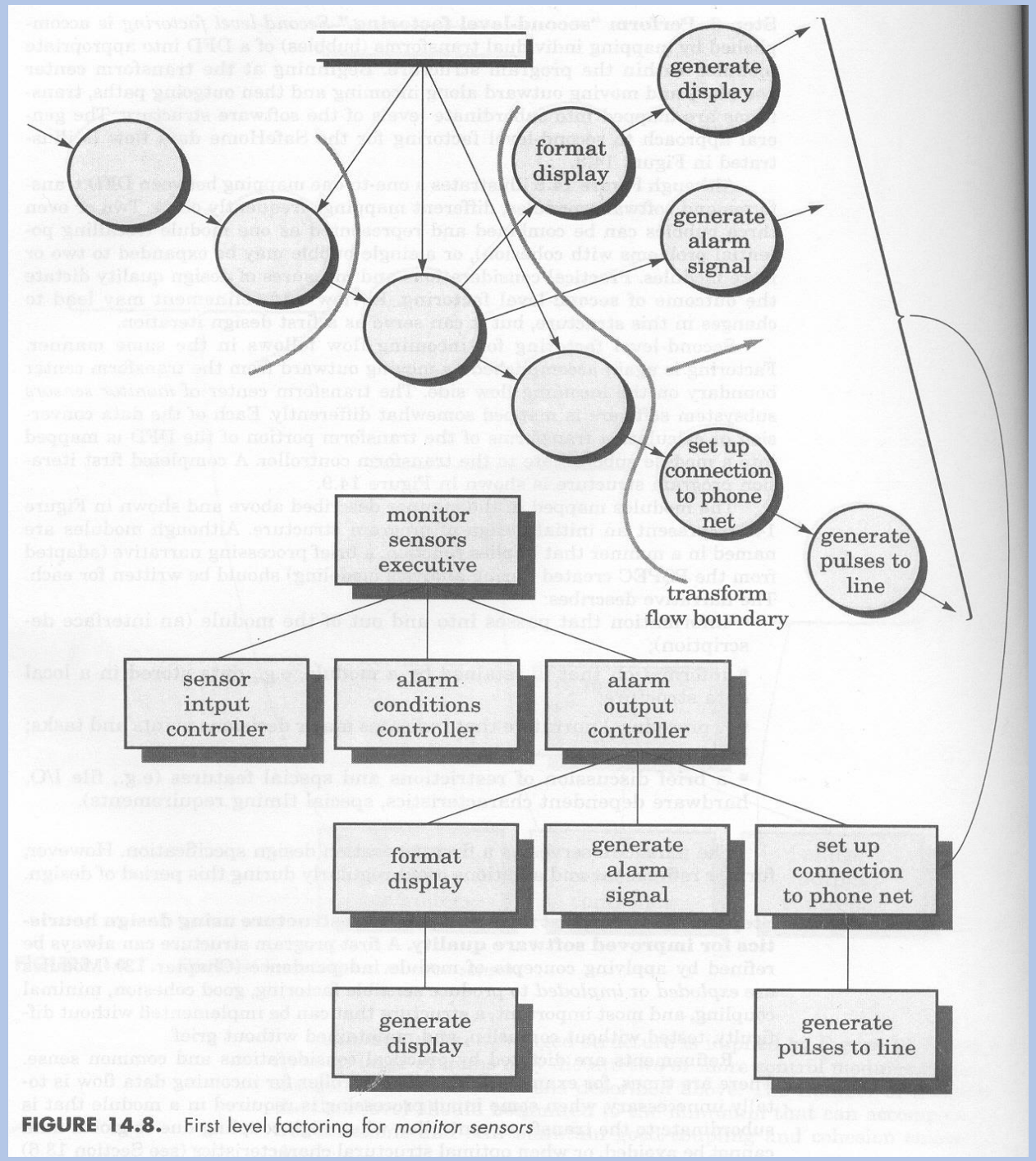


FIGURE 14.8. First level factoring for *monitor sensors*

Transform Mapping

6. Perform second level factoring : map individual transforms into appropriate module

   begin at transform boundary move outward on incoming & outgoing paths

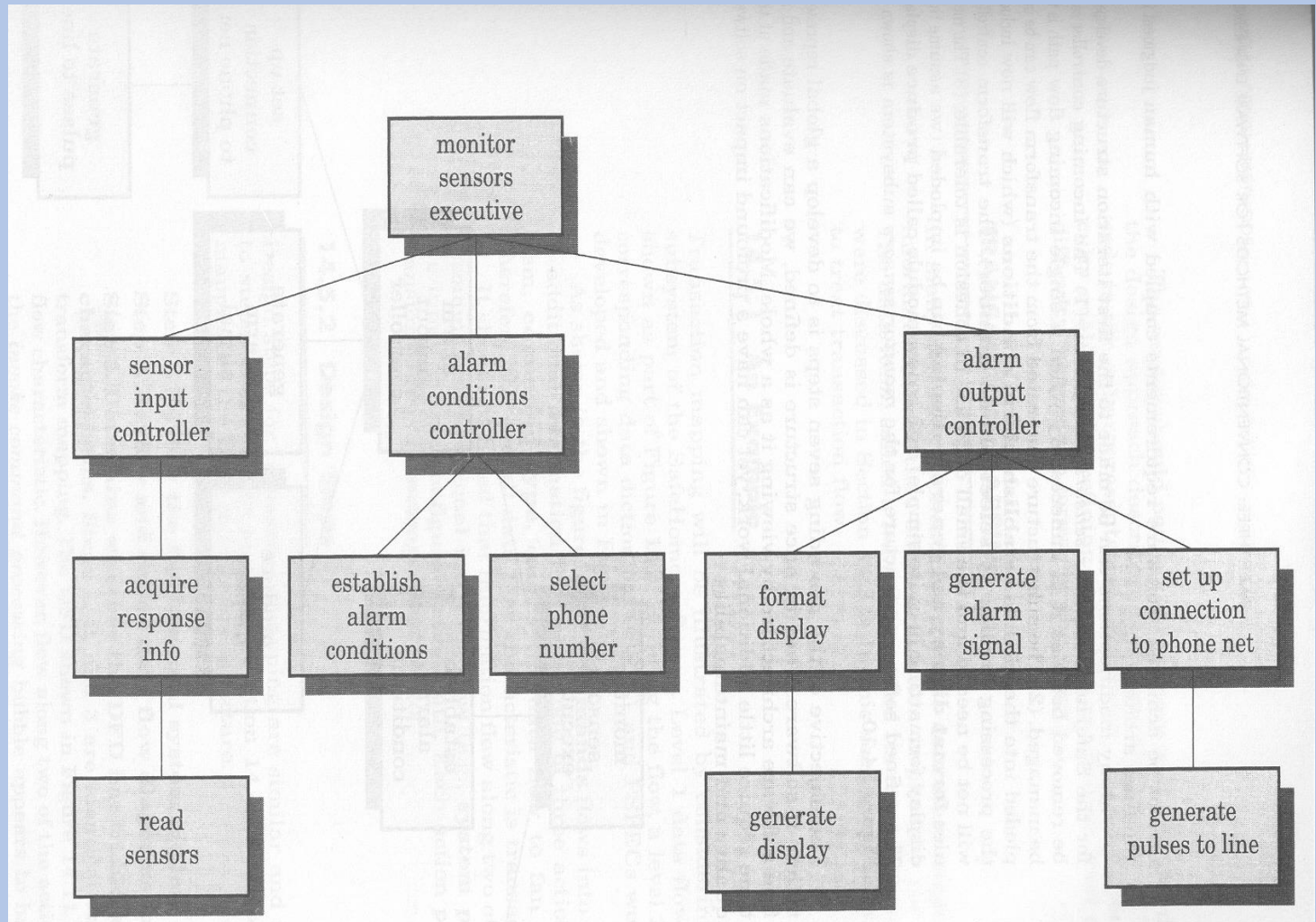


FIGURE 14.9. "First-cut" program structure for *monitor sensors*

Transform Mapping

7. Refine the first iteration program structure using design heuristics for improved s/w quality :
   - apply module independence
   - explode or implode modules for sensible factoring, good cohesion, minimum coupling
   - for a structure that can be implemented without difficulty, testable & maintainable
   - in our example incoming controller is removed, it was unnecessary
   - final structure will be

Transform Mapping

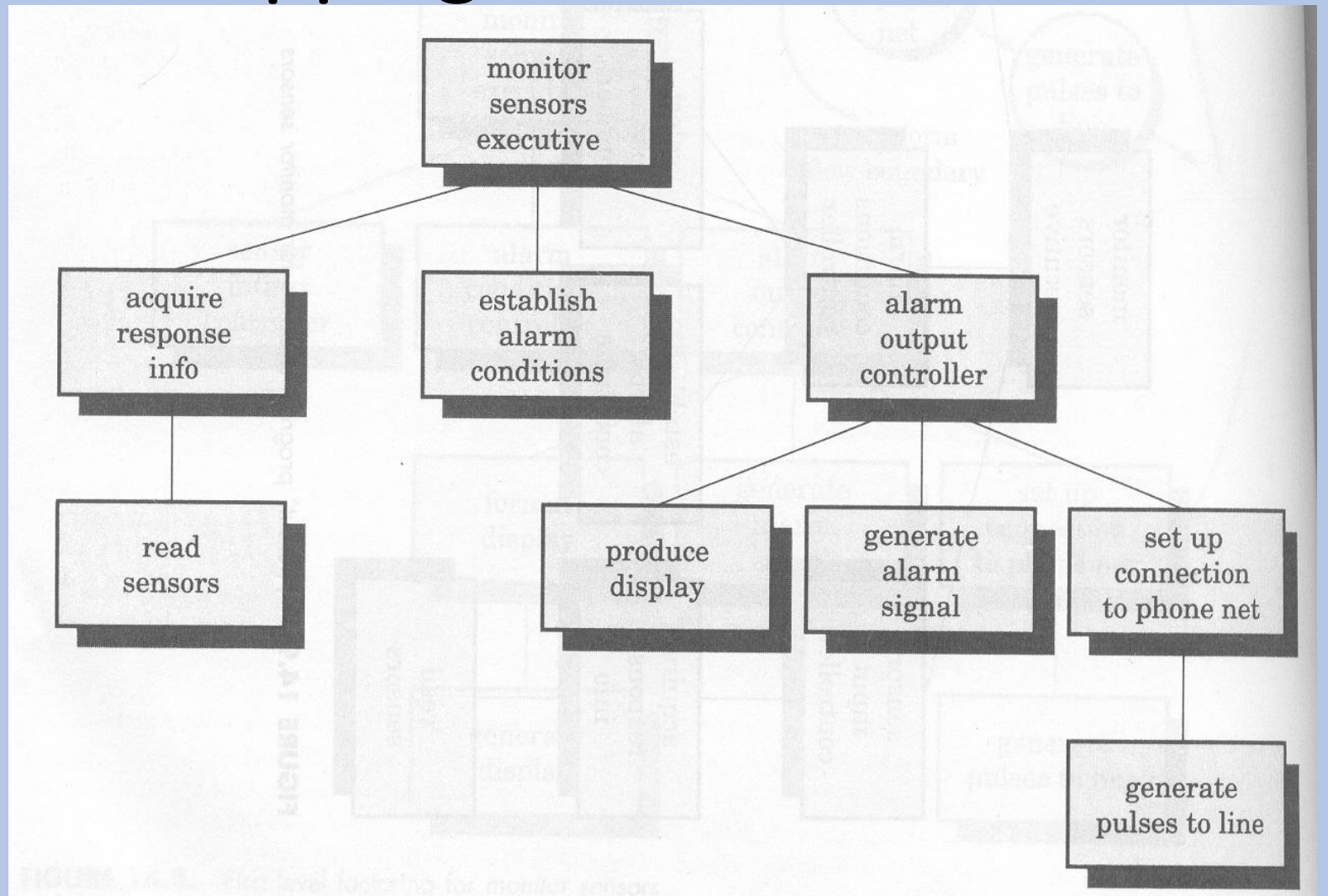


FIGURE 14.10. Refined program structure for *monitor sensors*

Transaction Mapping

- A single data item may trigger one of many information paths
- Our example will be user interaction subsystem of safehome s/w
- User command flows into the system
- Results in flow along one of three paths
- level 2 DFD is as shown

Transaction Mapping

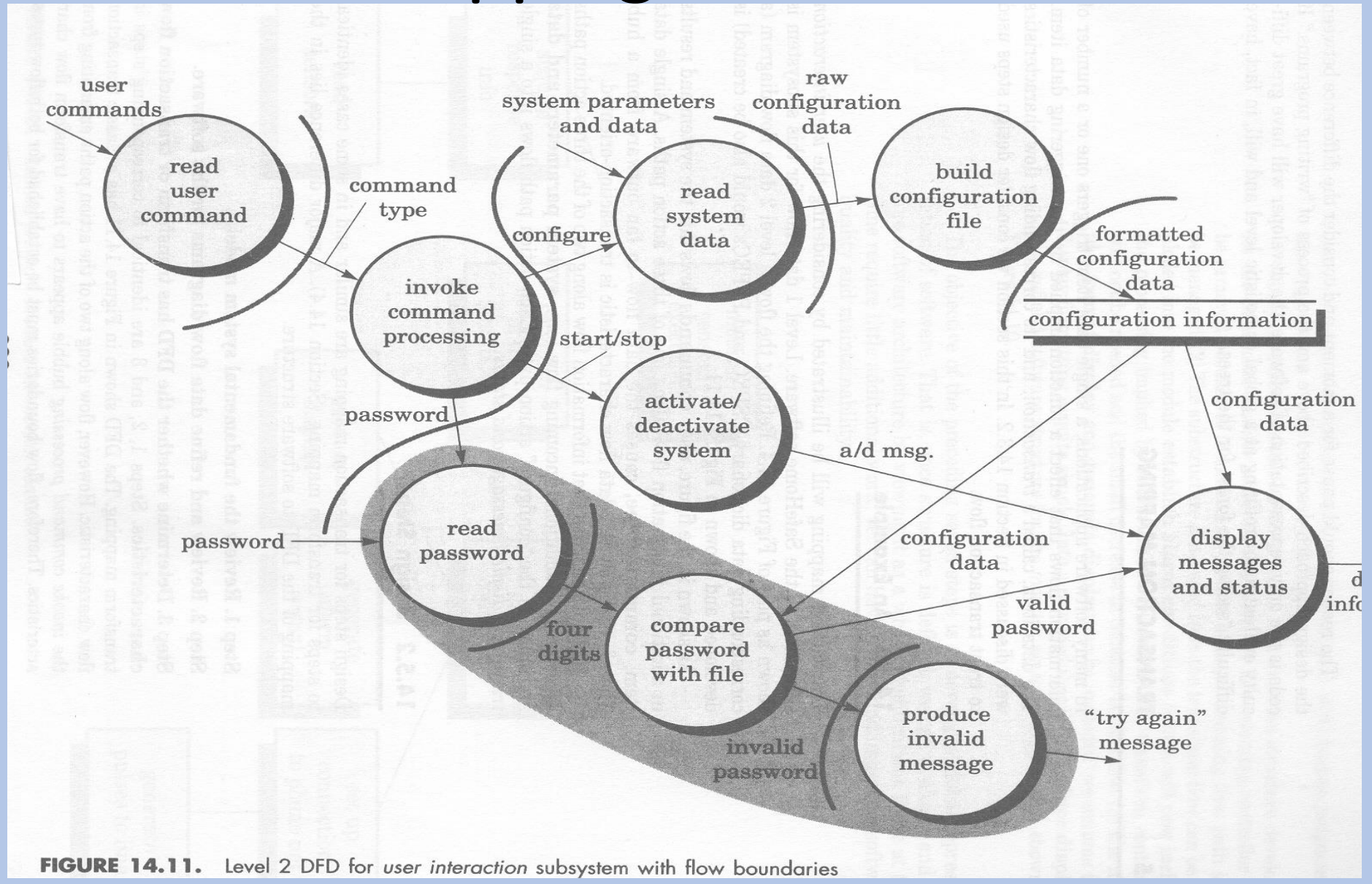


FIGURE 14.11. Level 2 DFD for *user interaction* subsystem with flow boundaries

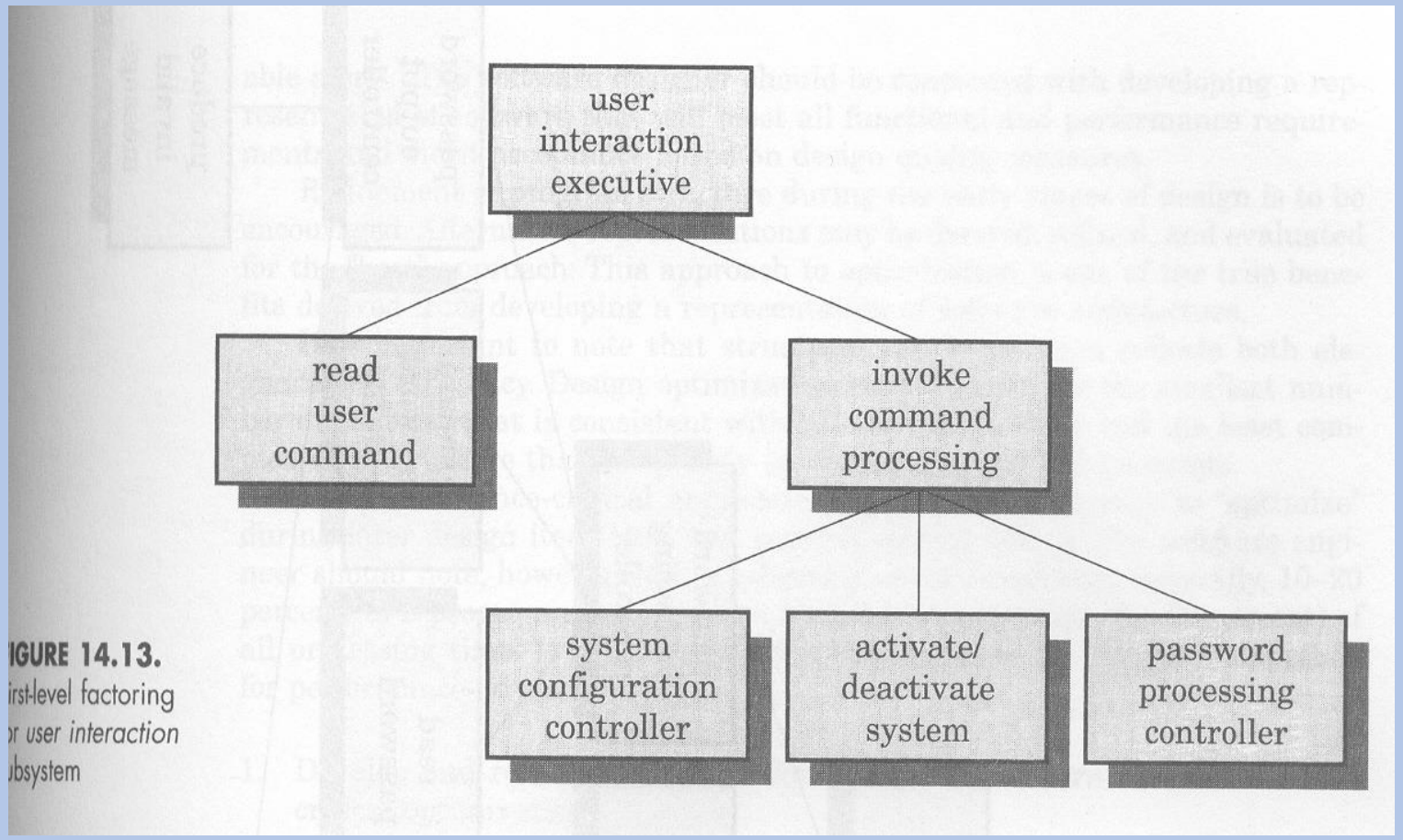
Transaction Mapping

1. Design Steps
2. Review fundamental system model
3. Review & refine data flow diagrams for the s/w
4. Determine if it has transform or transaction flow : invoke command processing has transform flow – flow boundaries for both flows
5. Identify transaction center & the flow characteristics along each action path : can be found from DFD – at the origin of no of action path – each action path evaluated for its individual flow characteristics

Transaction Mapping

5. Map the DFD in a program structure amenable to transaction processing : mapping into a programs structure
   Structure of dispatch branch contains dispatcher modules that controls subordinate modules
   Each action flow path mapped
   Ex. user interaction first level factoring as shown in fig

Transaction Mapping



IGURE 14.13.
irst-level factoring
r user interaction
ubsystem

Transaction Mapping

6. Factor & refine the transaction structure & structures of each action path : each action path has its own flow characteristics

   Password processing (shaded part) exhibit transform characteristics

   Resultant architecture will be as shown in fig

Transaction Mapping

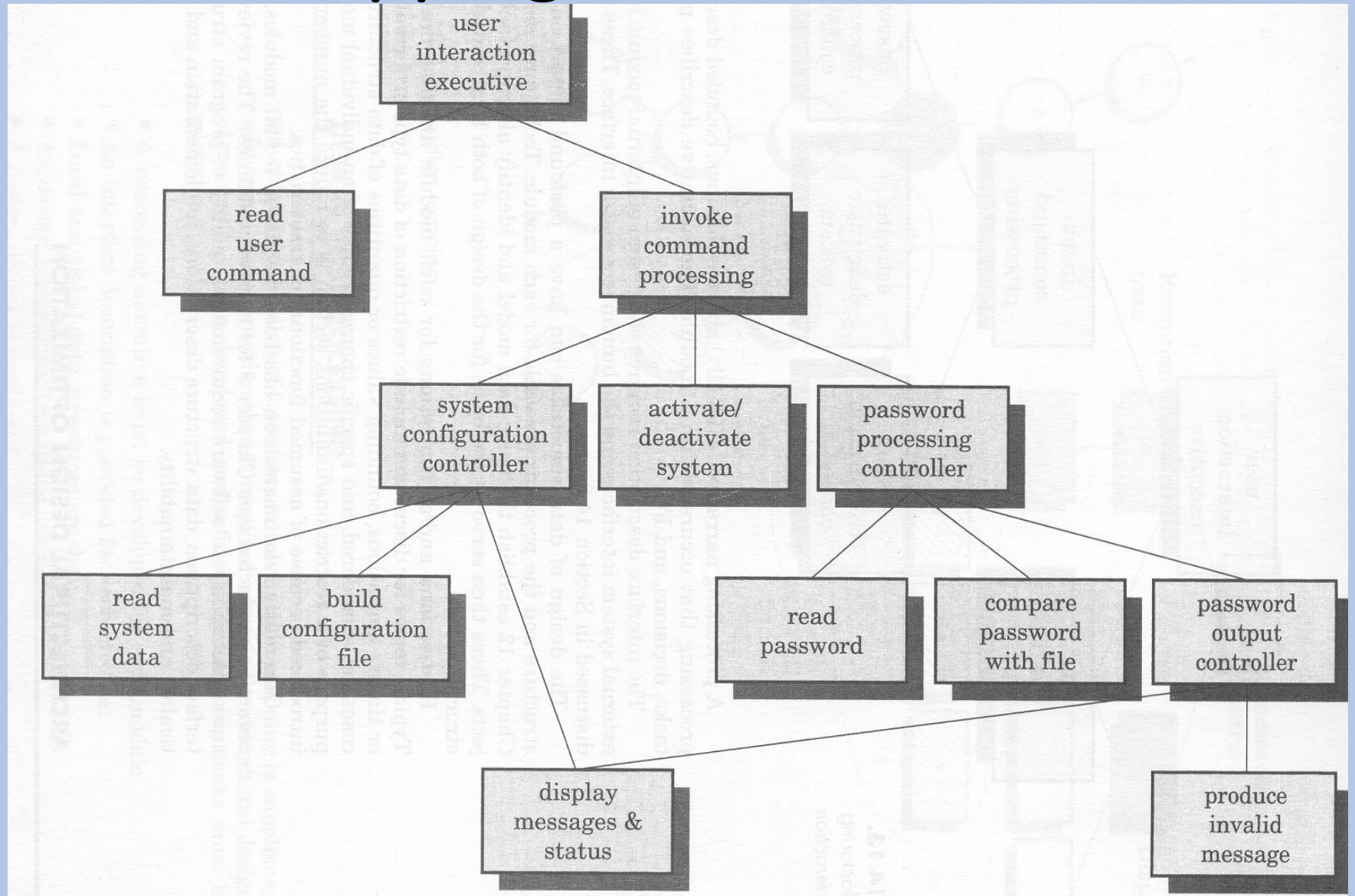


FIGURE 14.14. First-cut program structure for *user interaction* subsystem

Transaction Mapping

7. Refine the first iteration architecture using design heuristics for improved s/w quality : criteria such as module independence, practicality etc considered

Supplementary Tasks

- Processing narrative must be developed for each module – unambiguous bounded description, describes processing tasks, i/o, decisions
- Interface description provided for each module – design of internal module i/f, external system i/f, human-computer i/f
- Local & global data structure are defined – impact on archi & procedural detail
- Design restrictions/ limitations noted – data type/format, memory, timing, purpose is to reduce no of errors introduced by assumed characteristics
- Conduct design review – for quality & maintainability
- Refinement is considered – for optimization, structural simplicity, less no of modules with effective modularity