Transitional Analysis

Guidelines for transforming DFD models to Structure Chart models

Ernest Cachia
Department of Computer Science

The Reasons

- To move from an abstract system representation to a more physical one.
- To offer some guidelines to this procedure.
- To reduce ambiguity which may arise from subjective interpretations.
- To move from data flow concepts to program structure concepts.

General Steps Involved (1)

1. The type of data flow is established
   - What is the nature of the data flowing between processes?
2. Determine flow boundaries (switch points)
   - Input↔Output boundaries
   - Hub↔Action boundaries
3. Map the abstract DFD on to a particular program structure
   - Transformational structure
   - Transactional structure

General Steps Involved (2)

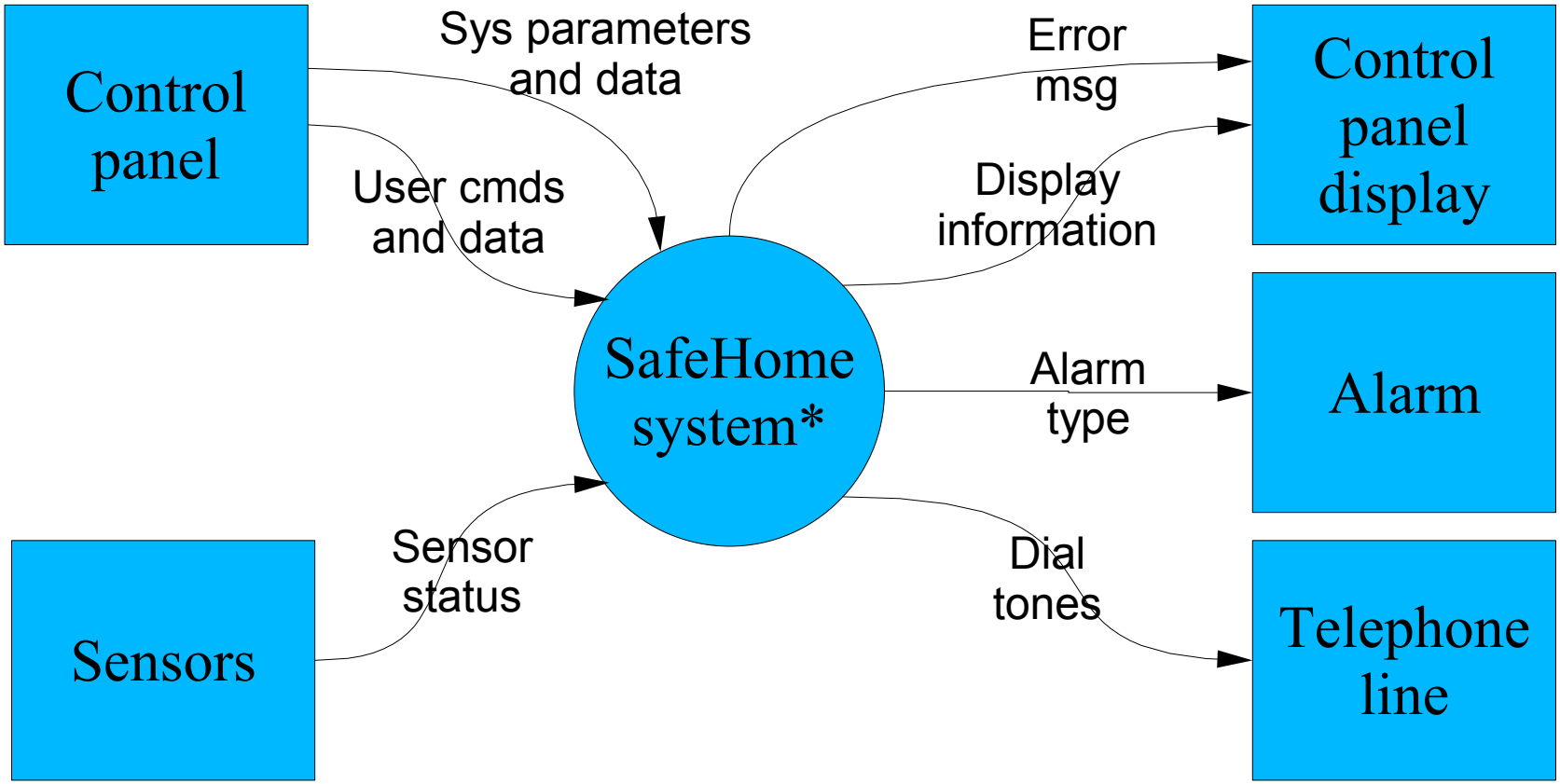
4. Define a valid control structure
   - Also known as “first-level” factoring
   - Depends on whether transformational or transactional models are used.
   - “Call-and-return” for transformational
   - “Call-and-act” for transactional
5. Refine (tune) the resulting structure
   - Also known as “second-level factoring”
   - Map IO flow bounded parts of DFD

General Steps Involved (3)

6. Supplement and tune the final architectural structure
   - Apply basic module independence concepts (i.e. Explode or implode modules according to coupling/cohesion requirements) to obtain an easier implementation.
   - Please also visit the slides on the web-site “www.sei.cmu.edu/ata/ATAM/index.htm” for a more comprehensive and interesting approach to architectural analysis known as “Architectural Trade-off Analysis – ATA”.

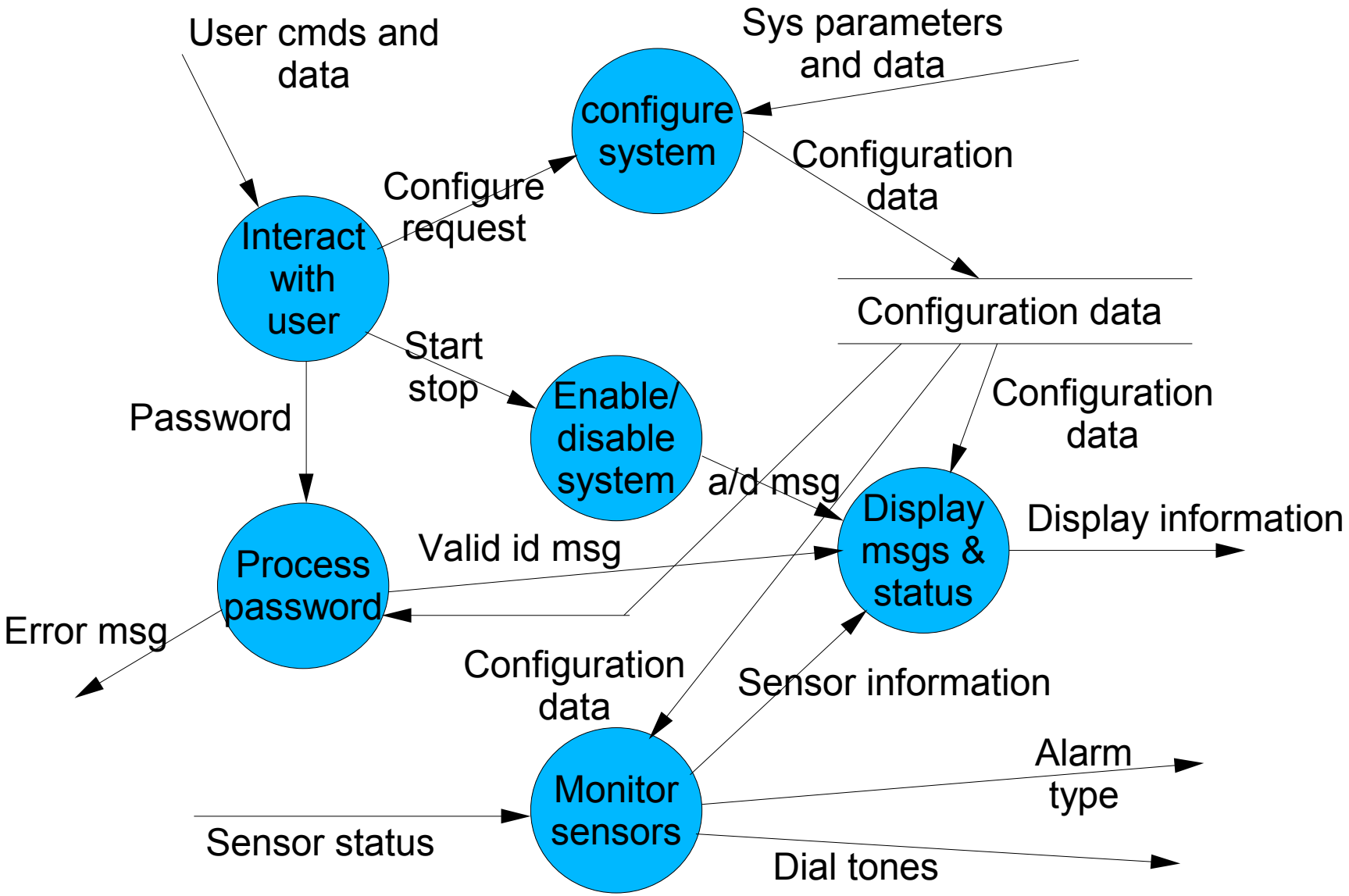
Transformational Analysis (aka Transformational Mapping)

Context level (level 0) Example

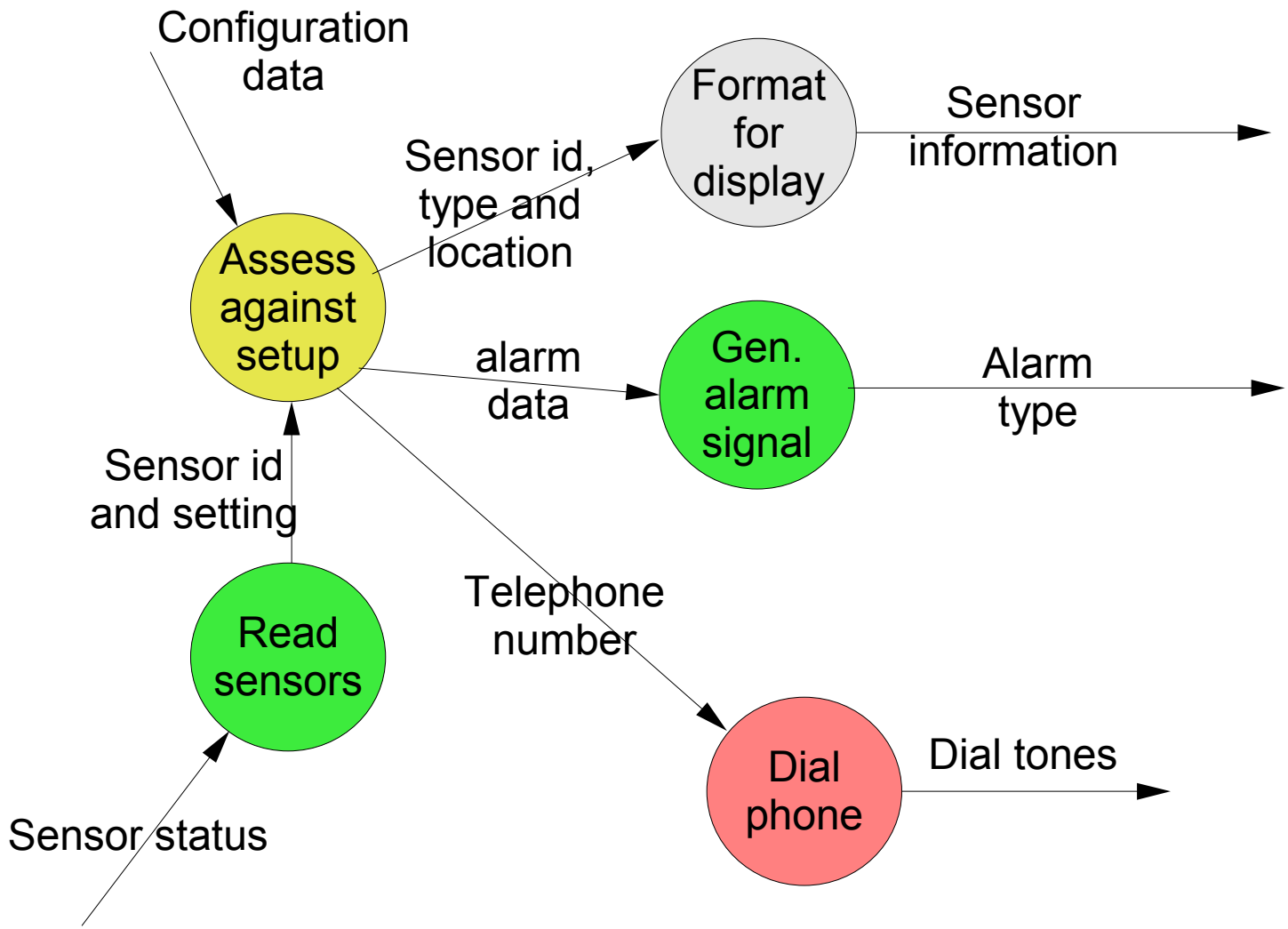


* Example taken from “Software Engineering – A Practitioner's Approach” by R. S. Pressman.

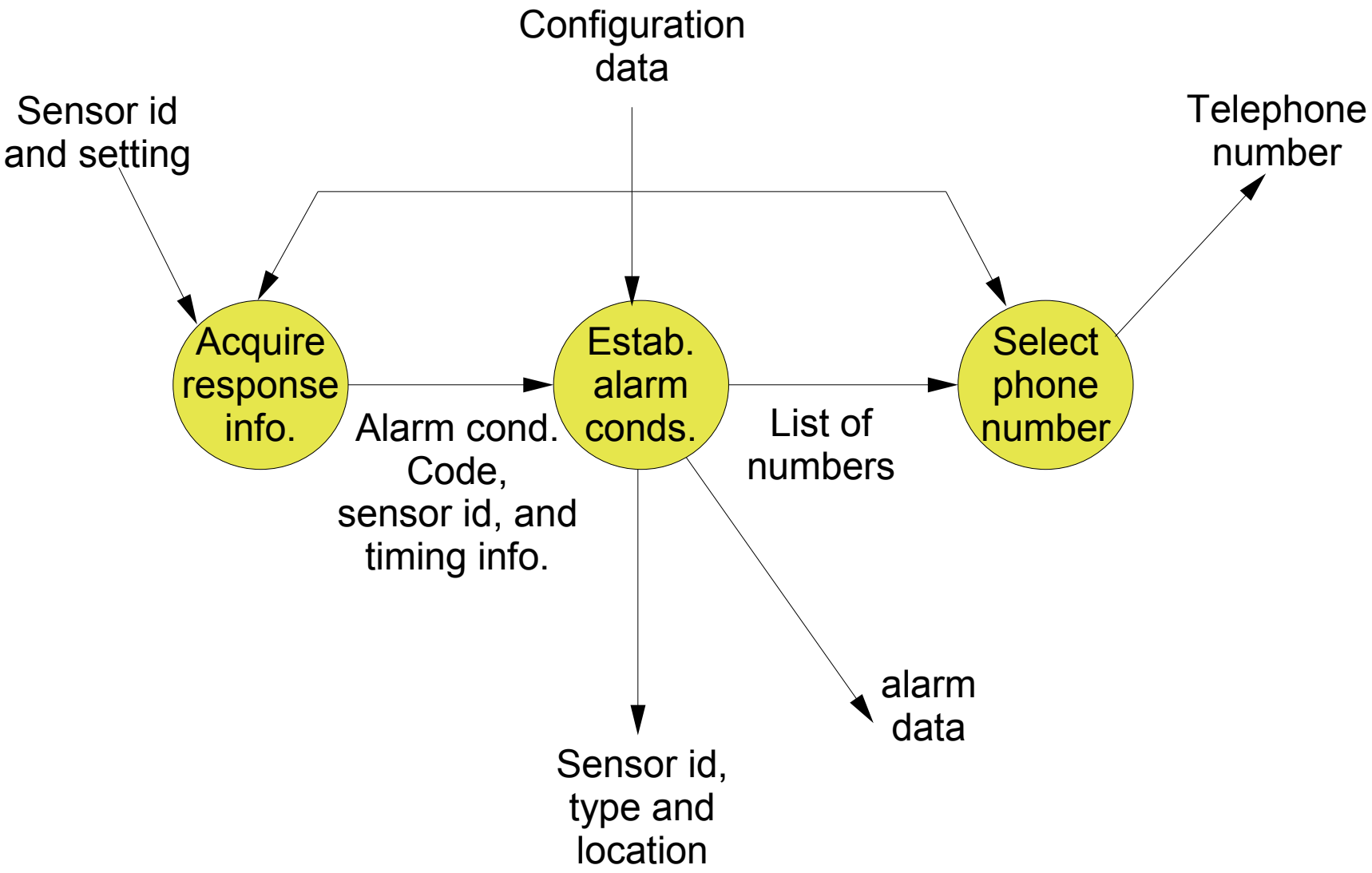
Level 1 Example



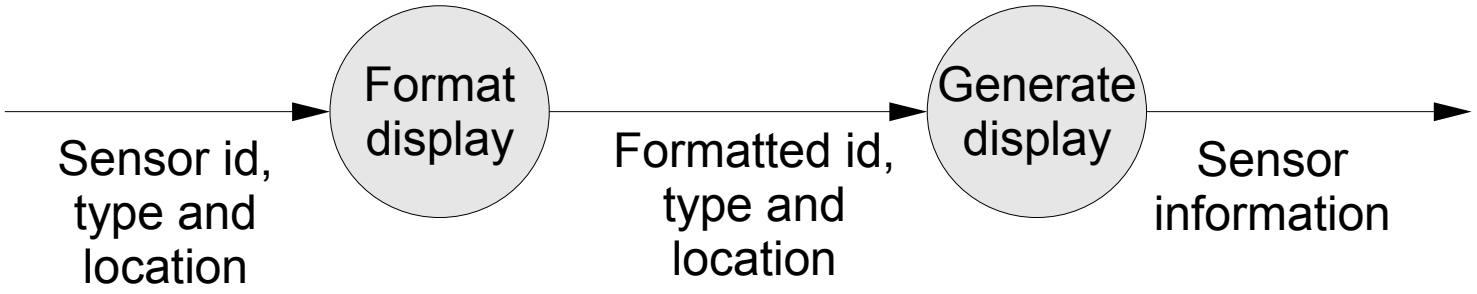
Level 2 Example (Monitor sensors)



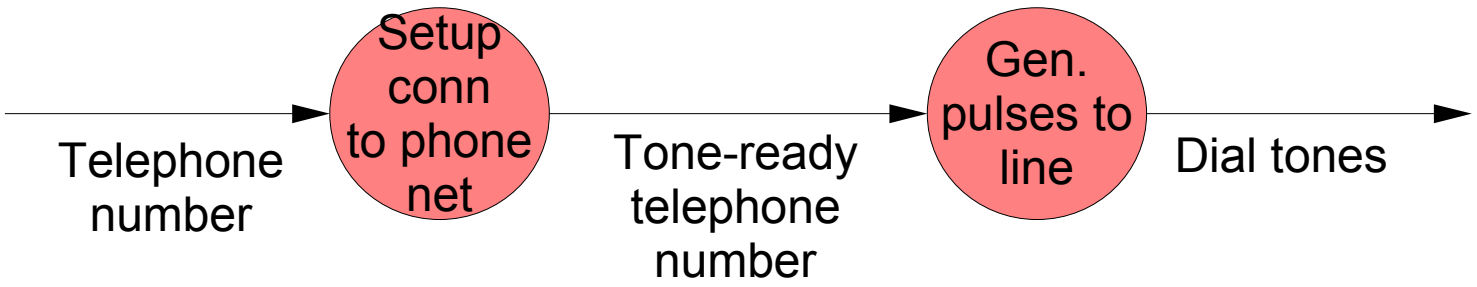
Level 3 Example (Assess against setup)



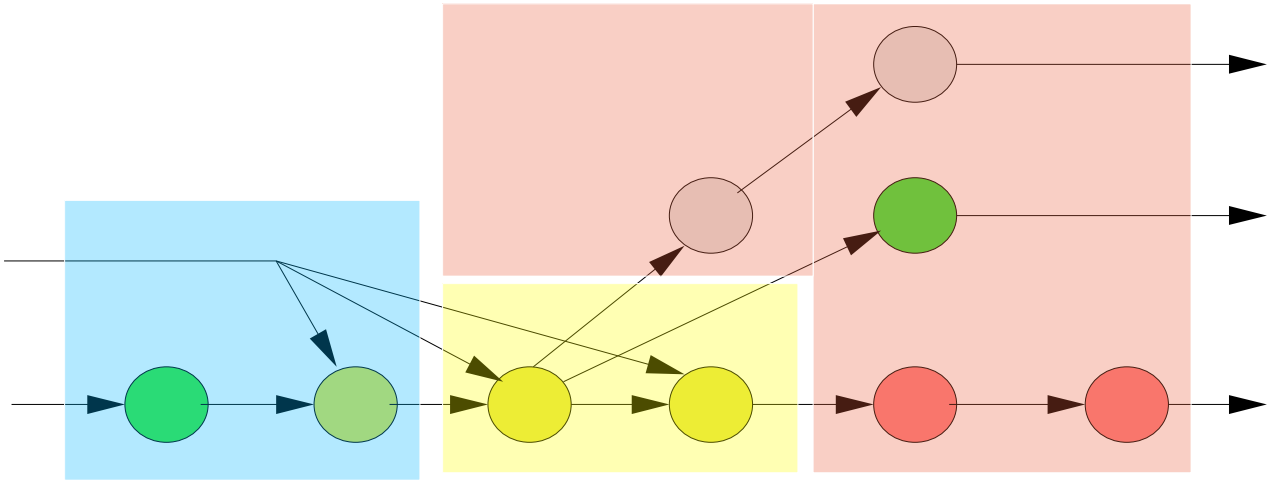
Level 3 Example (Format for display)



Level 3 Example (Dial phone)



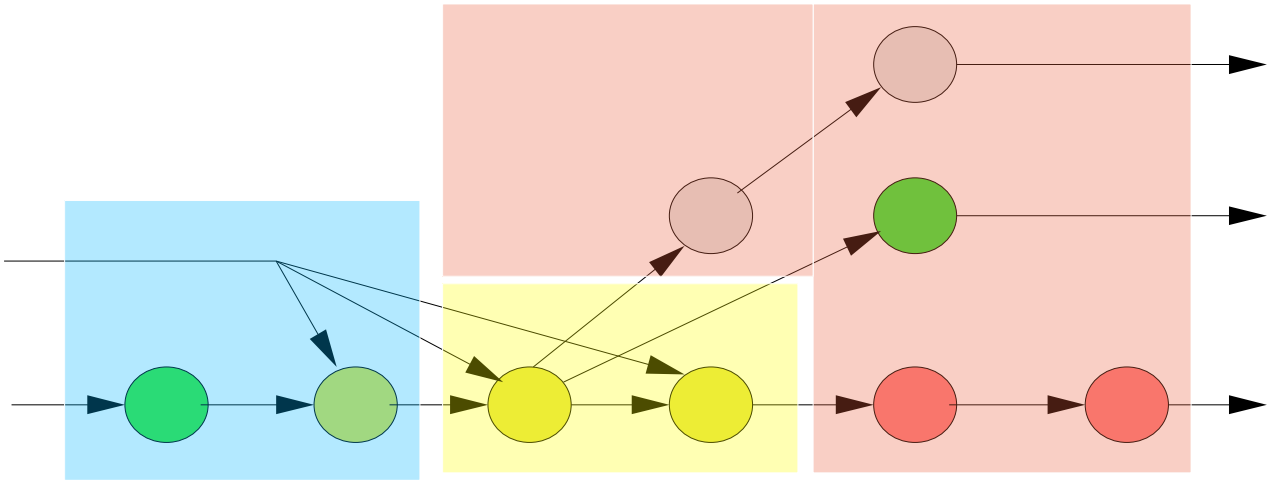
Putting Level 3 Together

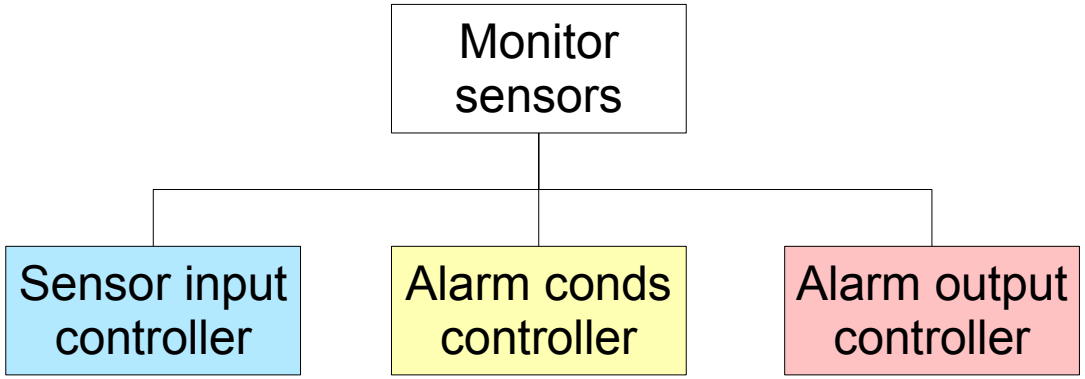


This DFD exhibits definite transform flow character.

- Afferent branch (input)
- Central transform (processing)
- Efferent branch (output)

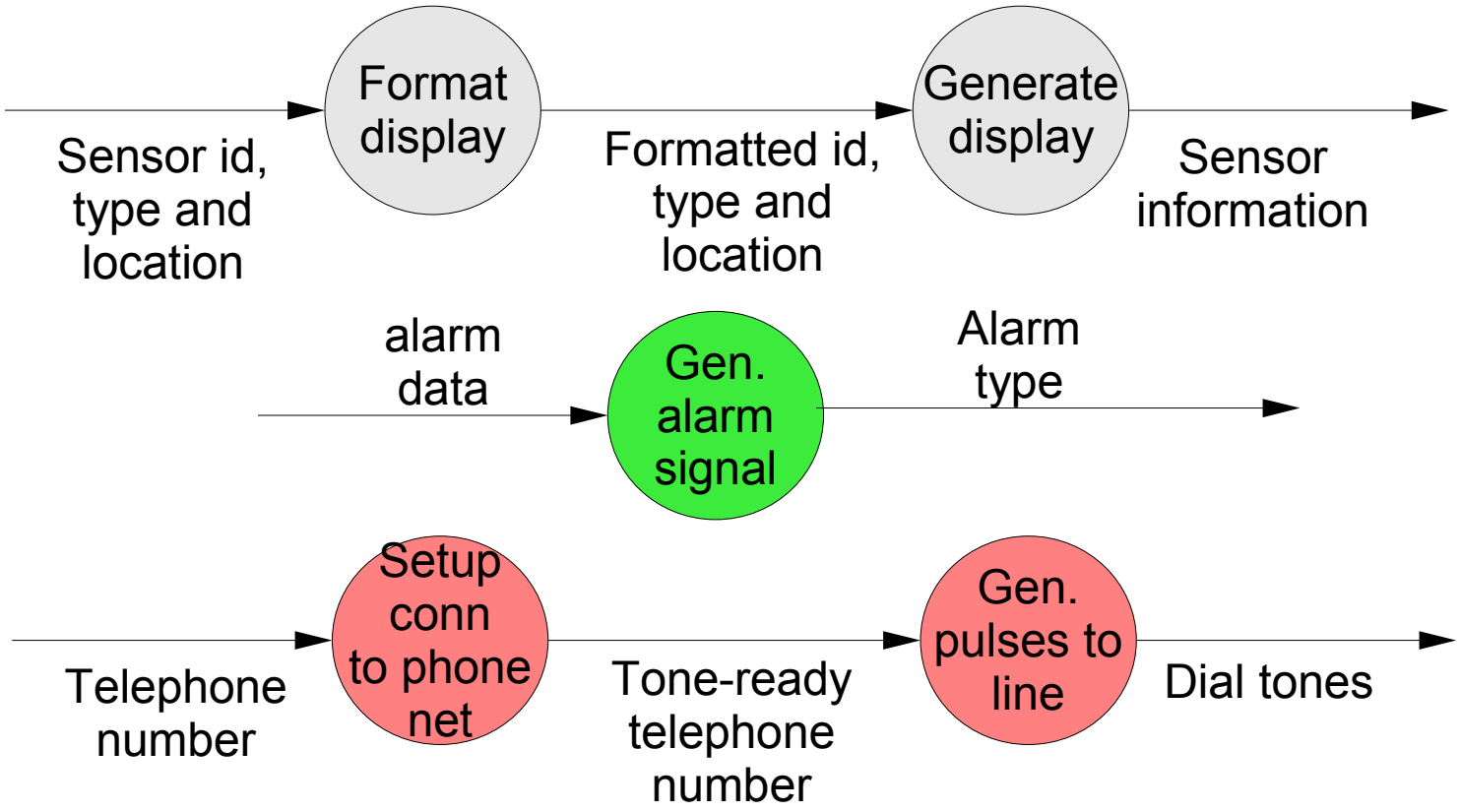
First-Level Factoring





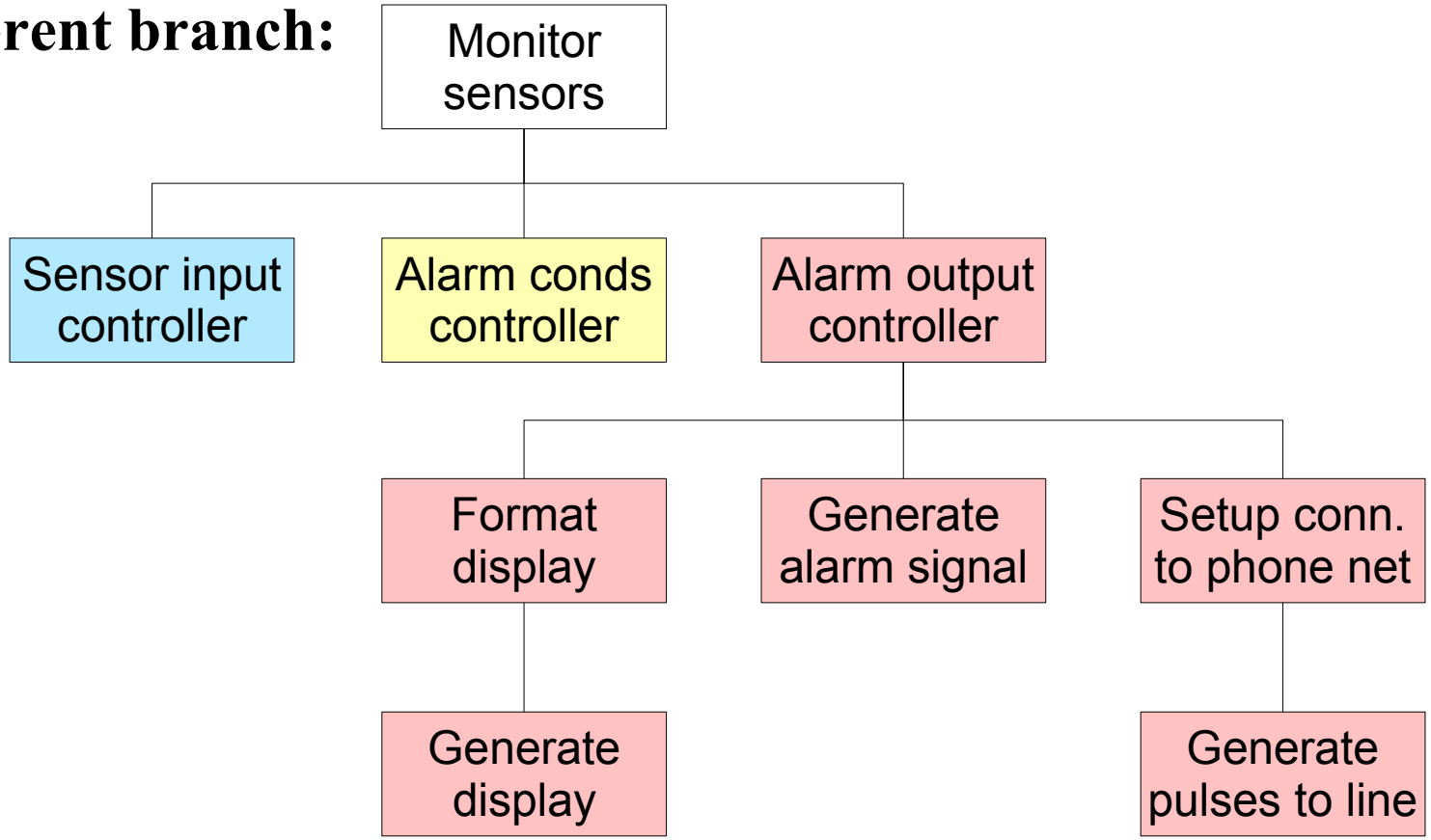
Second-Level Factoring (1)

These are all the processes in the efferent branch:



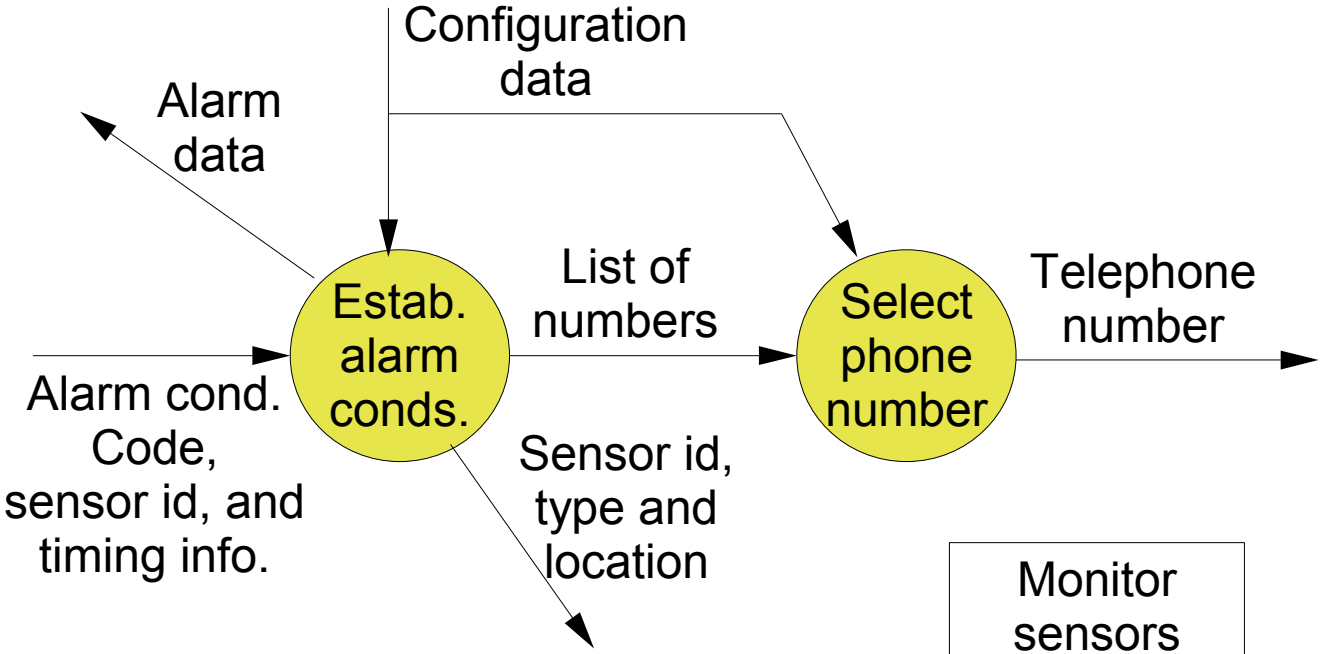
Second-Level Factoring (2)

For the efferent branch:

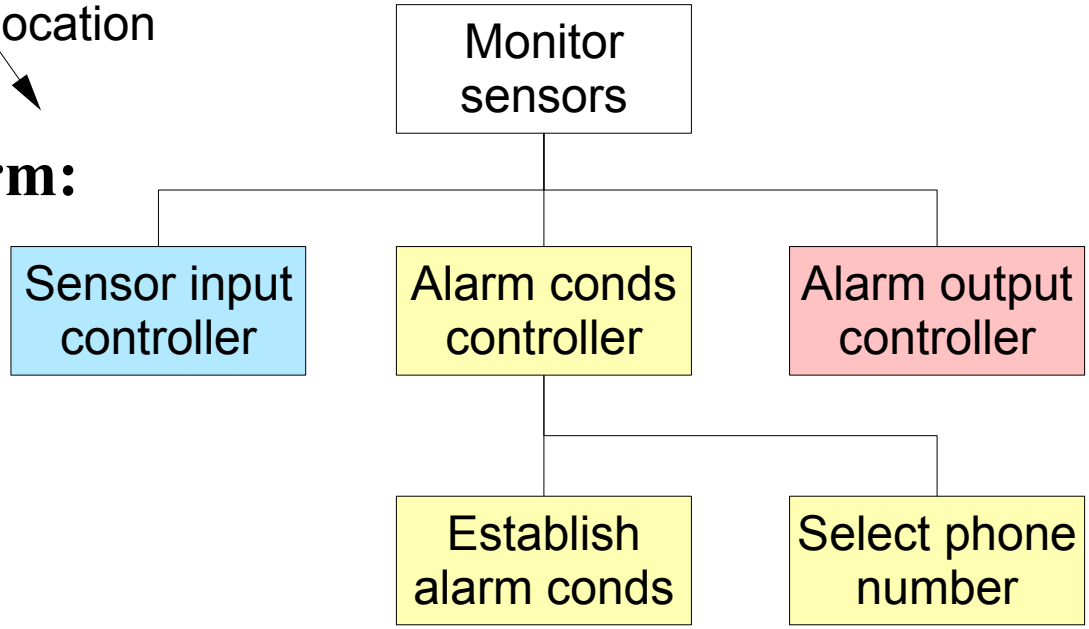


Now, do the same for the other branches (i.e. Afferent and Central)...

Second-Level Factoring (3)

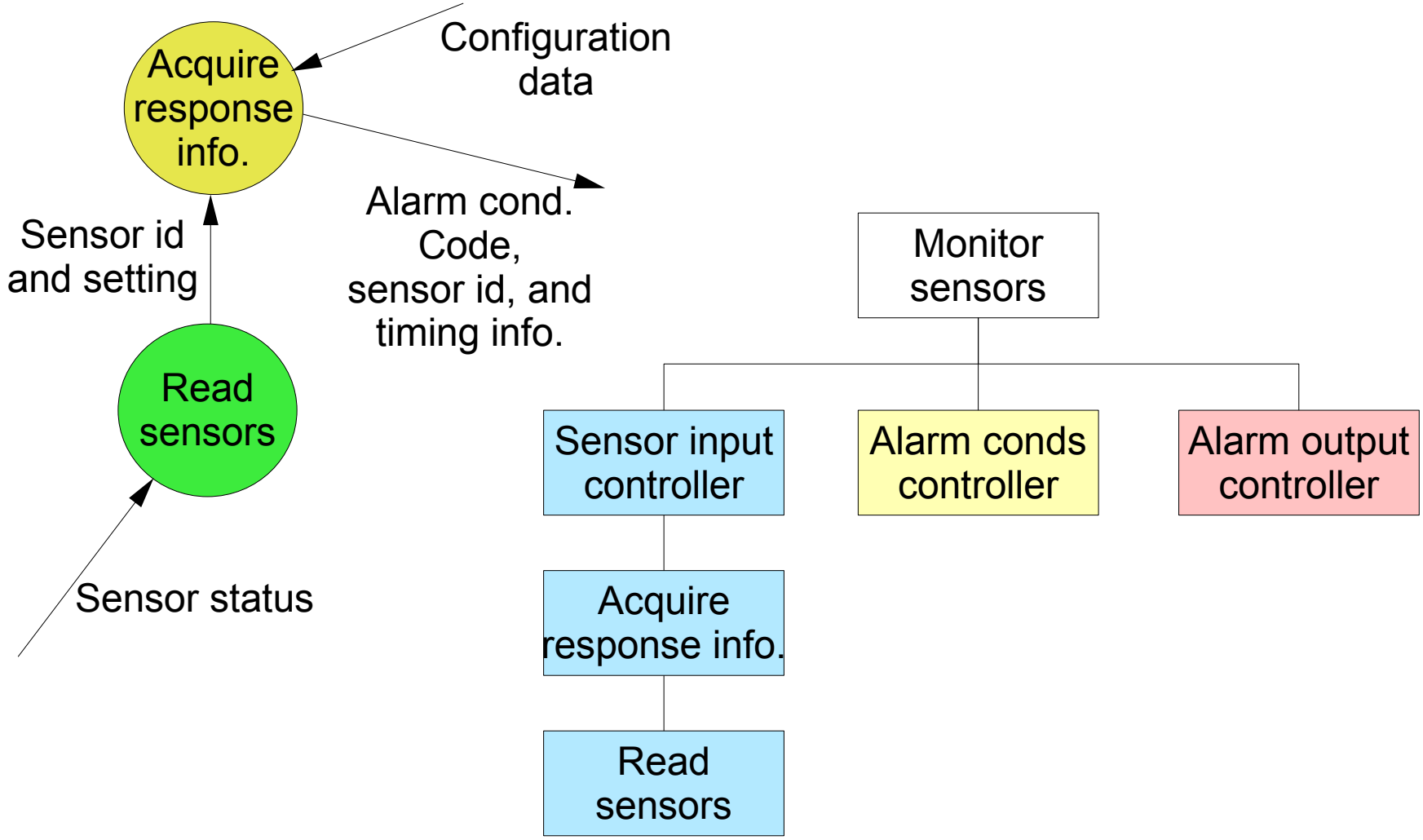


For central transform:



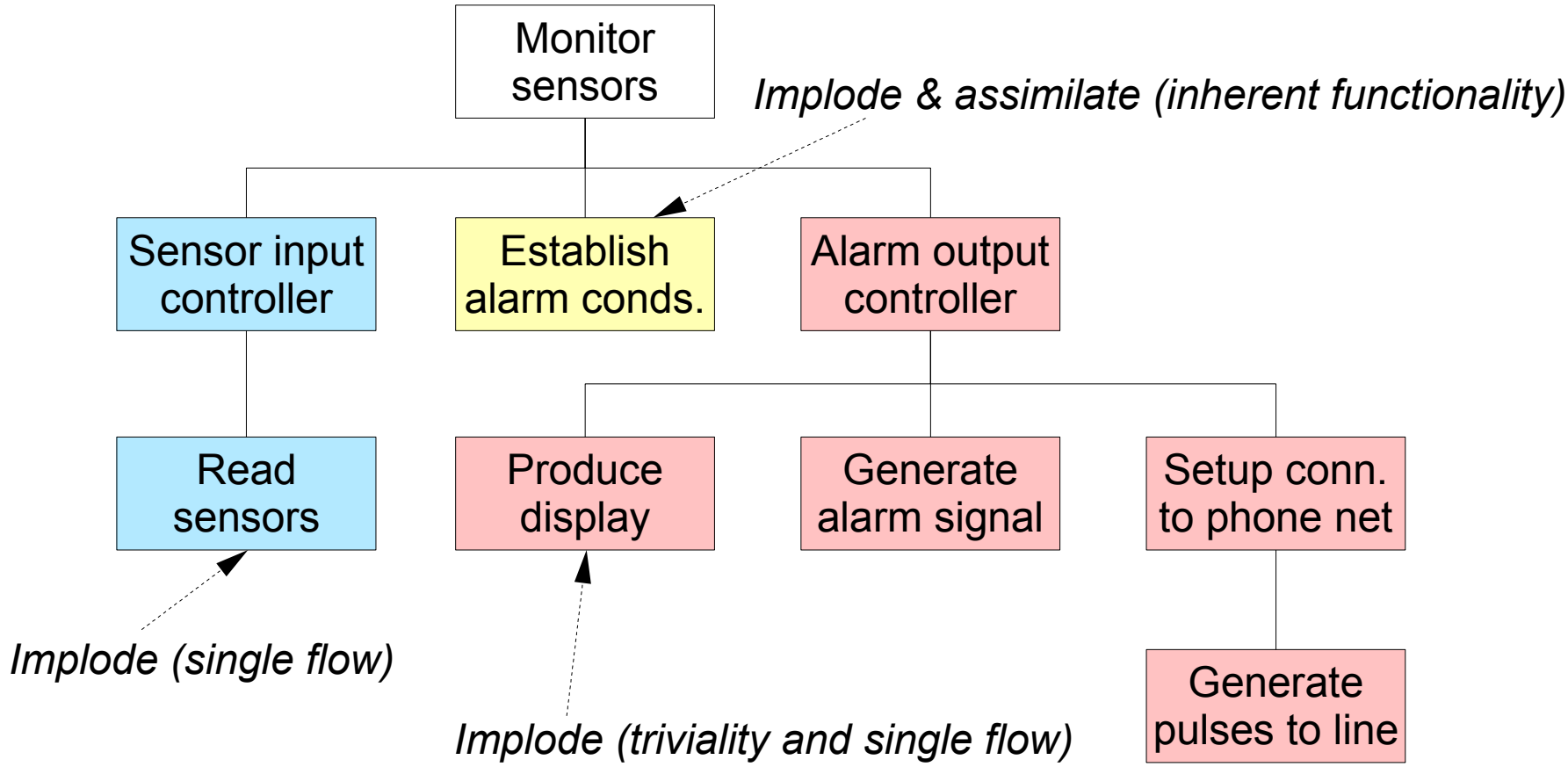
Second-Level Factoring (4)

Finally, for the afferent branch:



Design Refinement

- Some degree of refinement can sometimes be carried out on the initial “rough-cut” of the system's structure.

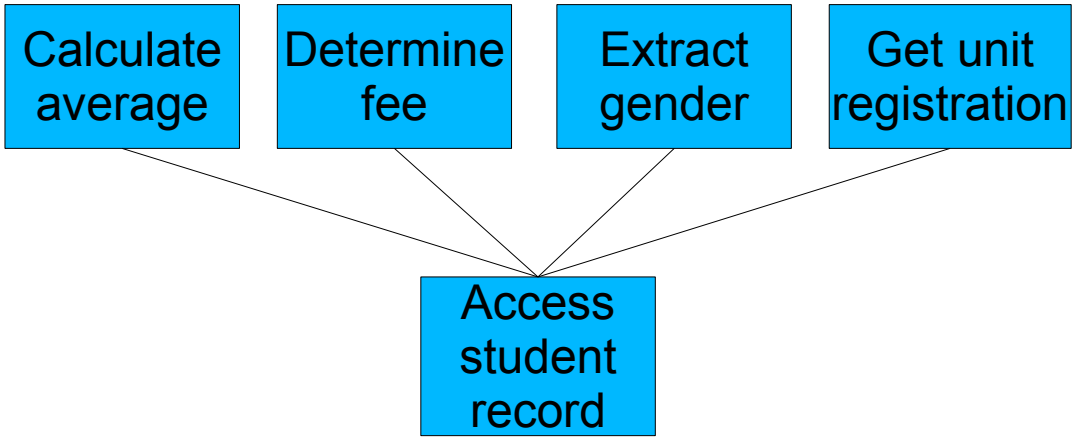


What next...?

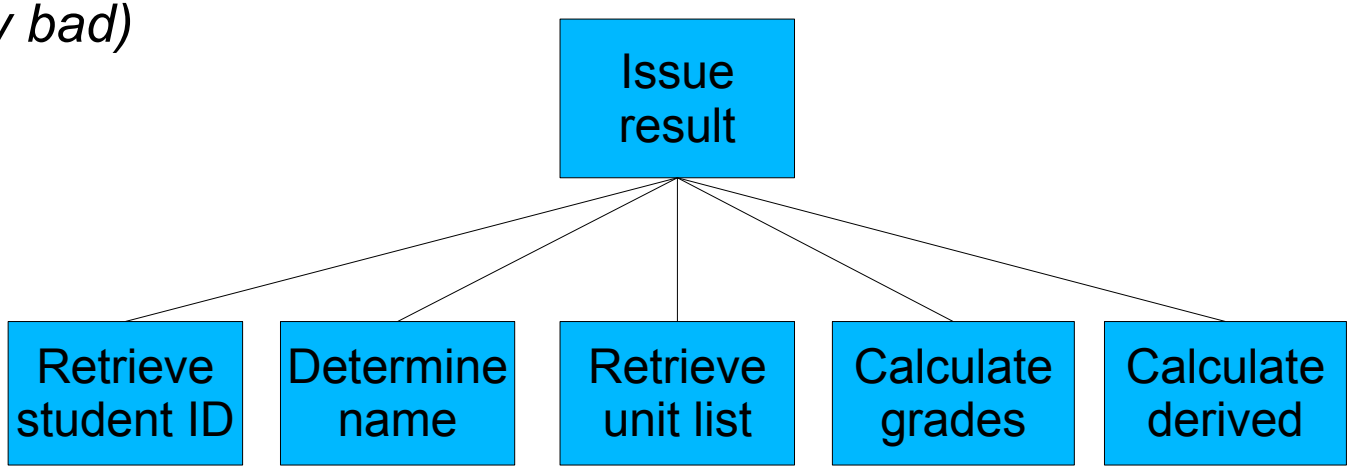
1. Check out your coupling and cohesion
   This was discussed earlier
1. Check out your “fan-in” and “fan-out”
   Fan-in should be high
   Fan-out should be low
   (see next slide)
1. Go through a predefined final check-list
   This will be organisation dependent
   (see example two slide on)

Fanning

High fan-in example:
(good)



High fan-out example:
(watch out – usually bad)



Final Check-list Example

Typical check-list content

(Example taken and adapted form Alan Dennis: "Systems Analysis & Design")

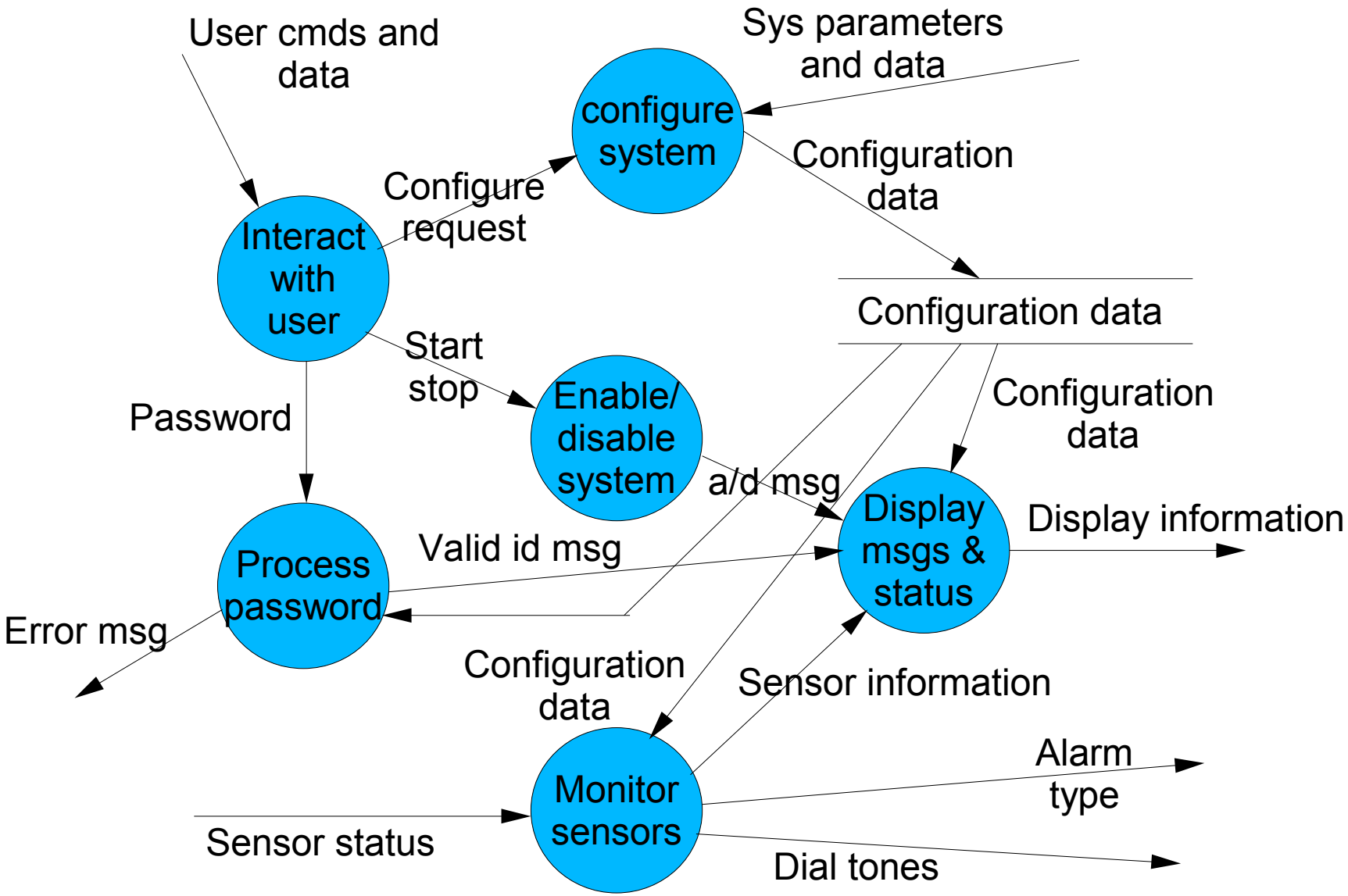
- Library modules have been created whenever possible (reuse and portability).
- The diagram has a high fan-in structure (reuse).
- Control modules have no more than 7 subordinates (efficiency).
- Each module performs only one distinct function (cohesive).
- Modules sparingly share information (de-coupled).
- Data couples passed are actually used by recipient (stamp coupling).
- Control couples are passed from "child" to "parent" module? (control coupling)
- Each module has a reasonable amount of code in it (a discernible system function).

Transactional Analysis (aka Transactional Mapping)

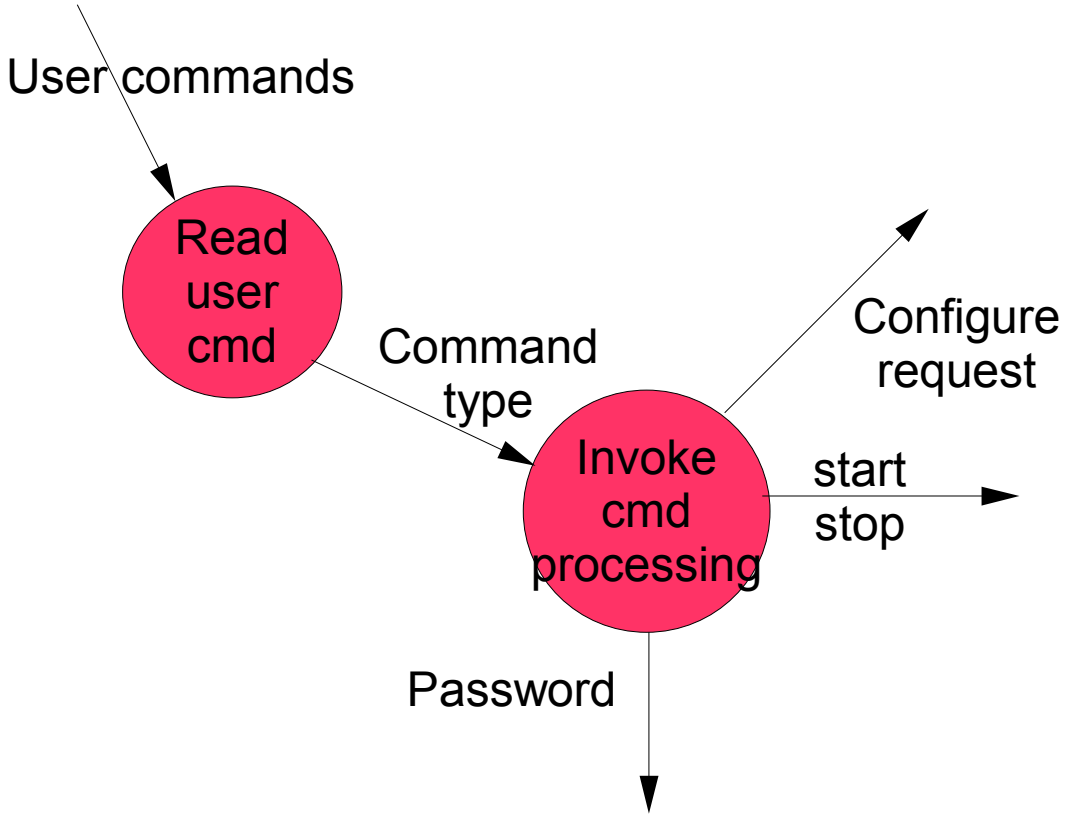
- First step of this analysis is exactly the same as for transformational – i.e. refine the DFD.
- The initial “SafeHome*” example will be used.
- The level one DFD will be decomposed to level two as necessary.
- A “transactional centre” will then be determined.

** Taken from “Software Engineering – A Practitioner's Approach” by R. S. Pressman.*

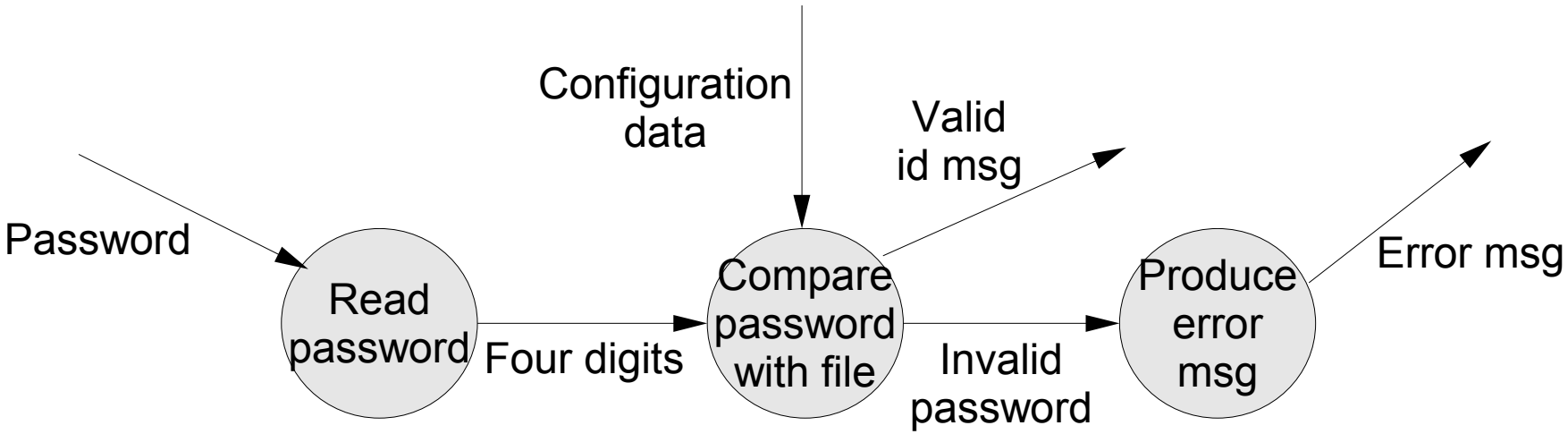
Level 1 (reproduced)



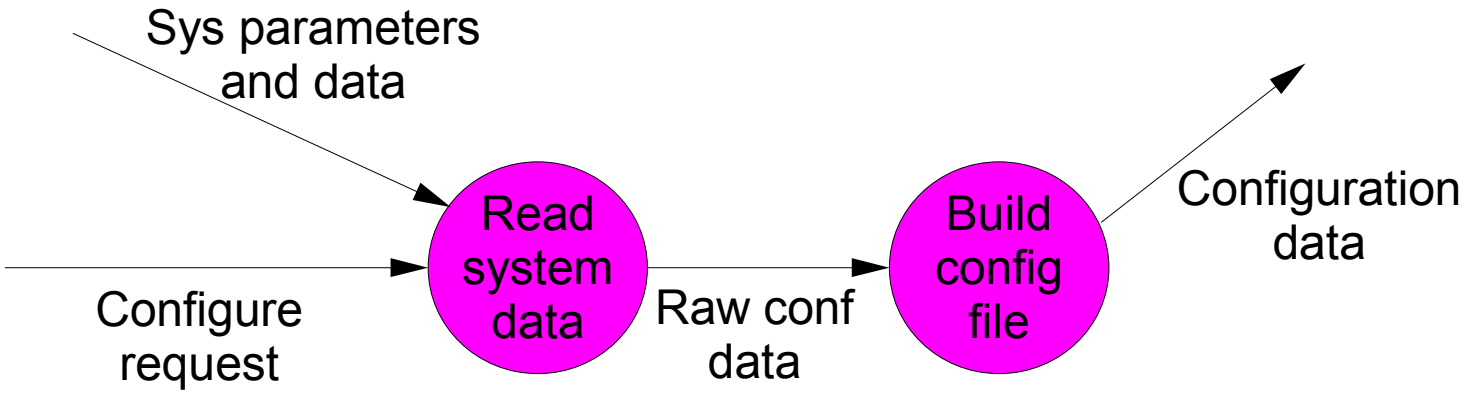
Level 2 Example (Interact with user)



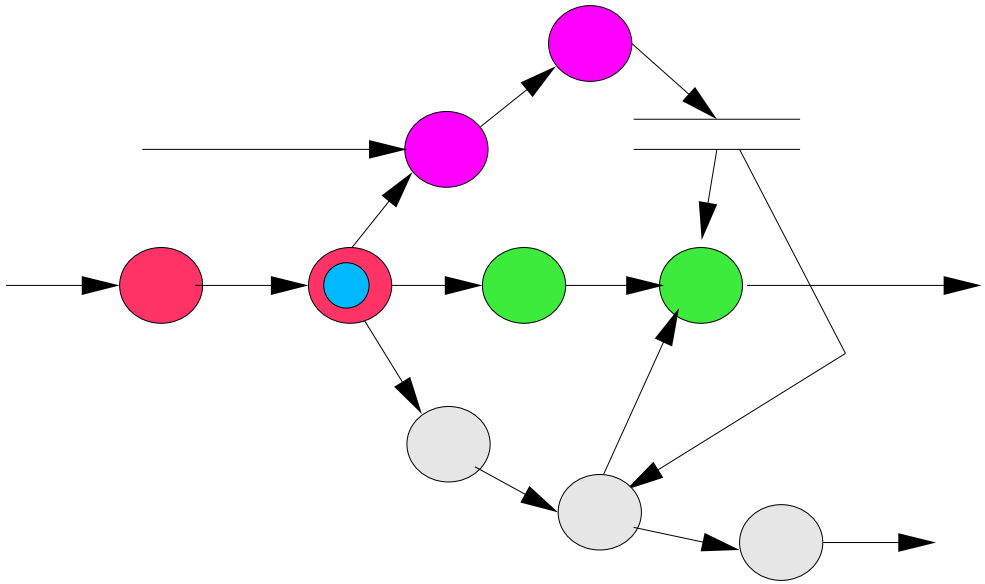
Level 2 Example (Process password)



Level 2 Example (Configure system)



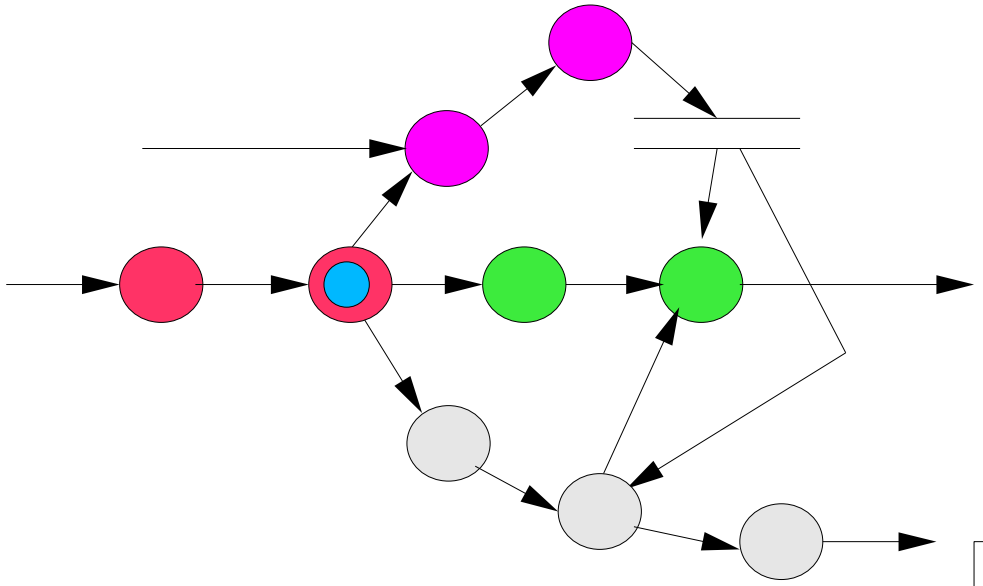
Putting Level 2 Together

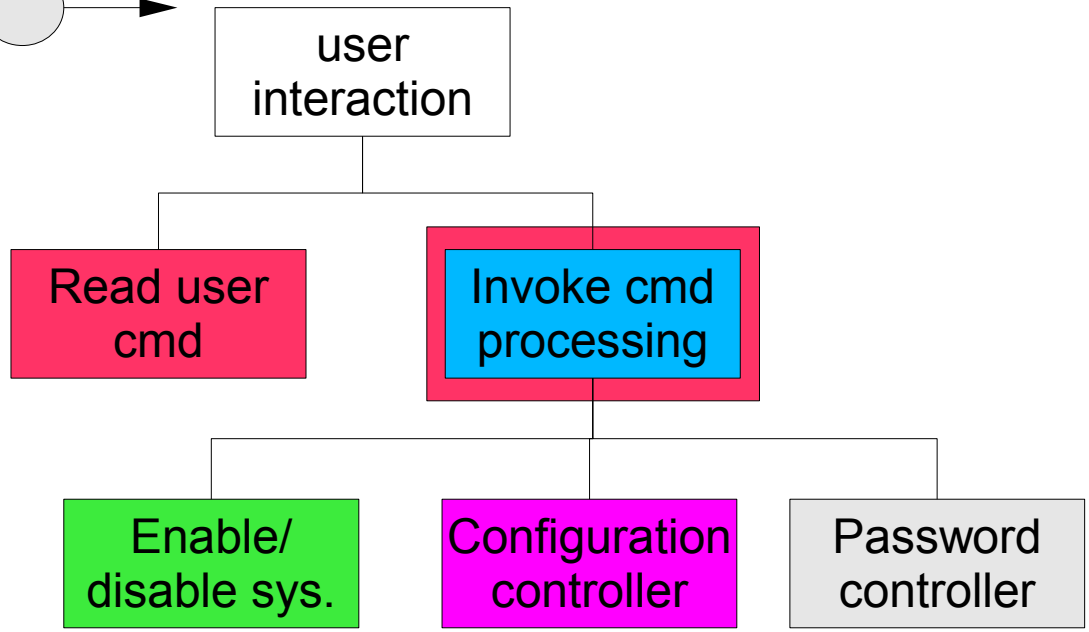


This DFD exhibits transaction flow character.

 Transaction centre (dispatching)

Transaction Structure Mapping





Refining The Structure

Fully refining the DFD could yield the following structure: