ARSITEKTURAL DESIGN

Bass, Clements, dan Kazman [Bass, 2003 via Pressman, 2010) mendefinisikan:

> **"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them".**

Arsitektur software menjelaskan susunan sistem yng terdiri dari komponen software, atribut dari komponen dan hubungan antar komponennya. Komponen dapat berupa modul, database, middleware, atau class. Atribut adalah ciri dan fungsi modul. Hubungan antar komponen adalah cara antar komponen tersebut berkomunikasi, sperti modul satu memanggil modul lain.

Fungsi dari desain arsitektur adalah memampukan software developer untuk :

1. Analisis efektifitas dari desain untuk memenuhi kebutuhan/ requirement yg diminta. Berguna untuk komunikasikan gambaran sistem dg stakeholder yg berkaitan dng pembangunan sistem
2. Mempertimbangkan alternatif lain ketika ada perubahan desainterjadi. Arsitektur memberikan gambaran bagaimana sistem itu saling terkait dan bekerja.
3. Mengurangi risiko yang berkaitan dng pembangunan software atau coding. Arsitektur memiliki pengaruh besar dalam rekayasa perangkat lunak karena menjelaskan hasil keputusan dalam bentuk desain

Struktur Arsitektur

Struktur Fungsional	komponen mewakili fungsi atau proses, konektor adalah antarmuka untuk menggunakan atau melewatkan data, properti menggambarkan sifat ari komponen dan organisasi dari antarmuka
Struktur implementasi	komponen mewakili package, class, object, procedure, function, atau method. Konektor berarti menggunakan, mengirim data/kontrol, berbagi data. Properti fokus pada kualitas karakter
Struktur konkurensi	komponen mewakili unit-unit yang tersusun secara paralel/konkuren kerjanya. Konektor menjelaskan: sinkronisasi-dengan, prioritas-lebih-tinggi-dari, kirim-data-ke, kerjasama-dg, tak-berfungsi-tanpa. Properti menjelaskan prioritas, kemampuan-mendapatkan lebih dulu waktu eksekusi.
Struktur fisik	komponen adalah hardware di mana software ada. Konektor adalah antarmuka hardware dengan hardware lain. Properti adalah kapasistas, bandwidth, kinerja atau atribut lain pada hardware
Struktur pengembangan	struktur definisikan komponen, produk kerja, dan sumber informasi lain yang dibutuhkan selama pengembangan software. Konektor mewakili relasi antar produk kerja dan properti identifikasi karakteristik tiap item.

Setiap struktur di atas mewakili arsitektur software dari sudut pandang yang berbeda, mengekspos informasi yang digunakan tim pengembang selama proses desain dan coding berlangsung.

Klasifikasi Gaya Arsitektur

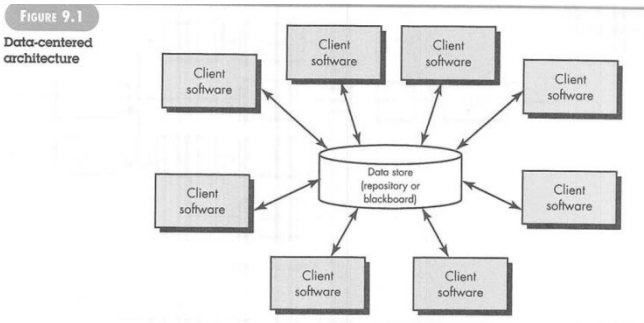


FIGURE 9.1
Data-centered architecture

Data-centered architecture

Suatu data store menjadi pusat di antara komponen lain yang mengaksesnya dalam rangka untuk update, tambah habis atau ubah data. Data store tersebut adalah repository pusat. Tiap komponen yang mengakses data berdiri sendiri sehingga memungkinkan adanya tambahan komponen tanpa mengganggu komponen lain. Contoh arsitektur ini seperti pada Gambar 9.1

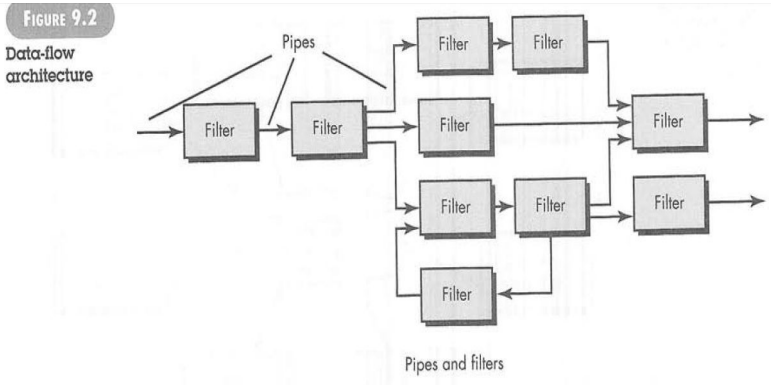
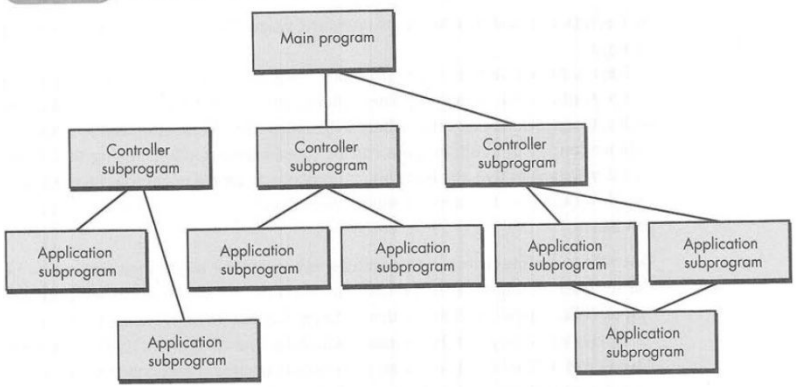


FIGURE 9.2
Data-flow architecture

Data-flow Architecture

Dimanfaatkan untuk menggambarkan input data yang diubah melalui serangkaian penghitungan dan manipulasi untuk menjadi output. Seperti pada Gambar 9.2 pipa dan filter menggambarkan aliran data dan komponen yang mengubah aliran data sehingga input menjadi output.

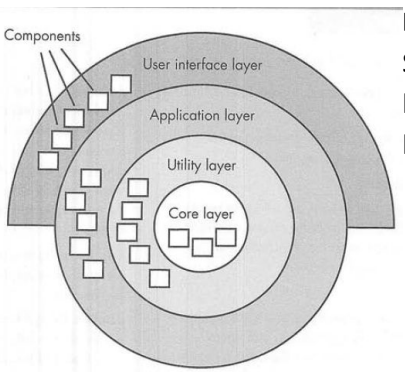
FIGURE 9.3 Main program/subprogram architecture



Call and Return Architecture

Menggambarkan struktur program yang disusun secara hirarki. Program dibagi menjadi beberapa sub program yang terdiri dari program utama dan beberapa sub program. Komponen mewakili sub program atau program utama. Gambar 9.3 menjelaskan hirarki dari program utama dan subprogram lain.

FIGURE 9.4
Layered architecture



Layered Architecture

Setiap lapisan menjalankan operasinya dan makin ke dalam komponennya makin mendekati perintah mesin. Yang terluar adalah lapisan yang paling dekat dengan pengguna.

Contoh Transformasi dari DFD ke Struktur Call n Return Architecture
Kasus : SafeHome Security Function

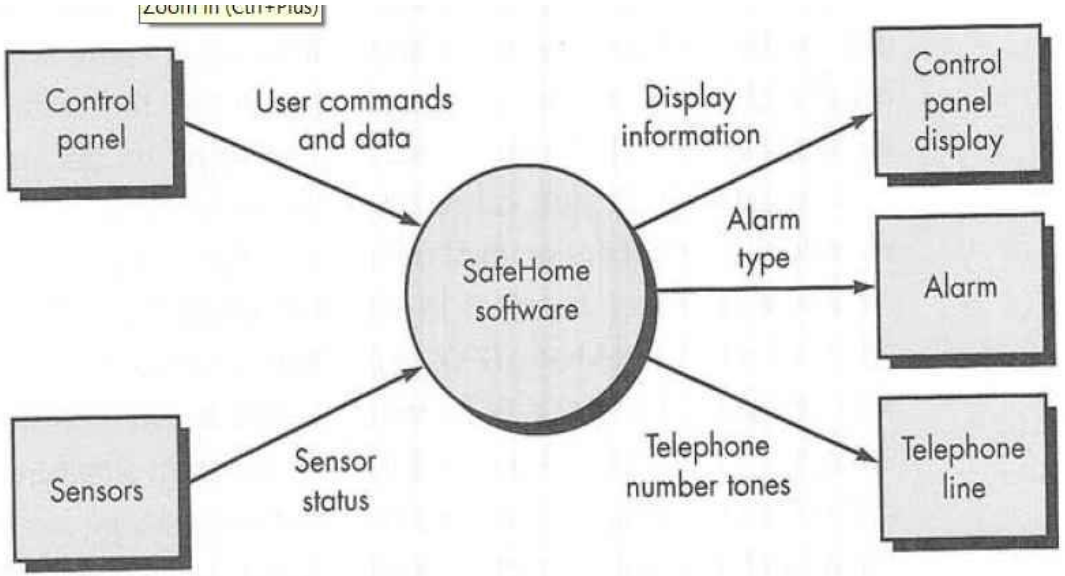


FIGURE 9.10
Context-level DFD for the *SafeHome* security function

FIGURE 9.11

Level 1 DFD for the *SafeHome* security function

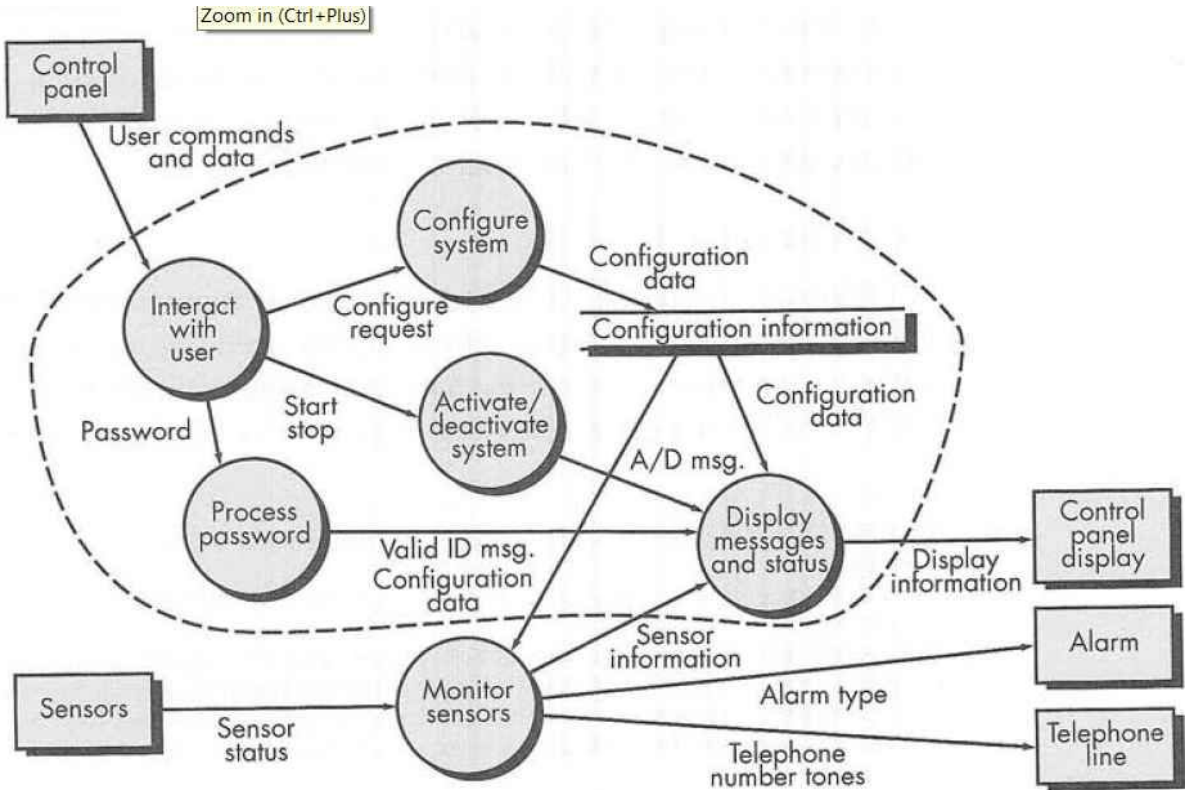


FIGURE 9.12

Level 2 DFD that refines the *monitor sensors* transform

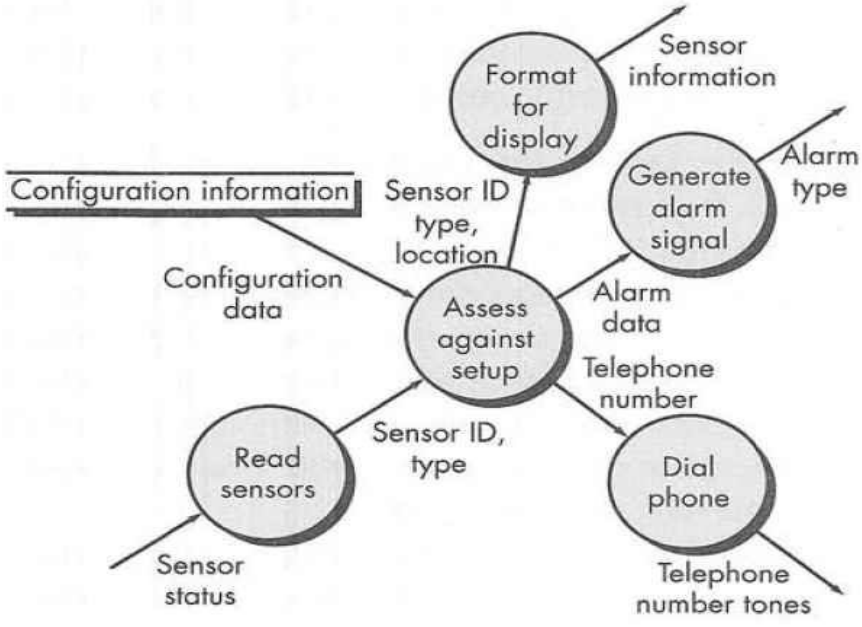


FIGURE 9.13 Level 3 DFD for *monitor sensors* with flow boundaries

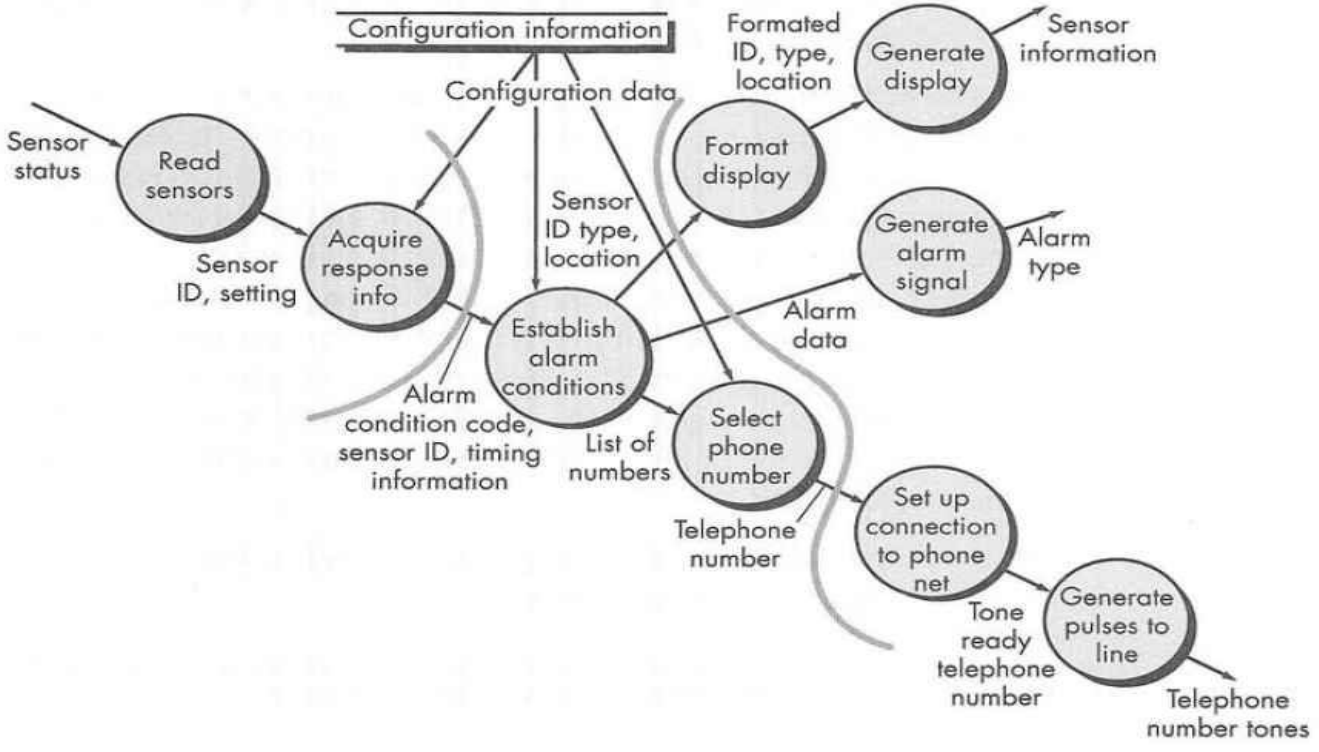


FIGURE 9.14

First-level factoring for *monitor sensors*

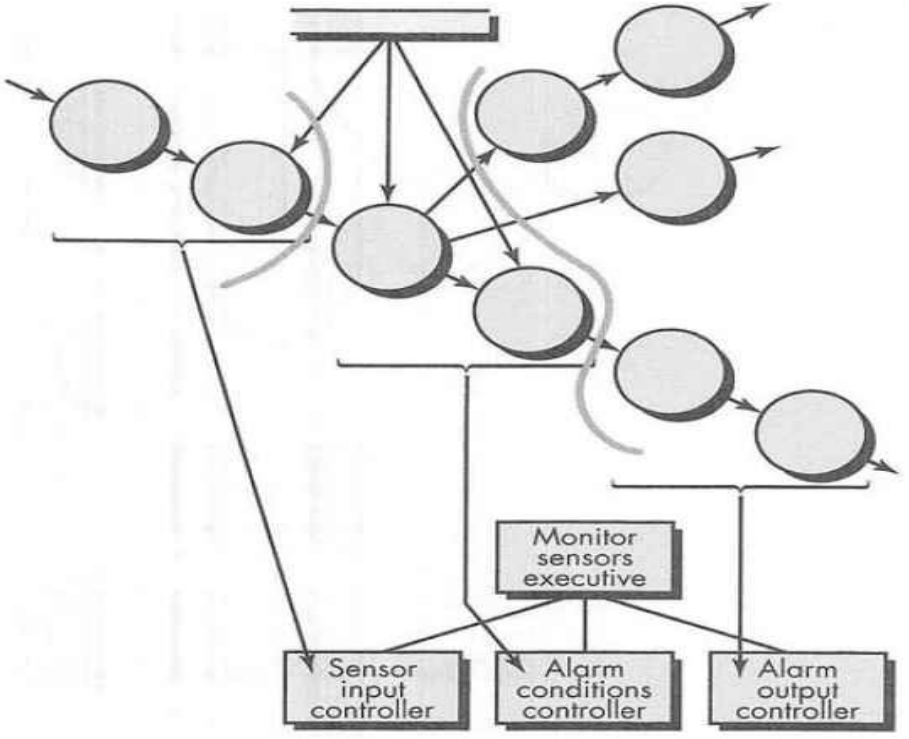


FIGURE 9.15

Second-level factoring for *monitor sensors*

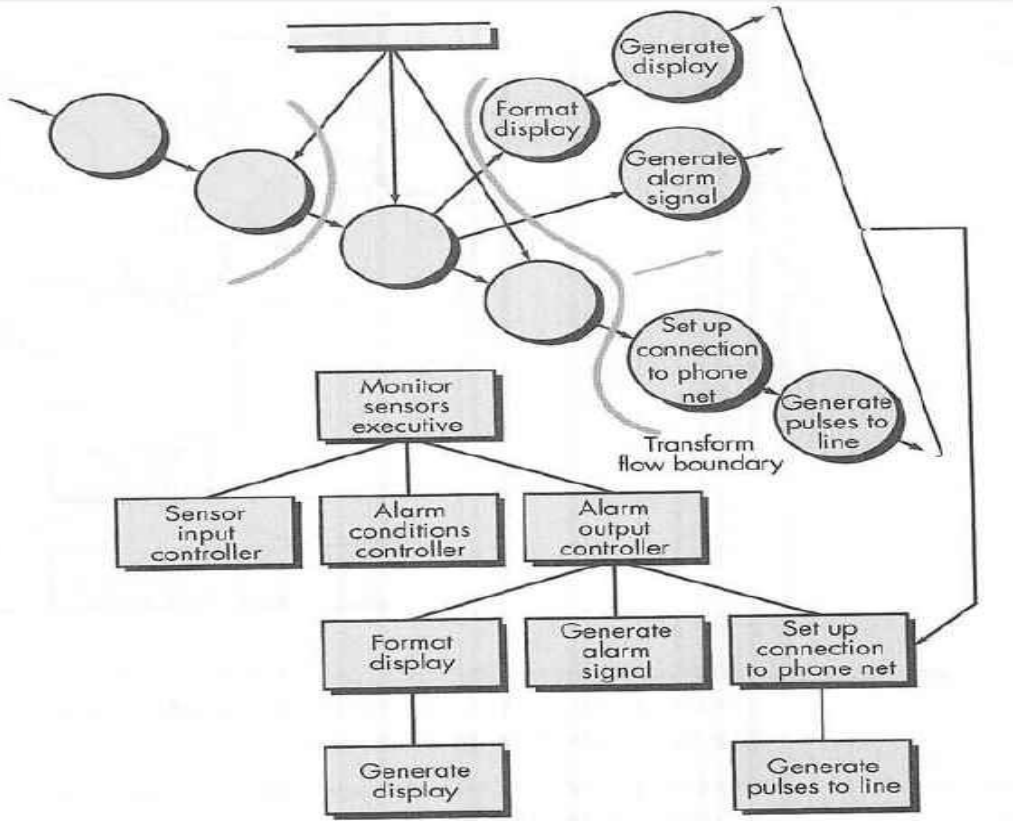


FIGURE 9.16

First-iteration structure for *monitor sensors*

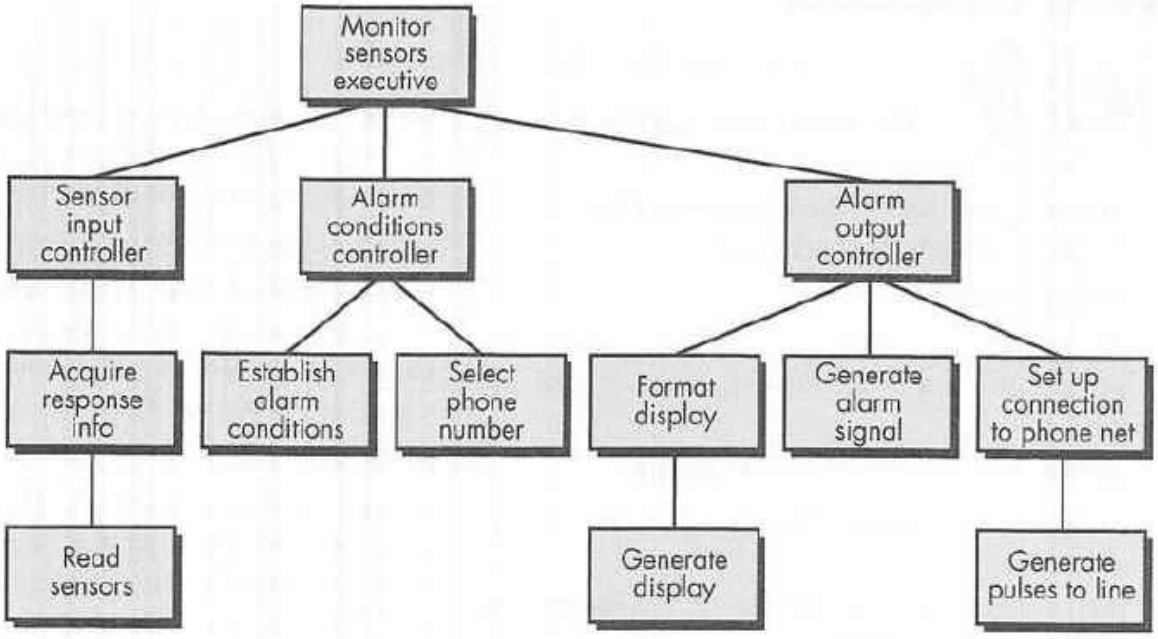
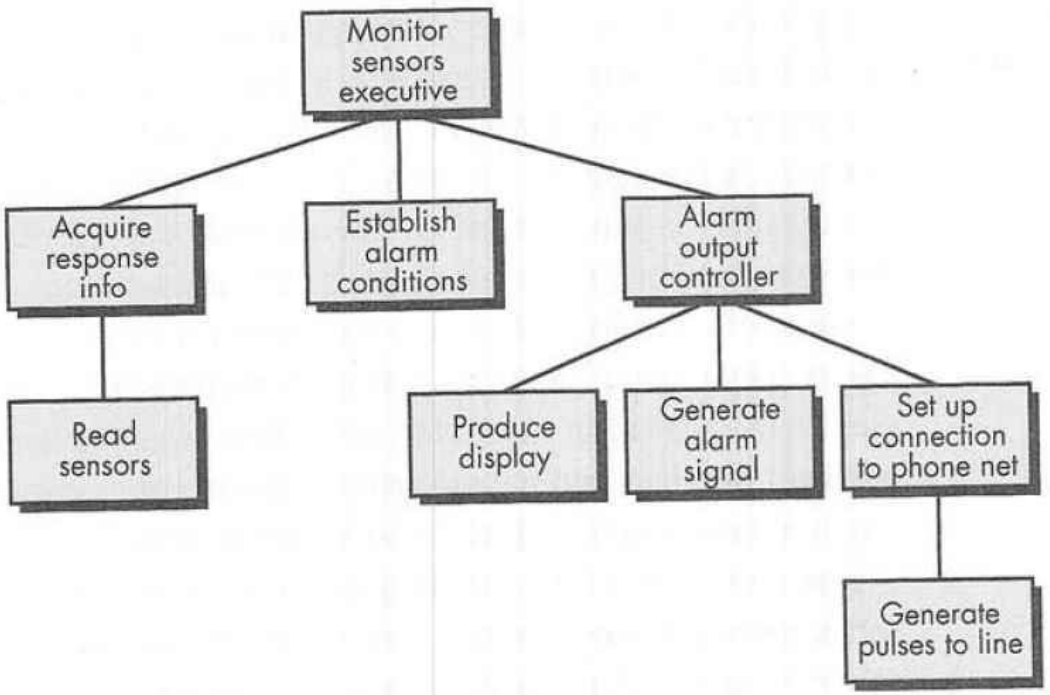


FIGURE 9.17

Refined program structure for *monitor sensors*



Pressman, Roger S.(2010). *Software Engineering : A Practitioner's Approach (7th ed)*. Boston: McGraw Hill Higher Education. Bab 9