

Evaluating Program Design

Evaluation Criteria: Modularity

- Coupling
- Cohesion
- Other Guidelines

Program Design

“Goal is to establish the internal structure of system in a way that minimizes cost of system over full life cycle.”

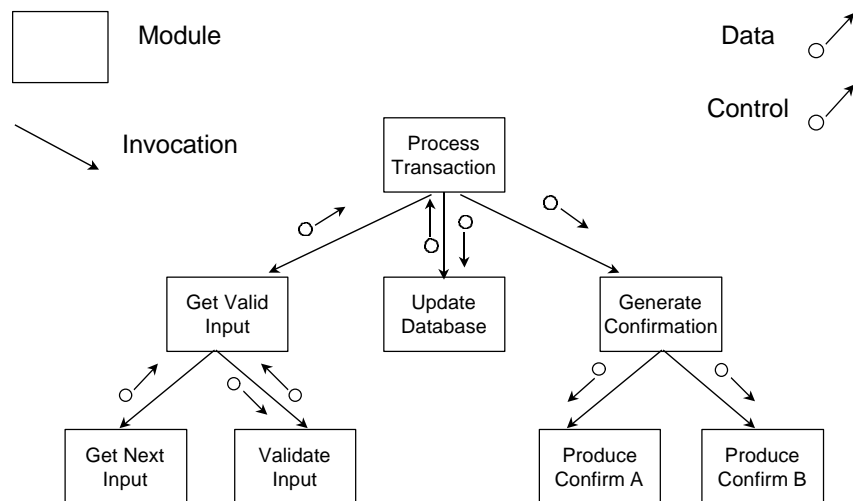
- modularity is the key
- program modules must:
 - perform a single function
 - be correctable individually
 - be small in size

Program Modules

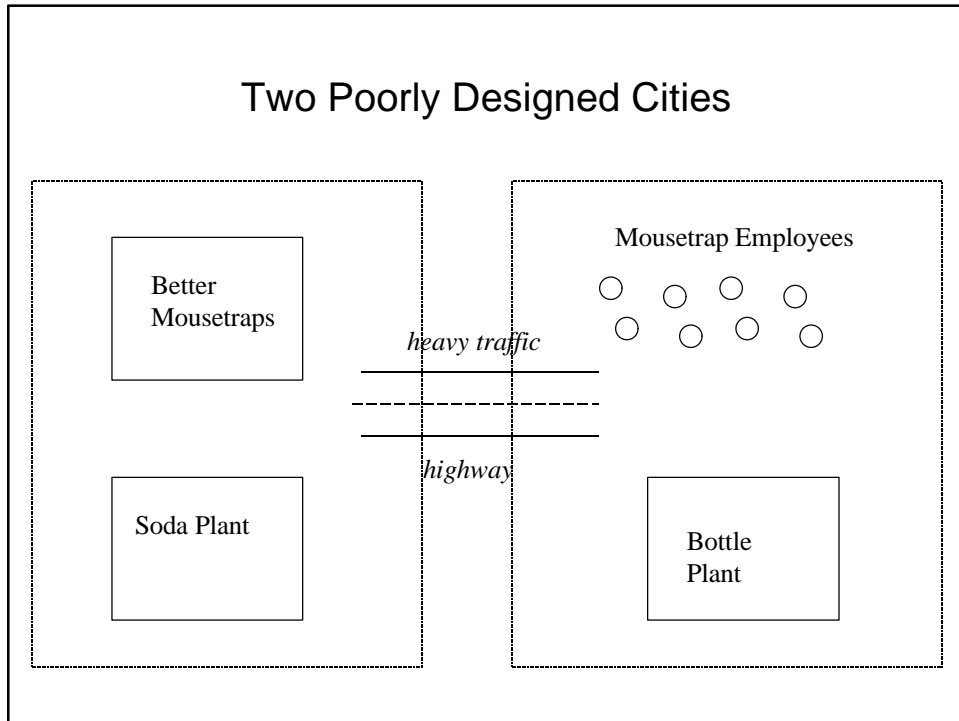
A named package of program instructions with single entry and single exit.

- examples
 - COBOL: program, section, paragraph
 - C: program, function
 - Pascal: procedure, function
 - FORTRAN: subroutine, function
 - ACCESS: query, form, report, macro, module

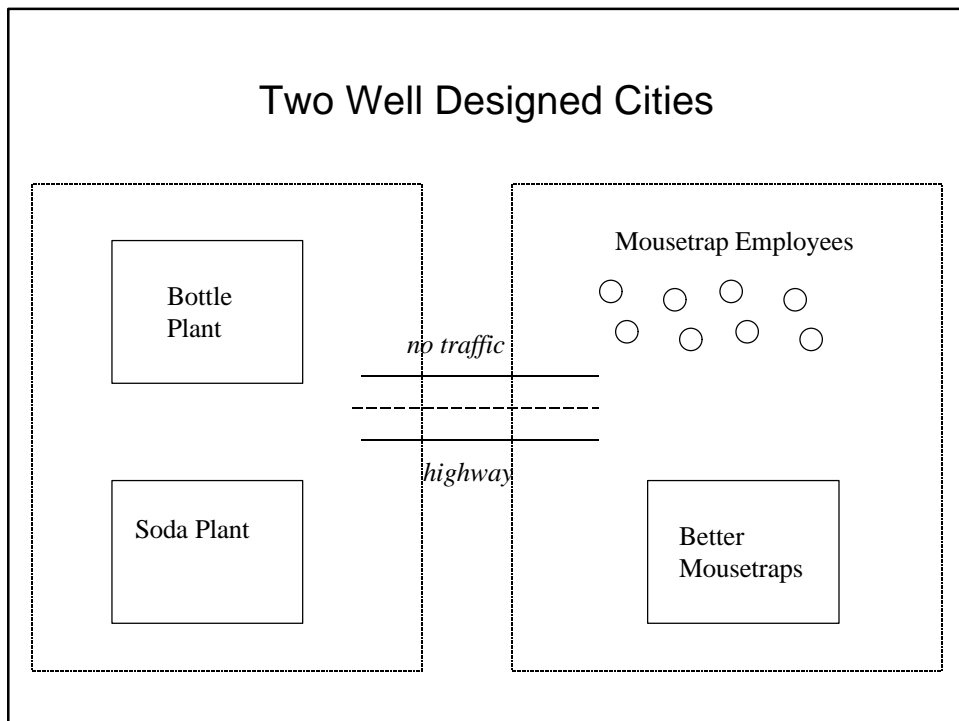
Program Design: Structure Chart Notation



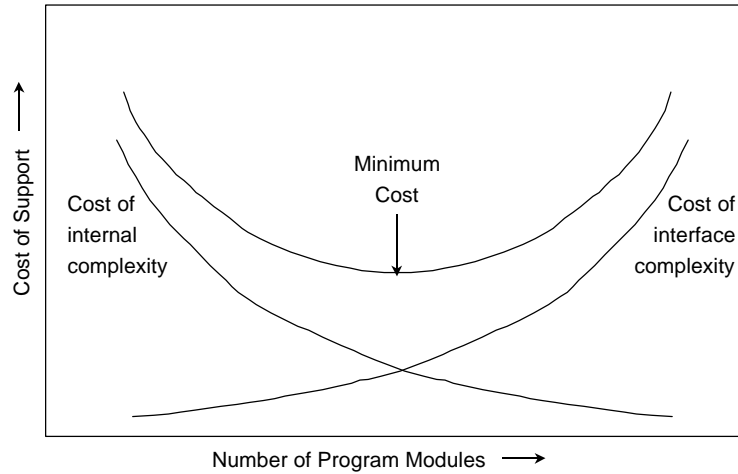
Two Poorly Designed Cities



Two Well Designed Cities



Balancing number and size of modules

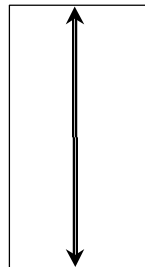


Program Design

Coupling measures the strength of connection between modules

- content (worst)
- common
- control ◦ →
- stamp ◦ →
- data (best) ◦ →

intertwined
spaghetti



well-defined
connections

Goal: Low/Loose Coupling

- simple connectivity among modules lead to easy to understand and less prone to ripple effects
- eliminates unnecessary connection
- reduce the number of necessary connection
- reduce dependence of necessary connection

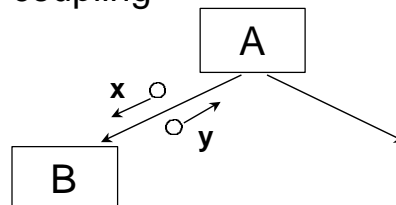
Principles of Connections among Modules : Coupling

- Narrow, not wide (reduce number of couple and flags)
- Direct, not indirect (no tramping, no bundling - more understandable to maintenance people)
- Local, not remote (no global variable)
- Obvious, not obscure (no implicit change in other modules, expect what they expect)
- Flexible, not rigid (reusable with minimum modification)

Normal Coupling

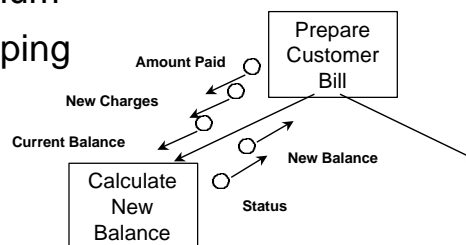
(considered good or OK)

- module A calls module B
- A tosses a piece of data as a parameter
- B returns to A when finished processing
- all info passed via parameters (x,y)
- data, stamp, control coupling



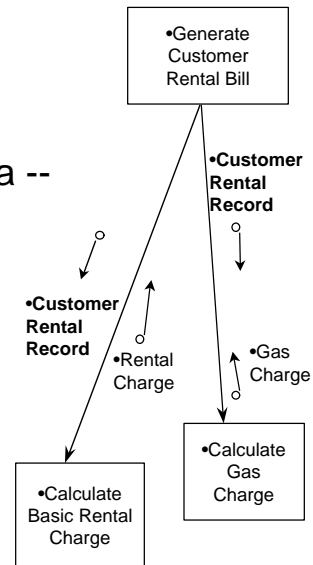
Data Coupling

- necessary operands and results
- flow up or down hierarchy
- necessary data communication
- keep it to minimum
- Beware of tramping

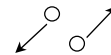


Stamp Coupling

- Passing composite piece of data -- data with meaningful internal structure
- Bit of indirectness involved - maintainer need to look into the data structure
- Good designers use stamp coupling well
- Warning
 - Beware of 'bundling' for nothing

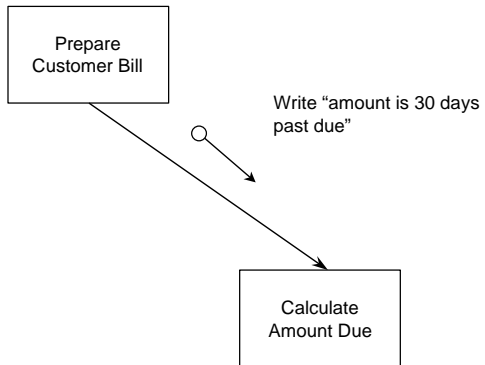


Control Coupling



- Passing a piece of information intended to control the internal logic of the other --decisions and instructions
- Indirect and obscure compared to data couple in which decisions are made within the module based on data passed

Control Coupling



It is OK as it is, but in general bad things follow

- beware when they flow down hierarchy

- beware when they flow long distances

Parameters

- Data couple : to inform: noun
- Descriptive flag: describing a piece of data
: adjective
: EOF, invalid
- Control flag : imperative: read next record
: possible design problem,
 - such as decision split, inversion of authority
- Warning
 - Beware of hybrid coupling, such as coding control information into data couple

Unnormal Coupling (bad)

- Common (global) coupling
 - all operands and results stored in common area
 - all are accessible to every module
 - critical problem for maintenance
 - any module can alter data in global area
 - global data referred to by name
 - introduces time remoteness

Another Bad One

- Content coupling
 - one module refers to data or statement contained inside another module
 - Ex: goto statement
 - critical problem for maintenance

Coupling through database?

- Is this common coupling?
- What are the difference?
 - Non-volatile
 - Well defined and, application-oriented objects with integrity rules
 - Validation schemes
 - Narrowness of database coupling
 - Documented
 -

What Type of Coupling?

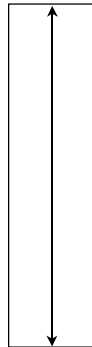
- Call PrintLabel Using Name, Street, City
- Call PrintLabel Using CustomerAddress
- Call PrintLabels

Program Design

Cohesion measures strength of association within a module

- functional (best)
- sequential
- communicational
- procedural
- temporal
- logical
- coincidental (worst)

single-minded



scatter-brained

Types of Cohesion

- Functional
- Sequential
- Communicational
- Procedural
- Temporal
- Logical
- Coincidental

Functional Cohesion (Best)

- All module elements (data and code) contribute to the execution of one and only one problem-related task
 - Examples
 - compute_cosine_of_angle, read_transaction_record
 - No time orientation
 - first, next, initialize
 - No extraneous access
 - only has access/calls to necessary data/functions
 - Often functional modules can be used in other designs without any modification
 - Easiest (and cheapest) to maintain

Sequential Cohesion

- A module whose activities are arranged such that the output of one serves as the input to the next
 - Example of “PREP_CAR_REPAINT” module
 - CLEAN CAR
 - FILL IN HOLES IN CAR
 - SAND CAR BODY
 - APPLY PRIMER
 - How are the steps related?
 - Serving one function? (what happens if PAINT CAR added at the end?)
 - Why worse than functional cohesion?
 - OK coupling but may not be reusable?

Communicational Cohesion

- All elements within a module use the same input or output data (nothing to do with sequence)
 - Example

```
module CUSTOMER_INFO uses customer_no
    find customer_name
    find customer_balance
    find customer_favorite_color
return customer_x...
```
 - OK Coupling but Reuse maybe problematic
 - Modules which don't want to know all of "customer_x.." will have to call CUSTOMER_INFO and throw away data

Procedural Cohesion

- A module whose elements are involved in different and possibly unrelated activities in which control flows from one activity to the next
- Example of "DOALL_GUEST_PREP" module
 - CLEAN DISHES
 - PREPARE DINNER
 - MAKE PHONE CALL
 - TAKE SHOWER
 - SET TABLE
- Related by execution order without explicit purpose

Temporal Cohesion

- A module whose elements are involved in activities related by time
 - Similar to procedural
 - collection of partial functions: hard to name
 - poor coupling: content or common
 - difference: order is more important in procedural
 - Example of “Finalize Evening” module
 - PUT OUT CAT
 - TURN OFF TV
 - BRUSH TEETH
 - Other examples
 - initialize, finalize

Logical Cohesion

- A module whose elements contribute to activities of the same general category; however, the specific activities are selected from outside the module via a control flag
- Example TRAVEL module
 - go by car
 - go by train
 - go by plane
 - Often, a programmer will attempt to optimize by combining loops and other code: Creates a maintenance problem

Coincidental Cohesion

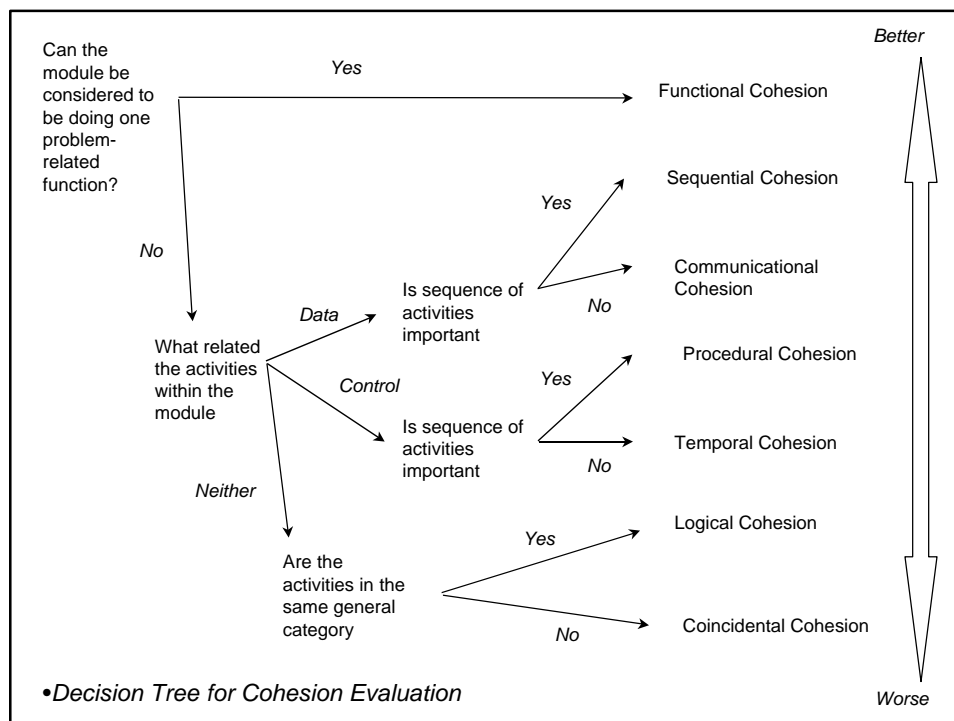
- A module whose elements contribute have no meaningful relationship to one another
 - Example
 - FIX CAR
 - BAKE CAKE
 - WALK DOG
 - GET OUT OF BED
 - GO TO MOVIES
 - After recognizes the usefulness of modules, a 20,000 line FORTRAN program was re-engineered to use modules
 - Every 75 lines of code was put into a subroutine
 - Similar to logical, but without even a broad category

Determining Cohesion by Name

- Functional:
 - verb-object: DEDUCT FEDERAL_TAXES
- Sequential:
 - assembly line: VALIDATE AND THEN UPDATE
- Communicational:
 - non-sequential : AVG & MAX SALARY
- Procedural:
 - flow chart names: DO ALL EDIT PREPARING
- Temporal:
 - time related: module for initialization of fields
- Logical:
 - general purpose: WASH ANY CAR
- Coincidental:
 - unknown meaning: MODULE-XX

Recapping Cohesion

- Sequential: toss data from the start till the end
- Communicational: share input but produce different output or vice versa
- Procedural: tossing control from proc to proc
- Temporal: end-of-day routine
- Logical: logical category, procs may have similarities as well as differences
- Coincidental: collection of unrelated functions



What Type of Cohesion?

- GENREPT
 - produce report: either a sales report, a project status report, or a customer transaction report
- STARTIT
 - open files, obtain first transaction, print page headings
- SYNCH
 - check syntactic correctness of vehicle guidance parameters
- Answers
 - logical, temporal, functional

Cohesion in Windows Interface

- Think of one interface as one module
 - Functional:
 - Sequential:
 - Communicational:
 - Procedural:
 - Temporal:
 - Logical:
 - Coincidental:

Additional Design Guidelines

Factoring
System Shape
Error Reporting
Fan In/Out

Summary of Guidelines

- Cohesion
 - Each module should be functional, sequential or communicational
- Coupling
 - Module coupling should be data or stamp
- Error Reporting
 - Have detection and reporting with same module
 - Use separate module with error codes and printing
- Factoring
 - Keep it high (i.e., partition into hierarchy)

Summary of Guidelines (cont.)

- Decision splitting
 - Keep recognition part of decision close to execution
- Fan In
 - Keep it high
- Fan Out
 - Restrict the number of subordinate to ≤ 7
- System Shape
 - Make it balanced (not input or output driven)

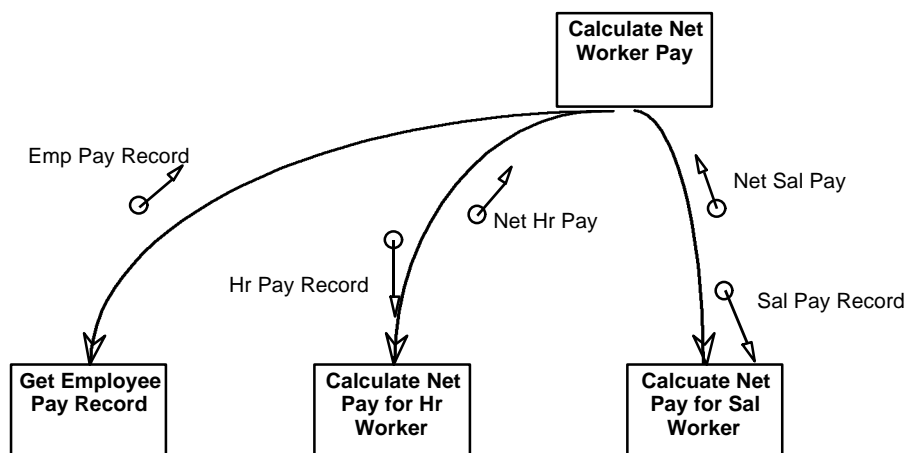
Summary of Guidelines (cont.)

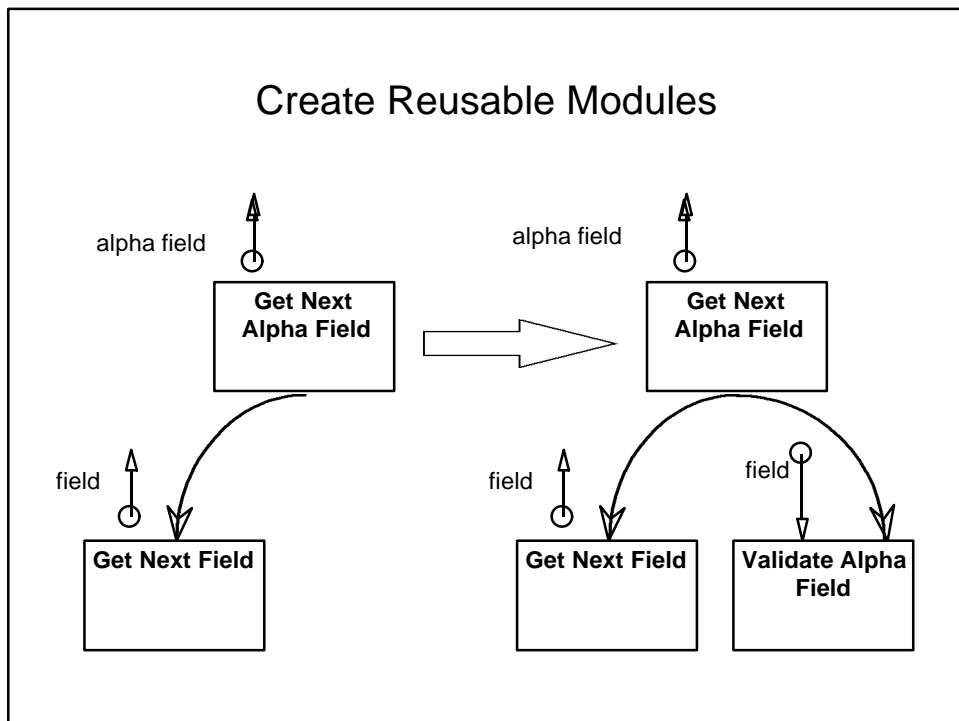
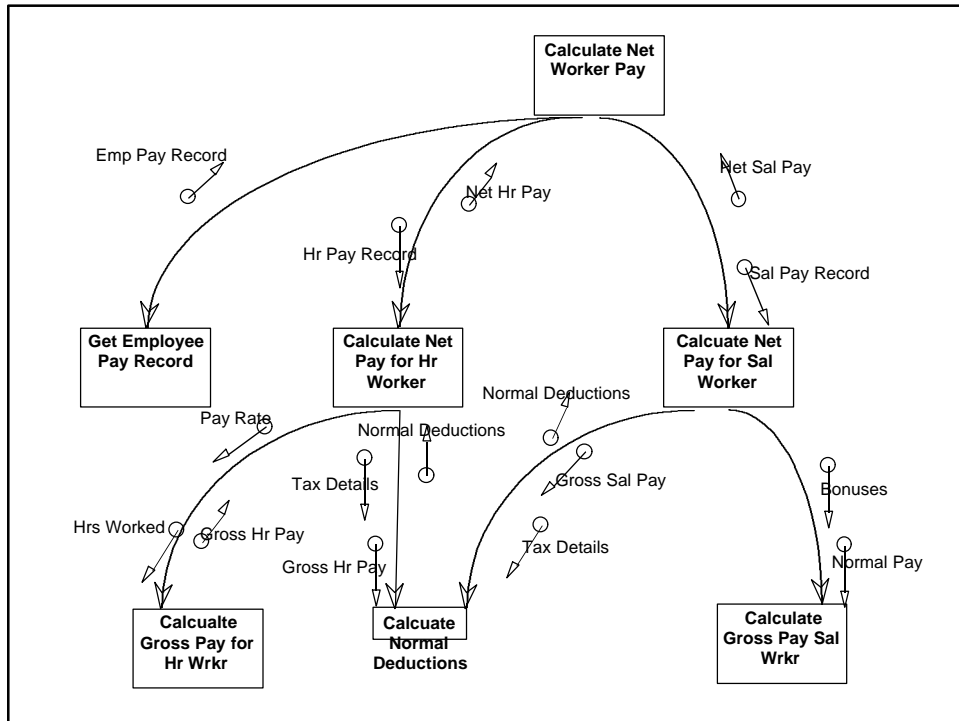
- Data Structure
 - Match program modules to incoming and outgoing data structures
- Editing
 - Edit in successive levels, with simplest editing being done at lowest levels
- Redundancy
 - Avoid it by factoring
- State Memory
 - Avoid it

Factoring

- ...is the separation of a function contained in one module, into a module of its own.
- Why factor?
 - Reduce module size
 - Stop factoring when well-defined cohesive functions can't be found within a module being considered
 - Simplify understanding
 - Prevent redundancy (by allow sharing, e.g., fan in)
 - Separate work (calculate) from management (deciding)
 - Create more useful modules (reusable)
 - Simplify implementation

Factor Calculate Net Pay?

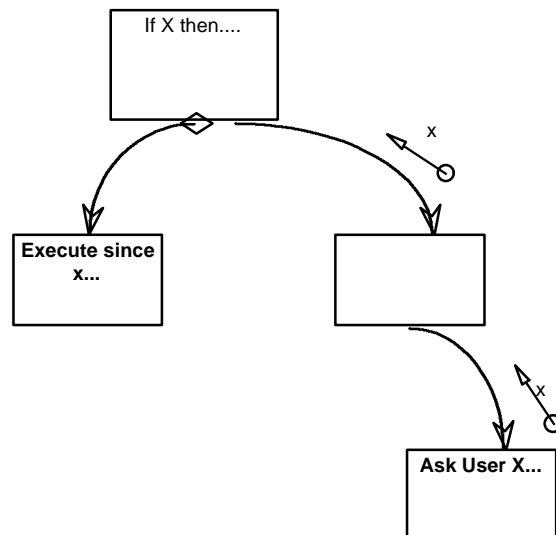




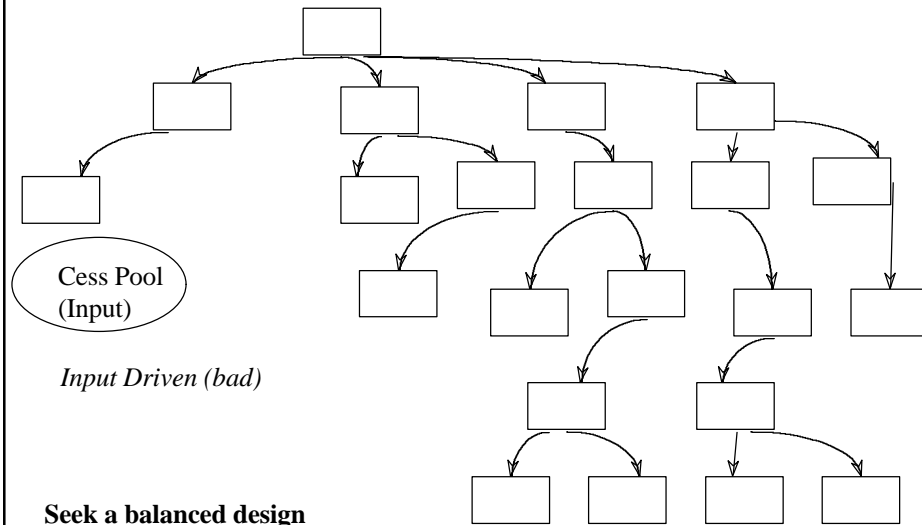
Decision Splitting

- A decision has two parts
 - recognition of what action to take
 - execution of that action
- Example
 - If customer account is not known
 - Then reject whole customer record
- Avoid separation (split) of recognition from execution

Decision Split Example



System Shape

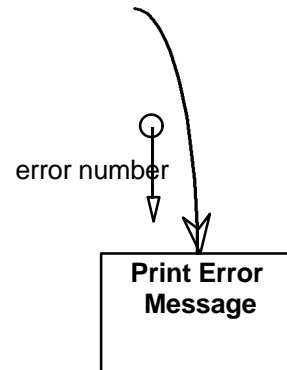


System Shape

- **Input Driven**
 - Coupling is usually poor
 - Many modules at top are concerned with physical format of input
 - Will not be reusable
- **Output Driven**
 - (not as common)
 - Many modules at top are concerned with physical format of input
- **Balanced**
 - Most modules are sheltered from Input and output formats

Error Reporting

- Report from module that
 - Detects error
 - Knows error message
- Separate an Error Module
 - Send in error code (can be enumerated type), output message
 - Easier to keep error messages similar



Fan In/Out

- In
 - Good, reusable modules
 - However, ensure good cohesion
- Out
 - Keep ≤ 7 subordinate modules

