Introduction

A. Much disagreement about benefits and disadvantages of hierarchical structures for computer software.

B. Many different things meant by “hierarchical structure”.

C. Nontrivial hierarchical structure always implies restrictions placed on the programmer.

   1. Restrictions may result in disciplined programming and a quality product.

   2. A given set of restrictions may not be appropriate for all situations. They may exclude good designs.

Definition of Structure

A. Division into parts

B. Relation between parts

C. Graphs can describe a structure

Definition of Hierarchical Structure

A. A structure with no loops in its relation's graph.

B. Before you know what someone means by a hierarchical structure, you must know the parts and the relation.

C. Hierarchies not necessarily trees.

The Uses Hierarchy

A. Parts: programs

Relation: uses

B. Definition of uses:

Given program A with specification S and program B, we say that A uses B if A cannot satisfy S unless B is present and functioning correctly.

3. Example: hardware for division uses power supply

but calls divide by 0 routine

C. Virtual-machine analogy.

D. Found in T.H.E., also in many examples of structured programming.

The "Gives Work" Hierarchy

A. Parts: processes

   Relation:give an assignment to

   Time: run time

B. Found in T.H.E.

C. Useful in guaranteeing termination and preventing deadlock; neither necessary nor sufficient.

D. In the T.H.E. system uses and gives work hierarchies coincide.

The Resource Allocation Hierarchy

A. Parts: processes

   Relation: "allocate a resource to" or "owns the resources of"

   Time: run time

B. Applicable with dynamic resource administration only.

C. "Allocate to" vs. "controls": The question of pre-emption.

D. Advantages
   1. Interference reduced or eliminated.
   2. Deadlock possibilities reduced.

E. Disadvantages
   1. Poor utilisation when load unbalanced.
   2. High overhead when resources are tight (especially with many levels).

The Courtois Hierarchy

A. Parts: operations

   Relation: takes more time and occurs less frequently than.

   Time: run time

B. Economics analogy.

C. T.H.E. comparison.

The Module Decomposition Hierarchy

A. Parts: modules

   Relation: part of

   Time: early design time

B. Never a loop in "part of" --module decomposition always a hierarchy.

The Created Hierarchy

A. Parts: processes

   Relation: created

   Time: run time

B. Must be a hierarchy (father is older than son).

C. Why a tree?--team work in creating progeny is accepted practice.

D. Sometimes implies unnecessary restrictions.

   e.g. Father cannot die until all progeny die.

Forcing different structures to coincide

may lead to an unrealistic design

Background on Extension and Contraction

A. Program families:

Different installations require different capabilities.

1. Systems with different capacities.

2. Systems with different run-time adaptability.

   Spectrum: ONE to FIXED to VARYING

3. Users who want to program vs. turnkey users.

Background on Extension and Contraction

B. Flexible systems: Easy to extend or subset.

1. Ability to remove access programs to make room for other access programs.

2. Fail-soft response to loss of capacity.

3. The difference between flexibility and generality.

Alternatives Available to the Software Producer

A. The supersystem: Generality costs!

B. A system for the "average" user: (who never exists).

C. A set of independently developed systems (with subtle differences).

D. A subsettable supersystem--each family member offers a subset of the services provided by the largest member.

   1. Individual installations only pay for what they need.
   2. Ability to extend by adding programs, without changing existing programs.
   3. Incremental implementation possible.
   4. Ability to contract by deleting whole programs not modifying programs.

Uses Hierarchy Reviewed

A. Parts: Programs, not modules.

B. Relation: "Requires the presence of".

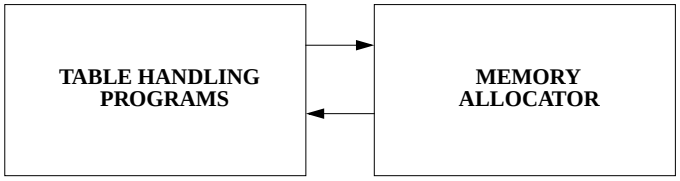
C. Difference between "uses" and "calls".

D. Why important

   1. Determine possible subsets.
   2. Determines possible fail-soft modes.
   3. Allows phased integration, testing, and delivery.

Design Error: Loops in Uses Relation

request and release
memory for tables



use tables to keep track
of memory assignments

Two dangers:

1. Memory allocator and table generator use each other
   - Neither works until both work
   - If either is removed, system no longer works

2. Memory allocator builds own tables
   - Code duplication

Basic Steps in the Design of a Subsettable System

A. Requirements definition: Identify the subsets first.

B. List programs belonging to each module.
   1. Access programs.
   2. Internal programs--cannot be used directly by programs outside the module.
   3. Main programs--cannot be used--top level in uses hierarchy.

C. For every pair of programs, three possibilities.
   1. A may use B
   2. B may use A
   3. Neither may use the other.

D. List programs at level 0: Programs that use no other programs.

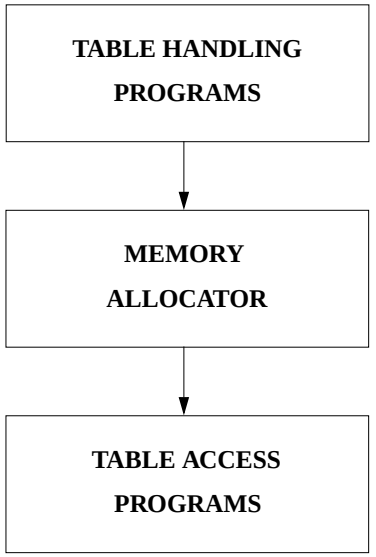
E. Try to work up from there.
   1. Level l programs use only level 0 programs
   2. Level 2 programs use only level 0 or level 1 programs, etc.

Four Conditions for Allowing Program A to Use Program B

1. A is simpler because it uses B.
2. B is not more complex because it is not allowed to use A.
3. There is a useful subset containing B and not A.
4. There are no useful subsets containing A and not B.

Conflict Removal: Sandwiching

If A and B use each other, can one be split into two parts, i.e., a simple version and a complex version?



MESSAGE: A LEVEL IS NOT A MODULE
These are not "layers of abstraction"

Another Example: T.H.E. Conflict

Should synchronisation use memory allocation?

Shouldn't memory allocation use synchronisation?

Another Example: Multics Conflict

Virtual Memory should use file system

File System should use virtual memory

Each Level is a Virtual Machine

A. Definition: A set of variables and operations, implemented in software.

B. Applications programs are simpler because they use virtual machine programs.

C. Resources used to implement a virtual machine not available to a program that uses the virtual machine.

D. Upper level machines are LESS POWERFUL than lower level machines.

E. Upper level machines are more convenient and safer than lover level.

Deriving Subsets from the Uses Relation

A. Any level is a subset.

B. Can also omit parts of levels.

Evaluation Criteria for Uses Hierarchy

A. Simple.

B. Avoid duplication or almost alike programs.

C. All desirable subsets.

   1. Without the subset constraints, anything will work.

Principle of Minimal Steps

A. Example: synchronisation and message passing.

B. Example: parameter passing and run-time type-checking.

Relation to Courtois Hierarchy

- Lower = faster, more frequent

- Higher = slower, less frequent

- Exceptions

The ONE, FIXED, VARIABLE Conflict

The super system allows things to be created and deleted.

A subset eliminates those programs but allows a fixed number of things.

A smaller subset has only one of those things.

Hierarchy as the Solution to the “uses” Dilemma

Compromise between desire to avoid duplication and independence.