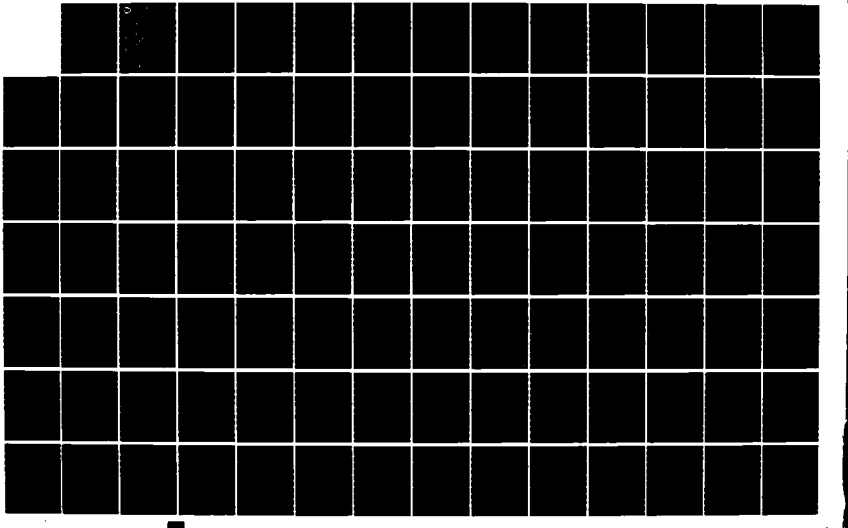
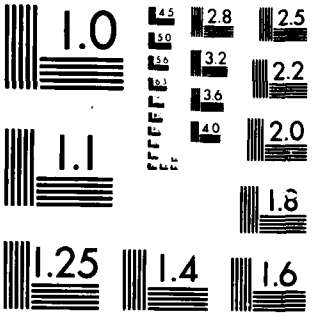
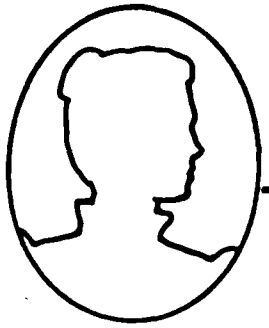
AD-A165 299 ADA (TRADEMARK) TRAINING CURRICULUM SOFTWARE ENGINEERING METHODOLOGIES M201 TEACHER'S WORKBOOK(U) SOFTECH INC WALTHAM MA 1986 DAAB07-83-C-K506 1/2

UNCLASSIFIED F/G 5/9 NL







Ada® Training Curriculum

1986

Software Engineering Methodologies
M201
Teacher's Workbook

Superseded AD-A-144 240

U.S. Army Communications-Electronics Command
(CECOM)

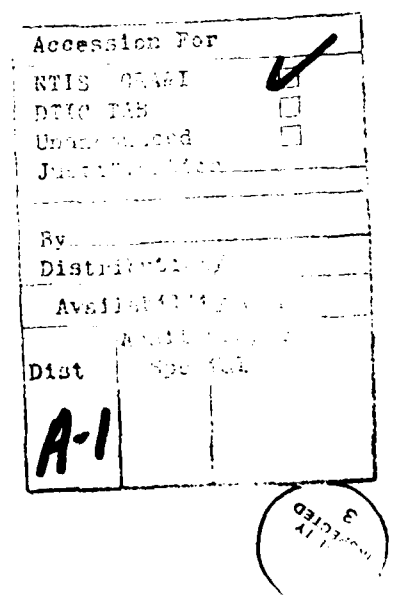
Contract DAAB07-83-C-K506

Prepared By:

SOFTECH, INC.
460 Totten Pond Road
Waltham, MA 02154

Section 1

SADT EXERCISE



INSTRUCTOR NOTES

THIS IS THE START OF THE EXERCISES. THEY ARE PROVIDED TO ALLOW THE STUDENTS TO GET A FEEL FOR SOME OF THE DIFFERENT METHODOLOGIES THEY ARE GOING TO STUDY. EMPHASIZE THAT THE INTENT IS NOT TO MAKE THEM EXPERTS IN ONE LESSON. FURTHERMORE, REMIND THEM THAT A METHODOLOGY MAY SEEM DIFFICULT TO USE AT FIRST, AND NOT TO TURN AWAY FROM IT BASED ON ONE EXPERIENCE. PROVIDE THE TIME AND LEVEL OF DETAIL NECESSARY TO GET THE GENERAL IDEAS ACROSS.

SADT OVERVIEW

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

GO OVER SOME OF THE GENERAL CONCEPTS OF SADT:

VERBS:	FUNCTIONS OR ACTIVITIES
NOUNS:	DATA ON ARROWS/CONNECTIONS
CONTROL DATA:	AFFECTS FUNCTIONALITY BUT DATA NOT TRANSFORMED
INPUT:	DATA THAT IS TRANSFORMED
FUNCTIONS:	A CONTINUOUS PROCESS: A ONE-TIME OCCURRENCE; A SERIES OF DISCRETE ACTIONS; A SET OF SIMILAR ACTIONS OCCURRING ASYNCHRONOUSLY; RELATED BUT DISSIMILAR ACTIONS.
DATA:	A CONTINUOUSLY CHANGING VARIABLE; A (SERIES OF) DISCRETE OBJECTS; VALUES OF A VARIABLE; A SET OF SIMILAR OBJECTS OR VARIABLES CHANGING ASYNCHRONOUSLY; A SET OF RELATED BUT DISSIMILAR OBJECTS OR VARIABLES.

PROBLEM

PURPOSE:

DESCRIBE, IN SADT FORM, THE WAY AN AMERICAN FAMILY FEEDS ITSELF.

VIEWPOINT:

THE PARENTS.

CONTEXT:

FOCUS ON THE "AT HOME" ACTIVITIES.

INSTRUCTOR NOTES

EXPLAIN TO THE CLASS HOW TO GET STARTED. REMIND THEM THAT AS THE DIAGRAMMING TAKES PLACE, ONE CONTINUALLY ITERATES-CHANGING THE DATA, FUNCTIONS, MAYBE EVEN VIEWPOINT, THE POINT IS TO PUT SOMETHING DOWN, DON'T SPEND ALOT OF TIME DOING IT, AND PASS THE RESULTS AROUND FOR REVIEW.

METHOD

1. DRAW TOP DIAGRAM.

   a. LIST DATA.

   b. LIST ACTIVITIES.

   c. DRAW DIAGRAM.

2. DRAW SUMMARY DIAGRAM. (REVISE TOP DIAGRAM IF NEEDED.)

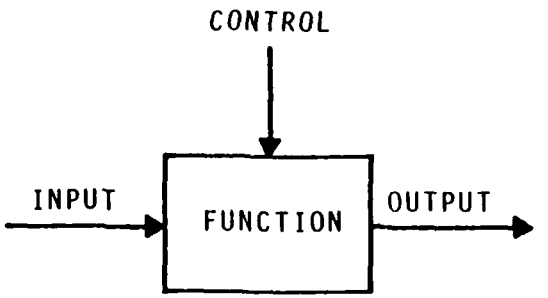
3. DECOMPOSE A BOX ON TOP DIAGRAM. (REVISE TOP DIAGRAM IF NEEDED.)

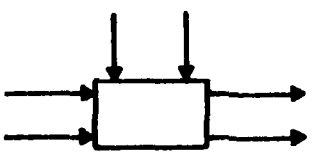
INSTRUCTOR NOTES

REVIEW HOW THE BOXES AND ARROWS ARE CONNECTED AND WHAT THEY MEAN.

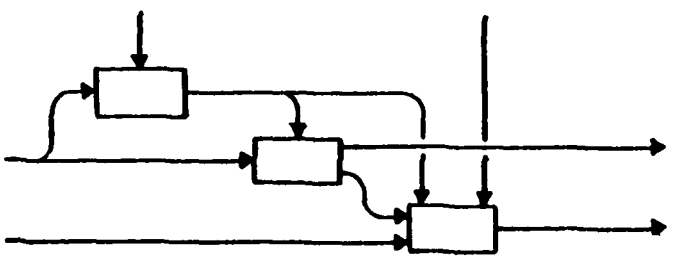
ACTIVITY

DATA





SUMMARY DIAGRAM IS ONE BOX WITH SEVERAL ARROWS



OTHER DIAGRAMS HAVE SEVERAL BOXES AND ARROWS. THE NUMBER OF EXTERNAL ARROWS OF THE CHILD MUST EXACTLY MATCH THE ARROWS OF ITS PARENT BOX.

INSTRUCTOR NOTES

HERE ARE SOME DATA AND ACTIVITIES IF THE CLASS IS HAVING TROUBLE:

DATA	ACTIVITIES
MENU	SHOP
SCHEDULE	PLAN
MONEY	COOK
BUDGET	PREPARE
FAMILY POLICY	CLEAN UP
COOKED FOOD	SERVE
DIRTY DISHES	WASH
TABLE	DRY
SHOPPING LIST	SET TABLE
PREFERENCES	CHOP
GARBAGE	MAINTAIN PANTRY

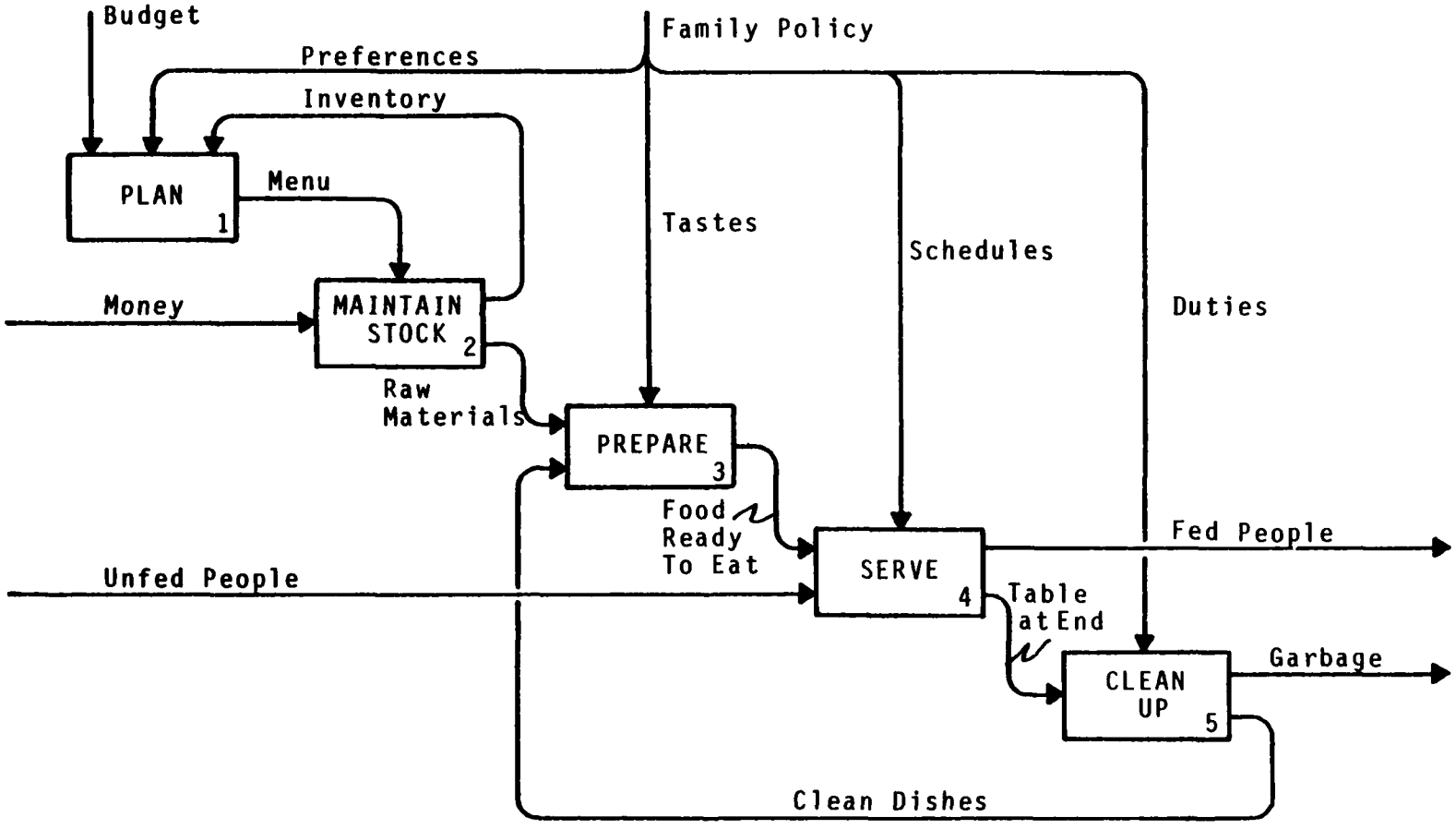
DATA AND ACTIVITIES

DATA LIST:

ACTIVITY LIST:

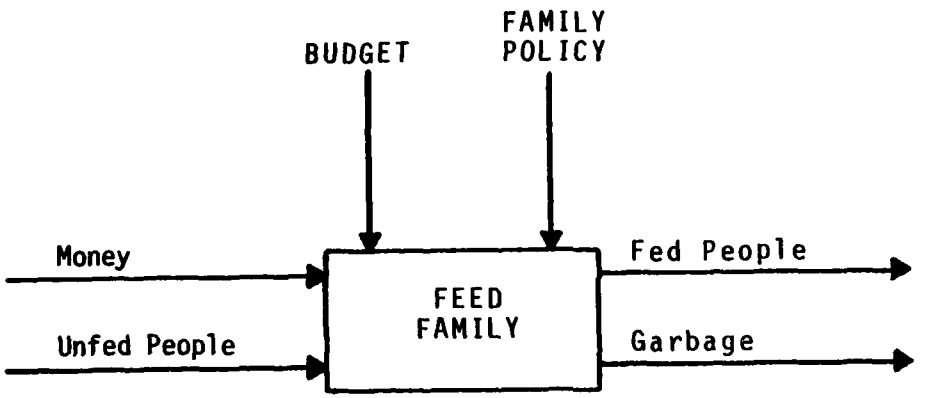
INSTRUCTOR NOTES

HERE'S THE KIND OF TOP DIAGRAM YOU SHOULD COME UP WITH:



TOP DIAGRAM

HERE'S THE KIND OF SUMMARY DIAGRAM YOU SHOULD COME UP WITH:



NOTE:

YOU'LL REVISE THE TOP DIAGRAM IN THE PROCESS OF MAKING THE SUMMARY DIAGRAM.

SUMMARY DIAGRAM

INSTRUCTOR NOTES

NOW PICK A BOX ON THE TOP DIAGRAM AND DECOMPOSE IT. GO THROUGH DATA AND ACTIVITY LISTS BEFORE DRAWING THE DIAGRAM. TRY TO POINT OUT THE PRINCIPLES THAT DECOMPOSITION CHANGES THE NEXT HIGHER LEVEL, BECAUSE IT CREATES A BETTER UNDERSTANDING OF A PART OF THE SYSTEM. TRY TO LEAD CLASS INTO CHANGING THE INTERFACES (ARROWS) BETWEEN THE MAJOR SUBSYSTEMS (BOXES) ON THE TOP DIAGRAM, AS A RESULT OF THIS DECOMPOSITION. THIS WILL MAKE A GOOD EXAMPLE OF HOW INTERFACES CHANGE DURING SYSTEM ARCHITECTURE DEVELOPMENT. DON'T BE AFRAID TO REDRAW THE TOP DIAGRAM BASED ON THIS EXPERIENCE.

(YOU MAY NOT GET TO THIS. IF NOT, JUST SKETCH THE DIAGRAM FOR THEM.)

1ST LEVEL DIAGRAM

INSTRUCTOR NOTES

REVIEW IMPORTANT ASPECTS OF SADT.

THE REVIEW QUESTIONS ARE ORIENTED TOWARD GETTING THE CLASS TO CRITICALLY LOOK AT THE SADT DIAGRAMS AND SEE WHAT IT'S REAL POWER IS.

SOME POSSIBLE ANSWERS:

1. *QUESTIONS ABOUT HOW DATA IS TRANSFORMED, WHAT CONTROLS THE TRANSFORMATION, THE* DEPENDENCIES BETWEEN VARIOUS DATA AND FUNCTIONS.

2. HOW OFTEN A FUNCTION OCCURS, THE ACCURACY OF A COMPUTATION, WHAT IS AUTOMATED/WHAT IS NOT AUTOMATED.

SUMMARY AND REVIEW QUESTIONS

- HELPS UNDERSTANDNG OF THE SYSTEM
- EMPHASIZES DECOMPOSITION
- CONCENTRATES ON FUNCTIONALITY AND DATA
- TOP-DOWN ANALYSIS
- WHAT CLASS OF QUESTIONS CAN BE ASKED AND ANSWERED USING THE DIAGRAMS YOU HAVE GENERATED? WHAT QUESTIONS CAN'T BE ANSWERED?

INSTRUCTOR NOTES

Section 2

ENTITY AND BACHMAN DIAGRAMMING EXERCISE

INSTRUCTOR NOTES

ENTITY DIAGRAMMING FOCUS IS ON THE INFORMATION OF A SYSTEM, AS OPPOSED TO THE FUNCTIONS, AS WE SAW IN THE SADT EXERCISE. TO MAKE THE INFORMATION GAINED MORE USABLE FOR HANDLING BY A COMPUTER (E.G., THE DESIGN OF THE DATABASE) WE APPLY THE BACHMAN TECHNIQUE TO OUR ENTITY DIAGRAMS. THIS EXAMPLE WILL PROCEED IN THE SAME LIGHT. WE WILL FIRST CONSTRUCT AN ENTITY DIAGRAM OF A PROBLEM, THEN USE THAT RESULT TO CONSTRUCT A BACHMAN DIAGRAM.

ENTITY AND BACHMAN DIAGRAMMING OVERVIEW

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

HAVE THE STUDENTS READ OVER THE PROBLEM TO THEMSELVES. REMIND THEM TO LOOK FOR THINGS THAT MIGHT BE ENTITIES, AND THEIR RELATIONSHIPS. ALSO REMIND THEM THAT NOUNS FORM THE ENTITIES AND VERBS THEIR RELATIONS.

PROBLEM

A SMALL COMMUNITY HAS ESTABLISHED A LIBRARY:

- FOR THE USE OF LOCAL RESIDENTS
- CONTAINING ONLY BOOKS

THE BOOKS ARE:

- CLASSIFIED BY CALL NUMBER
- SHELVED SEQUENTIALLY BY CALL NUMBERS
- CATALOGED BY AUTHORS - LISTING AUTHORS AND THE SPECIFIC BOOKS WRITTEN BY THOSE AUTHORS
- CATALOGED BY SUBJECTS - REFERENCING BOOKS WRITTEN ON THAT SUBJECT
- LOANED (FOR TWO WEEKS) ONLY TO LIBRARY CARD HOLDERS

INSTRUCTOR NOTES

TELL THE STUDENTS NOT TO FORGET THAT CLASSES OF ENTITIES AND RELATIONS ARE DEVELOPED AFTER IDENTIFYING THE LOWEST LEVEL OBJECTS.

PROBLEM (CONTINUED)

THE LIBRARY CARDS:

- ARE ISSUED TO RESIDENTS WHO FURNISH THEIR NAME, ADDRESS AND PHONE NUMBER TO THE LIBRARY
- EXPIRE TWO YEARS AFTER THEY ARE ISSUED

THE LIBRARIANS MAINTAIN AND HAVE ACCESS TO A THIRD CATALOG WHICH LISTS ALL CALL NUMBERS FOR ALL THE BOOKS IN THE LIBRARY.

INSTRUCTOR NOTES

GO OVER THE STEPS FOR DOING ENTITY DIAGRAMMING. REVIEW THE DEFINITION OF AN ENTITY (A TANGIBLE OBJECT OR ABSTRACT CONCEPT), CLASSES (PROPERTIES ENTITIES POSSESS WHICH CAN BE USED TO FORM GROUPS), AND RELATIONS (AN ASSOCIATION BETWEEN TWO ENTITIES).

METHOD

1. IDENTIFY ENTITIES.

2. GROUP ENTITIES INTO ENTITY CLASSES.

3. IDENTIFY RELATIONS.

4. GROUP RELATIONS INTO RELATION CLASSES.

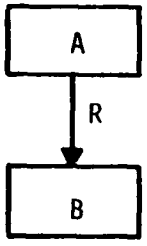
5. DRAW DIAGRAM.

INSTRUCTOR NOTES

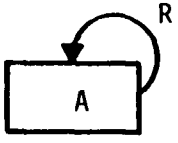
REVIEW THE SYNTAX THAT WILL BE USED TO DRAW THE DIAGRAM.

ENTITY CLASS

RELATION CLASS



A IS RELATED TO B THROUGH R.



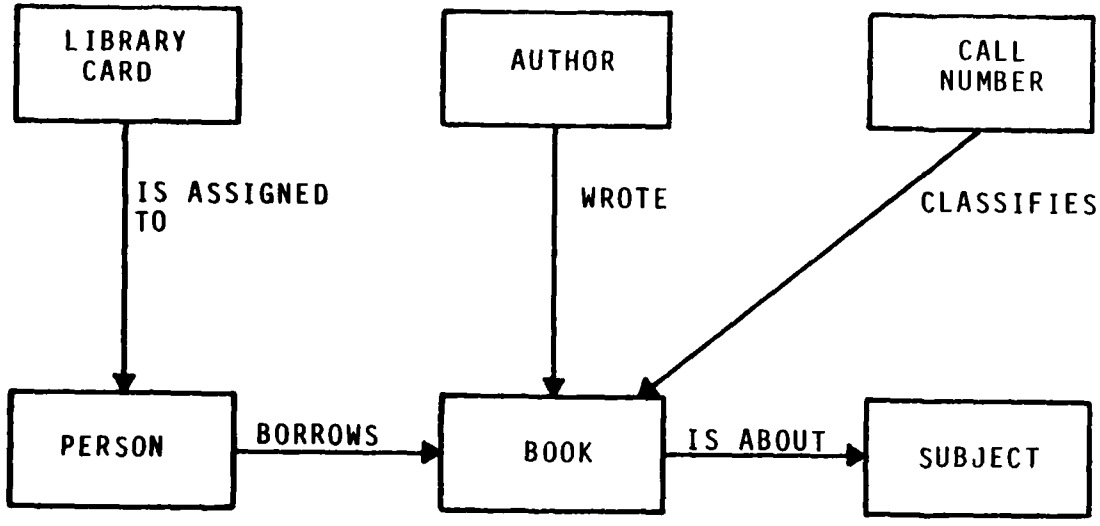
A IS RELATED TO ITSELF THROUGH R.

ENTITY CLASS = COLLECTION OF ENTITIES WITH SIMILAR PROPERTIES

RELATION CLASS = COLLECTION OF SIMILAR RELATIONS BETWEEN ENTITIES

INSTRUCTOR NOTES

NOW, GO BACK AND READ ALOUD THE PROBLEM. YOU SHOULD UNDERLINE NOUNS, FORMING THE ENTITIES. THEN DO THE SAME WITH THE VERBS, FORMING RELATIONS. THEN ASK THE CLASS TO DRAW THE ENTITY DIAGRAM.



CREATING AN ENTITY DIAGRAM

DRAW THE ENTITY DIAGRAM FOR THE LIBRARY PROBLEM:

INSTRUCTOR NOTES

AFTER SUCCESSFULLY GETTING THE CLASS TO DRAW THE CORRECT ENTITY DIAGRAM, (AND MAKING SURE EVERYONE USES THE SAME SOLUTION AS IS GIVEN), WE NEXT MOVE TO BACHMAN DIAGRAMMING.

BACHMAN DIAGRAMMING EXERCISE

(PART OF EXERCISE 3)

INSTRUCTOR NOTES

GO OVER THE STEPS OF THE BACHMAN METHOD. REMIND THE CLASS THAT THE KEY IS TO ASSESS THE RELATION CLASS RATIOS, ALLOW ONLY (1:1) and (1:N) RELATION CLASS RATIO'S, AND CONVERTING THE OTHERS TO THESE TYPES. ASK THE CLASS WHY? (BECAUSE THEY ARE HARD TO IMPLEMENT, AND THESE DIAGRAMS ARE TO HELP WITH IMPLEMENTING THE ENTITY DIAGRAMS INTO COMPUTER SYSTEMS).

METHOD

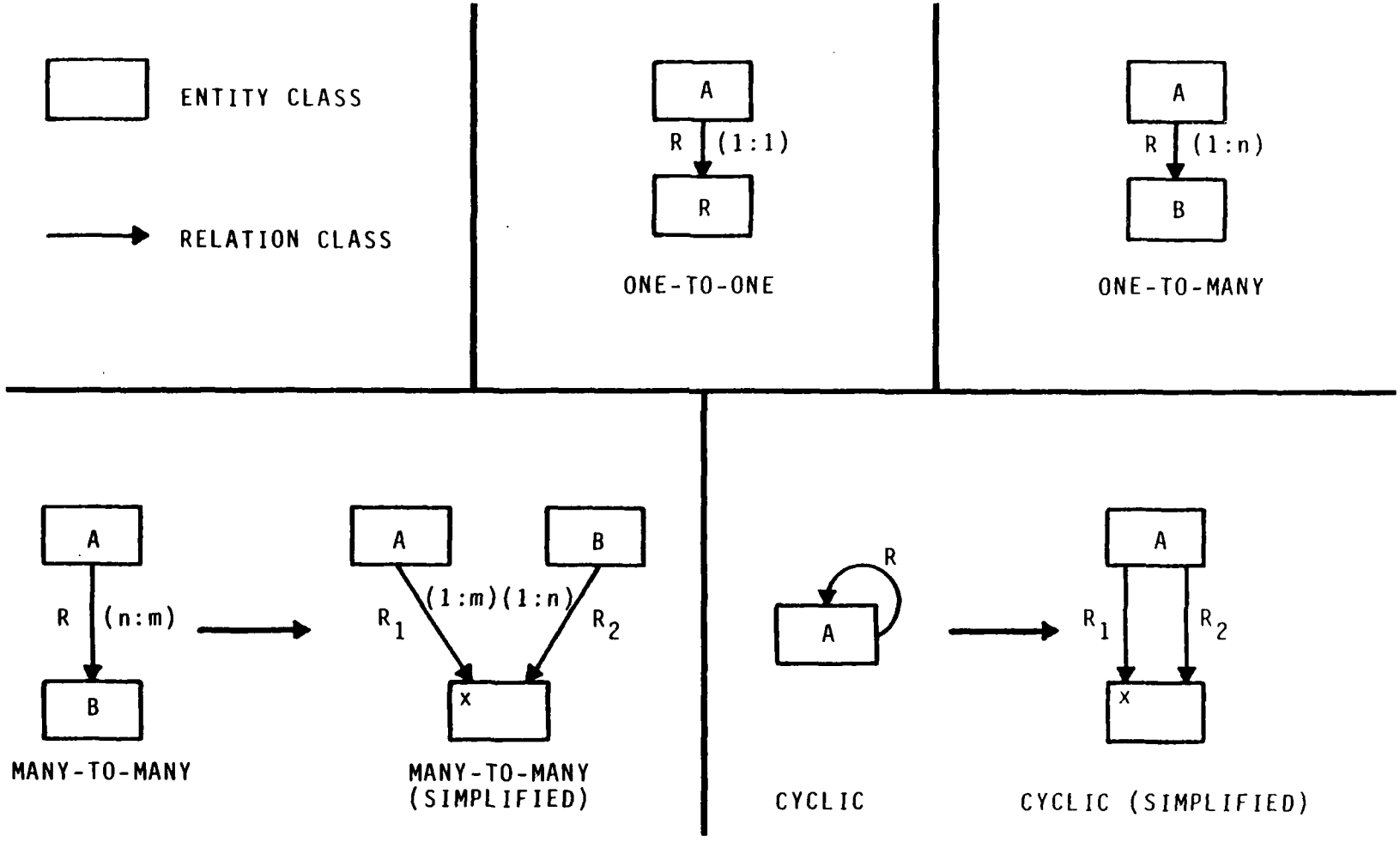
1. LABEL RATIOS.

2. DECIDE HOW TO CONVERT TO BACHMAN.

3. REDRAW IF NECESSARY.

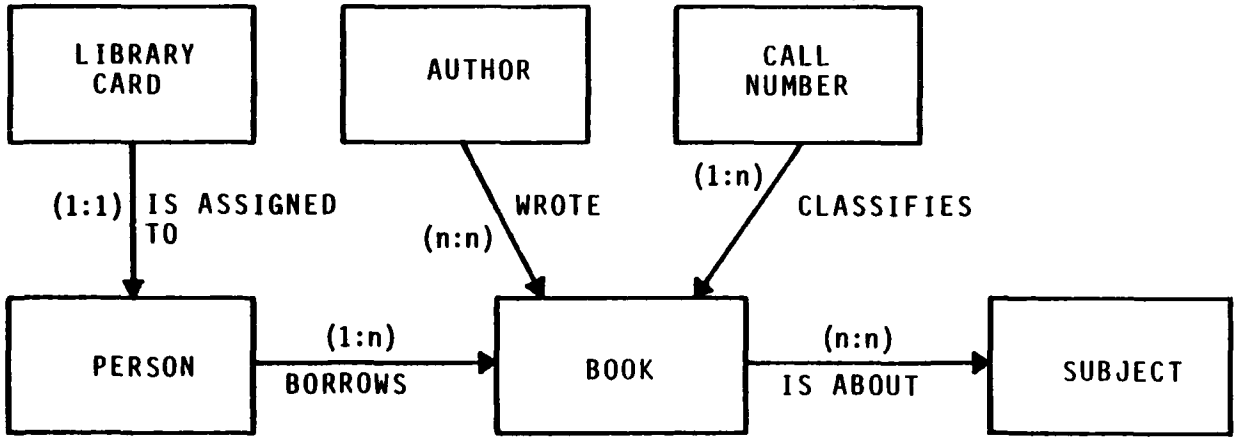
INSTRUCTOR NOTES

REVIEW THE SYNTAX OF THE BACHMAN DIAGRAMS AND RATIOS.


ENTITY CLASS
RELATION CLASS
A
R
(1:1)
R
ONE-TO-ONE
A
R
(1:n)
B
ONE-TO-MANY
A
R
(n:m)
B
MANY-TO-MANY
A
B
R1
(1:m)(1:n)
R2
x
MANY-TO-MANY
(SIMPLIFIED)
R
A
CYCLIC
A
R1
R2
x
CYCLIC (SIMPLIFIED)

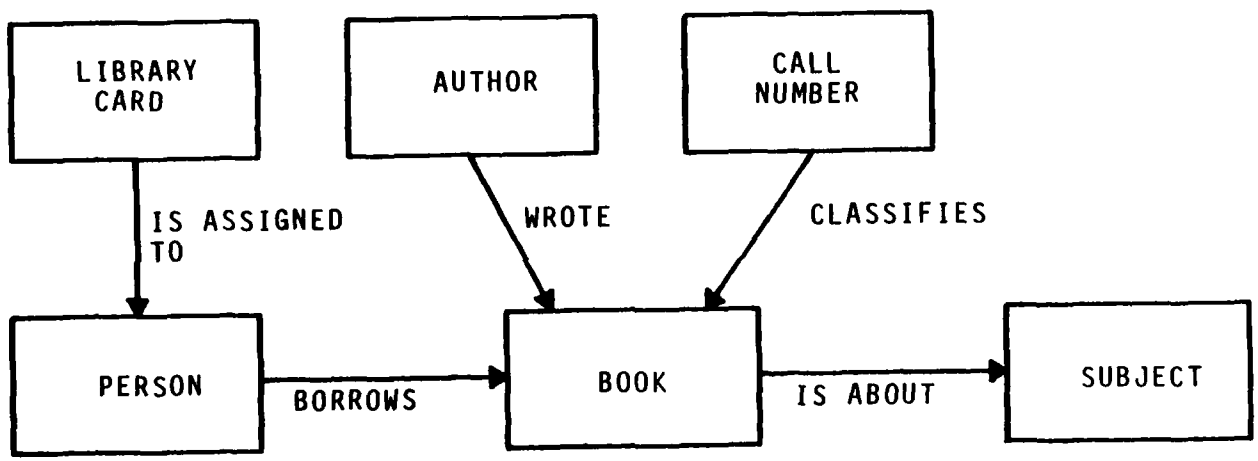
INSTRUCTOR NOTES

FIRST, LABEL THE RATIOS:

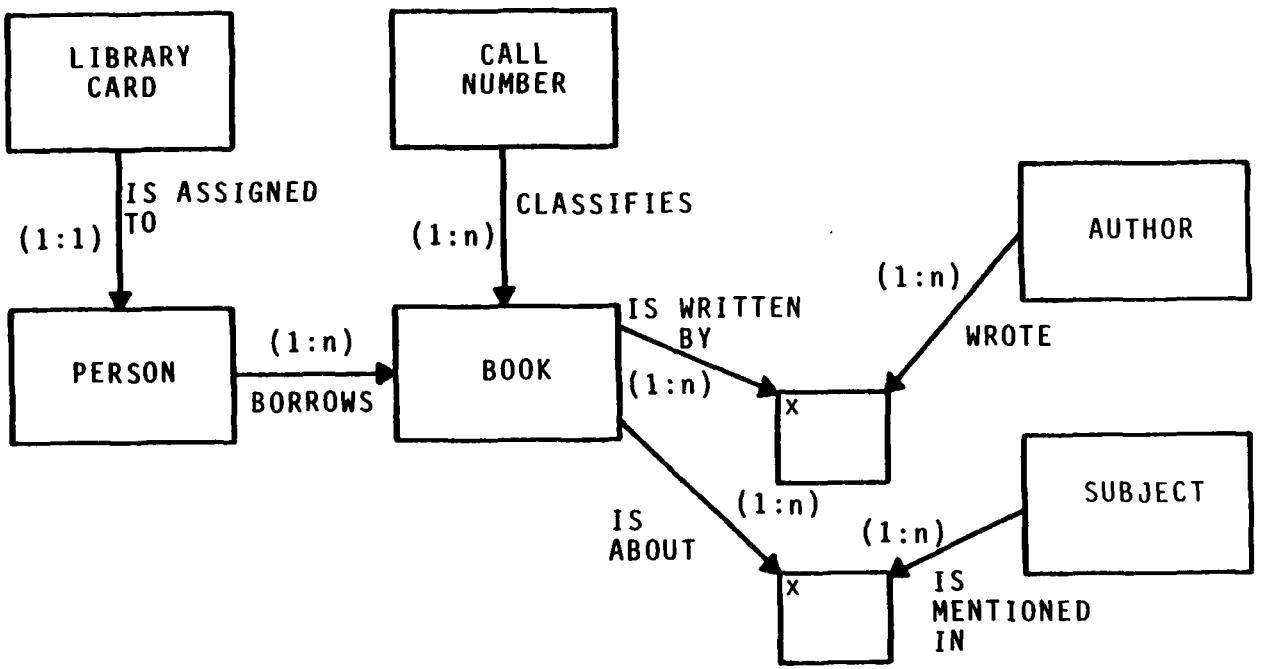


CREATING A BACHMAN DIAGRAM

LABEL THE RATIOS FIRST:



NOW REPLACE THE UNACCEPTABLE RATIOS WITH THOSE THAT ARE ALLOWED.



CREATING A BACHMAN DIAGRAM

NOW DRAW THE CORRESPONDING BACHMAN DIAGRAM:

REVIEW IMPORTANT ASPECTS OF THESE TWO TECHNIQUES.

THE REVIEW QUESTION WILL GET THE CLASS TO THINK ABOUT WHAT CLASS OF INFORMATION ENTITY DIAGRAMMING CAN PROVIDE DURING ANALYSIS. EMPHASIS THAT ENTITY DIAGRAMMING PROVIDES A DIFFERENT VIEW OF THE SYSTEM UNDER ANALYSIS THAN SADT.

SOME POSSIBLE ANSWERS:

1. NO, IT PROVIDES YOU WITH A DIFFERENT PERSPECTIVE, IF YOU ARE INTERESTED IN THE PROCESS USE SADT TO MODEL IT.

SUMMARY AND REVIEW QUESTIONS

- ENTITY DIAGRAMMING CONCENTRATES OF INFORMATION, NOT FUNCTION
- IDENTIFIES ENTITIES AND THEIR RELATIONSHIPS
- BACHMAN USES ENTITY DIAGRAMMING AND RELATION CLASS RATIO'S TO HELP WITH COMPUTER IMPLEMENTATION
- DOES ENTITY DIAGRAMMING HELP YOU UNDERSTAND THE PROCESS OF SELECTING AND CHECKING OUT A BOOK AND WHY?

INSTRUCTOR NOTES

Section 3

OBJECT ORIENTED DESIGN EXERCISE

INSTRUCTOR NOTES

OBJECT ORIENTED DESIGN PROVIDES A MEANS TO BEGIN A DESIGN. IT IS BASED ON THE FOLLOWING PRINCIPLES TO PRODUCE, AS CLOSE AS IS POSSIBLE, UNITS TO IMPLEMENT: MODULARITY (COLLECT LOGICALLY AND PHYSICALLY RELATED RESOURCES INTO ONE COHESIVELY STRONG UNIT); ABSTRACTION (IGNORE THE UNDERLYING DETAILS, CONCENTRATE ON THE MAJOR ATTRIBUTES); AND INFORMATION HIDING (SUPPORT ABSTRACTION BY NOT CONSIDERING HOW AN OPERATION IS IMPLEMENTED). THIS EXAMPLE WILL SHOW HOW EACH OF THESE PRINCIPLES WORK.

OBJECT ORIENTED DESIGN

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

OBJECT ORIENTED DESIGN ATTEMPTS TO KEEP THE SOLUTION AS CLOSE AS POSSIBLE TO THE REAL WORLD PROBLEM. THEREFORE, NOUNS ARE OBJECTS AND VERBS ARE OPERATING ON THEM. (RECALL SADT'S USE OF NOUNS AND VERBS).

THE OVERALL STEPS ARE:

1. DEFINE THE PROBLEM.
2. DEVELOP AN INFORMAL STRATEGY.
3. FORMALIZE THE STRATEGY
    a. IDENTIFY OBJECTS AND THEIR ATTRIBUTES.
    b. IDENTIFY THE OPERATIONS ON THE OBJECTS.
    c. ESTABLISH THE INTERFACES.
    d. IMPLEMENT THE OPERATIONS.

KEY TECHNIQUES

- IDENTIFY THE OBJECTS
- IDENTIFY OPERATIONS ON THE OBJECTS
- DETERMINE MODULES (FUNCTION CLASSES)
- DOCUMENT HIDDEN DESIGN DECISIONS
- ESTABLISH MODULE INTERFACES

> HINT: OUTLINE AN APPROACH FIRST, THEN DRAW A PICTURE TO VERIFY THE APPROACH.

INSTRUCTOR NOTES

THIS IS A VERY INFORMAL PROBLEM DESCRIPTION. EXPLAIN TO THOSE WHO AREN'T FAMILIAR OR WHO HAVE FORGOTTEN WHAT ARE BINARY TREES AND SUBTREES, WHAT LEAVES CONSIST OF, AND THE CONCEPT OF A STACK. NORMALLY A PROBLEM DESCRIPTION IS MUCH MORE DETAILED.

PROBLEM

COUNT THE LEAVES OF A BINARY TREE IN A WAY INDEPENDENT OF THE PHYSICAL STRUCTURES OF ALL THE DATA.

INSTRUCTOR NOTES

INFORMAL STRATEGY:

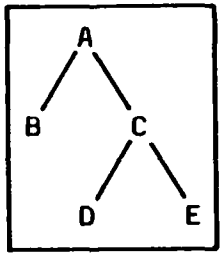
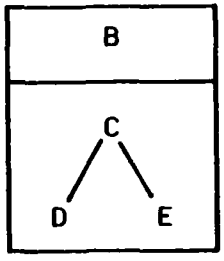
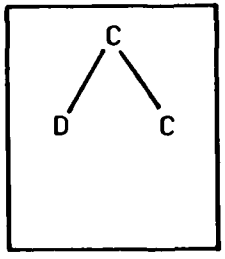
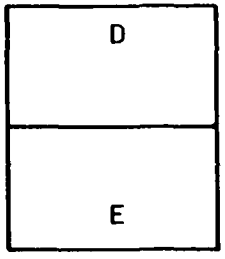
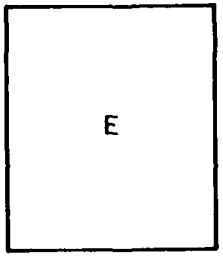
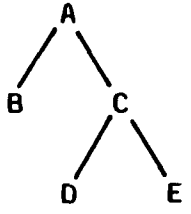
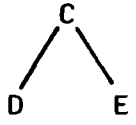
- KEEP A STACK OF TREE PARTS
- REPEAT UNTIL THE STACK IS EMPTY:
  - POP THE STACK
  - IF THIS IS A LEAF NODE, BUMP THE COUNTER
  - IF NOT, BREAK INTO TWO PARTS AND PUT EACH BACK ON THE STACK

NOTE: STACK STARTS WITH ONE OBJECT - THE WHOLE TREE

DEVELOP AN APPROACH

INFORMAL STRATEGY

EXAMPLE - PICTURE

STEP	1	2	3	4	5
STACK					
TREE		B		D	E
COUNT	0	1	1	2	3

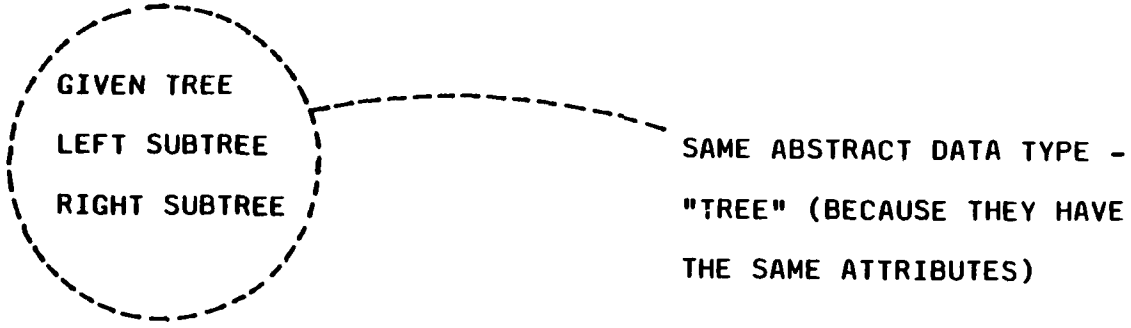
DEVELOP AN INFORMAL STRATEGY: DRAW A PICTURE

INSTRUCTOR NOTES

NOW START FORMALIZING THE STRATEGY

EXAMPLE - IDENTIFY OBJECTS AND ATTRIBUTES

- LEAF COUNT
- STACK



IDENTIFY OBJECTS AND THEIR ATTRIBUTES

INSTRUCTOR NOTES

CONTINUE FORMALIZING THE STRATEGY - INDENTIFY THE OPERATIONS ON THE OBJECTS.

- LEAF COUNT
  - ZERO
  - BUMP
  - PRINT
- STACK
  - INITIALIZE
  - PUSH
  - POP
  - IS EMPTY?
- TREE
  - GET INITIAL
  - IS LEAF?
  - SPLIT
  - THROW AWAY

IDENTIFY THE OPERATIONS ON THE OBJECTS

INSTRUCTOR NOTES

CONTINUE TO FORMALIZE THE STRATEGY: ESTABLISH THE MODULES.

MODULES:

COUNTER

STACK

TREE

NOTE: THESE WERE CHOSEN BECAUSE OF GROUPING OF OPERATIONS (OR ATTRIBUTES) EACH EXHIBITS. REVIEW INSTRUCTOR NOTE ON PREVIOUS SLIDE.

DETERMINE MODULES

DOCUMENT THE HIDDEN DESIGN DECISIONS:

MODULE	DESIGN DECISION HIDDEN
COUNTER	FORM OF THE COUNTER VARIABLE.
STACK	PHYSICAL STRUCTURE OF A STACK.
TREE	PHYSICAL STRUCTURE OF A TREE.

NOTE: THIS IS USED MORE BY THE SOFTWARE COST REDUCTION TECHNIQUE.

DOCUMENT HIDDEN DESIGN DECISIONS

MODULE	DESIGN DECISION HIDDEN

INSTRUCTOR NOTES

ESTABLISH THE INTERFACES. USE EITHER A "STANDARD PDL" LIKE THAT SHOWN, OR USE Ada DIRECTLY. NOTE THAT THE NEXT STEP IS TO IMPLEMENT THE DETAILS. ALSO, NOTE THE DIFFERENCE BETWEEN FUNCTIONS AND PROCEDURES.

```
MODULE NAME IS
      .
      .
      .
END NAME;
```

MODULE BOUNDARY

```
PROCEDURE NAME (PARAM-1:  IN     TYPE;
                          OUT
                PARAM-2:  IN     TYPE;
                          OUT
                   .
                   .
                   .
                PARAM-N:  IN     TYPE);
                          OUT
```

PROCEDURE DECLARATION

```
TYPE NAME;
```

TYPE DECLARATION

```
FUNCTION NAME  (PARAM-1:  IN     TYPE;
                          OUT
                PARAM-2:  IN     TYPE;
                          OUT
                   .
                   .
                   .
                PARAM-N:  IN     TYPE;
                          OUT

                RETURN TYPE
```

FUNCTION DECLARATION

```
module COUNTER_PACKAGE is
   type COUNTER_TYPE;
   procedure ZERO  (COUNTER: in     COUNTER_TYPE);
   procedure BUMP  (COUNTER: in out COUNTER_TYPE);
   procedure PRINT (COUNTER: in     COUNTER_TYPE);
   ...
end COUNTER_PACKAGE;

module TREE_PACKAGE Is
   type TREE_TYPE;
   procedure GET_INITIAL (TREE: out TREE_TYPE);
   function IS_LEAF      (TREE: in  TREE_TYPE) return boolean;
   procedure SPLIT       (TREE: in  TREE_TYPE) LEFT: out TREE_TYPE; RIGHT: out TREE_TYPE_);
   procedure THROW_ARRAY (TREE: in  TREE_TYPE);
   ...
end TREE_PACKAGE;

module STACK_PACKAGE is
   type STACK_TYPE;
   procedure INITIALIZE (TREE: in  TREE_TYPE);
   procedure PUSH       (TREE: in  TREE_TYPE);
   procedure POP        (TREE: out TREE_TYPE);
   procedure IS_EMPTY   return boolean;
   ...
end STACK_PACKAGE;
```

ESTABLISH THE INTERFACES

INSTRUCTOR NOTES

REVIEW IMPORTANT POINTS OF OBJECT ORIENTED DESIGN.

THIS METHODOLOGY WOULD NEED TO BE USED WITH SOME OTHER DESIGN TECHNIQUES (SCRP OR STRUCTURED DESIGN) IN ORDER TO ADDRESS A PROGRAM OF THAT SIZE. THE OTHER TECHNIQUES WOULD MODULARIZE THE SOFTWARE TO A POINT THAT OBJECT ORIENTED DESIGN TECHNIQUES COULD BE APPLIED TO INDIVIDUAL MODULES.

SOME POSSIBLE ANSWERS:

1. A QUALIFIED NO, THE METHODOLOGY WORKS WELL AT ESTABLISHING THE INTERFACE TO AND BETWEEN SMALL APPLICATIONS OR SMALL PARTS OF A MEDIUM TO LARGE APPLICATION. THIS METHODOLOGY COULD BE USED AFTER SOME INITIAL HIGH LEVEL APPLICATION STRUCTURING IS DONE BY SOME OTHER METHODS.

SUMMARY AND REVIEW QUESTIONS

- EMPHASIZES ABSTRACTION/SEPARATION
- PRODUCES MODULES
- RESULTS VERY CLOSE TO IMPLEMENTATION
- HIDES DESIGN DECISIONS
- DOES THIS METHODOLOGY SEEM USABLE ON AN APPLICATION THAT WOULD RESULT IN 100,000 LINES OF SOURCE CODE OR MORE AND WHY?

INSTRUCTOR NOTES

Section 4

STRUCTURED DESIGN EXERCISE

INSTRUCTOR NOTES

STRUCTURED DESIGN IS CONCERNED WITH MODULARIZATION, INTERCONNECTIVITY, AND THE FLOW OF DATA. IT IS NOT CONCERNED WITH THE MODULE INTERNALS. STRUCTURED DESIGN IS SIMILAR TO, BUT NOT AS FORMAL AS OBJECT ORIENTED DESIGN. FOR THIS EXERCISE WE WILL REPEAT THE OBJECT ORIENTED DESIGN PROBLEM TO GET A COMPARISON OF THE TWO METHODS.

STRUCTURED DESIGN OVERVIEW

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

FOR THIS EXAMPLE WE WILL USE ONLY THESE THREE TECHNIQUES. OTHERS, SUCH AS A SEQUENCE, SELECTION AND ITERATION, TRANSACTION CENTERS, ETC. WON'T BE USED. ALSO, WE WON'T BOTHER WITH GENERATING THE DATA FLOW DIAGRAMS WHICH ONE WOULD NORMALLY USE TO BEGIN WITH AS A FIRST APPROACH.

KEY TECHNIQUES

- IDENTIFY THE MODULES
- IDENTIFY THE CONTROL RELATIONSHIPS
- IDENTIFY THE DATA TRANSFER

HINT: DRAW A PICTURE TO GET STARTED

INSTRUCTOR NOTES

REVIEW QUICKLY WHAT THIS PROBLEM MEANS AGAIN. THE CLASS SHOULD HAVE A GOOD UNDERSTANDING OF IT AFTER DOING THE PREVIOUS EXERCISE.

PROBLEM

COUNT THE LEAVES OF A BINARY TREE IN A WAY INDEPENDENT OF THE PHYSICAL STRUCTURES OF ALL THE DATA.

INSTRUCTOR NOTES

REMEMBER, THESE SYMBOLS ARE NOT LANGUAGE SPECIFIC.

STRUCTURE CHART SYNTAX

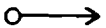
MODULE

(SOFTWARE FUNCTION)

INTERFACE

(WHERE DATA IS PASSED BETWEEN MODULES)

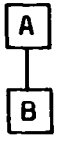
DATA



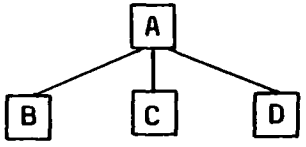
(APPLICATION)

(CONTROL)

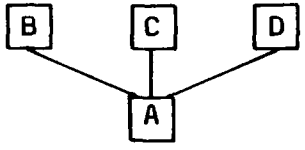
STRUCTURE AND CONTROL



MODULE A CALLS MODULE B



MODULE A CALLS B, C, AND D

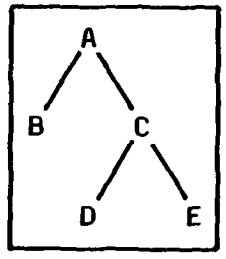
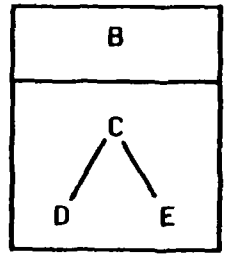
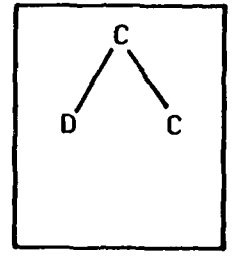
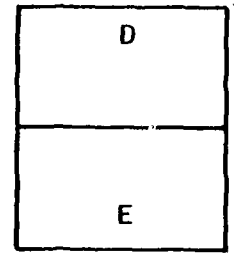
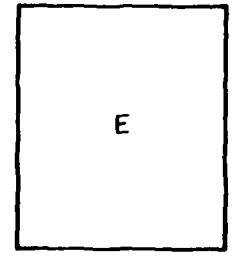
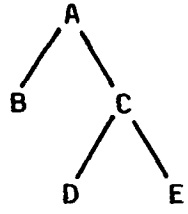
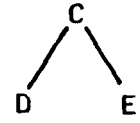


MODULE A IS CALLED BY B, C, AND D

INSTRUCTOR NOTES

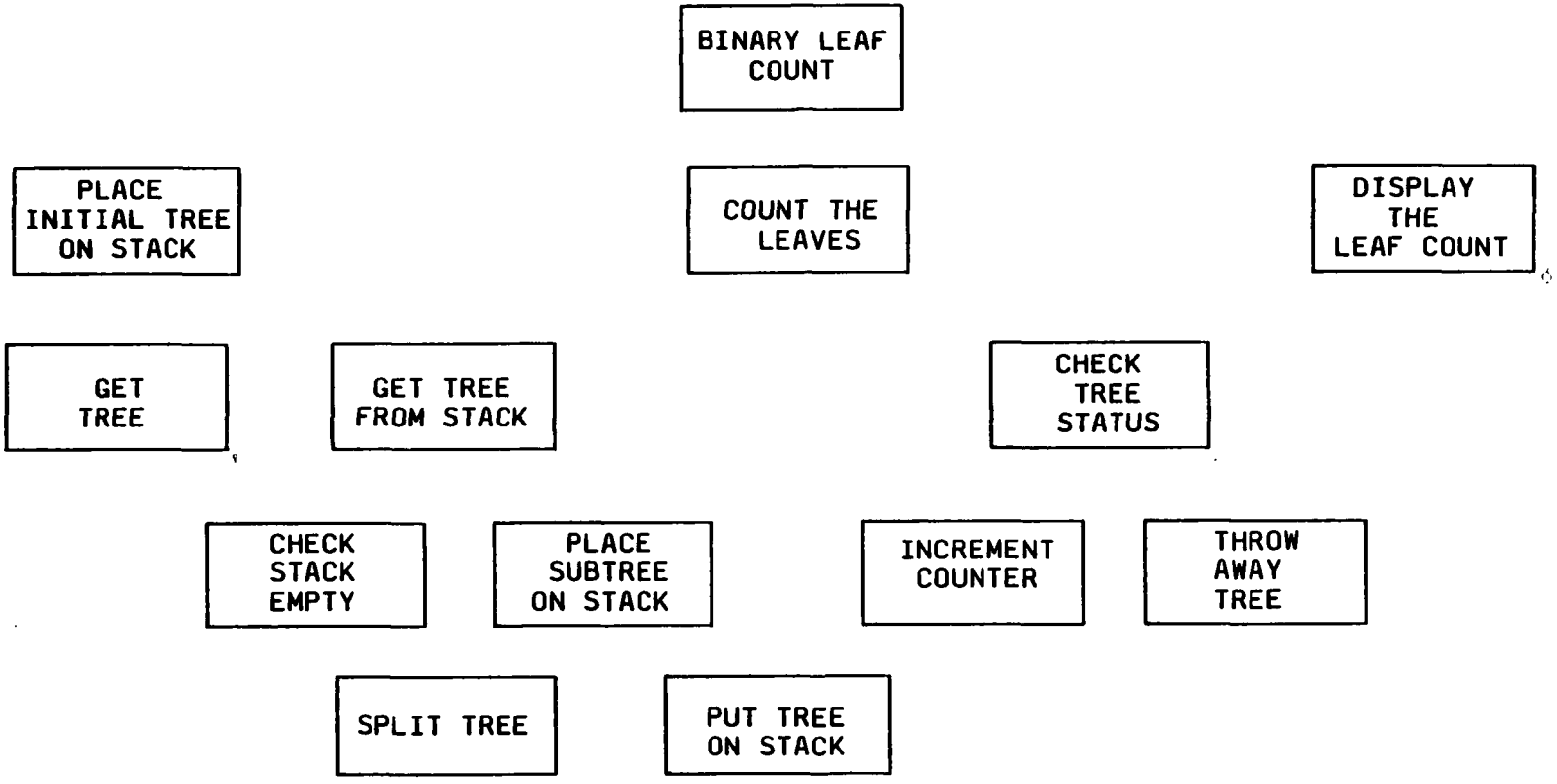
REVIEW THE PROCESS TAKING PLACE, INDENTIFYING POTENTIAL MODULES, CONTROL, AND DATA.

EXAMPLE - PICTURE

STEP	1	2	3	4	5
STACK					
TREE		B		D	E
COUNT	0	1	1	2	3

INSTRUCTOR NOTES

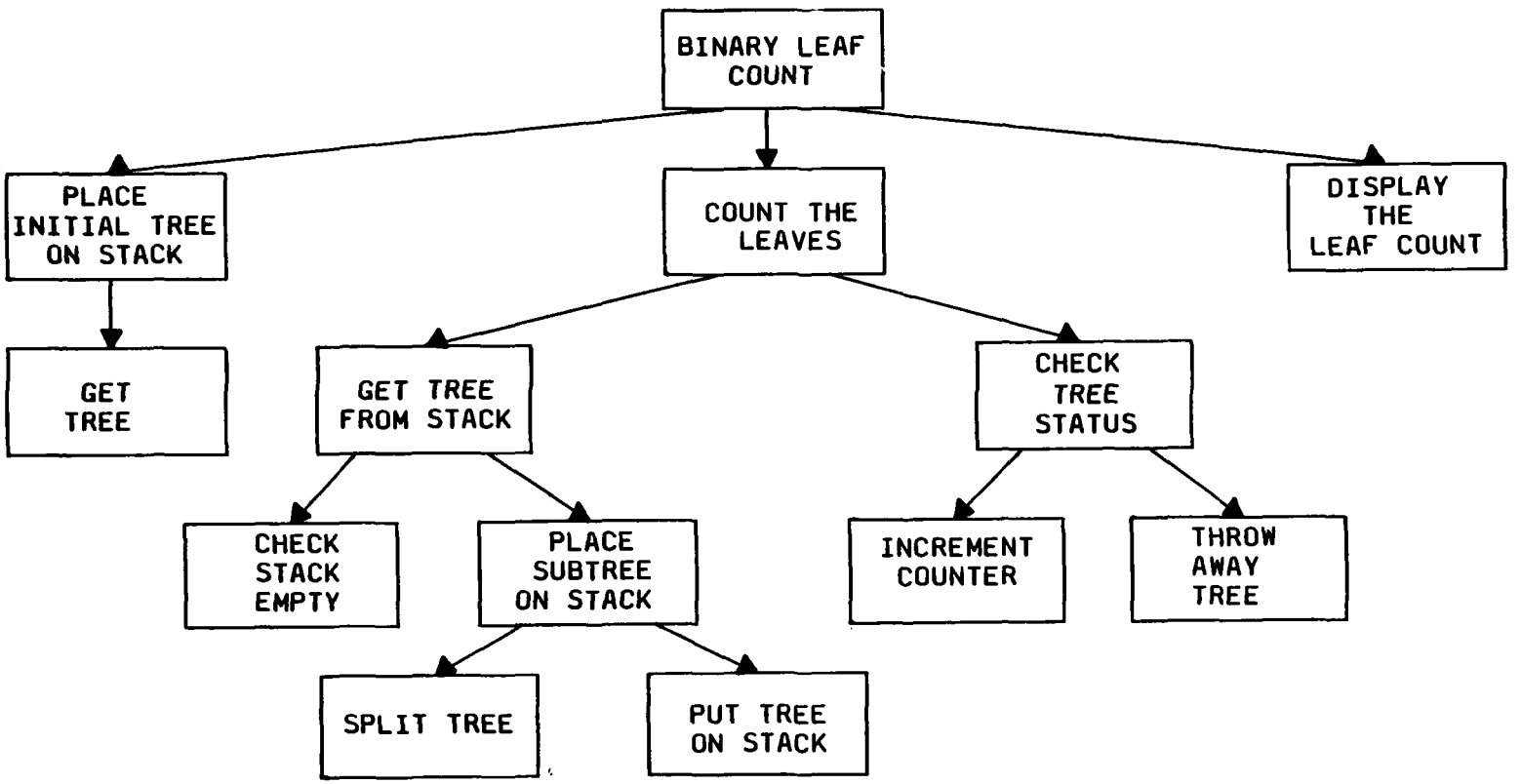
GIVE THE CLASS A FEW MINUTES, AND THEN DRAW THESE MODULES.



NOTE: YOU MAY WANT TO PLACE MODULES IN SOME RANDOM ORDERING. THE ABOVE PLACEMENT STARTS TO SHOW CONTROL RELATIONSHIPS.

IDENTIFY THE MODULES

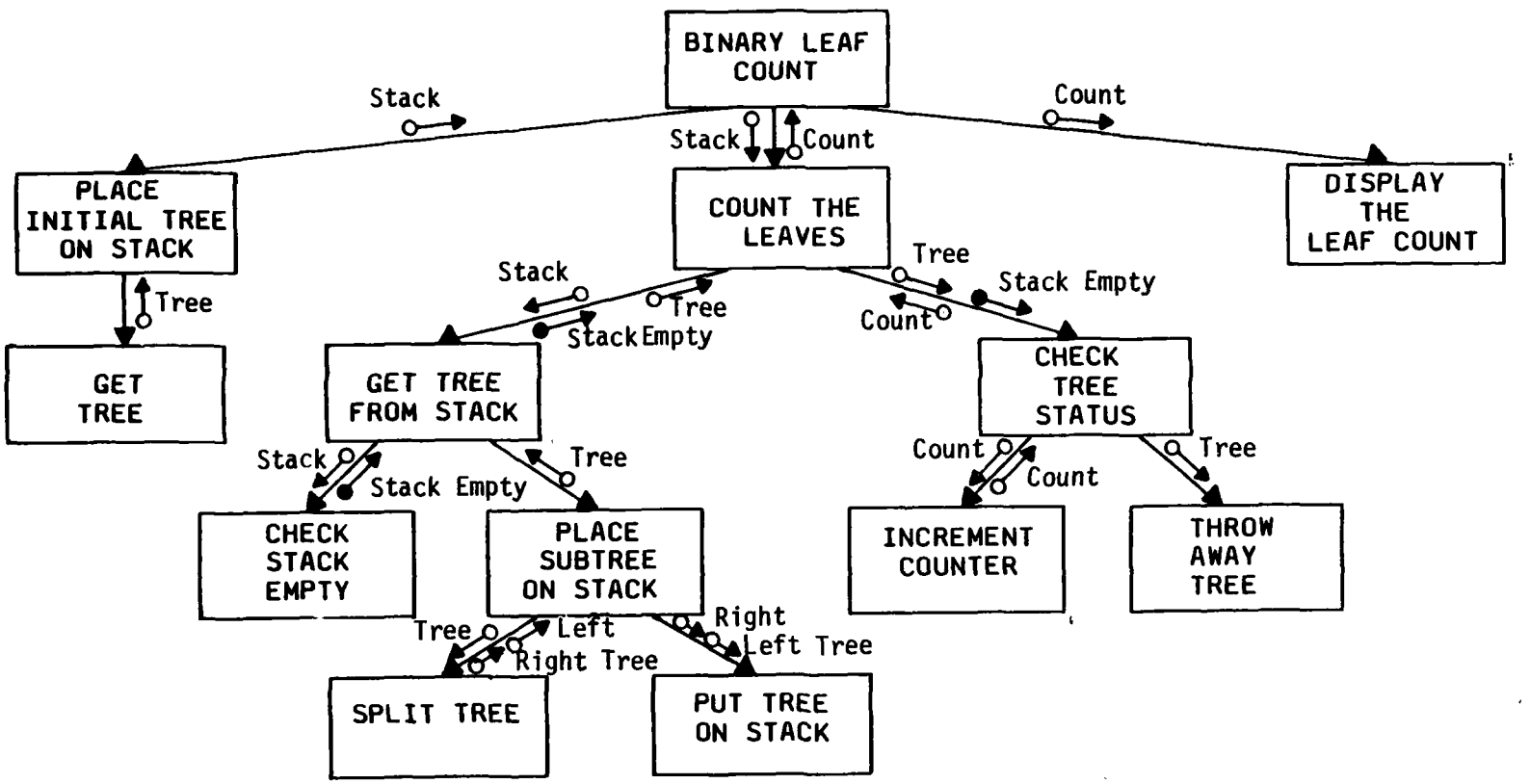
DRAW THE MODULE CONTROL AND STRUCTURE ARROWS



NOTE: YOU MAY WANT TO PLACE MODULES IN SOME RANDOM ORDERING. THE ABOVE PLACEMENT STARTS TO SHOW RELATIONSHIPS.

IDENTIFY THE CONTROL RELATIONSHIPS

DISCUSS SOME OF THE DATA TRANSFERS, INCLUDING CONTROL



IDENTIFY DATA TRANSFER

INSTRUCTOR NOTES

DISCUSS THESE POINTS WITH THE CLASS:

- STRUCTURED DESIGN IS MORE ON AN ART THAN AN OBJECT ORIENTED DESIGN.
- OBJECT ORIENTED DESIGN CONCENTRATES ON THE FUNCTIONAL ASPECTS OF THE PROBLEM VIA ABSTRACTION AND INFORMATION HIDING, AND TRYS TO GET ITS SOLUTION VERY CLOSE TO AN IMPLEMENTATION.
- STRUCTURED DESIGN SUPPORTS THE IDENTIFICATION OF SOFTWARE MODULES, CONTROL, AND DATA, BUT IS ABSTRACTED AWAY FROM THE REAL WORLD.
- STRUCTURED DESIGN ISN'T AS GOOD FOR REAL-TIME APPLICATIONS AS OBJECT ORIENTED DESIGN.

SOME POSSIBLE ANSWERS:

1. PRIMARILY IN FOCUS - S.D. FOCUS IS ON THE FUNCTIONS PERFORMED - O.O.D. FOCUS IS ON THE INTERFACES.
2. MOST PEOPLE WILL PICK S.D. DUE TO ITS GRAPHICAL NATURE AND FUNCTIONAL FOCUS WHICH IS MORE NATURAL TO MOST PEOPLE,

REVIEW QUESTIONS

HOW DOES STRUCTURED DESIGN DIFFER FROM OBJECT ORIENTED DESIGN?

COMPARING THE TWO TECHNIQUES WHICH GIVE THE BEST VIEW OF THE OVERALL ARCHITECTURE OF THE SOFTWARE AND WHY?

INSTRUCTOR NOTES

REVIEW IMPORTANT ASPECTS OF STRUCTURED DESIGN.

SUMMARY

- STRUCTURED DESIGN PROVIDES A WAY TO REPRESENT SOFTWARE ARCHITECTURAL DESIGN
- PROVIDES GUIDANCE ON HOW TO DETERMINE MODULARITY
- FAIRLY EASY TO USE

INSTRUCTOR NOTES

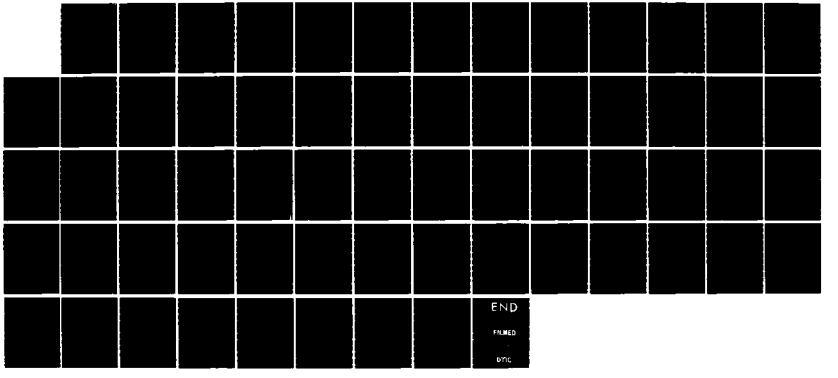
Section 5

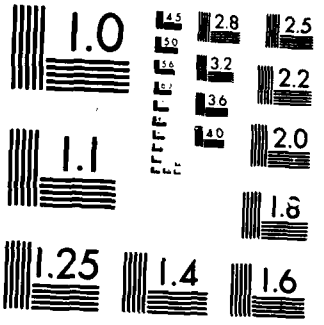
JACKSON EXERCISE

AD-A165 299 ADA (TRADEMARK) TRAINING CURRICULUM SOFTWARE ENGINEERING METHODOLOGIES M201 TEACHER'S WORKBOOK(U) SOFTECH INC WALTHAM MA 1986 DAAB07-83-C-K506

2/2

UNCLASSIFIED F/G 5/9 NL





Section 5

JACKSON EXERCISE

JACKSON OVERVIEW

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

SPEND SOME TIME EXPLAINING THIS PROBLEM TO THE CLASS.

NOTE: NET MOVEMENT IS TECHNICAL JARGON FOR QUANTITY ON-HAND.

PROBLEM #1: PARTS MOVEMENT

GOAL: DESIGN A PROGRAM THAT WILL REPORT ON THE TOTAL QUANTITY OF EVERY PART ON-HAND IN THE WAREHOUSE.

INPUT: "PARTS FILE" IS A FILE OF RECORDS, SORTED BY PART NUMBER. EACH RECORD CONTAINS INFORMATION ABOUT THE ISSUE (DISTRIBUTION) OR RECEIPT (ACCEPTANCE) OF THE PART.

OUTPUT: THE REPORT MUST HAVE ONE HEADING BEFORE ANY OTHER INFORMATION. EACH LINE AFTER THE HEADING SHOWS A PART WITH ITS NET MOVEMENT.

INSTRUCTOR NOTES

WALKTHROUGH THE BASIC STEPS OF THE JACKSON METHOD WITH THE STUDENTS. GO THROUGH EACH STEP OF THE METHOD WITH THE STUDENTS. NOTE THAT THIS EXERCISE IS LONG AND WILL TAKE SOME TIME.

METHOD

ANALYSIS

1. READ AND UNDERSTAND THE PROBLEM.

MODELING

2. DRAW STRUCTURE DIAGRAMS FOR INPUT, THEN OUTPUT.

CONVERTING

3. FIND THE POINT WHERE BOTH STRUCTURES MATCH.

4. DRAW ONLY THE STRUCTURE DIAGRAM OF THE PROGRAM.

COMPLETING

5. ANNOTATE THE STRUCTURE DIAGRAM OF THE PROGRAM, FILLING IN EXISTING BOXES AND ADDING NEW BOXES THAT TALK ABOUT PROGRAMMING DETAILS.

INSTRUCTOR NOTES

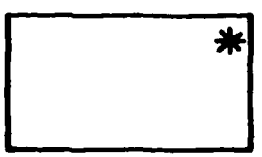
THESE ARE THE STRUCTURES TO BE USED AND THEIR REPRESENTATIONS.

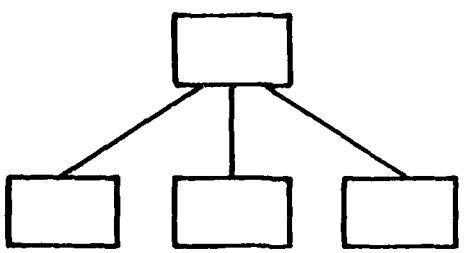
SYNTAX SUMMARY

LEFT-TO-RIGHT READING

SEQUENCE

SELECTION



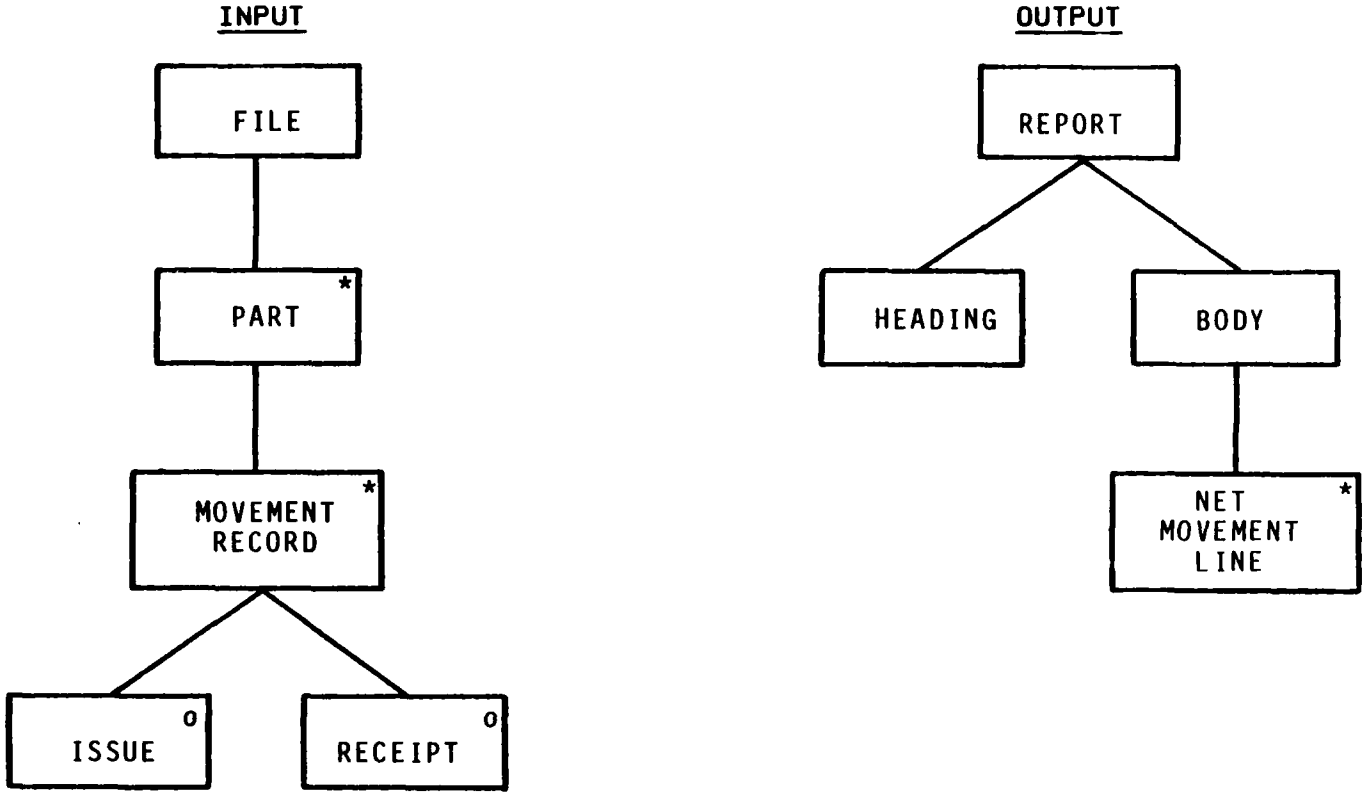


REPETITION

HIERARCHY

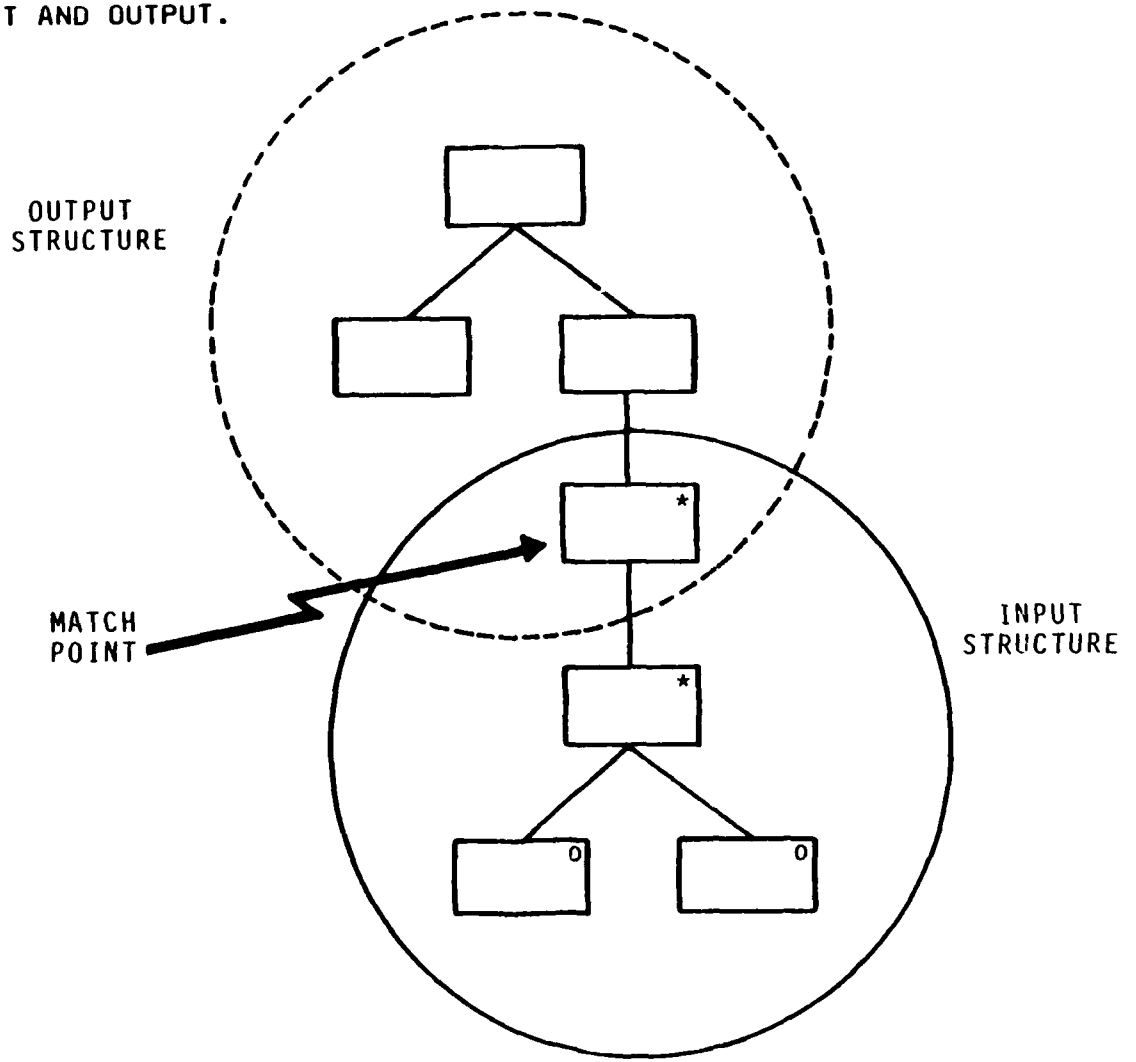
INSTRUCTOR NOTES

MODELING: DEFINE INPUT AND OUTPUT DATA STRUCTURES USING THE GRAPHIC NOTATION. GIVE THE CLASS A FEW MINUTES, DISCUSSING THE PROBLEM WITH THEM IF THEY GET STUCK



STEP 2: MODELING

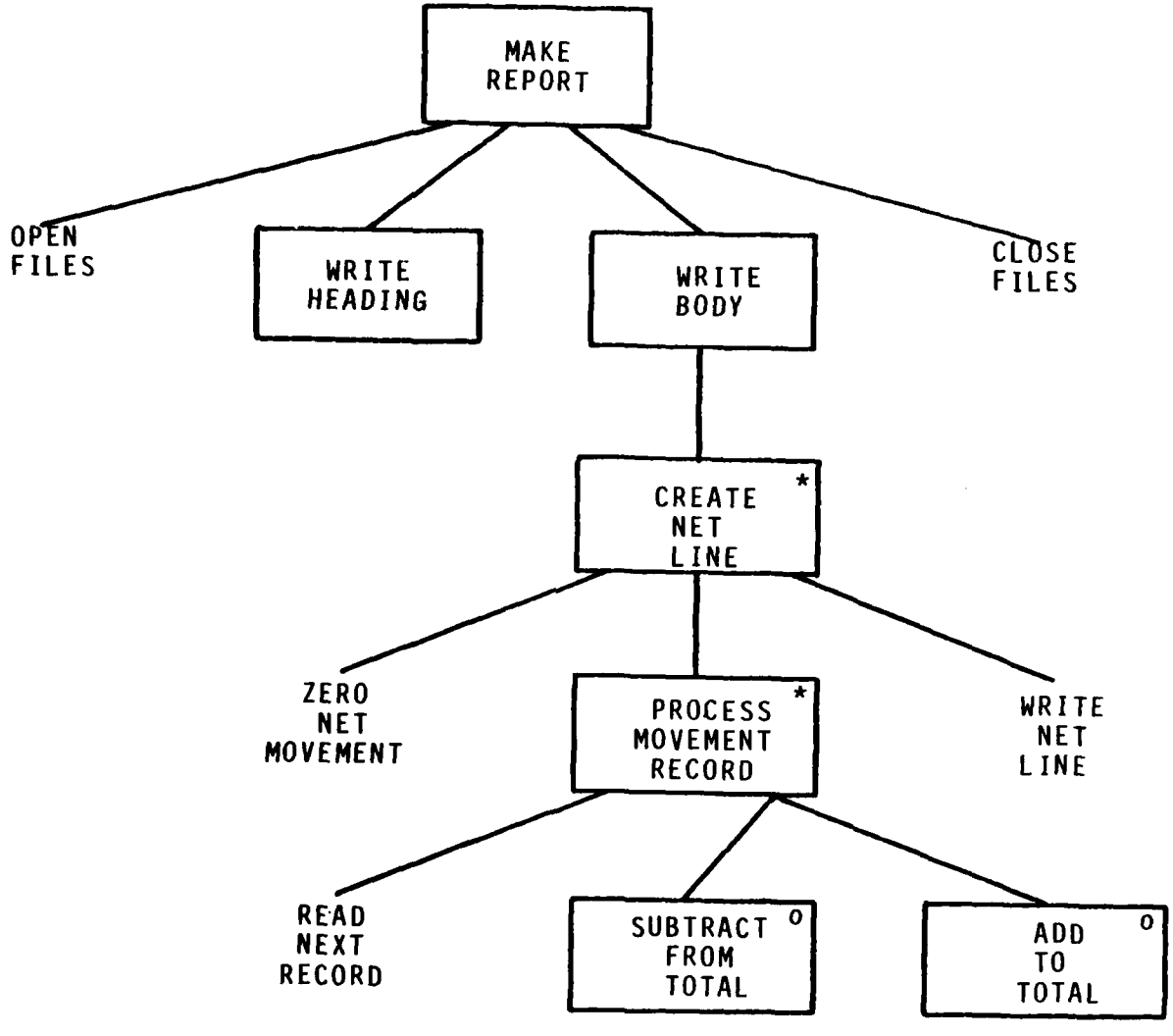
CONVERTING: CREATE A GENERAL PROGRAM STRUCTURE THAT MATCHES THE STRUCTURES OF BOTH THE INPUT AND OUTPUT.



STEP 3: CONVERTING

INSTRUCTOR NOTES

COMPLETING: LIST ELEMENTARY OPERATIONS AND ASSIGN THESE TO MODULES.



STEP 4: COMPLETING

INSTRUCTOR NOTES

YOU MAY NOT GET THROUGH ALL THIS. WALK CLASS THROUGH WHATEVER REMAINS.

PROBLEM #2

THE SAME MOVEMENT RECORD FILE EXISTS. THIS TIME, HOWEVER, YOU MUST (WHILE READING THE INPUT FILE ONLY ONCE AND USING NO INTERMEDIATE FILES) DEVELOP TWO SEPARATE REPORTS:

- REPORT #1 GIVES THE TOTAL ISSUE FOR EACH PART
- REPORT #2 GIVES THE TOTAL RECEIPT FOR EACH PART

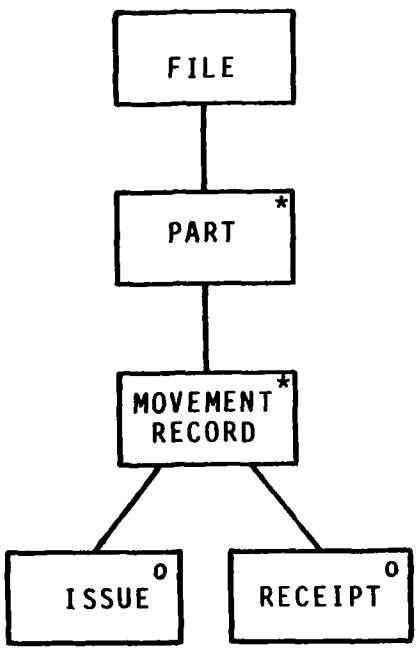
BOTH REPORTS HAVE A SINGLE HEADING, AND EACH TOTAL MUST BE PUT ON A SEPARATE LINE WITH ITS CORRESPONDING PART NUMBER.

HINT: FIND THE STRUCTURE CLASH AND RESOLVE IT.

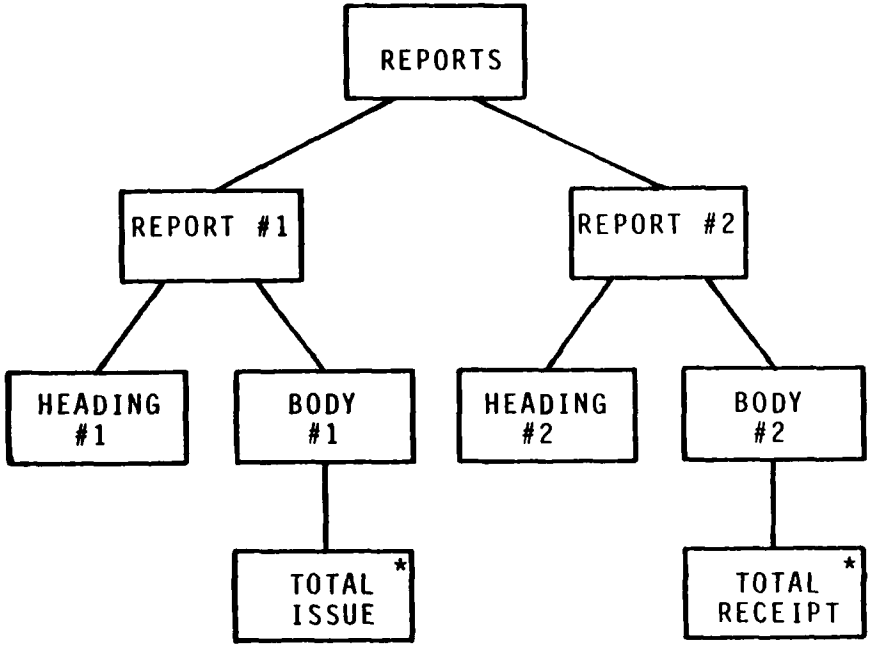
INSTRUCTOR NOTES

GO OVER THE PROBLEM AND THESE SOLUTIONS. MAKE SURE THE CLASS UNDERSTANDS THEM.

MODEL

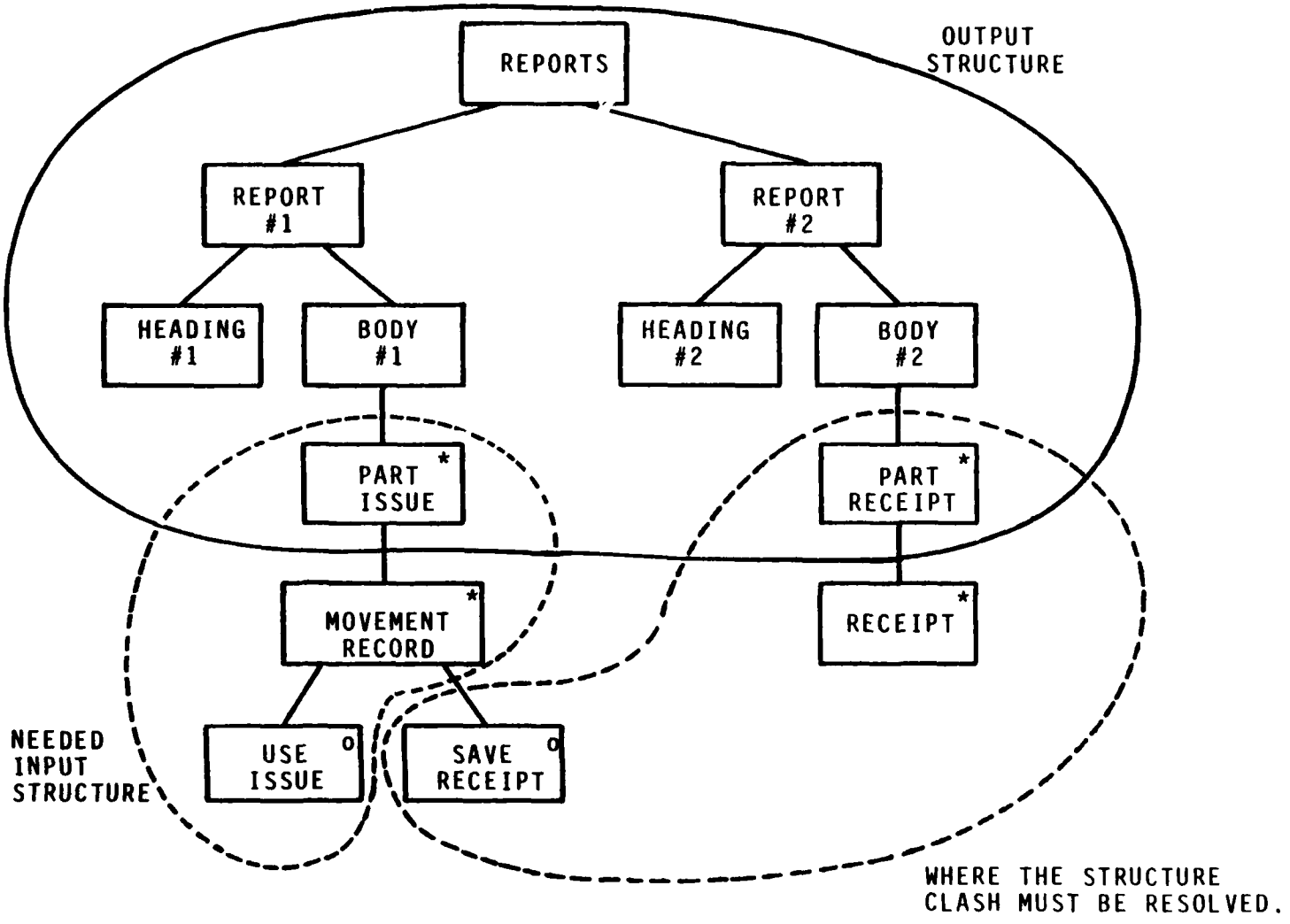


INPUT FILE

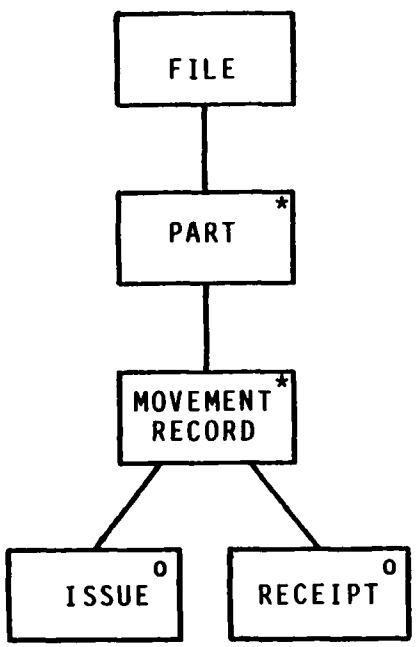


REQUIRED REPORTS

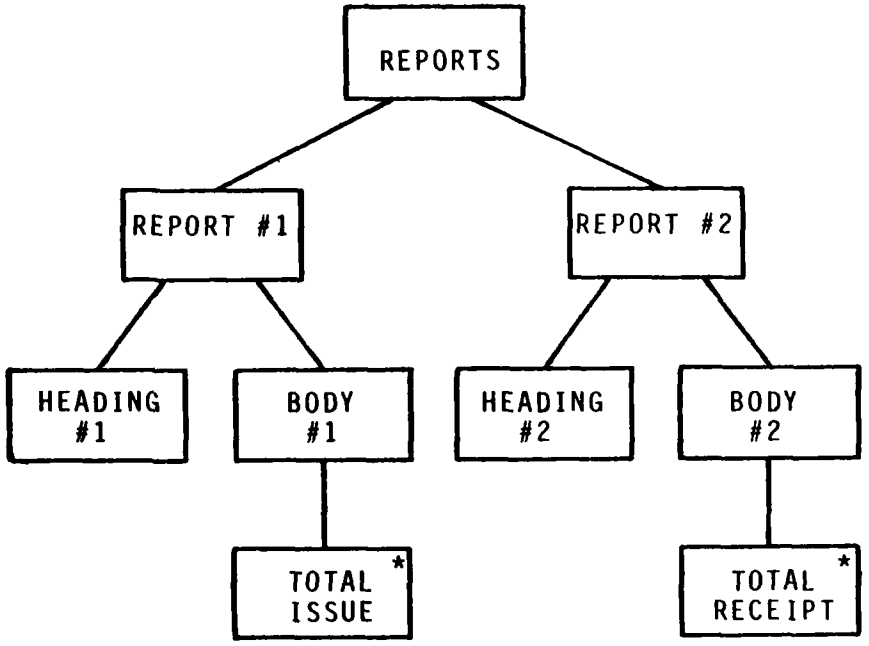
INSTRUCTOR NOTES



MODEL

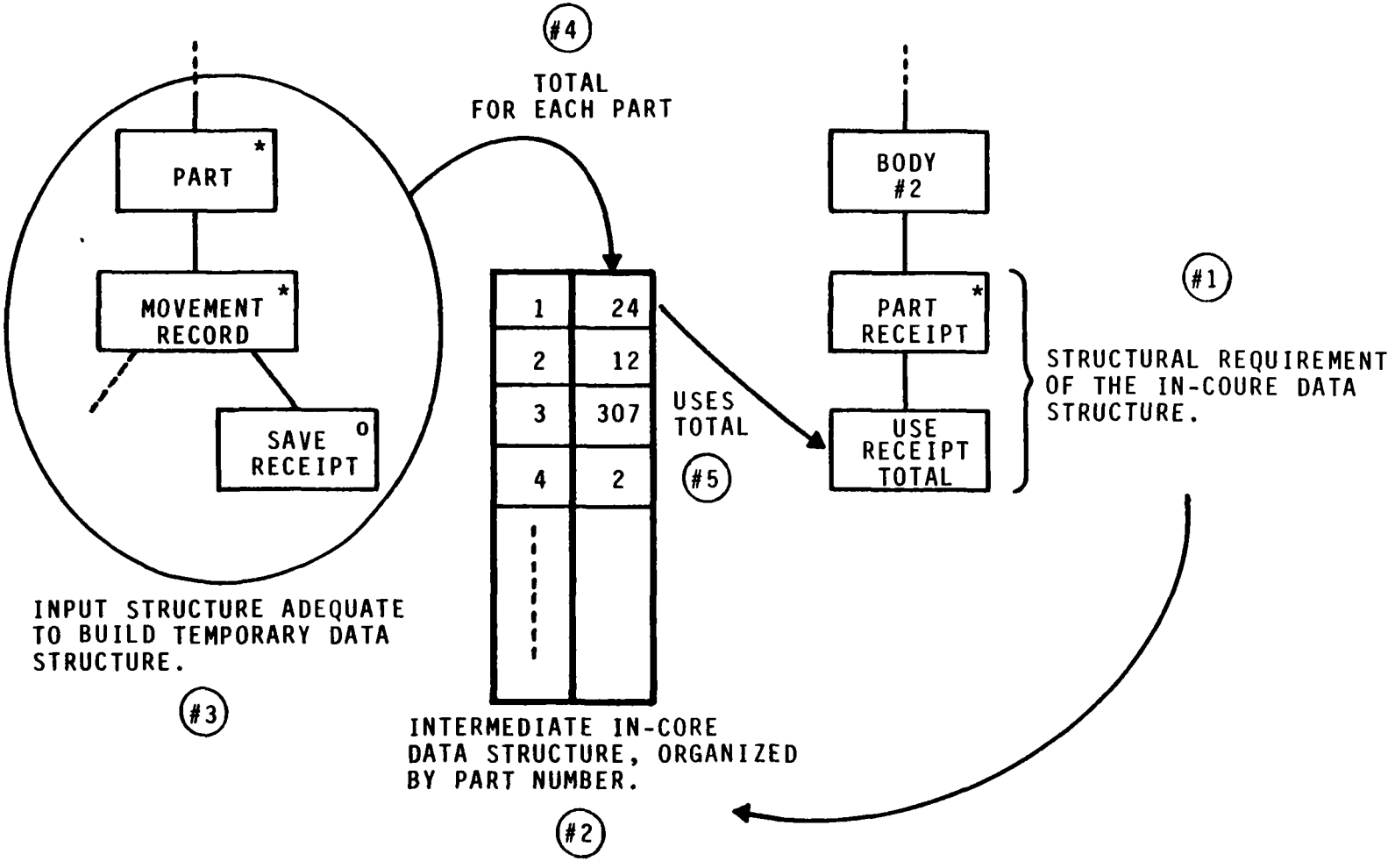


INPUT FILE



REQUIRED REPORTS

INSTRUCTOR NOTES

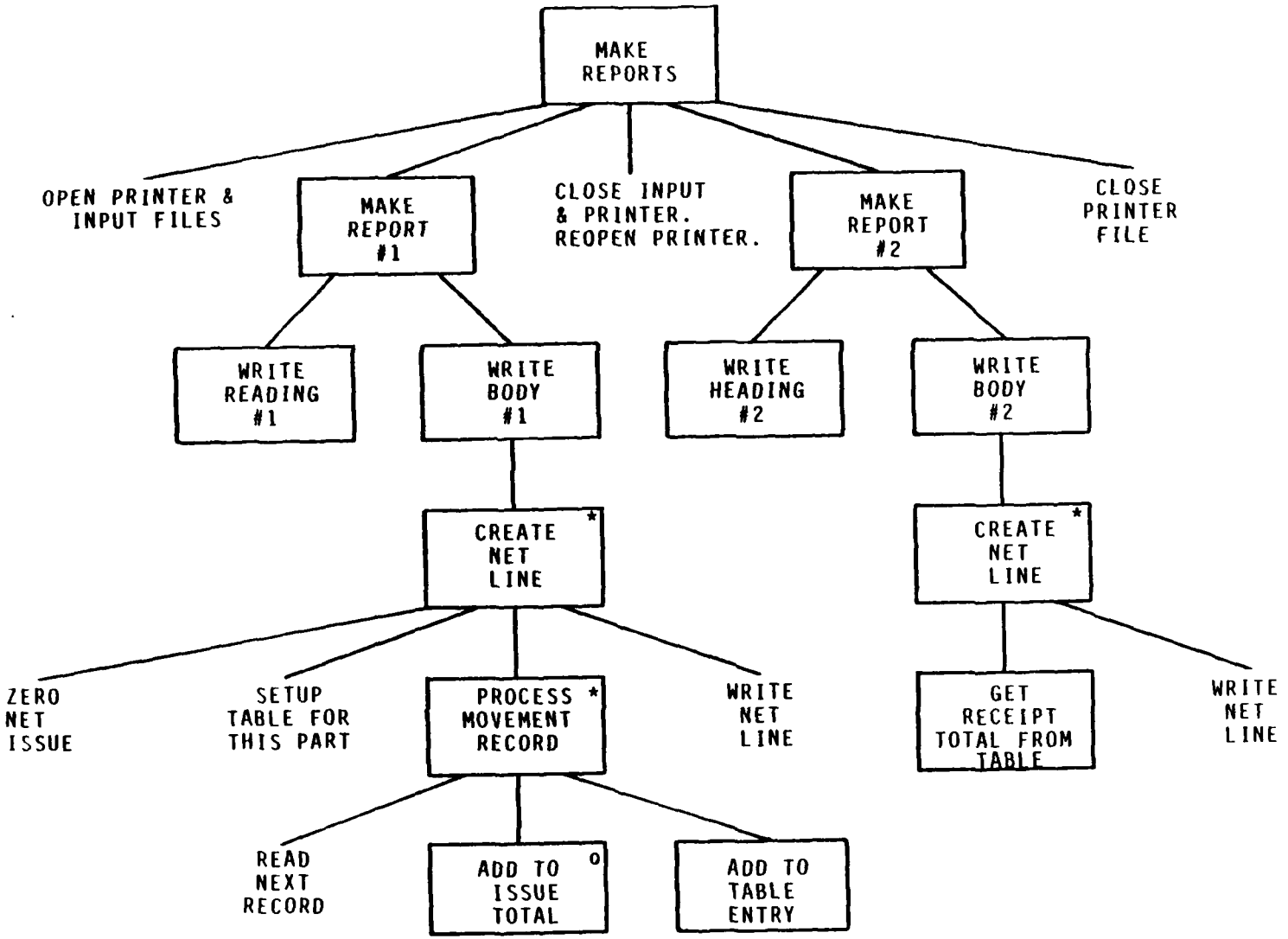


BEGIN CONVERTING

INSTRUCTOR NOTES

GO OVER THE COMPLETED STRUCTURE OF THE PROBLEM. DON'T SPEND TOO MUCH TIME ON THIS.

COMPLETED STRUCTURE



INSTRUCTOR NOTES

REVIEW IMPORTANT ASPECTS OF JACKSON METHOD.

USE THE QUESTION TO STIMULATE A DISCUSSION OF THE DIFFERENCES BETWEEN THE METHODS WE HAVE COVERED SO FAR. FOCUS THE DISCUSSION BY PROVIDING ADDITIONAL QUESTIONS ABOUT HOW THE CONCEPTS DIFFER, GRAPHICS DIFFER AND THE MODE OF PRESENTATION DIFFER.

SOME POSSIBLE ANSWERS:

1. ITS FOCUS DIFFERS - JACKSON ASSUMES THE STRUCTURE OF THE SOFTWARE IS DETERMINED BY THE STRUCTURE OF THE INPUT AND OUTPUT DATA. STRUCTURED DESIGNS FOCUS IS FUNCTIONAL DATA. STRUCTURED DESIGNS FOCUS IS FUNCTIONAL. OBJECT ORIENTED DESIGN FOCUS IS INTERFACE.

SUMMARY AND REVIEW QUESTION

- IDENTIFIES DATA STREAMS AND STATE INFORMATION EARLY IN DESIGN PROCESS
- FOCUS IS ON DATA STRUCTURES
- MODELS REALITY, NOT FUNCTION
- HOW DOES JACKSON'S TECHNIQUES DIFFER FROM STRUCTURED DESIGN AND OBJECT ORIENTED DESIGN?

INSTRUCTOR NOTES

Section 6

FINITE-STATE MAP EXERCISE

INSTRUCTOR NOTES

FINITE STATE MAPS COULD BE CONSIDERED AN ELEMENT OF STRUCTURED PROGRAMMING. THEY MODEL THE BEHAVIOR OF A (PORTION OF) SYSTEM BY SHOWING ITS STATES AND TRANSFORMATIONS. FSM THEREBY HELPS ONE UNDERSTAND HOW THE SYSTEM INTERACTS AND CAN STRUCTURE A PROGRAM TO TAKE ADVANTAGE OF THE SYSTEMS DYNAMIC BEHAVIOR, INCREASING EXECUTION EFFICIENCY AND REDUCING DEBUG TIME. THE EXAMPLE WILL SHOW THE FOUNDATION OF FSM.

SADT OVERVIEW

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

THIS EXAMPLE CAN BE THOUGHT OF AS A COMMUNICATION PROTOCOL PROBLEM, WHICH IS ONE WHICH FSM IS VERY USEFUL IN ATTACKING. STRESS TO THE CLASS THE NECESSITY OF THE "BLACK BOX" VIEW AND HOW THE PROBLEM CAN BE DECOMPOSED IN THIS WAY. ALSO RELATE THIS CONCEPT BACK TO THE IDEAS OF INFORMATION HIDING, SEPARATION OF CONCERNS, AND MODULARITY.

PROBLEM

DRAW A FINITE-STATE MAP THAT DESCRIBES DIALING A LONG DISTANCE NUMBER FROM YOUR HOME PHONE. INCLUDE ERROR SITUATIONS. LABEL:

- START AND GOAL STATES
- MAIN PATH

HINT: THINK OF THE PHONE AS A BLACK BOX, AND DETERMINE THE KINDS OF INPUT REQUIRED TO DIAL LONG DISTANCE. DRAW THE BOX IF NEEDED.

INSTRUCTOR NOTES

REVIEW THESE STEPS OF THE FSM METHOD. THIS EXAMPLE WON'T DO ALL STEPS SEPARATELY, BUT MERGE SOME TOGETHER.

FSM METHOD

1. DRAW BLACK BOX

2. IDENTIFY STATES

3. IDENTIFY TRANSITIONS

4. DRAW FSM DIAGRAM

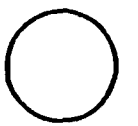
5. LABEL START AND GOAL STATES

6. LABEL MAIN PATH

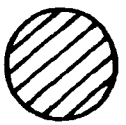
INSTRUCTOR NOTES

REVIEW THE FSM SYNTAX.

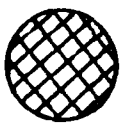
SYNTAX SUMMARY



STATE

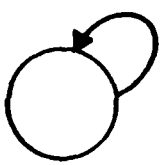


START STATE

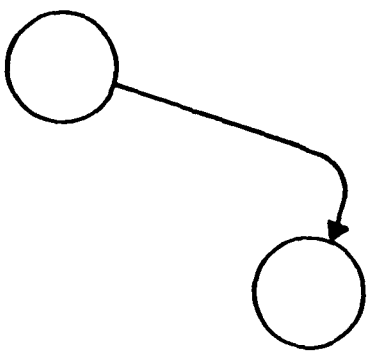


GOAL STATE

TRANSITION



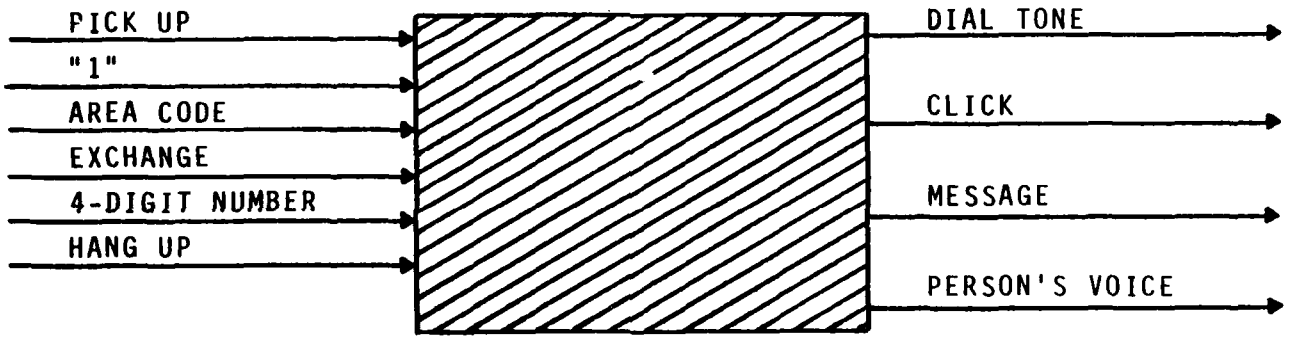
TRANSITIONS CAN GO TO THE SAME STATE.



TRANSITIONS FREQUENTLY CONNECT TO STATES.

INSTRUCTOR NOTES

DRAW THIS BOX AFTER GIVING THE STUDENTS A FEW MINUTES TO TRY IT THEMSELVES.



DRAW A BLACK BOX VIEW

INSTRUCTOR NOTES

GET THE CLASS TO PROVIDE THESE STATES. ADD OTHERS IF CLASS FEELS THEY ARE APPROPRIATE.

STATES

1. HANG UP
2. DIAL TONE
3. LONG DISTANCE
4. CONNECT AREA
5. CONNECT EXCHANGE
6. CONNECT PHONE
7. INVALID AREA
8. INVALID EXCHANGE
9. CHANGED NUMBER
10. NOT IN SERVICE
11. BUSY
12.
13.
14.
15.

LIST THE STATES

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

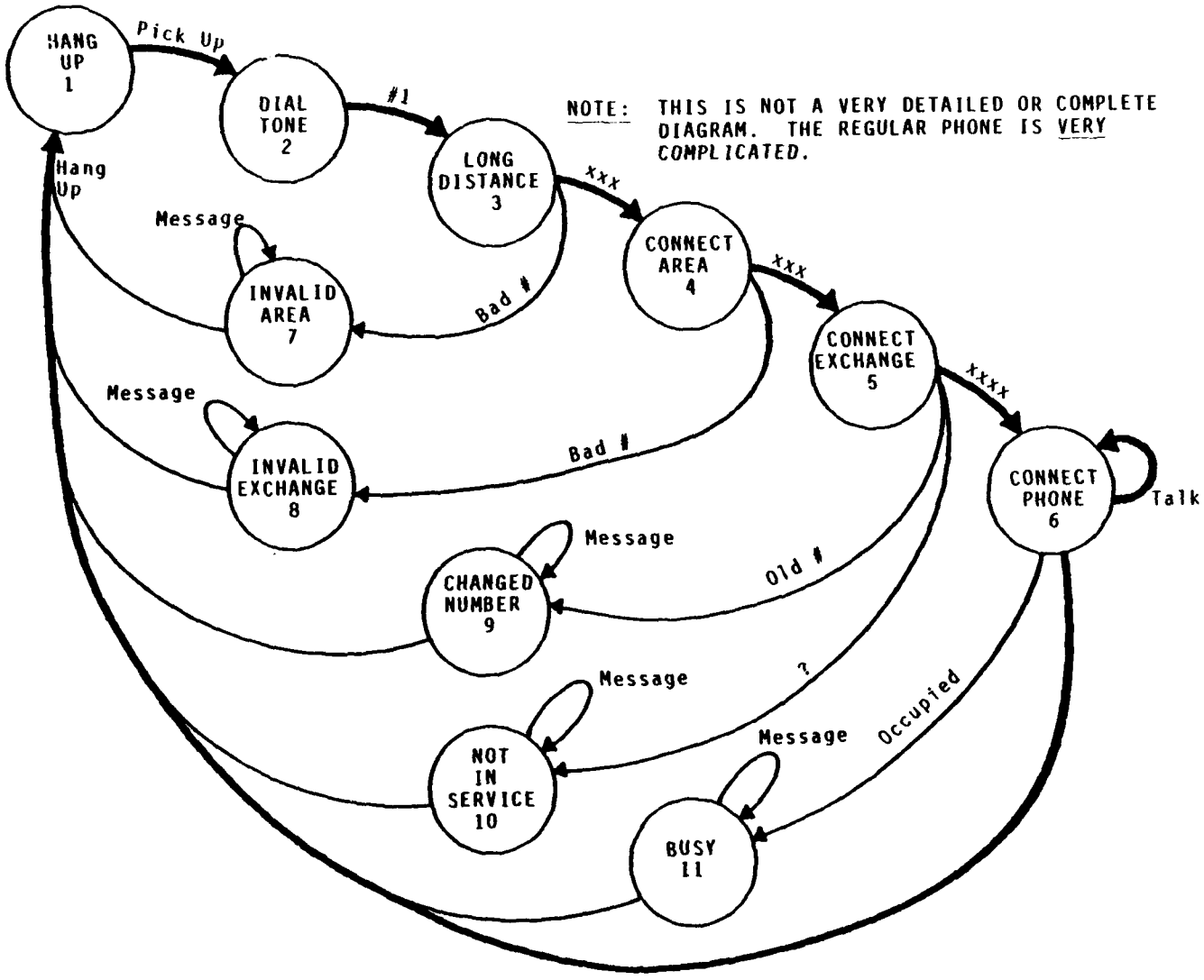
12.

13.

14.

15.

INSTRUCTOR NOTES



NOTE: THIS IS NOT A VERY DETAILED OR COMPLETE DIAGRAM. THE REGULAR PHONE IS VERY COMPLICATED.

DRAW THE DIAGRAM

INSTRUCTOR NOTES

REVIEW THE MAIN POINTS OF FINITE-STATE MAPS.

GIVE THE FOLLOWING ASPECTS AS SUGGESTIONS

- o DATA FLOW
- o DATA STRUCTURING
- o ALGORITHMS PROCESSING
- o CONTROL FLOW
- o CONTROL INTERACTION

SOME POSSIBLE ANSWERS:

1. CONTROL FLOW AND CONTROL INTERACTION ONLY.

SUMMARY AND REVIEW QUESTION

- A TECHNIQUE USED IN STRUCTURED PROGRAMMING
- MODELS SYSTEM BEHAVIOR
- CONCENTRATES ON STATES AND THEIR TRANSFORMATIONS
- WHAT ASPECTS OF A PROGRAM DOES FINITE STATE MACHINES METHOD HELP IDENTIFY?

INSTRUCTOR NOTES

Section 7

CORRECTNESS EXERCISE

INSTRUCTOR NOTES

CORRECTNESS MEANS A PROGRAM OR PROCEDURE WILL WORK FOR ANY (I.E., ALL!) COMBINATION OF INPUTS. REMIND THE STUDENTS THAT A PROGRAM MAY BE CORRECT BUT NOT BE USABLE. CORRECTNESS IS IMPORTANT TO LOWER DEBUG TIME AND COST. THE FOLLOWING EXAMPLE PROVIDES A TASTE OF CORRECTNESS.

SADT OVERVIEW

EXERCISE	METHODOLOGY	MODELING EMPHASIS	TECHNIQUE EMPHASIS
1	SADT	FUNCTIONAL	CONSTRAINT
2	ENTITY AND BACHMAN DIAGRAMMING	DATA	REAL-WORLD
3	OBJECT ORIENTED DESIGN	DATA	HIDING
4	STRUCTURED DESIGN	DATA	STRUCTURE
5	JACKSON	DATA THEN FUNCTION	STRUCTURE
6	FINITE-STATE	STATES	TRANSITIONS
7	CORRECTNESS	GUARDS	ASSERTIONS

INSTRUCTOR NOTES

TELL THE CLASS THE PROCEDURE WILL LOOP UNTIL THERE IS NO MORE LADDER TO CLIMB.

PROBLEM

WRITE A PROCEDURE (IN PSEUDO-ENGLISH) TO CLIMB A LADDER BY RAISING THE LOWER FOOT BY TWO RUNGS EACH TIME.

INSTRUCTOR NOTES

GO OVER THE TECHNIQUES, STRESSING THE IMPORTANCE OF COMPLETELY UNDERSTANDING THE PROBLEM.

CORRECTNESS TECHNIQUES

1. DRAW A PICTURE.

2. POSE AND ANSWER STANDARD QUESTIONS.

3. DOCUMENT ASSERTIONS.

4. WRITE PROCEDURE.

INSTRUCTOR NOTES

DRAW A PICTURE OF A LADDER AND FEET MOVING UP THE RUNGS.

ALSO REMIND THE STUDENTS ABOUT ASSERTIONS (COMMENT ABOUT WHAT HAPPENED SO FAR), LOOP INVARIANTS (THINGS THAT DON'T CHANGE IN A LOOP), AND GUARDS (THE OPPOSITE CONDITION FOR STOPPING). THIS WAY THEY WILL BE THINKING ABOUT THEM WHEN IT COMES TIME TO ANSWER THE QUESTIONS.

DRAW A PICTURE

INSTRUCTOR NOTES

THESE ARE QUESTIONS THAT NEED TO BE ANSWERED:

1. WHAT INITIALIZATION IS NEEDED? STAND IN FRONT OF LADDER. PUT ONE FOOT ON FIRST RUNG.

2. WHAT'S THE CONDITION FOR STOPPING? UNTIL THERE ARE NO MORE RUNGS TO CLIMB.

3. WHAT'S THE GUARD? WHILE THERE IS A RUNG TO CLIMB.

4. WHAT'S THE LOOP INVARIANT? ONE FOOT IS ALWAYS AHEAD OF THE OTHER.

5. WHAT'S THE STEP, I.E., WHAT PROGRESS IS MADE ON EACH ITERATION? MOVE THE LOWER FOOT TO THE NEXT EMPTY RUNG.

6. WHAT'S THE GOAL OF THE LOOP? TO GET TO THE TOP OF THE LADDER.

7. WHAT (IF ANYTHING) NEEDS TO BE DONE AFTER THE LOOP HAS FINISHED? MOVE THE LOWER FOOT TO THE RUNG WITH THE UPPER FOOT.

QUESTIONS TO ASK YOURSELF:

1. WHAT INITIALIZATION IS NEEDED? _____

2. WHAT'S THE CONDITION FOR STOPPING? _____

3. WHAT'S THE GUARD? _____

4. WHAT'S THE LOOP INVARIANT? _____

5. WHAT'S THE STEP, I.E., WHAT PROGRESS IS MADE ON EACH ITERATION? _____
_____

6. WHAT'S THE GOAL OF THE LOOP? _____

7. WHAT (IF ANYTHING) NEEDS TO BE DONE AFTER THE LOOP HAS FINISHED? _____
_____

NOW WRITE THE PROCEDURE ON THE NEXT PAGE ...

INSTRUCTOR NOTES

ASSERTIONS:

1. THERE MUST BE AN EMPTY RUNG TO CLIMB.

2. NORMAL CASE IS THAT ONE FOOT IS LOWER THAN THE OTHER.

3. EITHER FOOT MAY START.

4. ENDING IS SIGNIFIED BY BOTH FEET ON TOP RUNG.

DOCUMENT THE ASSERTIONS

1.

2.

3.

4.

5.

ANSWER:

CLIMB LADDER:

PLACE FEET TOGETHER ON GROUND FACING THE LADDER.
MOVE EITHER FOOT TO THE FIRST RUNG.
WHILE (THERE IS A RUNG TO CLIMB) LOOP
MOVE THE LOWER FOOT TO THE NEXT EMPTY RUNG.
END LOOP;
MOVE THE LOWER FOOT TO THE RUNG WITH THE UPPER FOOT.

GO OVER THE SOLUTION AND EXPLAIN WHY IT IS CORRECT.

WRITE THE PROCEDURE

INSTRUCTOR NOTES

REVIEW IMPORTANT ASPECTS OF PROGRAM CORRECTNESS ANALYSIS.

SOME POSSIBLE ANSWERS:

1. AS DESCRIBED IN THIS COURSE. THE FOLLOWING PHASES (SUBPHRASES) BENEFIT:

   - o IMPLEMENTATION - (CODING) - CHECK ON CORRECTIONS OF IMPLEMENTATION
   - o IMPLEMENTATION - (TESTING) - HELP DEVELOPMENT OF TEST CASES AND ANALYSIS OF PROBLEMS
   - o MAINTENANCE - HELPS IN UNDERSTANDING WHAT THE INTENT OF A PROGRAM IS

SUMMARY AND REVIEW QUESTION

- IT IS EASIER TO BUILD CORRECTNESS IN THAN TO ...
  - TEST IT IN
  - GET THE DEFECT OUT

- WHAT PHASES OR SUBPHASES OF THE SOFTWARE LIFE CYCLE BENEFIT FROM THE USE OF PROGRAM CORRECTNESS TECHNIQUES AND WHY?

END

FILMED

4-86

DTIC