

AD-A163 351

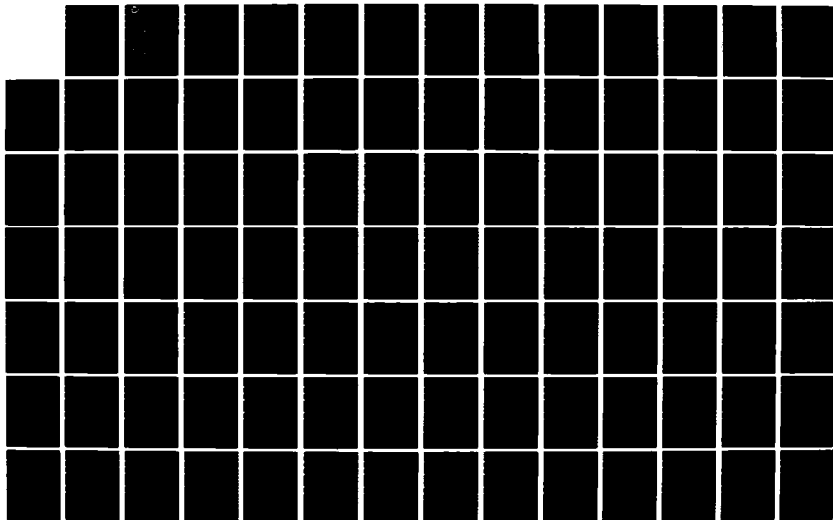
ADA (TRADEMARK) TRAINING CURRICULUM: ADA ORIENTATION
FOR MANAGERS L101 TEACHER'S GUIDE(U) SOFTECH INC
WALTHAM MA 1986 DAAB07-83-C-K514

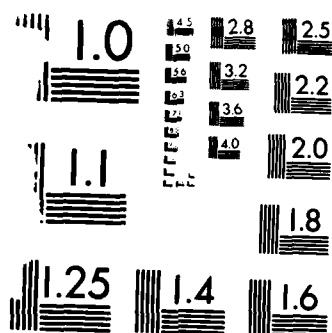
1/3

UNCLASSIFIED

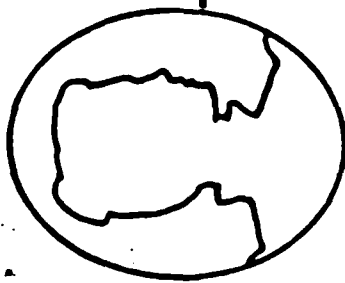
F/G 5/9

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A



Ada® Training Curriculum

1986



AD-A165 351

Ada® Orientation For Managers

L101

Teacher's Guide

86 3 11 146

*Suggested
A141846*

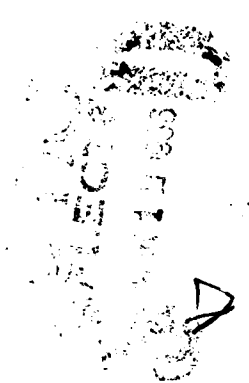
U.S. Army Communications-Electronics Command
(CECOM)

Contract DAAB07-83-C-K506

Prepared By:

SOFTECH, INC.

460 Totten Pond Road
Waltham, MA 02154



→ This document consists of the following sections:

INSTRUCTOR NOTES

A BRIEF SUMMARY OF WHAT WILL BE COVERED IN THIS MODULE. THE MODULE IS INTENDED TO TAKE A DAY. THE MATERIAL IN THE MODULE COVERS MORE THAN THE ADA LANGUAGE. IT IS THE INTENT TO PROVIDE AN APPRECIATION OF AND INFORMATION RELATED TO THE ENTIRE ADA EFFORT. IN THE COURSE OF THE DAY, EACH OF THESE QUESTIONS WILL BE ANSWERED.

TOPICS TO BE ADDRESSED:

- WHY ADA?
- WHAT ADA IS NOT
- WHAT ADA IS
- WHAT ARE SOME TRANSITION ISSUES WITH ADA,
- WHERE IS ADA NOW AND TOMORROW. ←

Copyright by SofTech, Inc. 1984. This material may be reproduced by or for the U.S. Government pursuant to the copyright license under DAR clause 7-104.9(a) (May 81).

ADA ORIENTATION FOR MANAGERS

1. Title		2. Date	
3. Location		4. Time	
5. Duration		6. Frequency	
7. Purpose		8. Special	
9. Other		10. Other	



INSTRUCTOR NOTES

THROUGHOUT THIS SECTION, EXPECT QUESTIONS ABOUT THE PRESENTED STATISTICS. MOST HAVE THE SOURCE LISTED IN THE INSTRUCTOR NOTES. TRY TO AVOID GETTING SIDETRACKED ON THESE ISSUES.

TAKE 45 MINUTES FOR THIS SECTION.

Section 1

Why Ada?

VG 722.1

INSTRUCTOR NOTES

THIS SECTION SETS THE HISTORICAL MOTIVATION FOR THE ADA EFFORT; WHY ALL THE INVESTMENT
OF TIME AND MONEY BY THE DoD.

TOPICS OUTLINE

WHY ADA?

WHAT ADA IS NOT

WHAT ADA IS

WHAT ARE SOME TRANSITION ISSUES WITH ADA

WHERE IS ADA NOW AND TOMORROW

INSTRUCTOR NOTES

A LIST THAT CHARACTERIZES THE PRESENT STATE OF SOFTWARE DEVELOPED FOR EMBEDDED COMPUTER SYSTEMS.

COMPLEX MILITARY SOFTWARE SYSTEMS ARE

- USUALLY LATE
- MORE EXPENSIVE THAN ORIGINALLY ESTIMATED
- NOT GOING TO WORK TO THE ORIGINAL SPECIFICATIONS
- UNRELIABLE
- DIFFICULT AND COSTLY TO MAINTAIN

THIS IS THE SOFTWARE CRISIS

INSTRUCTOR NOTES

THE FOLLOWING SLIDES DEPICT GRAPHICALLY AND IN LIST FORM RELATED PROBLEMS AND UNDERLYING CAUSES OF THE SOFTWARE CRISIS.

RELATED PROBLEMS

INSTRUCTOR NOTES

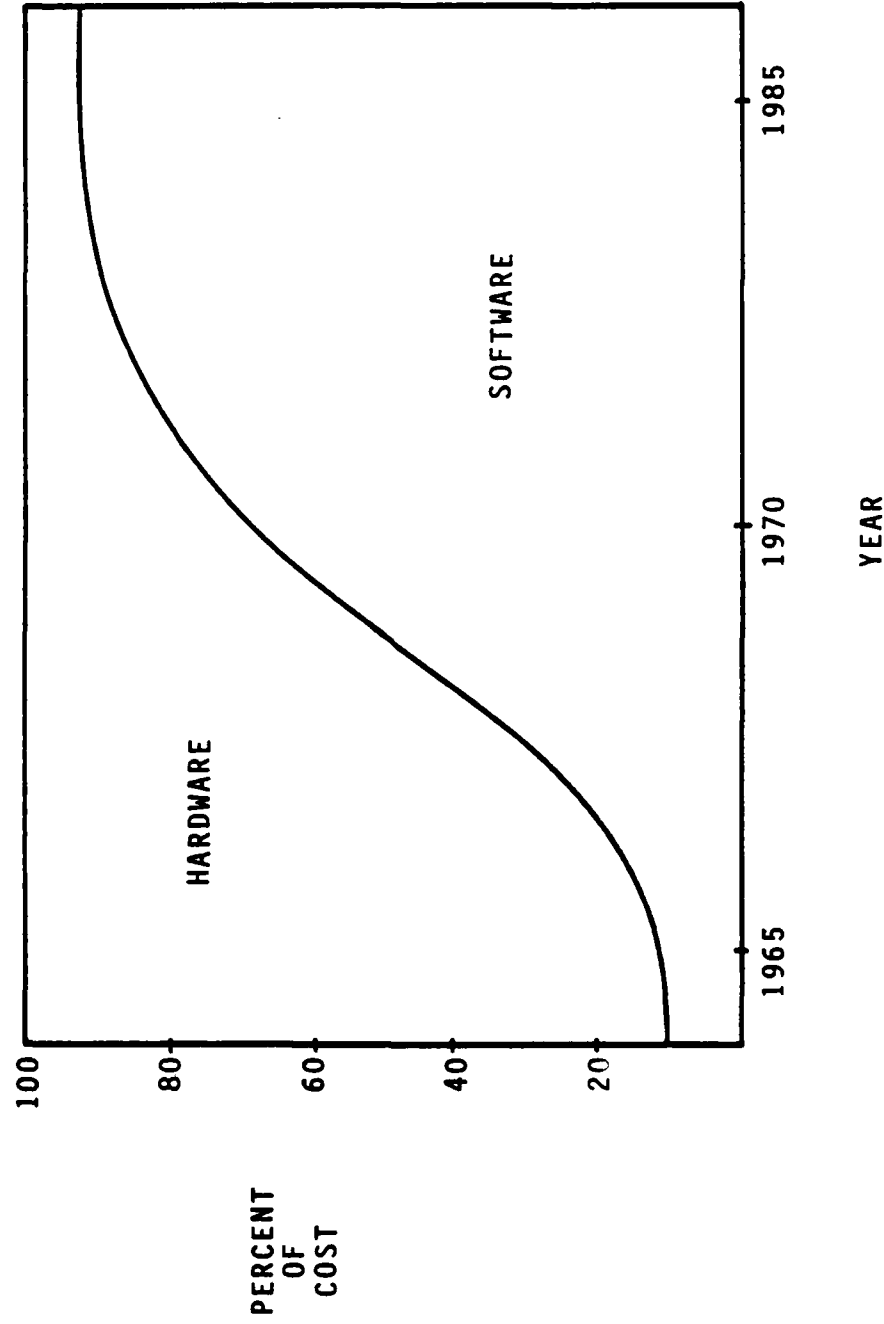
IN 1965, COST OF DEVELOPING A SOFTWARE SYSTEM WAS PRIMARILY A HARDWARE COST.

AROUND 1970 THIS BREAKDOWN OF TOTAL COST OF A SYSTEM WAS SPLIT FAIRLY EVENLY BETWEEN
HARDWARE AND SOFTWARE.

BUT SINCE THEN, SOFTWARE COSTS FOR A SYSTEM HAVE RISEN DRAMATICALLY WHILE HARDWARE COSTS
HAVE PLUMMETED AS A RESULT OF MICRO-CHIP TECHNOLOGICAL ADVANCES.

SOURCE: BARRY BOEHM, DEC. 1976, IEEE TRANSACTIONS

SOFTWARE COSTS INCREASING AS HARDWARE COSTS DECREASING

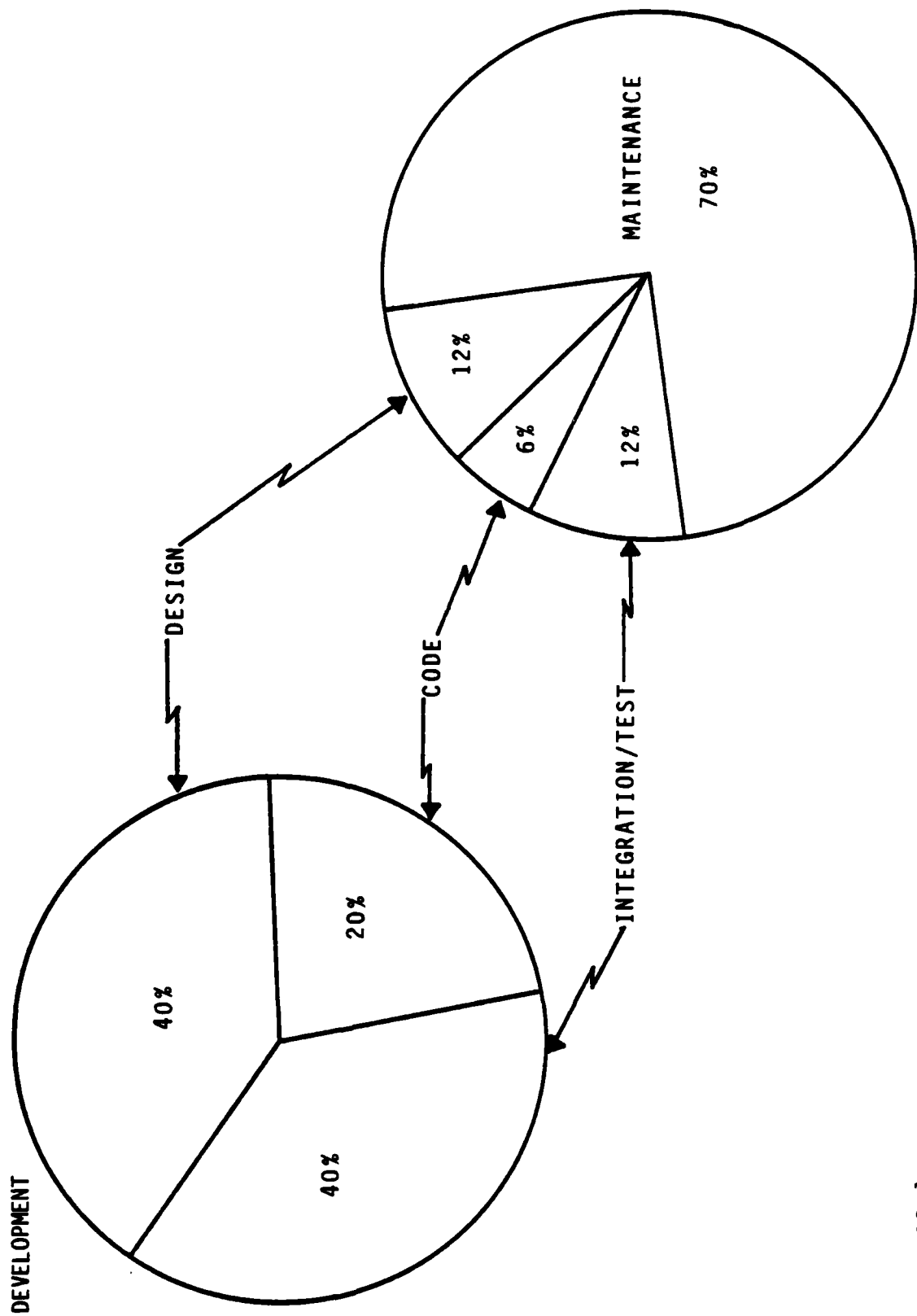


INSTRUCTOR NOTES

INCREASED SOFTWARE COSTS ARE THE SPECIFIC COST OF MAINTAINING/UPGRADING A SYSTEM ONCE IT IS OPERATIONAL.

SOURCE: MODEL ADA CURRICULUM BY GEORGIA TECH FOR THE GOVERNMENT
(BUT WAS PROBABLY NOT ORIGINAL TO THEM) -- PUBLIC INFO.

SOFTWARE MAINTENANCE NEARLY TRIPLES ORIGINAL DEVELOPMENT COSTS



INSTRUCTOR NOTES

AN ADDITIONAL COST WITH SOFTWARE LIES IN ERROR DETECTION AND CORRECTION.

FOR EXAMPLE:

IF A REQUIREMENTS ERROR IS FOUND AND CORRECTED DURING THE REQUIREMENTS PHASE, YOU CAN JUST CORRECT THE REQUIREMENTS DOCUMENT WITH LITTLE COST IMPACT OF THE ERROR.

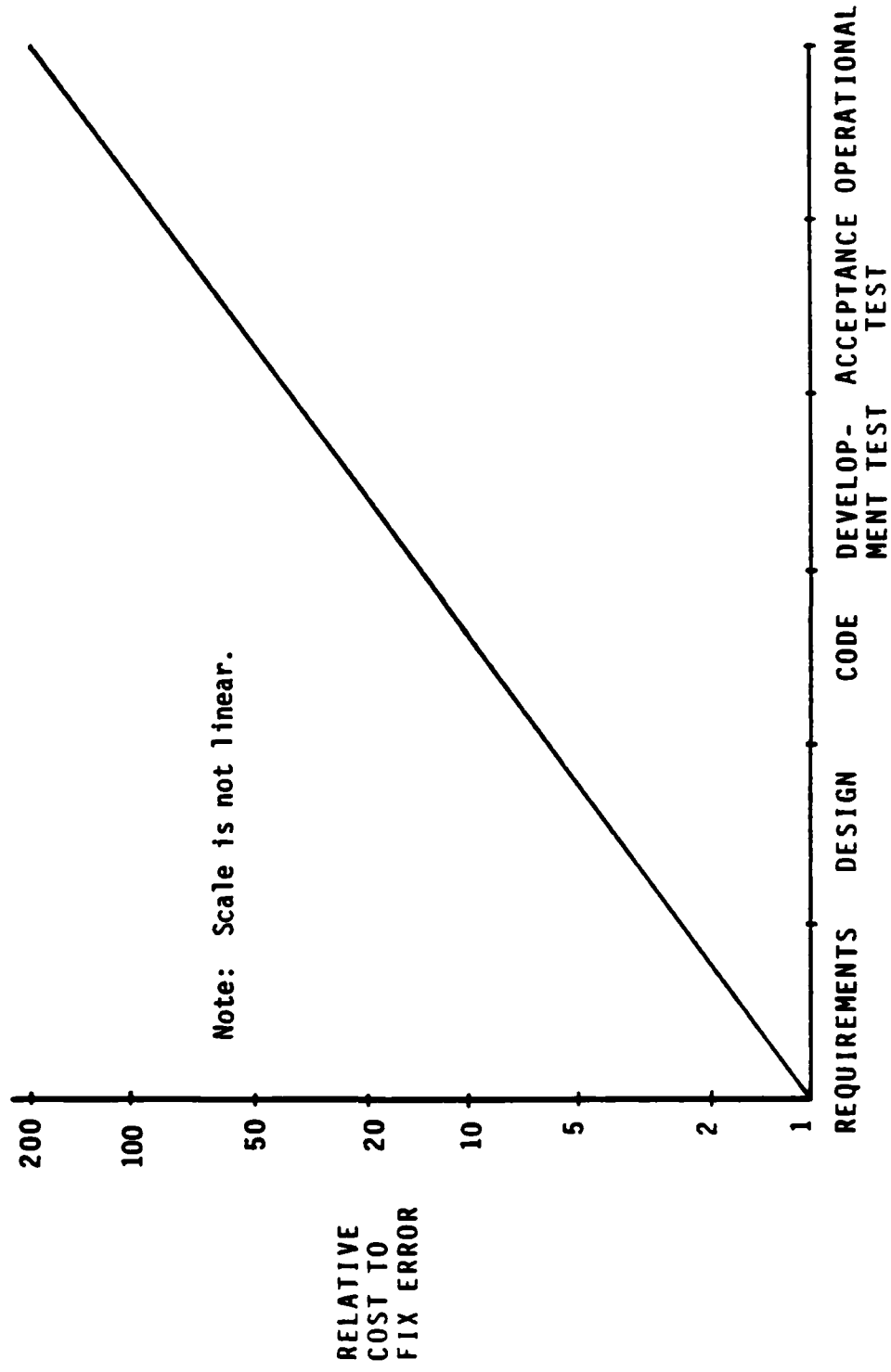
IF THE SAME ERROR IS NOT FOUND AND CORRECTED UNTIL MAINTENANCE, THE CORRECTION INVOLVES NOT ONLY DOCUMENT CHANGES SUCH AS SPECIFICATIONS, USER MANUALS, TRAINING MANUALS, BUT WILL ALSO INVOLVE VARIOUS AMOUNTS OF CODE MODIFICATIONS AND REVALIDATION. ERROR CORRECTION AT THIS POINT IN THE LIFE CYCLE IS TYPICALLY 100 TIMES WHAT IT WOULD HAVE BEEN IN THE REQUIREMENTS PHASE.

SOURCE: B. BOEHM, 1981, SOFTWARE ENGINEERING ECONOMICS
DATA IS FROM STUDIES BY IBM, TRW, GTE ON THIS TOPIC

VG 722.1

1-6i

COST OF ERROR CORRECTION



PHASE ERROR DETECTED AND CORRECTED

OTHER RELATED PROBLEMS

- SOFTWARE TASKS ARE NOW MORE COMPLEX WITH NO ADEQUATE TOOLS TO DEAL WITH THE PROBLEM
- SUPPORT TOOLS (ASSEMBLERS, LINKERS, DEBUGGER) MUST BE DEVELOPED FOR EACH LANGUAGE AND MACHINE
- LACK OF ADEQUATE MANAGEMENT AND SOFTWARE DEVELOPMENT TOOLS

OTHER RELATED PROBLEMS

- SOFTWARE IS NOT REUSABLE ON DIFFERENT SYSTEMS
- PROLIFERATION OF LANGUAGES AND ARCHITECTURES
- LANGUAGES NOT SUITED FOR CURRENT APPLICATIONS
- SUPPLY OF QUALITY SOFTWARE PERSONNEL NOT ABLE TO MEET
CURRENT SOFTWARE DEMANDS

INSTRUCTOR NOTES

DoD IS CONCERNED WITH MORE THAN JUST SAVING MONEY FROM ITS OWN BUDGET. THIS HAS NATIONAL AND INTERNATIONAL RAMIFICATIONS - AMERICAN TECHNOLOGICAL SUPERIORITY IS BEING CHALLENGED DAILY BY OTHER COUNTRIES: POWER AND MONEY CONSEQUENCES.

THE SOFTWARE CRISIS IS MORE THAN A DoD CONCERN

INTERNATIONAL IMPLICATIONS OF THE SOFTWARE CRISIS:

- LOSS OF COMPUTER TECHNOLOGY INDUSTRY LEAD
- LOSS OF REVENUE

INSTRUCTOR NOTES

TO COMBAT THIS, DoD HAS INSTIGATED A PLAN TO STANDARDIZE ARCHITECTURES (INSTRUCTION SETS), A PROGRAMMING LANGUAGE, AND A SUPPORT ENVIRONMENT.

IT IS A RETHINKING OF THE WAY IN WHICH SOFTWARE SYSTEMS WILL BE DEVELOPED IN THE FUTURE WITH THE ITEMS LISTED AS VEHICLES OF THAT CHANGE.

BY MODERN SOFTWARE ENGINEERING PRINCIPLES WE MEAN SUCH CONCEPTS AS ABSTRACTION, MODULARITY, INFORMATION HIDING.

DoD's RESPONSE

- A STANDARD LANGUAGE - THE ADA LANGUAGE DEVELOPMENT
- A STANDARD ENVIRONMENT - THE ADA PROGRAMMING SUPPORT ENVIRONMENT DEVELOPMENT
- INCORPORATION OF MODERN SOFTWARE ENGINEERING PRINCIPLES INTO SOFTWARE DEVELOPMENT METHODS AND THE PROGRAMMING LANGUAGE
- MIL-STANDARDS AND DIRECTIVES TO MANDATE CHANGES
- PROCUREMENT "INCENTIVES" TO ENCOURAGE CHANGES

INSTRUCTOR NOTES

THE APPROACH TO THE ADA DESIGN WAS INNOVATIVE. A LIFE-CYCLE APPROACH WAS THEN TAKEN. THE ADA LANGUAGE CAN BE VIEWED AS A PRODUCT MUCH LIKE BUILDING A MISSILE: FROM ANALYSIS OF A PROBLEM AND POSSIBLE SOLUTION, THROUGH REQUIREMENTS (IN THE SERIES OF LANGUAGE REQUIREMENT SPECS), TO OPERATIONAL (WITH ACTUAL COMPILER DEVELOPMENT AND VALIDATION).

IMPORTANT TO NOTE THAT THROUGHOUT THE PROCESS, UNIVERSITIES, INDUSTRY AND COMPILER IMPLEMENTORS WERE SOLICITED FOR INPUT (REVIEWS, OPINIONS).

DEVELOPMENT OF ADA LANGUAGE

ANALYSIS	1970-1975	IDENTIFICATION OF SOFTWARE PROBLEMS IN EMBEDDED MILITARY SYSTEMS (THE CRISIS)
REQUIREMENTS	1975-1977	STRAWMAN, WOODENMAN, TINMAN LANGUAGE REQUIREMENTS SPECIFICATIONS
		HOLWG: HOL REQUIREMENTS FOR EMBEDDED SYSTEMS DEFINED
		EXISTING LANGUAGES EVALUATED
		RESULTS: ONE LANGUAGE IS SUFFICIENT
		NO EXISTING LANGUAGE SATISFIES ALL REQUIREMENTS
		AN EXISTING LANGUAGE SHOULD BE USED AS A BASE
DESIGN		
PHASE I	1977-1978	PRELIMINARY LANGUAGE DESIGN - IRONMAN (RED, BLUE, YELLOW, GREEN)
PHASE II	1978-1979	FORMAL LANGUAGE DEFINITION - STEELMAN (RED, GREEN)
PHASE III	1979-1980	FINAL LANGUAGE DEFINITION BY CII HONEYWELL/BULL

INSTRUCTORS NOTES

COMPILER VALIDATION INSTITUTED TO RESTRICT THE PROLIFERATION OF ADA DIALECTS. ADA COMPILERS MUST BE VALIDATED EITHER YEARLY OR WHEN A NEW VERSION IS RELEASED. VALIDATION IS PERFORMED BY THE ADA VALIDATION OFFICE (PART OF THE ADA JOINT PROGRAM OFFICE). VALIDATION ONLY SAYS THAT A COMPILER CONFORMS TO THE ANSI STANDARD LANGUAGE DEFINITION OF ADA. IT DOES NOT SAY HOW EFFICIENT A COMPILER IS FOR A GIVEN APPLICATION AREA.

DR. DELAUER'S PROCLAMATION MANDATES THE USE OF ADA ON ALL NEW PROJECTS AFTER 1 JANUARY 1984. THE DoD IS TRYING TO DEMONSTRATE ITS SERIOUSNESS IN THE ADA EFFORT.

THE TOTAL NUMBER OF VALIDATED COMPILERS COVERS 11 VENDORS AND MANY COMBINATIONS OF HOST AND TARGET COMPUTERS. THERE ARE 4 VAX 11/750 SYSTEMS AND 7 VAX 11/780, 782, 785 SYSTEMS, TO NAME THE MOST COMMON COMPUTER.

LANGUAGE DEVELOPMENT (CONTINUED)

TESTING	1980-1982	LANGUAGE REFINEMENT BY INTERNATIONAL REVIEWERS COMPILER VALIDATION TEST FACILITY ANSI STANDARDIZATION REQUESTED
OPERATIONAL	1982 —→	COMPILER DEVELOPMENT BY DoD, PRIVATE INDUSTRY, ACADEMIA
		PARALLEL PROJECTS
	FEB 1983	ANSI STANDARDIZATION OF ADA LANGUAGE
	MAR 1983	NYU (ADA/ED) VALIDATED TRANSLATOR
	JUN 1983	ROLM VALIDATED COMPILER DR. DELAUER'S PROCLAMATION
	DEC 1984	ALS VALIDATED
	OCT 1985	35 VALIDATED COMPILERS

INSTRUCTOR NOTES

SIMILAR FORMAT AS THE LANGUAGE.

OF NOTE: THE SPECIFICATION FOR THE ENVIRONMENTS IS NOT AS RIGOROUS AS FOR THE LANGUAGE
SINCE WE KNOW LESS OF WHAT SHOULD BE IN AN ENVIRONMENT.

MAIN ENVIRONMENT PROJECTS: ALS (ADA LANGUAGE SYSTEM)

AIE (ADA INTEGRATED ENVIRONMENT)

- IN 1985, AIE WAS DOWNGRADED TO ACS (ADA COMPILATION SYSTEM)

DEVELOPMENT OF ADA ENVIRONMENTS

ANALYSIS	1977-1978	LANGUAGE ALONE NOT SUFFICIENT TO IMPROVE SOFTWARE DEVELOPMENT
REQUIREMENTS	1978-1979	PRELIMINARY ENVIRONMENT REQUIREMENTS (SANDMAN, PEBBLEMAN)
DESIGN	1980	FORMAL ENVIRONMENT DEFINITION (STONEMAN)
IMPLEMENTATION	1981 —————>	COMPILER PLUS ENVIRONMENT DEVELOPMENT PROJECTS FUNDED BY DoD, PRIVATE INDUSTRY, UNIVERSITIES
TESTING	1982 —————>	KAPSE INTERFACE TEAM (KIT)/KIT FOR INDUSTRY AND ACADEMIA (KITIA): TASK IS TO DEFINE STANDARD INTERFACES FOR ALS AND AIE
OPERATIONAL/ MAINTENANCE	1983 —————>	

INSTRUCTOR NOTES

THIS SLIDE IS INCLUDED BECAUSE MANY STUDENTS WANT TO KNOW, "WHAT DOES ADA STAND FOR?"

CHARLES BABBAGE WAS AN INVENTOR LIVING IN THE EARLY 1800'S. HE IS CREDITED WITH DEVELOPING ON PAPER THE FIRST COMPUTER, THE DIFFERENCE ENGINE.

YOU MAY POINT OUT THAT SOME PEOPLE THINK "ADA" STANDS FOR "ANOTHER DAMN ACRONYM."

WHY ADA?

- THE LANGUAGE WAS NAMED IN HONOR OF ADA AUGUSTA, THE COUNTESS OF LOVELACE, CONSIDERED TO BE THE FIRST PROGRAMMER.

-- ASSOCIATE OF CHARLES BABBAGE

-- WROTE AN ALGORITHM FOR BABBAGE TO COMPUTE BERNOULLI NUMBERS

-- THE NAME IS NOT AN ACRONYM!

INSTRUCTOR NOTES

Section 2

What Ada is Not

INSTRUCTOR NOTES

POSSIBLE APPROACH WOULD BE TO ASK THE STUDENT WHAT THE TERM "ADA" MEANS TO THEM BEFORE
SHOWING THE NEXT SLIDE.

ALLOW 15 MINUTES FOR THIS SECTION.

TOPICS OUTLINE

WHY ADA?

WHAT ADA IS NOT

WHAT ADA IS

WHAT ARE SOME TRANSITION ISSUES WITH ADA

WHERE IS ADA NOW AND TOMORROW

INSTRUCTOR NOTES

PURPOSE IS TO DISPEL SOME POSSIBLE MISCONCEPTIONS REGARDING ADA.

ADA...

- ALONE WILL NOT SOLVE YOURS OR DoD's SOFTWARE PROBLEMS
- ALONE WILL NOT INCREASE PRODUCTIVITY
- IS NOT ANOTHER PROGRAMMING LANGUAGE LIKE FORTRAN, COBOL OR PASCAL
- IS NOT "JUST" A LANGUAGE

INSTRUCTOR NOTES

Section 3

What Ada Is

INSTRUCTOR NOTES

WHAT DO WE MEAN BY THE TERM 'ADA'?

ALLOW 3 HOURS 10 MINUTES FOR THIS SECTION:

30 MINUTES -- UP TO PROGRAMMING WITH ADA

90 MINUTES -- TO CATALOGUE OF ADA FEATURES

70 MINUTES -- TO END OF SECTION 3.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

TOPICS OUTLINE

WHY ADA?

WHAT ADA IS NOT

WHAT ADA IS

WHAT ARE SOME TRANSITION ISSUES WITH ADA

WHERE IS ADA NOW AND TOMORROW

INSTRUCTOR NOTES

THIS SECTION ATTEMPTS TO GIVE A "FEEL" FOR WHAT IS MEANT BY THE ADA EFFORT. A LANGUAGE OVERVIEW THROUGH AN EXAMPLE IS PRESENTED TO PROVIDE MANAGERS A BRIEF LOOK AT ADA PROGRAMMING, PLUS A CATALOGUE TO SUMMARIZE THE SCOPE OF THE LANGUAGE AND EACH FEATURE'S IMPORTANCE. FINALLY A BRIEF BACKGROUND OF ADA ENVIRONMENTS IS PRESENTED, AGAIN PRIMARILY FOR JARGON.

THE INTENT HERE IS NOT TO TEACH MANAGERS HOW TO PROGRAM IN ADA. IT IS TO PROVIDE SOME FAMILIARITY WITH THE ADA JARGON THEY ARE LIKELY TO ENCOUNTER IN READING AND IN DEALING WITH PROGRAMMERS/DESIGNERS.

WHAT IS ADA

- A DEFINITION
- LANGUAGE OVERVIEW
- ENVIRONMENT OVERVIEW

INSTRUCTOR NOTES

IT IS IMPORTANT TO STRESS THAT ADA IS NOT JUST A LANGUAGE -- IT IS A TOOL, LIKE LINKERS, DEBUGGERS, OR METHODOLOGIES, TO DEAL WITH SOFTWARE PRODUCTIVITY.

RELIABLE SOFTWARE IMPLIES THE SOFTWARE PRODUCT WAS BUILT USING METHODS THAT DECREASE THE LIKELIHOOD OF ERRORS IN ANALYSIS, DESIGN, AND CODE. IT ALSO MEANS OUR PRODUCT CAN RECOVER FROM ERRORS OR FAILURE CONDITIONS DURING OPERATION.

MAINTAINABLE SOFTWARE IMPLIES THAT THE STRUCTURE AND ORGANIZATION OF THE SYSTEM ARE CLEAR. THUS MODIFYING THE SYSTEM IS RELATIVELY EASY AND THE CHANGE DOES NOT CAUSE NEW ERRORS.

COST REDUCTION OCCURS ONLY OVER THE LIFE OF THE PRODUCT. WE ARE PRIMARILY CONCERNED WITH PROJECTS OF LONG DURATION WHICH WILL BE MODIFIED AND ENHANCED CONTINUALLY. THERE IS NO COST SAVINGS DURING DEVELOPMENT (IN FACT, THERE COULD BE A COST INCREASE).

PORTABLE SOFTWARE IMPLIES THAT A SOFTWARE PRODUCT (OR COMPONENT) DEVELOPED ON ONE MACHINE ARCHITECTURE CAN BE MOVED TO A DIFFERENT SYSTEM AND CONTINUE TO PERFORM THE SAME.

ADA IS A TOOL

- TO HELP DESIGNERS DESIGN BETTER SYSTEMS
- TO HELP PROGRAMMERS CODE BETTER SYSTEMS
- TO HELP MANAGERS WITH CONFIGURATION MANAGEMENT
- TO DEVELOP SOFTWARE THAT IS
 - RELIABLE
 - MAINTAINABLE
 - LESS COSTLY OVER THE LIFE CYCLE
 - PORTABLE

INSTRUCTOR NOTES

AGAIN, THE POINT IS THAT THE ADA EFFORT IS THE COMBINED WORKINGS OF THE POINTS STATED ON THE SLIDE. IT PROVIDES A CHANCE TO RE-EVALUATE AND UPGRADE OUR METHODS OF SOFTWARE DEVELOPMENT.

ADA IS A COMBINATION OF

- A COMMON HIGH ORDER LANGUAGE THAT SUPPORTS MODERN SOFTWARE ENGINEERING
- COMMON SUPPORT TOOLS IN A PROGRAMMING DEVELOPMENT SUPPORT ENVIRONMENT

INSTRUCTOR NOTES

THE LANGUAGE OVERVIEW CONSISTS OF THREE SECTIONS: 1) AN INTRODUCTION TO HIGH ORDER LANGUAGES IN GENERAL; 2) WHAT ITS LIKE TO PROGRAM IN ADA; AND 3) A FORMAL CATALOGUE OF ADA FEATURES TO SUMMARIZE THE ENTIRE LANGUAGE.

SKIM OVER 1 IF STUDENTS ALREADY HAVE THE NECESSARY BACKGROUND IN HOLs.

LANGUAGE OVERVIEW

- COMPILER AND HOL BACKGROUND
- ADA PROGRAMMING EXAMPLE
- CATALOGUE OF ADA FEATURES

INSTRUCTOR NOTES

WHAT IS MEANT BY HOLS AND WHAT ARE THEIR ADVANTAGES.

COMPARISON OF HOL'S TO ASSEMBLY LANGUAGES BY EXAMPLE:

LET'S SAY WE WANT TO EXPRESS $A + B = C$. WHAT ONE MIGHT SEE IN AN HOL AND SOME ASSEMBLY LANGUAGE ARE SHOWN. NOTICE THAT THE HOL HAS MORE CLOSELY MAPPED OUR ORIGINAL EXPRESSION AND INTENT THAN THE ASSEMBLY LANGUAGE. BECAUSE OF THIS QUALITY OF HOL'S WE CAN REALIZE THE LISTED EXAMPLES.

HIGH ORDER LANGUAGES (HOLS)

- ALLOW PROGRAMMERS TO WRITE SOFTWARE IN A MORE ENGLISH-LIKE FORM

HOL: SET C = A + B

ASSEMBLY LANGUAGE: LOAD A

ADD B

STORE C

- ADVANTAGES

PROGRAMS EASIER TO READ

LANGUAGES FASTER TO LEARN

ALGORITHMS EXPRESSED MORE NATURALLY (AND MORE ACCURATELY)

LIFE CYCLE COSTS LOWER

INSTRUCTOR NOTES

PURPOSE IS TO PROVIDE A BRIEF PICTURE OF COMPILERS, SOME RELATED JARGON AND CONCEPTS.
ITEM TO NOTE: (BESIDES COVERING SLIDE CONTEXT) A COMPILER IS A PROGRAM LIKE A SOFTWARE PROGRAM THAT GUIDES A MISSILE. THIS IDEA IS IMPORTANT AS MANY MANAGERS MAY BE FACED WITH USING IMMATURE COMPILERS ON THEIR PRODUCTION PROJECTS. THEY MAY BE ABLE TO DEAL WITH THE SITUATION MORE EFFECTIVELY IF THEY REMEMBER THE MATURITY PROBLEMS OF THE SOFTWARE THEY ARE DEVELOPING.

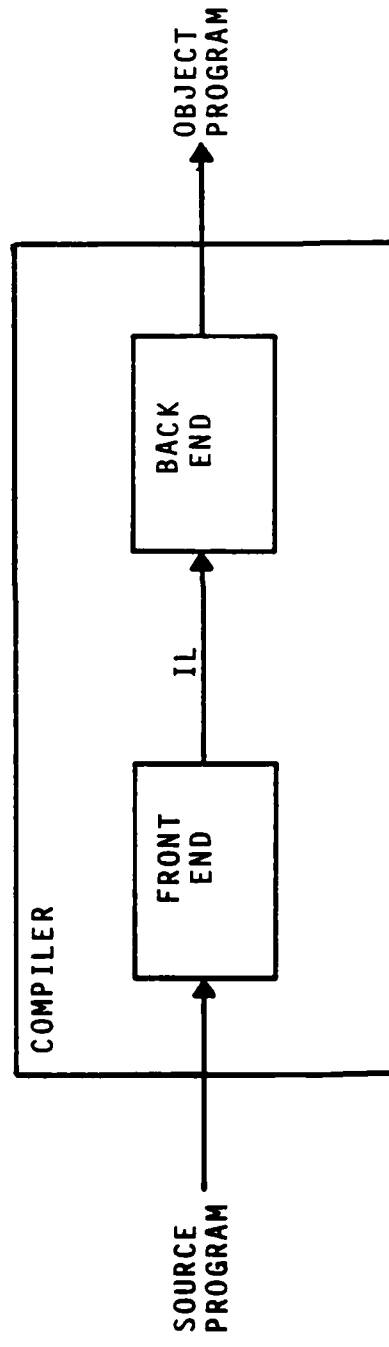
EXPECT SOME QUESTIONS SUCH AS, "HOW FAST IS AN ADA COMPILER COMPARED TO LANGUAGE X?", "WILL IT OPTIMIZE EFFICIENTLY?", AND "WHAT COMPILERS ARE AVAILABLE?" IF YOU HAVE NO STATISTICS IN THIS AREA, TELL THE STUDENTS THAT YOU DO NOT KNOW FOR SURE AND WHAT STATISTICS ARE AVAILABLE DO NOT REFLECT THE STATE OF ADA COMPILERS TO COME.

COMPILERS

- ARE COMPUTER PROGRAMS THAT TRANSLATE HOL PROGRAMS (THE SOURCE) INTO FUNCTIONALLY EQUIVALENT MACHINE LANGUAGE PROGRAMS (THE OBJECT)
- GENERALLY HAVE TWO MAJOR PARTS:
 - THE FRONT END THAT DETERMINES THE GRAMMATICAL STRUCTURE AND GENERATES AN INTERMEDIATE LANGUAGE (IL)
 - THE BACK END OR CODE GENERATOR

COMPILERS

- PICTORIALY:



OPTIMIZATIONS AT THE FRONT END AND BACK END HELP TO PRODUCE MORE EFFICIENT CODE BY REMOVING REDUNDANT CODE AND EXPRESSIONS

INSTRUCTOR NOTES

LAST TWO BULLETS ARE IMPORTANT TO NOTE.

DEFINE WHAT ABSTRACTION IS: FOR A GIVEN LEVEL OR VIEWPOINT, WE LOOK AT THE NECESSARY PROPERTIES BUT IGNORE (FOR THE MOMENT) THE UNNECESSARY PROPERTIES.

AN EXAMPLE: MESSAGE SWITCH SYSTEM

CAN BE VIEWED AS MESSAGES COMING INTO A PROCESSING AREA AND MESSAGES ROUTED AFTER PROCESSING (WE IGNORE HOW THE MESSAGE COMES IN AND GOES OUT, AND WHAT EXACTLY ARE THE PROCESSING STEPS)

OR, CAN BE VIEWED AS HARDWARE RECEIVER/TRANSMITTERS, WHICH CONVERT SERIAL/PARALLEL SIGNALS INTO PARALLEL/SERIAL SIGNALS, RECEIVING MESSAGES WHICH ARE QUEUED FOR PROCESSING ... (THIS IS A LOWER LEVEL OF ABSTRACTION)

EXAMPLES FOR LEVELS OF ABSTRACTION:

MACHINE	11010001
ASSEMBLY LANGUAGE	ST A, B
FORTRAN I	A = B+C**D/E
ALGOL 60	IF <COND A> THEN <DO SOMETHING> ELSE <DO ANOTHER>
PASCAL	DATA RECORDS
ADA	DATA STRUCTURES, E.G. QUEUES, STACKS

LANGUAGE EVOLUTION

- PROGRAMMING LANGUAGE DEVELOPMENT VIEWED AS INCREASING LEVELS OF ABSTRACTION

MACHINE LANGUAGE	NO ABSTRACTION
ASSEMBLY LANGUAGE	BIT ABSTRACTION
FORTRAN I, ALGOL 58	EXPRESSION ABSTRACTION
FORTRAN II, COBOL, ALGOL 60	CONTROL ABSTRACTION
PASCAL, PL/I, JOVIAL J73	DATA ABSTRACTION
ADA	DATA/ALGORITHM ABSTRACTION

- WITH EACH INCREASE IN ABSTRACTION, WE GAIN MORE POWERFUL TOOLS TO MORE ACCURATELY EXPRESS THE "REAL WORLD" IN OUR SOFTWARE ALGORITHMS. THIS ALLOWS US TO DEAL MORE EFFECTIVELY WITH INCREASINGLY COMPLEX PROBLEMS.
- OTHER LANGUAGES HAVE INDIVIDUAL FEATURES/CONSTRUCTS OF ADA, BUT ADA WAS DESIGNED TO HAVE IT "ALL" IN ONE PLACE

INSTRUCTOR NOTES

TAKE A 15-MINUTE BREAK HERE.

WHAT IS IT LIKE TO PROGRAM IN ADA?

INSTRUCTOR NOTES

THE PURPOSE OF THE EXAMPLE IS TO ILLUSTRATE WHAT IT'S LIKE TO WRITE AN ADA PROGRAM FROM BEGINNING TO END. THIS ALLOWS MANAGERS AN APPRECIATION OF THE PROCESS IN ADA. THIS EXAMPLE IS VERY ELEMENTARY BUT BECAUSE OF THAT, THE STUDENT CAN CONCENTRATE ON THE ADA AND NOT THE ALGORITHMS. THE FORMAT IS TO PARALLEL SOFTWARE DEVELOPMENT. FIRST DECOMPOSE THE PROBLEM FROM THE TOP, DOWN THROUGH SPECIFIC ALGORITHMS TO THE CONTROL STRUCTURE LEVEL. AFTER THUS ANALYZING THE PROBLEM, THE ADA CODE IS BUILT FROM THIS POINT BACK UP TO A COMPLETE ADA SYSTEM. THE ADA SYNTAX IS TOTALLY BY EXAMPLE (I.E. OSMOSIS). ADDITIONAL GOALS ARE TO GENERATE A FAMILIARITY WITH ADA, THE EASE WITH WHICH IT CAN BE READ, AND TO CREATE A NON-THREATENING APPRECIATION FOR THE LANGUAGE. TO BUILD THE ADA SYSTEM, WE START FIRST WITH CONTROL STRUCTURES, AS ACTION STATEMENTS IN ADA ARE VERY SIMILAR TO OTHER LANGUAGES. THE STATEMENT CODE FRAGMENTS ARE SIMILAR TO WHAT WILL BE USED IN THE FINAL CODE. IN THIS WAY THE RATIONALE IS SET FOR TYPES AND OBJECTS. NEXT, A LOOK AT TYPE AND OBJECT DECLARATIONS. AGAIN ACTUAL CODE RELATED TO THE EXAMPLE IS USED. CODE COMMENTS PROVIDE EXPLANATIONS OF THE ADA THUS AFTER THE COURSE IS FINISHED, THE STUDENT CAN REFER BACK TO THE COURSE NOTES WITH UNDERSTANDING. THE EXAMPLE NOW BUILDS TO ADA SUBPROGRAMS AND PARAMETERS. AT THIS POINT, THE COMPLETED CODE IS PRESENTED FOR ALL PROCEDURES AND FUNCTIONS. NEXT, THESE RESOURCES ARE COLLECTED INTO AN ADA PACKAGE. ADA PROVIDES THE FACILITIES TO CREATE OUR OWN USAGE PACKAGES. THIS BUILDS AN INTUITIVE FEEL FOR THE USEFULNESS OF THE PACKAGE CONCEPT IN ADA. FINALLY, THE MAIN LOGIC PROCEDURE IS PRESENTED WHICH USES THE RESOURCES OF TWO PACKAGES. WITHIN THE MAIN PROCEDURE, A SIMPLE I/O FORMAT IS PRESENTED TO ILLUSTRATE BOTH THE ABILITY TO CREATE ONE'S OWN I/O ROUTINES, SPECIALLY TAILORED, AND TO ALSO SHOW THE USE OF THE 'GET' AND 'PUT' PROCEDURES. AS A WHOLE THE ADA EXAMPLE ILLUSTRATES A BASIC PROGRAM STRUCTURE - I.E. A MAIN DRIVER PROCEDURE USING RESOURCES FROM ONE OR MORE PACKAGES WITH THE PACKAGES IN TURN CONSISTING OF NESTED SUBPROGRAMS. AS PART OF CODING ADA, THE SYSTEM MUST BE COMPILED TO TRANSLATE THE SOURCE TO OBJECT CODE FOR EVENTUAL EXECUTION. COMPILATION AND THE PROGRAM LIBRARY ARE PRESENTED FOLLOWED BY TWO EXAMPLES OF SYSTEM CHANGE.

IT IS CRUCIAL FOR THE INSTRUCTOR TO SET UP THE PURPOSE OF THIS EXAMPLE. OTHERWISE, CONTINUAL SYNTAX QUESTIONS MAY ARISE. (THIS MAY HAPPEN ANYWAY. IF SO, GENTLY REMIND THEM OF THE PURPOSE.)

EXAMPLE 1

A SYSTEM THAT RECORDS AND TRACKS TWO-DIMENSIONAL MOVEMENT ON A RADAR SCREEN NEEDS A PROCEDURE THAT, GIVEN THE LAST POSITION RECORDED, THE CURRENT POSITION, THE TIME BETWEEN THOSE READINGS, AND A NEW TIME INTERVAL, WILL PREDICT WHERE THE NEXT POINT SHOULD OCCUR. THE PREDICTION WILL ASSUME THAT NO CHANGE IN SPEED OR DIRECTION WILL OCCUR; THE VALUE THUS OBTAINED MIGHT LATER BE COMPARED TO THE ACTUAL READING TO DETERMINE PATTERNS OF CHANGE IN EITHER FACTOR. THE TRACKING PROGRAM THUS NEEDS ACCESS TO A NEXT-POINT CALCULATION ROUTINE, WHICH SHOULD BE ASSOCIATED WITH FACILITIES TO CALCULATE THE DISTANCE BETWEEN TWO POINTS AND TO DETERMINE VELOCITY. DUE TO THE SPECIFICS OF THE SYSTEM, A VENDOR-SUPPLIED PACKAGE CONTAINING SUCH ROUTINES WOULD BE UNSUITABLE.

INSTRUCTOR NOTES

OUR EXAMPLE PROCESS

STATEMENT OF REQUIREMENTS (COMPLETED)

DECOMPOSITION OF SOLUTION

ADA IMPLEMENTATION (CODE AND COMPILATION)

CHANGES TO THE SYSTEM

INSTRUCTOR NOTES

FOR THE EXAMPLE WE ARE NOT TRYING TO SHOW THE BEST OR ONLY WAY TO APPROACH THE PROBLEM BUT RATHER TO ILLUSTRATE THE THOUGHT PROCESS INVOLVED IN ADA SYSTEMS.

WE BEGIN AT A HIGH LEVEL OF ABSTRACTION OF THE PROBLEM AND CONTINUE TO DECOMPOSE TO THE STATEMENT LEVEL.

LET US SUMMARIZE THE OBJECTS TO BE DEALT WITH AND THE OPERATIONS NEEDED TO BE PERFORMED RELATIVE TO THE OBJECTS.

A PICTURE OF A SOLUTION IS SHOWN. IT HAS BEEN DECIDED TO HAVE A MAIN PROGRAM WHICH CONTROLS THE OVERALL LOGIC FLOW OF THE SYSTEM. A SMALL PACKAGE WILL IMPLEMENT THE VECTOR CALCULATIONS. THE MAIN PROCEDURE LOGIC IS PRESENTED AS PSEUDO-CODE FOR THE MOMENT. BUT THE POSSIBLE SOLUTION MUST BE FURTHER DECOMPOSED TO MORE FULLY UNDERSTAND THE VECTOR SERVICES.

THIS PROCESS WOULD THEN BE DONE FOR SUCCEEDING LEVELS OF DECOMPOSITION.

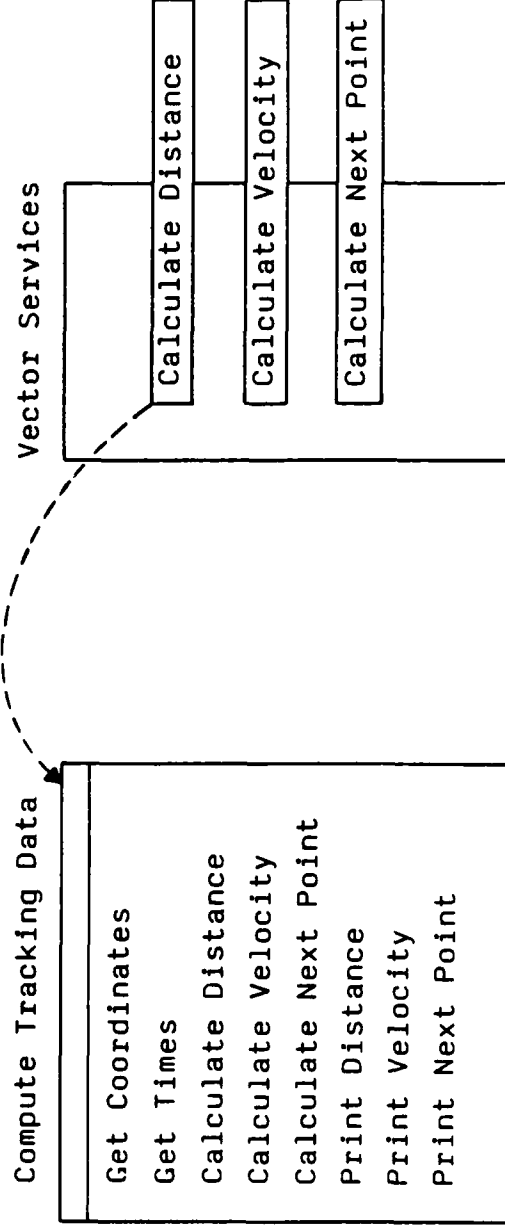
DECOMPOSITION OF SOLUTION: TRACKING PROGRAM

OBJECTS

POINTS
TIMES

OPERATIONS

CALCULATE DISTANCE
CALCULATE VELOCITY
CALCULATE NEXT POINT



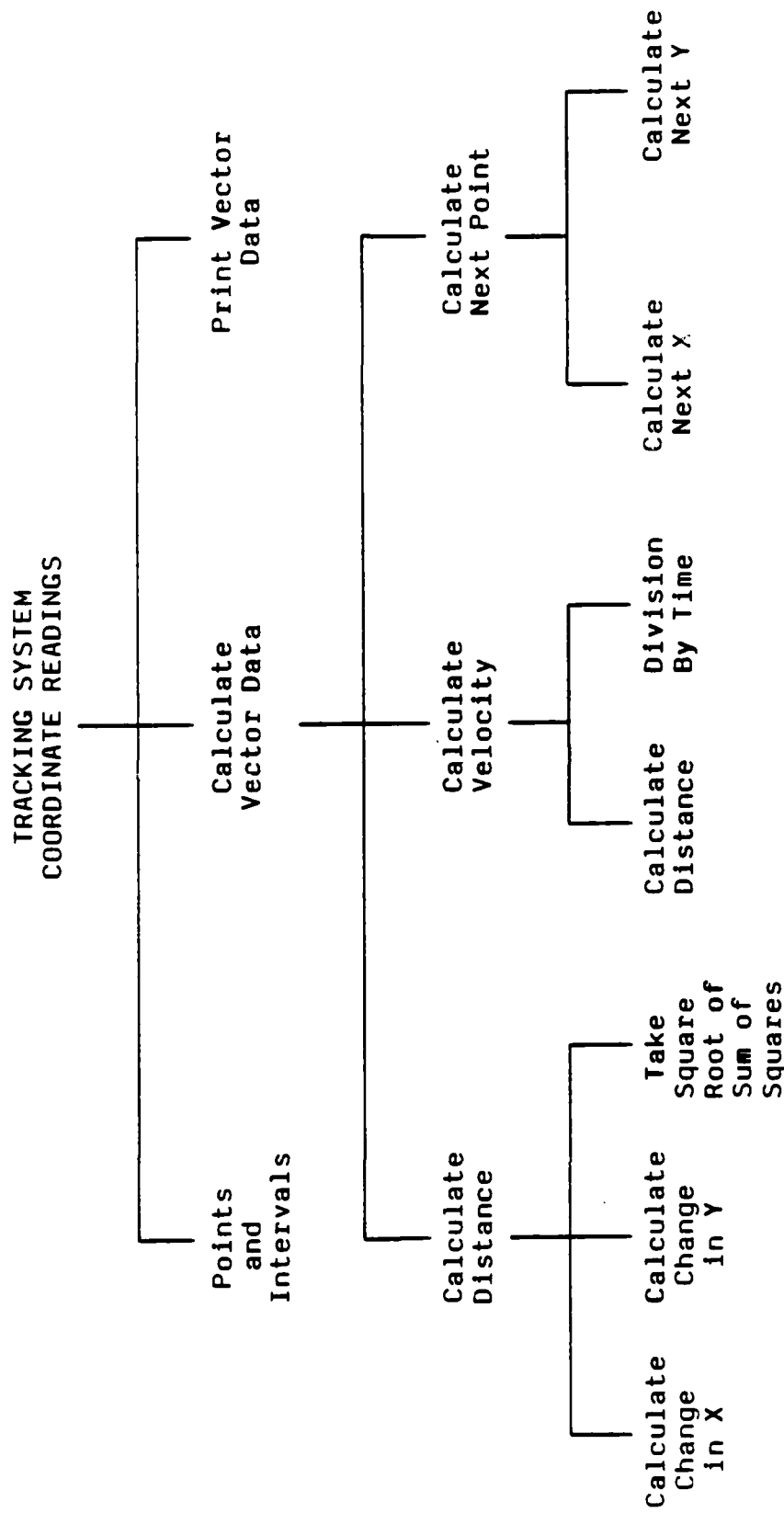
INSTRUCTOR NOTES

THE DIAGRAM SUMMARIZES THE LEVELS OF DECOMPOSITION OF THE SAMPLE DESIGN.

WE NOW TURN TO THE ACTUAL ADA CODING PHASE.

THE NAMES IN THE DIAGRAM ARE NOT THE NAMES OF THE RESULTING SUBPROGRAMS. HERE WE ARE DISCUSSING FUNCTIONS (NOT THE ADA TYPE).

DESIGN SOLUTION SUMMARY



INSTRUCTOR NOTES

THE LISTED ADA FEATURES WILL BE DISCUSSED AS PREPARATION IS MADE FOR THE CODING OF THE SOLUTION.

DO NOT GO INTO DETAIL ABOUT THE ALGORITHMS. THE POINT IS THE STRUCTURE OF THE SOLUTIONS.

ADA FEATURES USED IN SOLUTION

AS WE EXPRESS OUR SOLUTION FOR A TRACKING PROGRAM IN ADA, WE MUST LOOK AT:

- PACKAGES
- SUBPROGRAMS
- CONTROL STRUCTURES AND STATEMENTS
- TYPES AND DECLARATIONS

TABLE 1

3-16i

PACKAGES

SPECIFICATION { package Vector_Services is

type Coordinate_Type is (X,Y);

type Point_Type is array (Coordinate_Type) of Float;

subtype Time_Type is Duration;

function Distance_Between (Last_Point, This_Point : Point_Type)
return Float;

procedure Calculate_Velocity (From, To : in Point_Type;
In_Time : in Time_Type;
Velocity : out Float);

function Next_Point_After (Last_Point, This_Point : in Point_Type;
Time_Between_Last,
Time_Between_Next : Time_Type)
return Point_Type;

end Vector_Services;

DECLARATIONS }

INSTRUCTOR NOTES

THE ADA SYSTEM CAN NOW BE FURTHER DEVELOPED BY CODING THE MAIN LOGIC PROCEDURE. THE TRACKING RESOURCES ARE PROVIDED BY THE `Vector_Services` PACKAGE JUST SHOWN. THE 'WITH' STATEMENT MUST BE USED TO "HOOK TOGETHER" THE MAIN PROGRAM AND THE PACKAGE. THE RESOURCES FROM AN I/O PACKAGE CALLED `Text_IO` WILL ALSO BE USED.

PROCEDURE `Compute_Tracking_Data` HAS THE SAME FORMAT AS ANY OTHER PROCEDURE (EXCEPT IT HAS NO PARAMETERS). THIS SLIDE SHOWS THE DECLARATIONS FOR ALL DATA OBJECTS AND LOCAL ROUTINES TO BE USED IN THE STATEMENT PART. THE USE OF "is separate" WILL BE DISCUSSED IN LATER SLIDES.

POINT OUT THE OBJECT DECLARATIONS CREATING OBJECTS OF TYPES `Point_Type` AND `Time_Type` (SHOWN ON THE PREVIOUS SLIDE) AS WELL AS OBJECTS OF THE PREDEFINED TYPE `Float`. EACH OBJECT IS GIVEN A NAME THAT REPRESENTS ITS INTENDED FUNCTION. THE TYPE TEMPLATE NAME DETERMINES HOW THE OBJECT WILL "LOOK" AND FUNCTION (DON'T GO INTO DETAIL OR SYNTAX.)

(IF POSSIBLE, DISPLAY THIS SLIDE AND THE NEXT AT THE SAME TIME.)

MAIN PROGRAM LOGIC

```
with Text_IO, Vector_Services;
use Vector_Services;
procedure Compute_Tracking_Data is
```

```
    Last_Point, Current_Point, Next_Point : Point_Type;
    Time_Elapsed, Time_Projected : Time_Type;
    Distance, Velocity : Float;
```

← OBJECT DECLARATIONS

```
package Time_IO is new Text_IO.Fixed_IO (Time_Type);
package Flt_IO is new Text_IO.Float_IO (Float);
```

```
procedure Get_Point (P : out Point_Type) is separate;
procedure Put_Point (P : in Point_Type) is separate;
```

```
begin -- Compute_Tracking_Data
```

executable statements on next page

```
end Compute_Tracking_Data;
```

INSTRUCTOR NOTES

THIS SLIDE SHOWS THE EXECUTABLE PART OF Compute_Tracking_Data.

STATEMENTS TO READ IN THE POINTS AND TIMES WITH THE SERVICES OF TEXT_IO; THE DESIRED INFORMATION IS CALCULATED BY THE FUNCTION AND PROCEDURE CALLS; AND WE PRINT OUR RESULTS WITH THE SERVICES OF TEXT_IO.

SUBPROGRAMS ARE DISCUSSED ON A LATER SLIDE; HERE, JUST POINT OUT THE CALLS AND PARAMETERS.

MAIN PROGRAM LOGIC (Continued)

```
with Text_IO, Vector_Services;
use Vector_Services;
procedure Compute_Tracking_Data is

    declarations on previous page

begin
    -- Compute_Tracking_Data
    -- input points and times
    Text_IO.Put ("Enter coordinates of last position: ");
    Get_Point (Last_Point);
    Text_IO.Put ("Enter coordinates of current position: ");
    Get_Point (Current_Point);

    -- calculate distance, velocity, and new coordinates
    Text_IO.Put ("Time (in seconds) between readings : ");
    Time_IO.Get (Time_Elapsed);
    Text_IO.New_Line;
    Text_IO.Put ("Time (in seconds) until next reading : ");
    Time_IO.Get (Time_Projected);
    Text_IO.New_Line;

    Distance := Distance_Between (Last_Point, Current_Point);
    Calculate_Velocity (Last_Point, Current_Point, Time_Elapsed, Velocity);
    Next_Point := Next_Point_After (Last_Point, Current_Point,
                                     Time_Elapsed, Time_Projected);

    -- output calculation results
    Text_IO.Put ("Distance between point was ");
    Flt_IO.Put (Distance);
    Text_IO.Put_Line (" units.");

    Text_IO.Put ("Velocity was ");
    Flt_IO.Put (Velocity);
    Text_IO.Put ("units per second.");

    Text_IO.Put ("After ");
    Time_IO.Put (Time_Projected);
    Text_IO.Put ("seconds, the next point should be ");
    Put_Point (Next_Point);

    end Compute_Tracking_Data;
```

INSTRUCTOR NOTES

THEN IN THE SECOND PART OF THE PROGRAM UNIT, THE BODY, IS THE ACTUAL CODE THAT PERFORMS THE RESOURCES ACTIONS. THIS SLIDE SHOWS THE FIRST TWO SUBPROGRAM BODIES - THE OTHER TWO ARE ON THE FOLLOWING SLIDE.

NOTICE THAT PROCEDURE Sqrt WAS NOT LISTED IN THE SPECIFICATION. Sqrt IS A UTILITY WHICH WILL ONLY BE USED BY THE ALGORITHM Distance_Between. BY PLACING IT IN THE PACKAGE BODY, WE ENSURE THAT NO UNAUTHORIZED TAMPERING OF THE DATA SCORES CAN BE DONE.

BRIEFLY DISCUSS THE INDICATED CONTROL STRUCTURES, POINTING OUT RESERVED WORDS (UNDERLINING MAY BE HELPFUL). DO NOT GET BOGGED DOWN IN SYNTAX; FOCUS ON GENERAL STRUCTURE AND FUNCTION. POINT OUT NESTED CONTROL STRUCTURE, WITH INDENTATION SHOWING LOGICAL NESTING.

THE while LOOP IS AN ITERATIVE CONTROL STRUCTURE, ALLOWING REPETITION OF SOME SEQUENCE OF ACTION WHILE SOME CONDITION IS PRESENT. THE OTHER ITERATIVE CONTROL STRUCTURE IS THE for LOOP (NOT SHOWN), WHICH ALLOWS REPETITION FOR A SPECIFIED NUMBER OF TIMES.

THE VECTOR PACKAGE BODY

```

package body Vector_Services is

    function Sqrt (X : Float) return Float is
        Epsilon : constant := 0.000001;
        Root : Float := 1.0;
    begin -- Sqrt
        if X = 0.0 then
            return 0.0;
        else
            Root := (X/Root + Root) / 2.0;
            while abs (X/Root**2 - 1.0) >= Epsilon
            loop
                Root := (X/Root + Root) / 2.0;
            end loop;
            return Root;
        end if;
    end Sqrt;

-- IF_THEN_ELSE CONTROL STRUCTURE
-- LOOP CONTROL STRUCTURE

function Distance_Between (Last_Point, This_Point : Point_Type) return Float is
    Dx, Dy : Float;
begin -- Distance_Between
    Dx := abs (This_Point(X) - Last_Point(X));
    Dy := abs (This_Point(Y) - Last_Point(Y));
    return ( Sqrt( Dx**2 + Dy**2) );
end Distance_Between;

```

INSTRUCTOR NOTES

INDICATE PROCEDURE AND FUNCTION TEMPLATE STRUCTURE BY UNDERLINING RESERVED WORDS.

A PROCEDURE BEGINS EXECUTION THROUGH A PROCEDURE CALL (SHOWN IN MAIN PROCEDURE BODY), WHICH IS A STATEMENT. A FUNCTION CALL IS AN EXPRESSION (RETURNS A VALUE); THUS EVERY FUNCTION MUST SPECIFY A RETURN TYPE AND MUST EXPLICITLY RETURN A VALUE VIA A RETURN STATEMENT.

POINT OUT THE PARAMETER LISTS AND MODE INDICATIONS. A PARAMETER OF MODE IN IS PASSED TO THE SUBPROGRAM BUT CANNOT BE MODIFIED IN IT; AN OUT PARAMETER IS ONE THAT RETURNS A VALUE ASSIGNED TO IT IN THE SUBPROGRAM. A THIRD MODE, IN OUT, INDICATES A PARAMETER THAT IS PASSED IN, MODIFIED, AND PASSED OUT AGAIN. A FUNCTION PARAMETER MAY BE OF MODE IN ONLY.

IN Next_Point_After, A DESIGN DECISION WAS MADE TO CALCULATE THE CHANGES IN X AND Y VALUES WITHOUT A FURTHER SUBPROGRAM CALL (I.E. WE PUT OUR ALGORITHM IN LINE).

IN THE STATEMENTS WHICH CALCULATE THE NEXT POINT, INDICATE THAT THIS IS AN EXAMPLE OF WHAT STRONG TYPING DOES FOR YOU. TIME VALUES ARE OF TYPE Time_Type WHEREAS POINT COORDINATES ARE Float. ADA WILL CATCH THIS TYPE OF MISMATCH FOR NUMERIC TYPES; CONVERSIONS ARE ALLOWED. A LIKELY QUESTION TO ARISE: CAN CHARACTERS BE CONVERTED TO INTEGER (OR VICE VERSA) IN ADA?

THE VECTOR PACKAGE BODY (Continued)

```
procedure Calculate_Velocity (From, To : in Point_Type;
                             In_Time : in Time_Type;
                             Velocity : out Float) is
begin
    -- Calculate Velocity
    Velocity := Distance_Between(From, To)/Float(In_Time);
end Calculate_Velocity;

function Next_Point_After (Last_Point, This_Point : in Point_Type;
                           Time_Between_Last, Time_Between_Next : Time_Type)
    return Point_Type is
    Next_Point : Point_Type;
begin
    -- Next_Point_After
    if Time_Between_Last = 0 then
        return This_Point;
    else
        Next_Point(X) := Last_Point(X) + Float(Time_Between_Next/Time_Between_Last)
            * abs(This_Point(X) - Last_Point(X));
        Next_Point(Y) := Last_Point(Y) + Float(Time_Between_Next/Time_Between_Last)
            * abs(This_Point(Y) - Last_Point(Y));
        return Next_Point;
    end if;
end Next_Point_After;

end Vector_Services;
```

INSTRUCTOR NOTES

THE PRECEDING SLIDES SHOWED THE PACKAGE BODY AS IT WOULD "REALLY" APPEAR, WITH ALL CODE ACTUALLY LOCATED IN THE BODY. NOTE HOW CONFUSING THIS IS - WE CAN'T AS EASILY SEE WHAT FUNCTIONS, PROCEDURES, ETC. ARE CONTAINED IN THE BODY.

INSTEAD, WE MAY USE THE 'is separate' CLAUSE, WHICH TELLS THE COMPILER THAT THE ACTUAL CODE FOR THE SUBPROGRAM IS IN A SEPARATE PLACE FROM THE PARENT ADA UNIT. THIS PROCESS IS CALLED STUBBING; THE SEPARATE BODIES ARE CALLED SUBUNITS.

ALTERNATIVE PACKAGE BODY STRUCTURE

BODY

```
package body Vector_Services is

function Sqrt (X : Float) return Float is separate;

function Distance_Between (Last_Point, This_Point : Point_Type)
    return Float is separate;

procedure Calculate_Velocity (From, To : in Point_Type;
    In_Time : in Time_Type;
    Velocity : out Float) is separate;

function Next_Point_After (Last_Point, This_Point : in Point_Type;
    Time_Between_Last, Time_Between_Next : Time_Type)
    return Point_Type is separate;

end Vector_Services;
```

INSTRUCTOR NOTES

- FOR EACH 'SEPARATE' SUBPROGRAM (SUBUNIT) WE INDICATE THE PARENT UNIT AS FOLLOWS:

```
separate (Vector_Services)      -- We Add This Line

function Sqrt (X : Float) return Float is
    Epsilon : constant := 0.000001;
    Root    : Float    := 1.0;
begin -- Sqrt
    if X = 0.0 then
        return 0.0;
    else
        Root := (X/Root + Root) / 2.0;
        while abs (X/Root**2 - 1.0) >= Epsilon
            loop
                Root := (X/Root + Root) / 2.0;
            end loop;
        return Root;
    end if;
end Sqrt;
```

INSTRUCTOR NOTES

THESE ARE THE SUBUNITS STUBBED OUT OF THE MAIN PROCEDURE. NOTE THAT THIS CODE WOULD ADD CONSIDERABLE BULK TO THE MAIN PROCEDURE BODY IF USED IN LINE, WHILE CONTRIBUTING LITTLE TO THE LOGICAL STRUCTURE. STUBBING OUT THESE ROUTINES ALLOWS EASY MODIFICATION OF I/O FORMAT.

MORE SUBUNITS

```
separate (Compute_Tracking_Data)
procedure Get_Point (P : out Point_Type) is
begin -- Get_Point
    Text_IO.Put (" X = ");
    Flt_IO.Get (P(X));
    Text_IO.Put (" Y = ");
    Flt_IO.Get (P(Y));
    Text_IO.New_Line;
end Get_Point;
```

```
separate (Compute_Tracking_Data)
procedure Put_Point (P : in Point_Type) is
begin -- Put_Point
    Text_IO.Put ("(");
    Flt_IO.Put (P(X));
    Text_IO.Put (",");
    Flt_IO.Put (P(Y));
    Text_IO.Put (")");
end Put_Point;
```

INSTRUCTOR NOTES

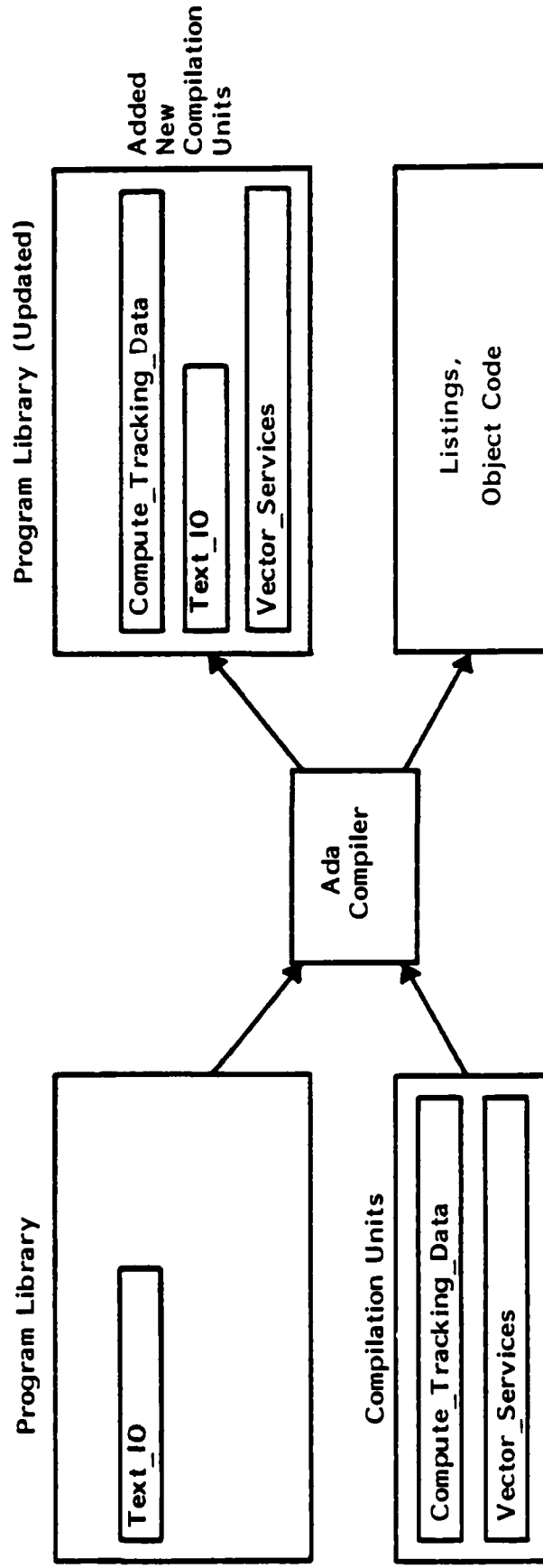
CODING OF THE ADA SYSTEM IS COMPLETED. NEXT THE TOPIC OF COMPILATION IN ADA IS DISCUSSED.

COMPILATION UNITS ARE PARTS OF ADA CODE THAT THE LANGUAGE SAYS CAN BE SUBMITTED BY THEMSELVES TO AN ADA COMPILER.

COMPILATION CONSISTS OF SUBMITTING OUR COMPILATION UNITS PLUS THE PROGRAM LIBRARY WHICH IS A FILE THAT WILL CONTAIN CERTAIN INFORMATION ABOUT A UNIT THAT SUBSEQUENT COMPILER SUBMISSION WILL NEED. ONCE COMPILED, THE SUBMITTED COMPILATION UNITS ARE ADDED TO THE PROGRAM LIBRARY.

COMPILATION OF OUR TRACKING SYSTEM

- SUBMIT ALL PROGRAM PARTS AT ONE TIME:



INSTRUCTOR NOTES

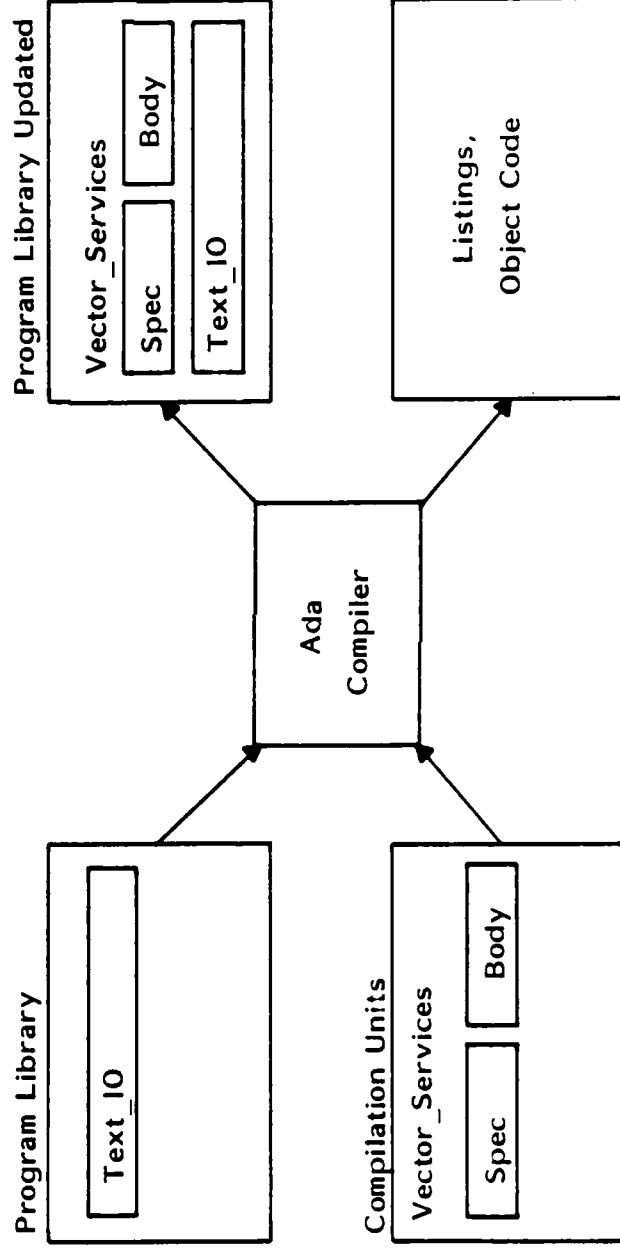
INSTEAD OF SUBMITTING ALL OUR PROGRAM PARTS AT ONE TIME, WE COULD SUBMIT THEM SEPARATELY. LET'S SAY PROGRAMMER 1 CODED OUR `Vector_Services` PACKAGE. INSTEAD OF WAITING FOR PROGRAMMER 2, WHO WILL HAVE HIS CODE COMPLETED LATER, WE CAN COMPILE THE `Vector_Services` PACKAGE. THE COMPILER WILL ADD THE NECESSARY INFORMATION ABOUT THE PACKAGE TO THE PROGRAM LIBRARY.

NOTE THAT THE SPECIFICATION AND BODY COULD ALSO BE COMPILED SEPARATELY, BUT THE SPEC MUST BE COMPILED FIRST.

ALTERNATE COMPILATION OF OUR TRACKING SYSTEM

- SUBMIT PROGRAM PARTS (COMPILATION UNITS) SEPARATELY:

RUN 1

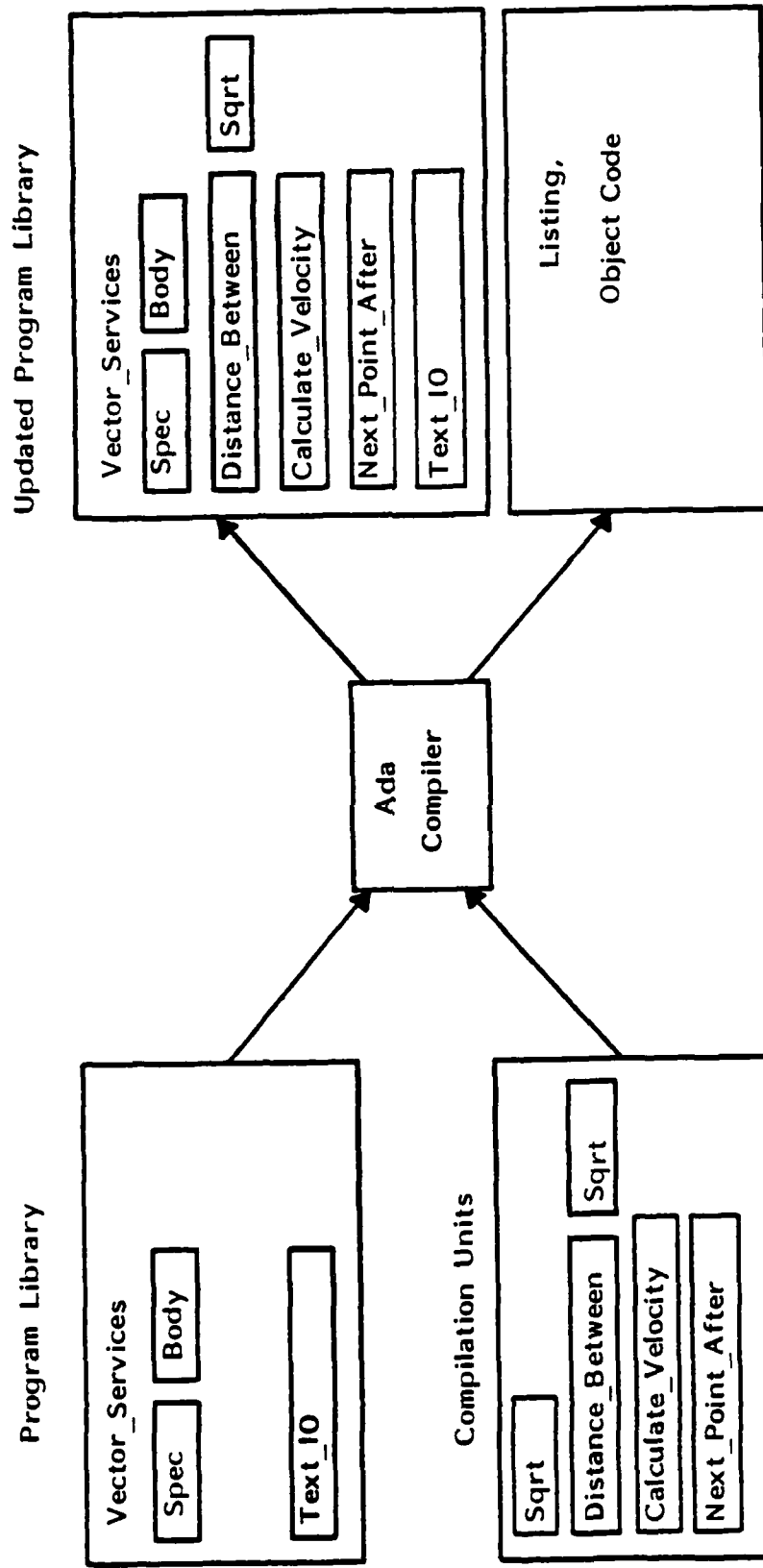


INSTRUCTOR NOTES

FOR OUR EXAMPLE, WE WILL COMPILE THE PACKAGE SUBUNITS AND ADD THEM TO THE PROGRAM LIBRARY.

AGAIN, ALL FOUR SUBUNITS NEED NOT BE COMPILED AT THE SAME TIME. HOWEVER, ANY SUBUNIT THAT DEPENDS UPON ANOTHER MUST BE COMPILED AFTER THE ONE UPON WHICH IT DEPENDS. FOR EXAMPLE, Distance_Between CALLS Sqrt, SO Sqrt MUST BE COMPILED BEFORE Distance_Between.

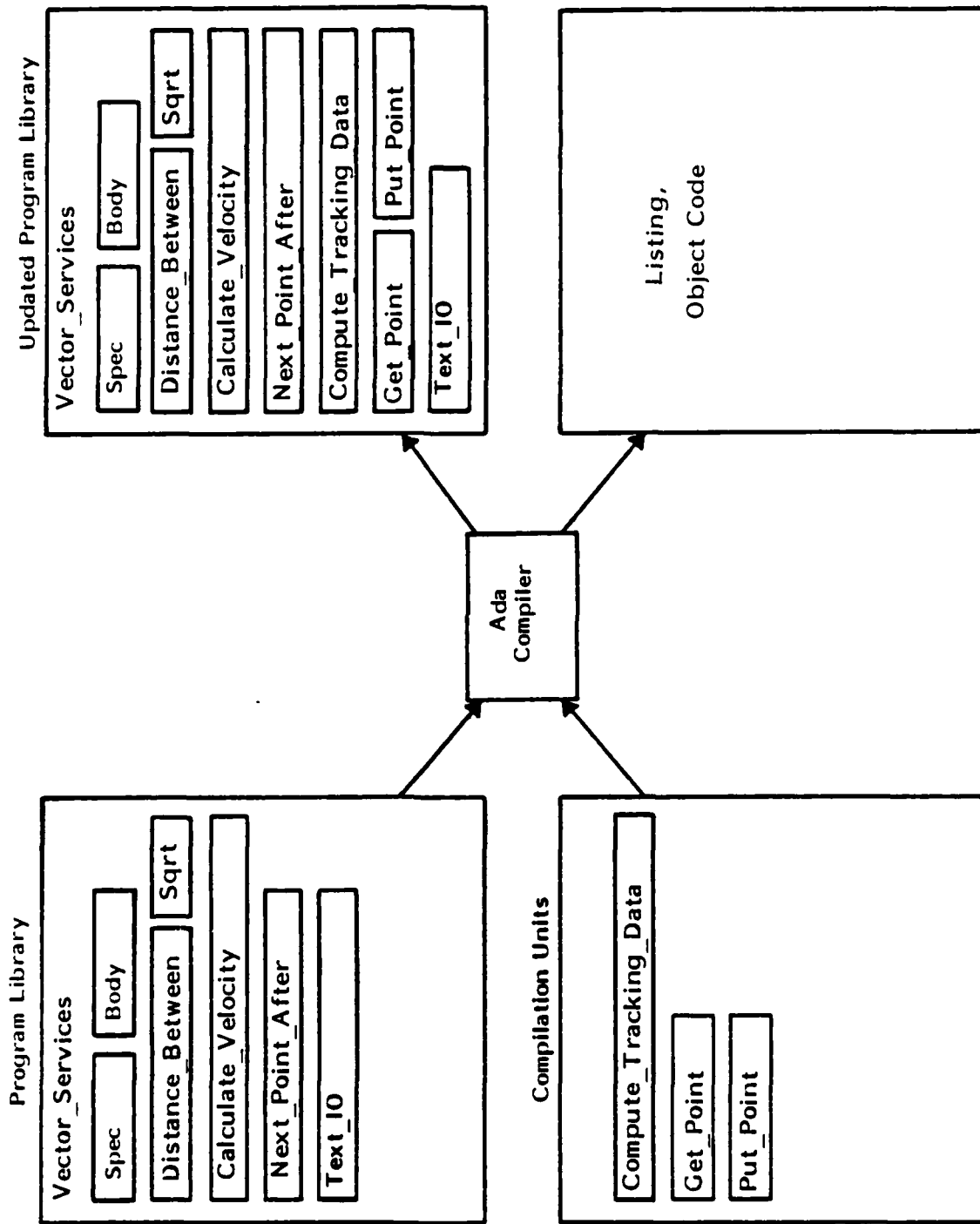
RUN 2



INSTRUCTOR NOTES

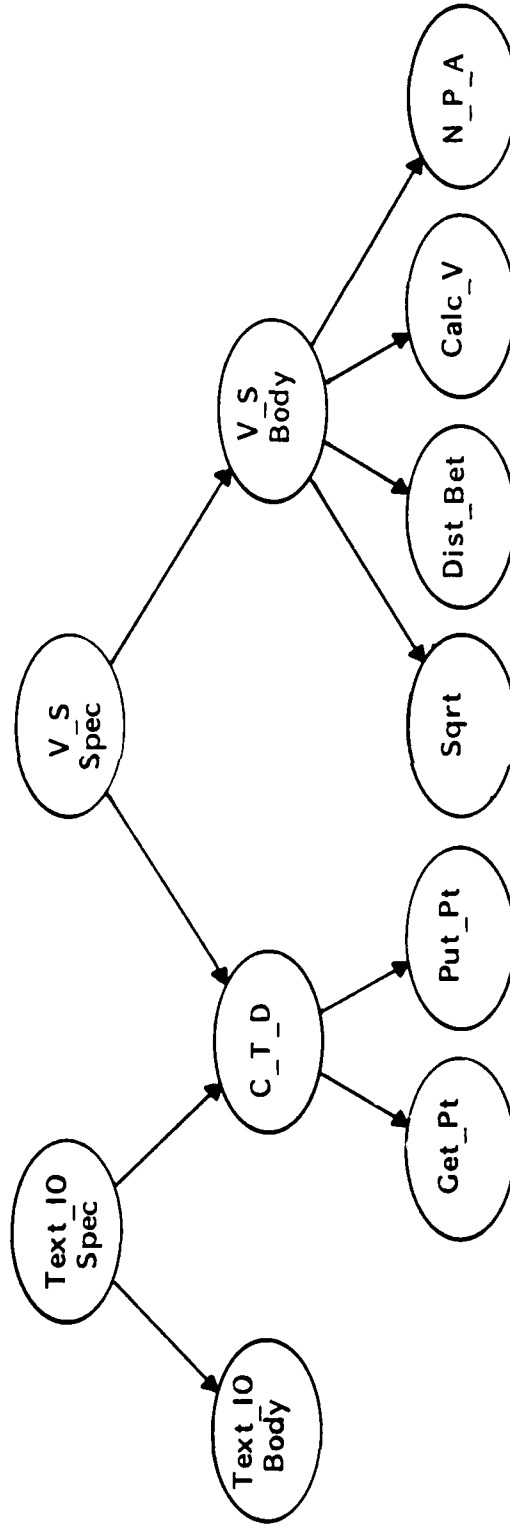
ONCE ALL THE RESOURCE PIECES (HERE, THE PACKAGE SPECS FOR TEXT_IO AND Vector_Services)
NEEDED BY THE MAIN PROCEDURE ARE IN PROGRAM LIBRARY,
WE CAN COMPILE Compute_Tracking_Data AND ITS SUBUNITS.

RUN 3



INSTRUCTOR NOTES

HERE IS THE DEPENDENCY DIAGRAM



ALL POSSIBLE ORDERINGS CAN BE DERIVED FROM THE ABOVE DIAGRAM.

AD-A165 351

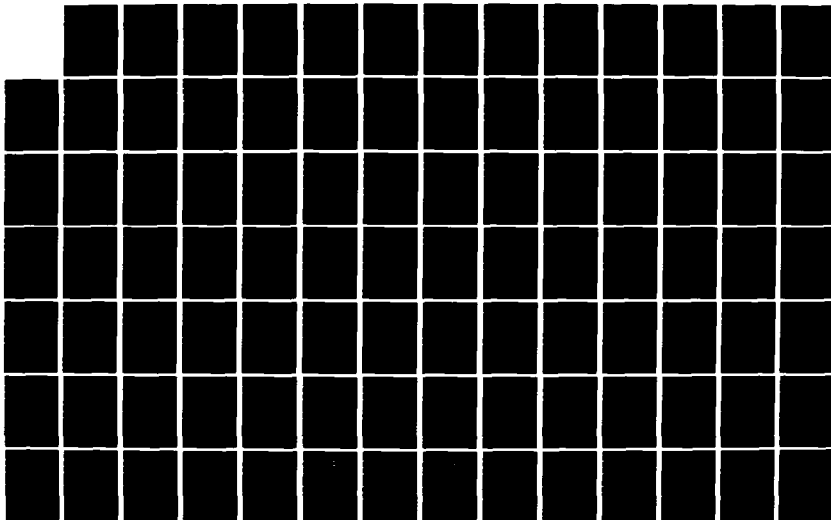
ADA (TRADEMARK) TRAINING CURRICULUM: ADA ORIENTATION
FOR MANAGERS L101 TEACHER'S GUIDE(U) SOFTECH INC
WALTHAM MA 1986 DAAB07-83-C-K514

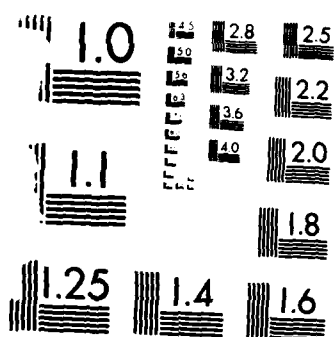
2/3

UNCLASSIFIED

F/G 5/9

NL





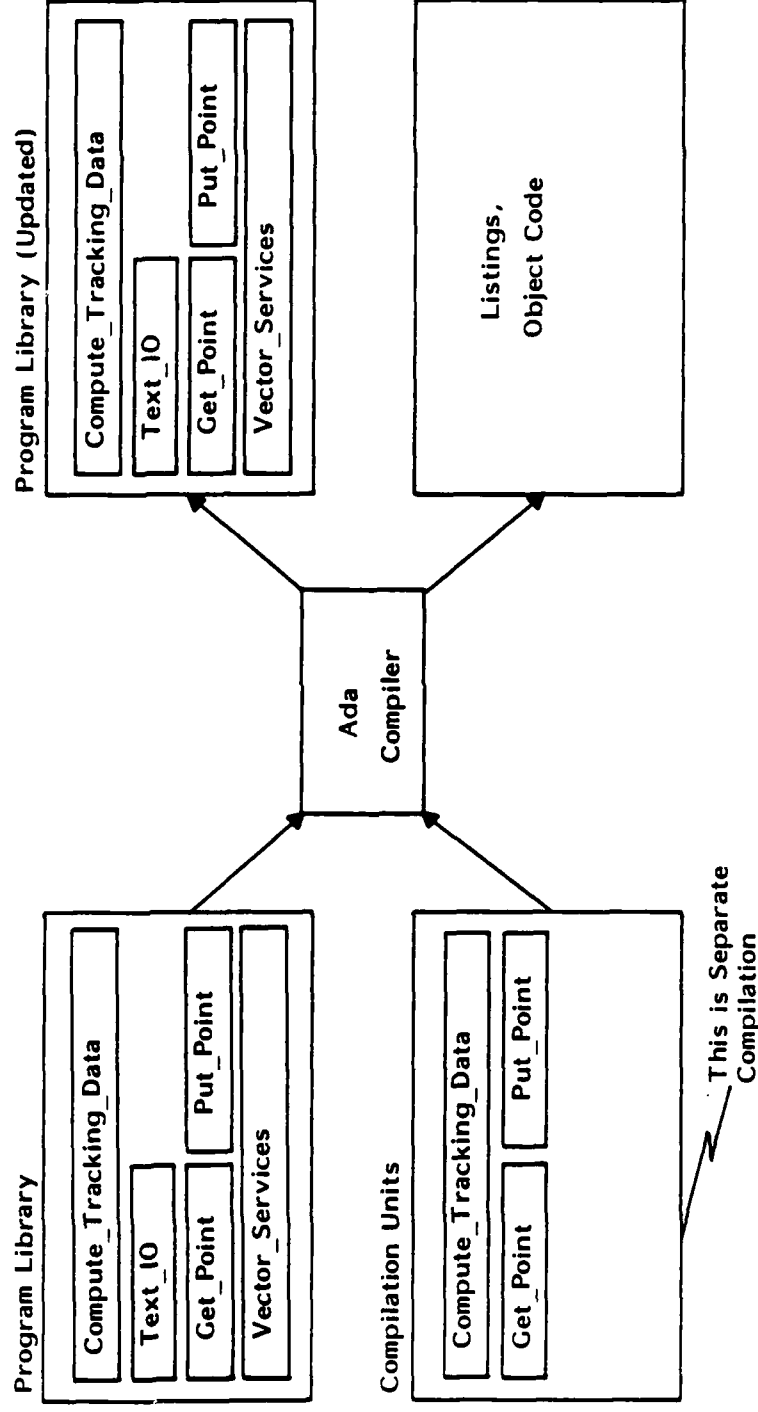
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

IN-CLASS EXERCISE

SUGGEST OTHER COMPILATION ORDER POSSIBILITIES

CHANGES TO THE SYSTEM: MAIN PROCEDURE

SUPPOSE WE NEED TO CHANGE ONE OF THE PRINTOUT FORMATS. SINCE THE PACKAGE WORRIES ABOUT ALL AND ONLY THE VECTOR CALCULATIONS, THE PACKAGE NEED NOT BE CHANGED OR RECOMPILED. HOWEVER, THE SUBUNITS OF THE MAIN PROCEDURE MUST BE RECOMPILED WHEN MAIN IS CHANGED, SINCE THEY ARE POTENTIALLY AFFECTED BY THE CHANGE.



INSTRUCTOR NOTES

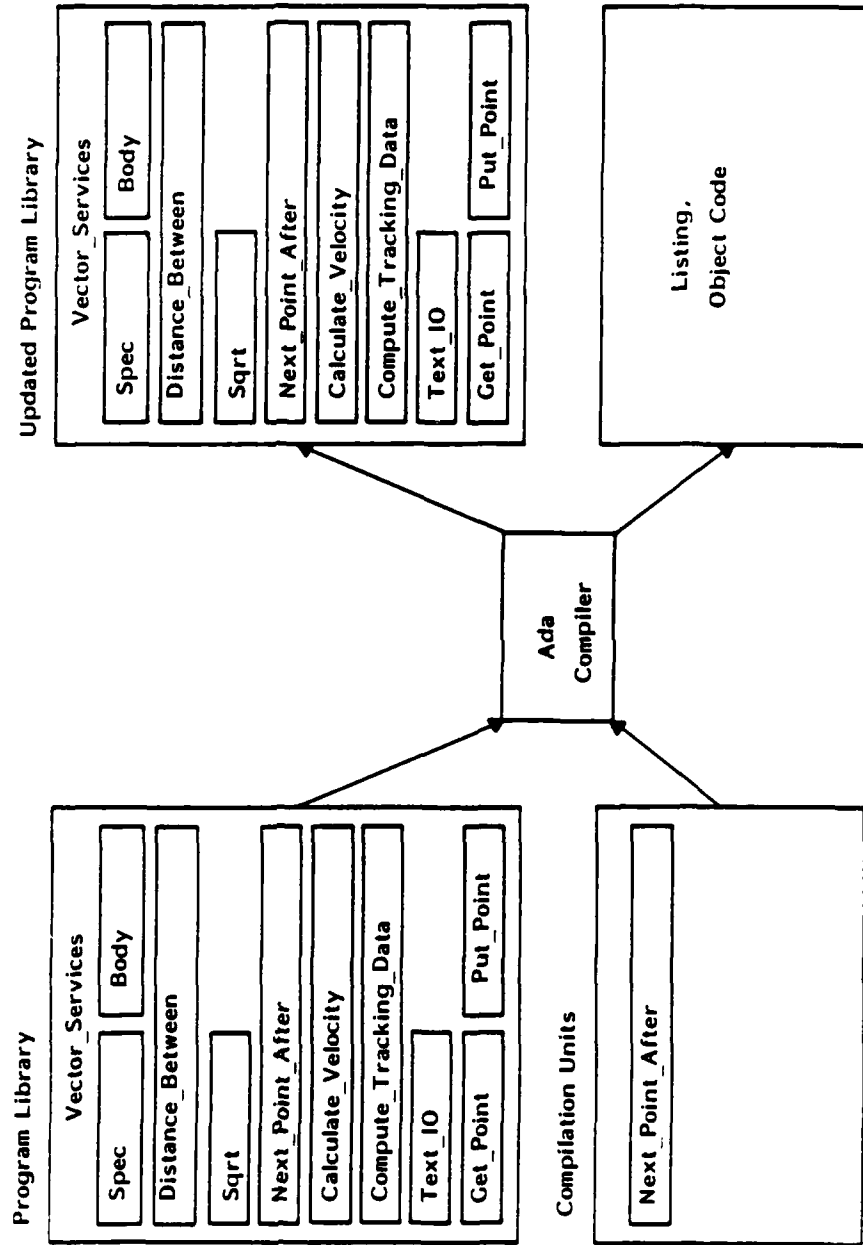
ANOTHER EXAMPLE OF EASE OF CHANGE.

POINT OUT THAT NEITHER PROCEDURE MAIN NOR THE PACKAGE Vector_Services NEED TO BE RECOMPILED.

IF THE SUBPROGRAMS HAND NOT BEEN STUBBED OUT, THE PACKAGE BODY (BUT NOT SPEC) WOULD HAVE REQUIRED RECOMPILATION.

CHANGES TO THE SYSTEM: A SUBUNIT

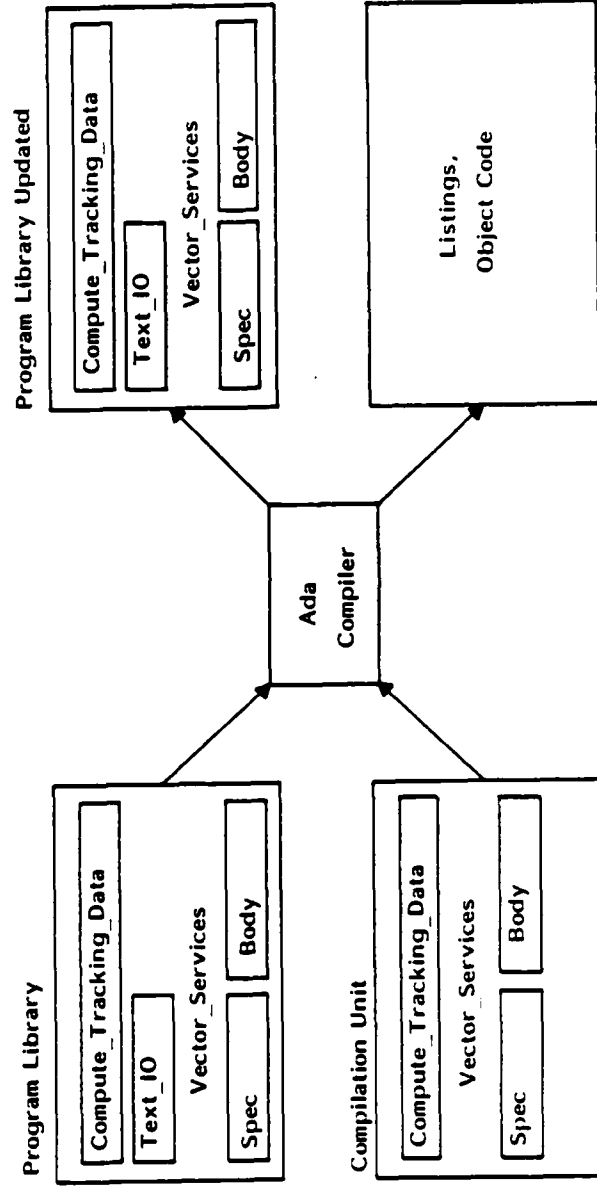
ASSUME WE FIND A BETTER ALGORITHM FOR ONE OF OUR VECTOR ROUTINES. SINCE WE COLLECTED THE ROUTINES IN A PACKAGE AND STUBBED OUT EACH ROUTINE, WE CAN MAKE THE CHANGE TO THE SUBPROGRAM ITSELF WITHOUT REQUIRING ANY CHANGES TO THE MAIN PROCEDURE, THE PACKAGE SPECIFICATION FOR `Vector_Services`, THE PACKAGE BODY, OR THE OTHER ROUTINES STUBBED FROM THE PACKAGE BODY.



CHANGES TO THE SYSTEM: ADDING A ROUTINE

SUPPOSE WE WANT TO ADD A ROUTINE TO COMPUTE THE ANGLE OF THE VECTOR. SINCE WE COLLECTED OUR VECTOR ROUTINES IN A PACKAGE, WE WANT TO ADD THIS ROUTINE TO THE PACKAGE SPECIFICATION AND BODY OF `Vector_Services`. WE MODIFIED `Vector_Services` AND OUR MAIN PROCEDURE DEPENDS ON THOSE RESOURCES.

AS A RESULT WE MUST ALSO RECOMPILE THE MAIN PROCEDURE.



INSTRUCTOR NOTES

THE FIRST THREE LANGUAGE REQUIREMENTS FROM THE STEELMAN DOCUMENT ARE GIVEN. OTHERS ARE EFFICIENCY, SIMPLICITY, IMPLEMENTATION. THESE LAST THREE COULD BE QUITE CONTROVERSIAL AS TO WHETHER ADA ACTUAL SATISFIES ITS OWN REQUIREMENTS.

LIST IS IN ORDER OF IMPORTANCE OF DESIGN CRITERIA. IT SHOULD BE NOTED THAT RELIABILITY IS MORE IMPORTANT THAN EFFICIENCY. ALSO THAT READABILITY IS MORE IMPORTANT THAN WRITABILITY - A PROGRAM IS READ MANY MORE TIMES IN ITS LIFE TIME THAN IT IS WRITTEN.

WHAT ADA IS IS IN LARGE PART DUE TO THE OVERRIDING DESIRE TO HAVE A LANGUAGE DESIGNED TO SATISFY SPECIFIC REQUIREMENTS. IT WAS NOT A HODGE-PODGE OF FAVORITE FEATURES.

THE ADA LANGUAGE WAS DESIGNED FOR

- GENERALITY

MEETS A WIDE SPECTRUM OF NEEDS

- RELIABILITY

PROVIDES COMPILE-TIME DETECTION OF CODING ERRORS
ENCOURAGES MODERN SOFTWARE ENGINEERING PRINCIPLES

- MAINTAINABILITY

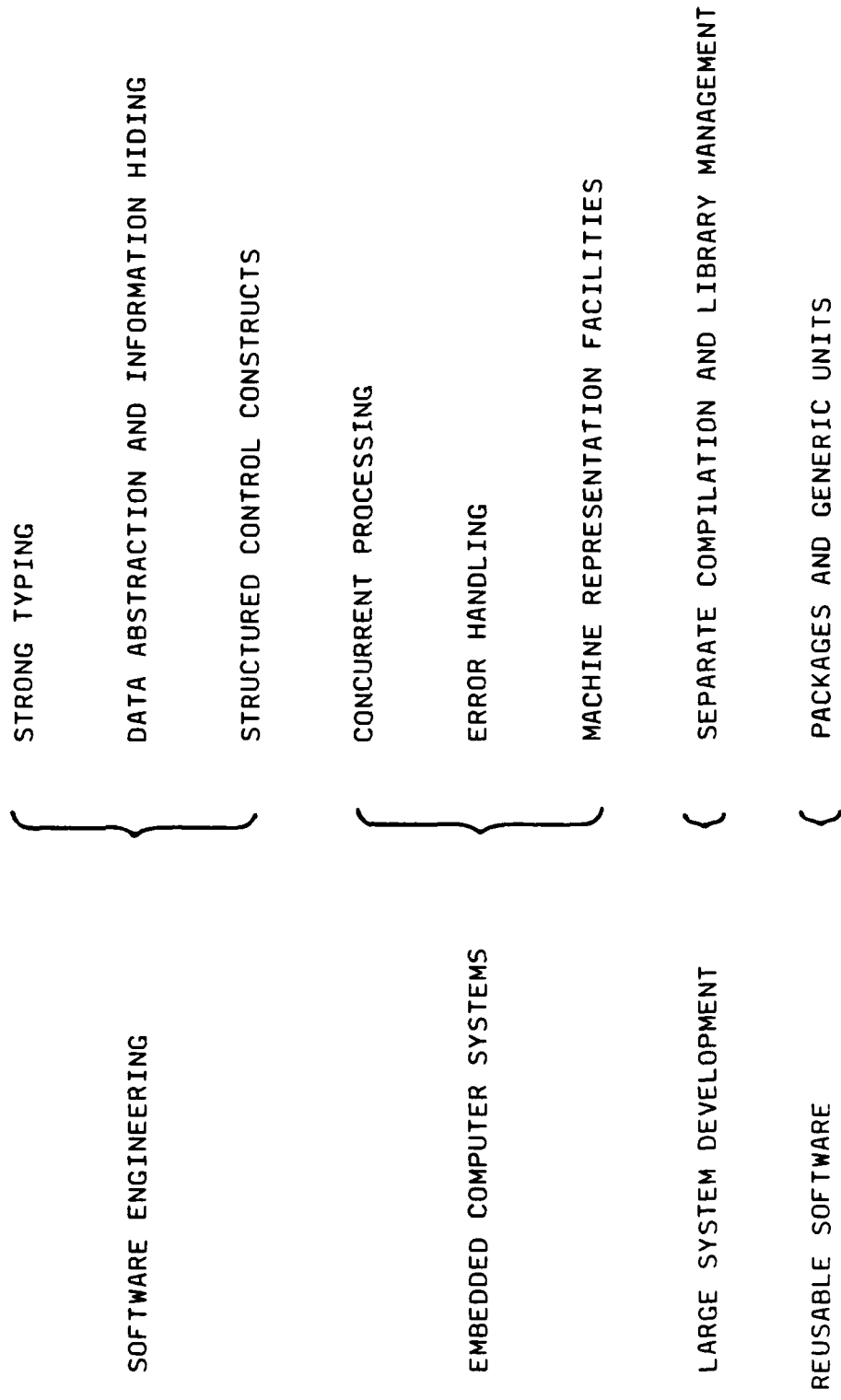
READABILITY IS MORE IMPORTANT THAN WRITABILITY
ENCOURAGES DOCUMENTATION

INSTRUCTOR NOTES

DoD'S PRIMARY CONCERN WAS DEVELOPING A LANGUAGE TO SUPPORT THE REQUIREMENTS OF EMBEDDED COMPUTER SYSTEMS (ECS). AN ECS IS A COMPUTER FOUND IN THE CONTEXT OF A LARGER SYSTEM OF POSSIBLY NON-COMPUTER ITEMS. FOR EXAMPLE: RADAR, MICROWAVE OVENS, MISSILES. THIS IS NOT DATA PROCESSING BUT REAL TIME SYSTEMS WHICH MUST INTERACT WITH AN EXTERNAL ENVIRONMENT. ECS NEED PARALLEL PROCESSING, REAL TIME CONTROL, ERROR HANDLING, UNIQUE I/O CONTROL. WE ARE GENERALLY DEALING WITH SYSTEMS THAT ARE LARGE, WILL BE IN EXISTENCE FOR MANY YEARS, UNDERGOING CONTINUING MODIFICATIONS. RELIABILITY AND SIZE CONSTRAINTS ARE CRITICAL FACTORS IN MOST ECS. FOR EXAMPLES, WE CANNOT AFFORD ERRORS IN NUCLEAR MISSILE SOFTWARE.

DoD WAS ALSO CONCERNED WITH SOFTWARE ENGINEERING METHODS AND PRINCIPLES. IN ADDITION THE LANGUAGE HAD TO SUPPORT DEVELOPMENT FOR LARGE SCALE PROJECTS AND ADDRESS THE REUSABILITY OF SOFTWARE.

DoD's REQUIREMENTS FOR THE LANGUAGE



CATALOGUE OF ADA FEATURES

- PACKAGES
- SUBPROGRAMS
- TASKS
- SEPARATE COMPILATION
- STATEMENTS
- DECLARATIONS
- TYPES
- GENERICS
- OVERLOADING
- EXCEPTIONS
- MACHINE REPRESENTATION SPECS
- I/O

PROGRAM UNITS

ADA SYSTEMS CAN CONSIST OF COMBINATIONS OF:

- PACKAGES

- SUBPROGRAMS

PROCEDURES

FUNCTIONS

- TASKS

- GENERICS

INSTRUCTOR NOTES

THE SEPARATION OF THE SPECIFICATION FROM THE BODY (THE WHAT FROM THE HOW) IS WHAT GIVES US THE RELIABILITY AND MAINTAINABILITY POINTS OF THE SLIDE. THIS IS CRUCIAL TO AN UNDERSTANDING OF ADA.

INTERFACE ERRORS ARE ONE OF THE MAJOR PROBLEMS IN INTEGRATING MODULES IN LARGE SYSTEMS. WITH THE SPECIFICATION INFORMATION, THE COMPILER CAN PERFORM VALIDITY CHECKS AT COMPILE-TIME RATHER THAN INTEGRATION TIME. IN OTHER WORDS, YOU CAN TEST THE INTERFACES OF THE DESIGN AS A WHOLE BEFORE CODING ANY OF THE ALGORITHMS. IT IS MORE COST EFFECTIVE TO CORRECT ERRORS AT THIS POINT THAN AT INTEGRATION AND TESTING. (RESHOW SLIDE "COST OF ERROR CORRECTION" - SLIDE 1-6).

SPECIFICATIONS CAN BE VIEWED AS LOGICAL INTERFACES.

PROGRAM UNIT STRUCTURE

ALL PROGRAM UNITS HAVE A SIMILAR FORM

- SPECIFICATION
 - DESCRIBES WHAT THE PROGRAM UNIT DOES
 - THIS INFORMATION IS 'VISIBLE' TO (CAN BE USED BY) THIS AND OTHER PROGRAM UNITS
- BODY
 - DETAILS HOW THE PROGRAM UNIT IMPLEMENTS AN ALGORITHM OR STRUCTURE
 - THIS INFORMATION IS 'HIDDEN' FROM (CANNOT BE USED BY) OTHER PROGRAM UNITS

RELIABILITY INCREASED BECAUSE INTERFACE ERRORS CAN BE DETECTED EARLIER

MAINTAINABILITY INCREASED BECAUSE CHANGES TO THE IMPLEMENTATION CAN BE DONE WITHOUT USER PROGRAM UNITS ALSO REQUIRING MODIFICATION

INSTRUCTOR NOTES

THIS IS ONE OF ADA'S STRONGEST FEATURES.

PACKAGES PROVIDE A MEANS TO PHYSICALLY GROUP LOGICALLY RELATED OBJECTS AND OPERATIONS IN SUCH A WAY THAT WHEN WE NEED TO CHANGE PORTIONS OF A SYSTEM WE CAN KNOW THE EXACT AREAS THAT WILL BE AFFECTED. THUS WE CAN REDUCE THE AFFECTED AREA TO A MINIMUM. THIS ALLOWS US CONTROL OF THE PROVERBIAL "RIPPLE EFFECT" ASSOCIATED WITH SYSTEM CHANGES.

PACKAGES

- BASIC STRUCTURING UNIT
- GROUPS LOGICALLY RELATED DATA AND PROGRAM UNITS
(ENCAPSULATION)
- ARE STRUCTURE REPRESENTATIONS NOT ALGORITHMS
- PROVIDES REUSABLE SOFTWARE COMPONENTS
- MAINTAINABILITY INCREASED BECAUSE EFFECT OF CHANGES
CAN BE LOCALIZED

SUBPROGRAMS

- BASIC EXECUTABLE PROGRAM UNITS
- TWO FORMS OF SUBPROGRAMS
 - PROCEDURE
 - CALLED BY A STATEMENT
 - FUNCTION
 - CALLED IN AN EXPRESSION, ALWAYS RETURNS ONE RESULT
- SUBPROGRAM PARAMETERS PASS VALUES

TASKS

- PARALLEL THREADS OF CONTROL
- CONCURRENCY REAL WITH MULTIPROCESSOR; APPARENT WITH SINGLE PROCESSOR
- MECHANISM FOR SYNCHRONIZATION AND DATA TRANSMISSION IS CALLED RENDEZVOUS
- PROVIDES DIRECT MAPPING OF REAL-TIME PROCESSING DESIGNS TO THE LANGUAGE

INSTRUCTOR NOTES

THIS IS A SAMPLE TASK PROGRAM UNIT. DON'T GO INTO ANY DETAIL, JUST INDICATE ITS SIMILARITIES TO A PROCEDURE.

TASKS

```
task Card_Reader is
    entry Get (C : out Card);
end Card_Reader;

task body Card_Reader is
    Latest_Card : Card;
begin -- Card_Reader
    loop
        Text_IO.Get (Latest_Card);
        accept Get (C : out Card) do
            C := Latest_Card;
        end Get;
    end loop;
end Card_Reader;
```

INSTRUCTOR NOTES

THIS IS ANOTHER STRONG FEATURE OF ADA.

THIS IS NECESSARY FOR LARGE SOFTWARE SYSTEM DEVELOPMENT BY ALLOWING MANY PROGRAMMERS TO WORK IN PARALLEL - CODING AND DEBUGGING AND TESTING IN PARALLEL WITH OTHER PROGRAMMERS.

SEPARATE COMPILATION

- DIFFERENT PROGRAM UNITS CAN BE COMPILED SEPARATELY
- THE BODY OF A PROGRAM UNIT CAN BE COMPILED SEPARATELY FROM ITS CORRESPONDING SPECIFICATION
- ALLOWS MANY PROGRAMMERS TO BE DEVELOPING A SYSTEM CONCURRENTLY

INSTRUCTOR NOTES

DON'T GO INTO THE INDIVIDUAL LISTS OF STATEMENTS. JUST SHOW THAT STATEMENTS EXIST TO HANDLE THE LISTED AREAS OF ACTION AND CONTROL. NOTE THAT THESE ARE ALL THE STATEMENTS IN ADA. STATEMENTS ARE SIMILAR TO OTHER LANGUAGES.

STATEMENTS

- EXAMPLE

```
if not Found then
    Text_IO.Put ("Name not found.");
end if;
```

- PROVIDE LOGIC CONTROL OR SPECIFIC ACTIONS

```
FLOW CONTROL:
    GOTO
    IF (CONDITIONAL)
    CASE (CONDITIONAL)
    LOOP & EXIT (ITERATIVE)
    RETURN
    EXCEPTION HANDLERS
    RAISE (EXCEPTIONS)
```

```
BASIC ACTIONS:    SUBPROGRAM CALLS
                   ASSIGNMENT
```

```
REAL-TIME ACTION:  ENTRY CALL
                   ACCEPT
                   ABORT
                   DELAY
                   SELECT
```

```
EXCEPTIONS:       RAISE
```

```
DECLARATION SCOPE: BLOCK
```


OBJECT DECLARATIONS

- EXAMPLE:

Interval : Time_Type;

- ASSOCIATE A NAME WITH AN OBJECT

- ALL OBJECTS MUST BE EXPLICITLY DECLARED

- PERMIT

- CONSTANTS

- VARIABLES

- DYNAMIC CREATION OF A GIVEN OBJECT AT RUNTIME

- CHOICE OF APPROPRIATE NAMES TO ACCURATELY REFLECT THE OBJECTS
USE CAN GREATLY IMPROVE THE UNDERSTANDABILITY OF A SYSTEM
AND THUS MAINTAINABILITY (THEREFORE DECREASED COSTS).

TYPES

- EXAMPLE:
 type List_Type is array (1 .. 15) of Scores_Type;
- A TEMPLATE TO DESCRIBE (NOT CREATE)
 - A SET OF VALUES
 - THE OPERATIONS APPLICABLE TO THOSE VALUES
- PREDEFINED AND USER-DEFINED TYPES
- STRONG TYPING ALLOWS ERROR DETECTION AT COMPILE TIME
 - THE TYPE OF A VARIABLE OR PARAMETER DOES NOT CHANGE ONCE DECLARED
 - EXAMPLE:
 Amount_Of_Gold : Pounds;
 Amount_In_Glass : Ounces;
 Amount_In_Glass := Amount_Of_Gold + 1; -- ILLEGAL
- PROVIDES INCREASED RELIABILITY BECAUSE LANGUAGE CAN BE USED
 - TO PROHIBIT OBJECTS OF DIFFERING LOGICAL TYPES FROM BEING MIXED
 - TO EXPLICITLY STATE DESIGN CONSTRAINTS

INSTRUCTOR NOTES

GENERICIS ARE SIMILAR TO MACROS, BUT MACROS ARE COMPILE-TIME CONCEPTS, AND GENERICIS ARE RUNTIME.

DIFFERENCE BETWEEN GENERICIS AND SUBPROGRAMS:

SUBPROGRAMS CAN PASS VALUES AS PARAMETERS

GENERICIS CAN PASS TYPES OF DATA AS PARAMETERS

REUSABLE PROGRAM UNITS/SOFTWARE COMPONENTS CAN BE AN EFFECTIVE METHOD OF REDUCING OVERALL SOFTWARE COSTS ... BUT REQUIRES THOUGHT AND PLANNING.

GENERIC

- PROBLEMS THAT DIFFER ONLY IN TYPES OF DATA NEED ONLY BE SOLVED ONCE

EXAMPLE:

SORT A LIST OF NAMES

SORT A LIST OF NUMBERS

- PARAMETERIZED TEMPLATES FOR SUBPROGRAMS OR PACKAGES (NOT EXECUTABLE)

- "INSTANTIATION" CREATES AN EXECUTABLE COPY OF THE PROGRAM UNIT AND
SUBSTITUTES THE PARAMETERS

- PROVIDES REUSABLE PROGRAM UNITS

GENERIC

```

generic
  Size : Positive;
  type Item is private;
  package Stack is
    procedure Push (E : in Item);
    procedure Pop (E : out Item);
    Overflow, Underflow : exception;
  end Stack;

  package body Stack is
    type Table is array (Positive range <>) of Item;
    Space : Table (1 .. Size);
    Index : Natural := 0;

    procedure Push(E : in Item) is
    begin
      if Index >= Size then
        raise Overflow;
      end if;
      Index := Index + 1;
      Space(Index) := E;
    end Push;

    procedure Pop(E : out Item) is
    begin
      if Index = 0 then
        raise Underflow;
      end if;
      E := Space(Index);
      Index := Index - 1;
    end Pop;
  end Stack;

```

INSTANCES OF THIS GENERIC PACKAGE CAN BE OBTAINED AS FOLLOWS:

```

package Stack_Int is new Stack(Size => 200, Item => Integer);
package Stack_Bool is new Stack(100, Boolean);

```


OVERLOADING

- CONCEPT OF ONE ENTITY NAME REPRESENTING TWO OR MORE ENTITIES
Put ("Distance Between Points is"); -- OUTPUT String VALUE
Put (Distance); -- OUTPUT NUMERICAL VALUE
- IDENTIFIER NAMES, SUBPROGRAMS, OPERATORS CAN BE OVERLOADED
- ALLOWS PROGRAMMERS TO CHOOSE NAMES APPROPRIATE TO THEIR USE
THE ABSTRACTION AS LONG AS AMBIGUITY CAN BE RESOLVED BY CONTEXT
OR QUALIFICATION

INSTRUCTOR NOTES

IN REAL TIME SYSTEMS YOU CAN'T AFFORD TO ALLOW A SYSTEM TO CRASH BECAUSE SOME "IMPOSSIBLE" STATE WAS REACHED AT SOME POINT IN THE PROGRAM. EXCEPTIONS ALLOW FOR POSSIBLE CORRECTION AND RESUMED EXECUTION, OR AT LEAST A GRACEFUL EXIT FROM EXECUTION.

NOTE:

EXCEPTIONS ARE NOT JUST FOR ERROR CONDITIONS. THEY CAN BE USED TO INDICATE WHEN SOME SPECIFIC STATE IS REACHED AND TO BRING THIS TO THE ATTENTION OF THE PROGRAM FOR HANDLING. (BACKGROUND, ONLY).

EXCEPTIONS

- AN EXCEPTION STOPS SEQUENTIAL EXECUTION WHEN A PARTICULAR CONDITION IS REACHED, AND TRANSFERS CONTROL TO SOME KNOWN LOCATION WHERE THE CONDITION MAY BE HANDLED
- A MECHANISM FOR FAULT-TOLERANT PROGRAMMING
 - ALTERNATIVE TO EXPLICIT ERROR CODE PARAMETERS
 - LANGUAGE ALLOWS DIRECT REPRESENTATION OF REAL WORLD ALGORITHM
- PREDEFINED AND USER-DEFINED EXCEPTIONS
- AID TO RELIABILITY

INSTRUCTOR NOTES

VG 722.1

3-491

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

EXCEPTION

```
begin
...
exception      -- EXCEPTION HANDLER
when Heat_Sensor_Failure =>
...
when others =>
...
end;
```

INSTRUCTOR NOTES

THE MOST IMPORTANT ASPECTS OF THIS FACILITY ARE LISTED IN THE FINAL THREE BULLETS.

BY ENCAPSULATING THE MACHINE DEPENDENT CODE, IT IS EASIER TO MAINTAIN THE SYSTEM OR TO RETARGET BECAUSE WE HAVE LOCALIZED THE AREA OF NECESSARY CHANGE.

MACHINE REPRESENTATION SPECIFICATIONS

for Vehicle_Record'Size use 1000;

- USED (SPARINGLY) TO SPECIFY THE PHYSICAL REPRESENTATION OF OBJECTS, TYPES, ETC.
- PRIMARILY USED FOR EMBEDDED COMPUTER SYSTEMS
- PERMITS INTERFACES WITH FEATURES OUTSIDE THE LANGUAGE (E.G. INTERRUPTS, I/O DEVICES)
- ALLOWS USER TO INTERFACE WITH HARDWARE PERIPHERALS WHILE REMAINING IN HIGH ORDER LANGUAGE
- RECOMMENDED PRACTICE: ENCAPSULATED (GROUP) FOR PORTABILITY, MAINTAINABILITY
- ONLY A SPECIALIZED FEW WILL NEED TO LEARN THIS FEATURE

INSTRUCTOR NOTES

THE LANGUAGE DEFINES A RICH SET OF FACILITIES, BUT DOES NOT DEMAND FULL SUPPORT.

IF YOUR PART OF A SYSTEM HAS SPECIFIC OR LIMITED I/O NEEDS, THEN YOU ONLY NEED TO HAVE THAT WHICH IS ABSOLUTELY NECESSARY TO YOUR PARTICULAR FUNCTION. YOU DON'T NEED TO HAVE ALL POSSIBLE FORMS/FORMATS OF I/O FOR ALL POSSIBLE USES. DECREASES COMPILE OVERHEAD.

INPUT/OUTPUT

- ACCESSED THROUGH PACKAGES (PREDEFINED AND USER-DEFINED)

- USER HAS COMPLETE CONTROL OF I/O

- PREDEFINED I/O

LOW-LEVEL I/O

HIGH-LEVEL I/O

- TEXT I/O

- DIRECT I/O

- SEQUENTIAL I/O

EMPHASIS OF ADA

- USEFUL FOR WIDE RANGE OF APPLICATIONS
EMBEDDED COMPUTER SYSTEMS
SYSTEMS PROGRAMMING
REAL TIME PROGRAMMING
DATA PROCESSING
- DEVELOPMENT BY PROJECT TEAMS
- SOFTWARE ENGINEERING PRINCIPLES ENCOURAGED AND ENFORCED
- MAINTAINABILITY AND RELIABILITY

INSTRUCTOR NOTES

EARLY IN THE ADA DEVELOPMENT PROCESS, IT WAS CLEARLY RECOGNIZED THAT A LANGUAGE ALONE WAS NOT SUFFICIENT TO ACHIEVE DoD's SOFTWARE GOALS.

AN ADA COMPILER IS USED WITHIN AN S/W DEVELOPMENT ENVIRONMENT.

THIS SECTION PROVIDES A RUDIMENTARY CONCEPT OF THE APSE.

ENVIRONMENT OVERVIEW

ENVIRONMENTS

- PROVIDE A SET OF AUTOMATED TOOLS TO AID SOFTWARE DEVELOPERS AT VARIOUS PHASES IN THE LIFE CYCLE

EXAMPLES:

COMPILERS

LINKERS

LOADERS

CODE AUDITORS

PROGRAMMING SUPPORT LIBRARIES

- PRE-ADA SITUATION WITH ENVIRONMENTS
MUST BE DEVELOPED FOR EACH MACHINE
PERSONNEL MUST LEARN A NEW SET OF TOOLS FOR EACH MACHINE
LIMITED TOOL SETS AVAILABLE

INSTRUCTOR NOTES

SPECIFICALLY ADA ENVIRONMENTS.

THE APSE WAS INTENDED TO BE HOSTED ON ONE PHYSICAL MACHINE (GENERALLY A SIZABLE MAINFRAME) WITH THE TARGET MACHINE OF THE DEVELOPMENT PROBABLY A MUCH SMALLER COMPUTER (WHICH WOULD NOT HAVE THE ADDRESS SPACE/PERIPHERALS NECESSARY).

THE DATABASE OF THE APSE IS AN IMPORTANT FEATURE. IT HOUSES ALL PROJECT SOURCE CODE, OBJECT CODE, AND DOCUMENTATION.

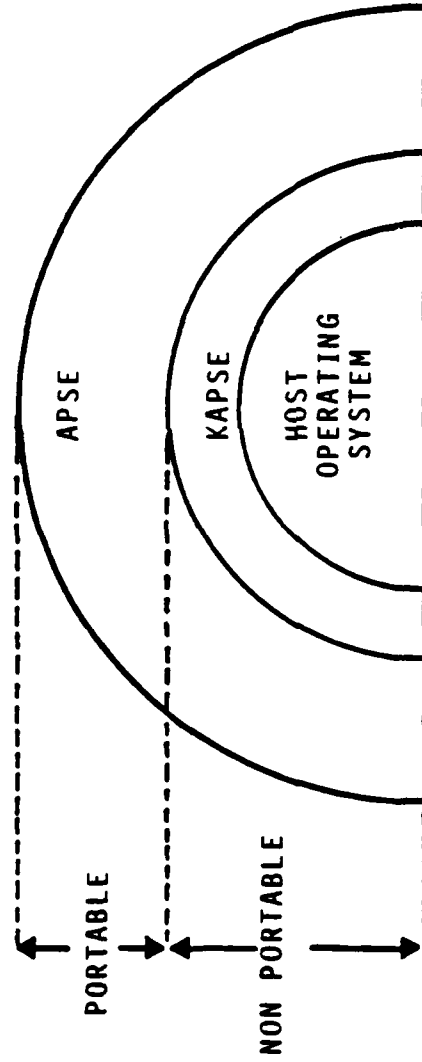
ADA ENVIRONMENTS

- GOAL IS TO PROVIDE AUTOMATED TOOL SUPPORT FOR ALL PROJECT PERSONNEL INVOLVED IN MANAGING, DEVELOPING, AND MAINTAINING SOFTWARE SYSTEMS
- INCLUDES TOOLS FOR ALL PHASES OF LIFE CYCLE
- ADVANTAGES
 - TOOL DEVELOPMENT COSTS REDUCED
 - PORTABILITY OF TOOLS, SOFTWARE, PROGRAMMERS
 - CAN BE USED THROUGHOUT THE LIFE CYCLE
- PORTABILITY ACHIEVED THROUGH A LOW-LEVEL INTERFACE TO THE HOST OPERATING SYSTEM (THE KAPSE) AND MINIMAL SET OF TOOLS (THE MAPSE)

INSTRUCTOR NOTES

CONCEPTUALLY THE STRUCTURE IS IN NESTED LEVELS. AT THE INNER MOST LEVEL IN THE OPERATING SYSTEM IS THE PHYSICAL DATABASE. ABOVE IT, IS THE KAPSE WHICH TAKES CARE OF ALL PHYSICAL TO LOGICAL INTERFACES OF THE ENTIRE APSE. ABOVE THE KAPSE, THE APSE SITS. IT CONTAINS THE SET OF TOOLS NECESSARY TO AID SOFTWARE DEVELOPMENT THROUGHOUT THE LIFE CYCLE.

ADA ENVIRONMENT STRUCTURE



KAPSE: KERNEL ADA PROGRAMMING SUPPORT ENVIRONMENT

APSE: ADA PROGRAMMING SUPPORT ENVIRONMENT

INSTRUCTOR NOTES

VG 722.1

3-57i

J 2 200 1PC 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

APSE

WHAT IS IN EACH PART OF ENVIRONMENT:

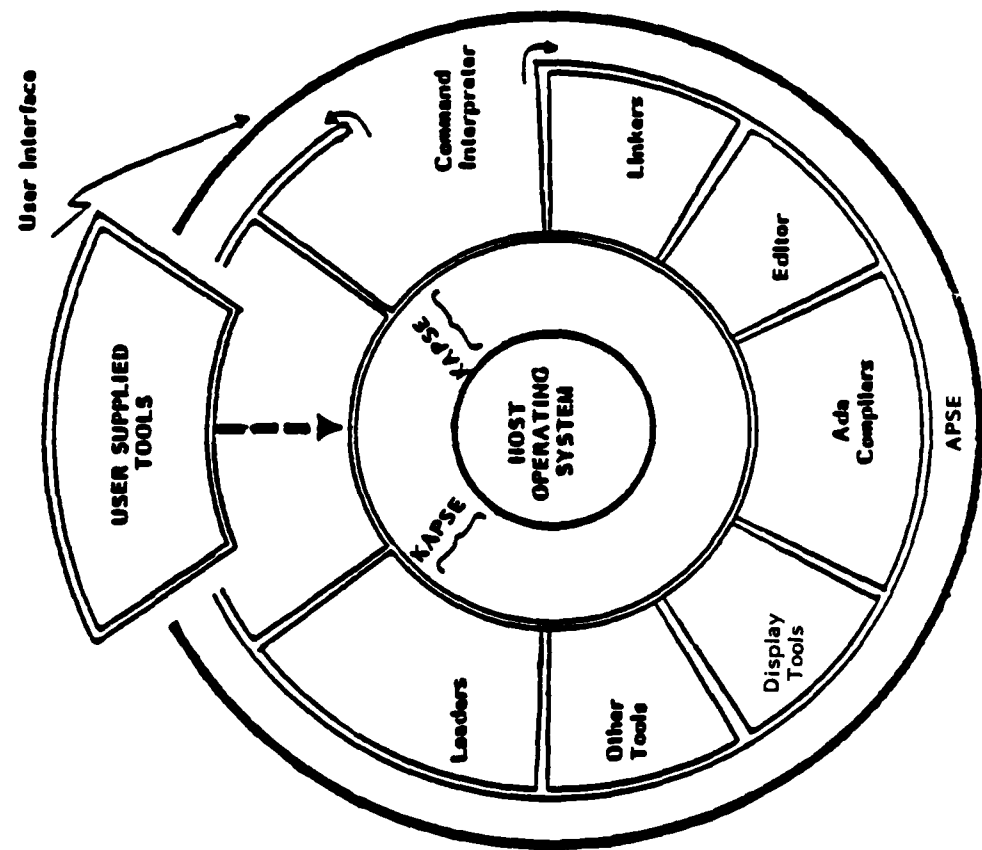
KAPSE: NO EXPLICIT TOOLS BUT SUPPORTS -
DATABASE ACCESS
I/O
TERMINAL TO TOOL ACCESS
RUNTIME SYSTEM

APSE: EXAMPLES OF TOOLS ARE:

| | |
|-------------|-----------------------------|
| COMPILERS | SYMBOLIC DEBUGGER |
| LOADERS | COMMAND INTERPRETER |
| LINKERS | FILE ADMINISTRATOR TOOLS |
| TEXT EDITOR | CONFIGURATION CONTROL TOOLS |

THE KAPSE SHOULD CONTAIN ALL LOW-LEVEL FEATURES NECESSARY TO REHOST ONTO ANOTHER SYSTEM.

APSE STRUCTURE



INSTRUCTOR NOTES

VG 722.1

4-i

Section 4

What are Some Transition Issues with Ada?

INSTRUCTOR NOTES

TO SUMMARIZE WHERE WE HAVE BEEN AND WHERE WE GO FROM HERE.

WE'VE NOW PRESENTED THE MOTIVATION FOR THE DoD ADA EFFORT PLUS WHAT IS MEANT BY THE TERM ADA, A BRIEF INTRODUCTION TO ADA LANGUAGE FEATURES AND THEIR USE IN OUR OVERALL SOFTWARE IMPROVEMENT PICTURE. IN ADDITION, A MANAGER NEEDS TO KNOW WHAT WILL BE DIFFERENT USING ADA OVER ANY OTHER LANGUAGE.

ALLOW 75 MINUTES FOR THIS SECTION.

TOPICS OUTLINE

WHY ADA?

WHAT ADA IS NOT

WHAT ADA IS

WHAT ARE SOME TRANSITION ISSUES WITH ADA

WHERE IS ADA NOW AND TOMORROW

WHAT ARE SOME TRANSITION ISSUES WITH ADA

- DIFFERENCES
- TRAINING
- SELECTION ISSUES
- A TRANSITION STRATEGY
- PDL

INSTRUCTOR NOTES

VG 722.1

4-3i

Abstract

DIFFERENCES WITH ADA

- NEED TO THINK (MUCH) MORE, BUT WILL NEED TO DEBUG (MUCH) LESS
- SOFTWARE "TALKS" ABOUT THE PROBLEM NOT ABOUT THE MACHINE
 COMPUTE_TRAJECTORY;
 not
 LOAD ACCUMULATOR
 ROTATE LEFT
 XOR WITH MASK
- MACHINE PERFORMANCE IS NOT ALWAYS THE ONLY PARAMETER TO OPTIMIZE. AT
 LEAST AS IMPORTANT ARE:
 - SCHEDULE/DEVELOPMENT COSTS
 - RESPONSIVENESS TO CHANGING REQUIREMENTS
 - RELIABILITY

INSTRUCTOR NOTES

VG 722.1

4-4i

1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525

DIFFERENCES WITH ADA (Continued)

- IT WILL BE EASY TO CODE/DESIGN IN FORMER LANGUAGE STYLE BUT HARDER TO LEARN AN "ADA STYLE"
- AN INCREASING IMPORTANCE OF DESIGN, CODING, AND DOCUMENTATION STANDARDS THAT ARE ENFORCED
- MODERN SOFTWARE ENGINEERING PRINCIPLES CAN BE USED MORE NATURALLY
- THE POWERFUL FEATURES OF ADA REQUIRE MORE RESPONSIBILITY, TRAINING AND KNOWLEDGE OF PROGRAMMERS, DESIGNERS, AND MANAGERS. THEY MUST BE USED WITH PURPOSEFUL INTENT
- THE IMPACT OF SYSTEM MODIFICATIONS RESULTING FROM REQUIREMENT CHANGES CAN BE GREATLY REDUCED, BUT CAREFUL PLANNING IS REQUIRED

INSTRUCTOR NOTES

DESIGNERS AND PROGRAMMERS ARE PROBABLY GOING TO HAVE PROBLEMS WITH THE LANGUAGE CONCEPTS LISTED AND A MANAGER CAN EXPECT TO HEAR A LOT ABOUT THEM.

PREDICTABLE DIFFICULTIES WITH THE LANGUAGE

- STRONG TYPING
- MULTI-TASKING APPROACH TO REAL TIME SYSTEMS
- MODULARITY
- STRUCTURED CONTROL
- ABSTRACTION AND INFORMATION HIDING

INSTRUCTOR NOTES

RELATED TO THE LIST IN THE PRECEDING SLIDE. A MANAGER WILL PROBABLY HEAR VARIATIONS OF THESE OBJECTIONS.

VG 722.1

4-6i

WHAT A MANAGER MIGHT HEAR

- "IT CAN'T BE DONE IN ADA"

USUALLY MEANS:

"I CAN'T FIGURE OUT HOW TO DO IT IN ADA"

- "IT'S INEFFICIENT"

MAY MEAN:

"I DON'T UNDERSTAND IT, AND I WON'T LET YOU FORCE ME TO USE IT"

"THE IMPLEMENTATION IS POORLY MATCHED TO OUR NEEDS"

"WE TRIED IT WITH A DIFFERENT LANGUAGE/COMPILER/CPU AND IT WAS INEFFICIENT"

INSTRUCTOR NOTES

ADA HAS VERY POWERFUL FEATURES. USED WITH PURPOSEFUL INTENT AND KNOWLEDGE, THE BENEFITS CAN BE ENORMOUS. ADA MAY ALSO REQUIRE NEW WAYS TO SOLVE A GIVEN PROBLEM.

REMEMBER:

THE AREAS OF GREATEST DIFFICULTY ARE THOSE OFFERING THE GREATEST POTENTIAL BENEFITS.

ORDER-OF-MAGNITUDE IMPROVEMENT REQUIRES NEW TECHNIQUES, AND NEW TECHNIQUES REQUIRE NEW USAGE PATTERNS.

INSTRUCTOR NOTES

LACK OF REAL ENVIRONMENT STANDARDIZATION IS A SOURCE OF CONCERN AND QUESTIONING. IF MANY ENVIRONMENTS ARE BEING DEVELOPED NOW, HOW DOES A MANAGER PICK THE "RIGHT" ONE -- I.E., THE ONE THAT HAS THE MOST TOOLS THAT EVENTUALLY BECOME THE "STANDARD"?

THE PROBLEM IS NOT TOO SERIOUS:

- DoD CONTRACTS ARE LIKELY TO PRESCRIBE AN ENVIRONMENT, AND PROVIDE IT GFE (FREE OF CHARGE).
- ENVIRONMENTS ARE EXPENSIVE TO PRODUCE. YOU WON'T SEE A CHAOTIC MARKET, WITH HUNDREDS OF ENVIRONMENTS.

VG 722.1

4-8i

APSE CONCERNS

- ENVIRONMENT HAS NOT BEEN STANDARDIZED LIKE THE ADA LANGUAGE
LIMITED TOOL EXPERIENCE TO KNOW WHAT IS DESIRABLE OR POSSIBLE
- CURRENT SITUATION
ENVIRONMENTS BEING DEVELOPED BY ARMY, AIR FORCE, PRIVATE INDUSTRY,
AND UNIVERSITIES
EACH IMPLEMENTOR HAS DEFINED ITS OWN ENVIRONMENT, RESULTING IN
SEVERAL "STANDARDS"
- EVENTUAL STANDARD ACHIEVED THROUGH SELECTION OF THE BEST OF CURRENT
IMPLEMENTATIONS
- COMMERCIALY AVAILABLE OFF-THE-SHELF MANAGEMENT, DEVELOPMENT AND
MAINTENANCE TOOLS SHOULD BE AVAILABLE IN FUTURE
- APSE CONFIGURATION MANAGEMENT TOOLS WILL ENCOURAGE AND ENFORCE
CONFIGURATION MANAGEMENT

INSTRUCTOR NOTES

A NEW "MIND SET" REFERS TO THIS: TO EFFECTIVELY USE ADA'S FEATURES REQUIRES CHANGING THE WAY WE VIEW A GIVEN PROBLEM. FOR EXAMPLE, REMEMBER HOW IN THE TRACKING SYSTEM EXAMPLE WE WERE MOSTLY PREOCCUPIED WITH MAKING THE SOFTWARE FLEXIBLE AND RE-USABLE. FOR MOST PROGRAMMERS THAT'S A CHANGE IN MIND SET. SIMILAR RE-ORIENTATION IS REQUIRED FOR MORE MICROSCOPIC LANGUAGE TECHNICALITIES.

ANALOGY: IT'S POSSIBLE TO DESIGN A MECHANICAL CALCULATOR AND THEN SIMULATE IT ELECTRONICALLY. BUT THAT'S NOT THE WAY TO DESIGN AN ELECTRONIC CALCULATOR!

THE TIME SPENT IN RETRAINING DOES EVENTUALLY CARRY COST BENEFITS. SOFTWARE SHOULD BECOME MORE RELIABLE, EASIER TO MAINTAIN. PLUS PROGRAMMERS NEED BE TRAINED ONLY ONCE IF ADA IS USED ON ALL PROJECTS. THUS A PROGRAMMER CAN EASILY SWITCH PROJECTS WITH LESS "GEAR UP" TIME.

ADA TRAINING

- MORE INITIAL TRAINING REQUIRED FOR ADA (THAN FORTRAN, BASIC, PASCAL)

MANY NEW CONCEPTS TO LEARN

DISCOVER NEW WAYS TO VIEW A PROBLEM (A NEW "MIND SET")
- LEARNING ADA IS AN ITERATIVE PROCESS (NEED TO LEARN SOME ADA, USE THE ADA, LEARN MORE ADA ...)

REINFORCEMENT OF LEARNING IS NECESSARY

PARALLEL PROJECTS USING ADA FOR REQUIREMENTS, DESIGN, CODE, TEST
- EXTENT OF RETRAINING (AND TIME REQUIRED) IS DEPENDENT ON AN ORGANIZATIONS AND AN INDIVIDUALS CURRENT LEVEL OF SOFTWARE METHODOLOGY AND HIGH ORDER LANGUAGE EXPERTISE

INSTRUCTOR NOTES

THE MODEL CURRICULUM WAS DEVELOPED BY SOFTECH, INC. FOR THE U.S. ARMY - CECOM.
PRESENTED AS A GUIDE TO START ASSESSING THE TRAINING NEEDS FOR A PARTICULAR ORGANIZATION.

(IF SOMEBODY ASKS) FOR DETAILS, SEE:

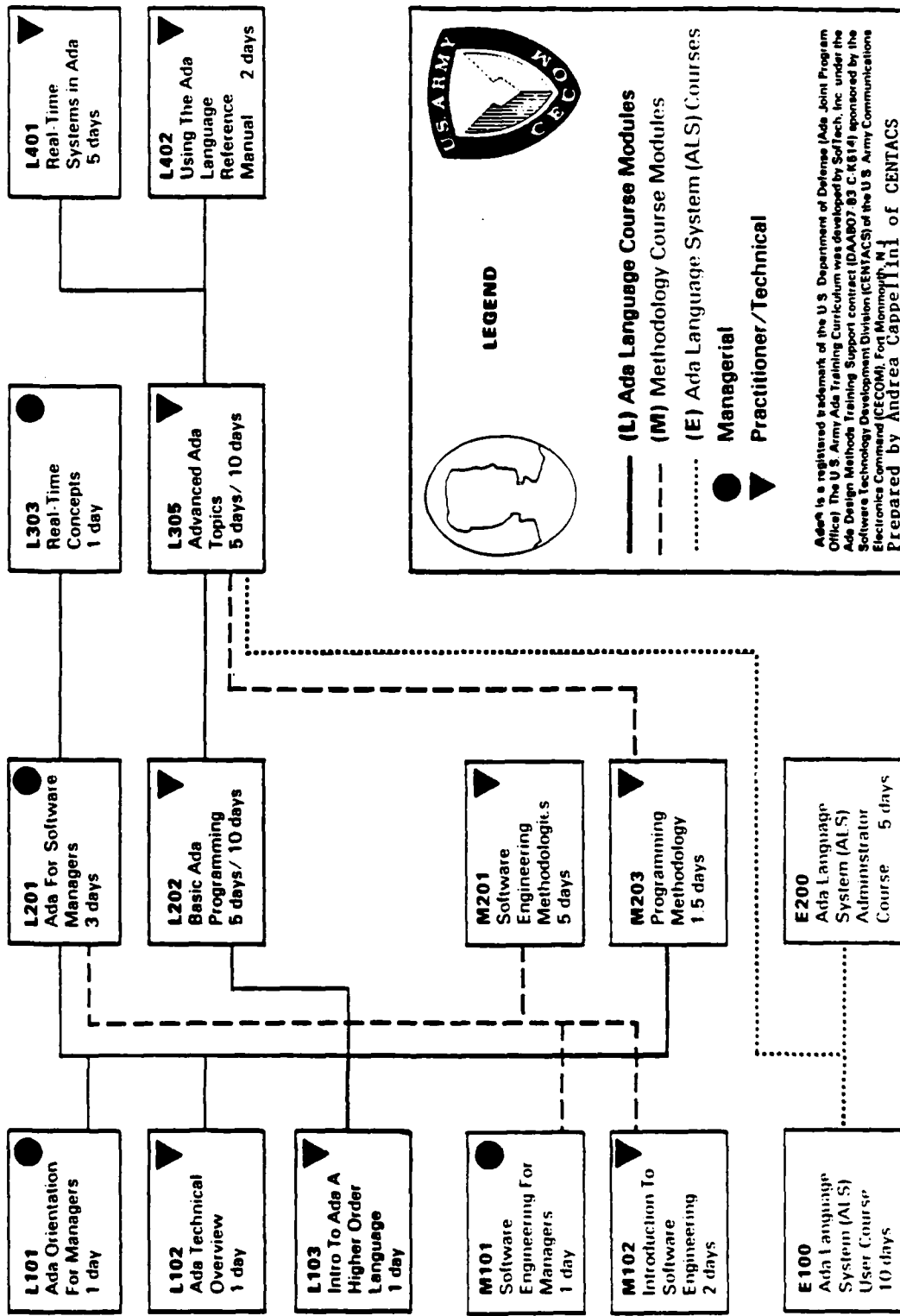
ADA SOFTWARE DESIGN METHODS FORMULATION. FINAL REPORT AVAILABLE FROM

NTIS

SPRINGFIELD, VA

DOCUMENT #AD A124 996

U.S. ARMY ADA TRAINING CURRICULUM



INSTRUCTOR NOTES

ARMY MODEL ADA CURRICULUM (Continued)

BACKGROUND

THE U.S. ARMY ADA TRAINING CURRICULUM DEFINES A COMPREHENSIVE SET OF TRAINING MODULES, OR BUILDING BLOCKS, WHICH CAN BE CONNECTED IN A VARIETY OF WAYS TO FORM THE COURSES AND TRAINING PROGRAMS THAT BEST SATISFY A GIVEN SET OF NEEDS.

THE MODULES DIFFER IN ONE OR MORE OF THE FOLLOWING DIMENSIONS.

1. AREA. KNOWLEDGE OF A PROGRAMMING LANGUAGE CANNOT BE EFFECTIVELY SEPARATED FROM KNOWLEDGE OF A SOFTWARE ENGINEERING METHODOLOGY AND THE TOOLS THAT SUPPORT IT (THE SOFTWARE "ENVIRONMENT"). MODULES WHOSE IDENTIFIER STARTS WITH THE LETTER L ARE CONCERNED WITH THE ADA LANGUAGE PROPER: THE LETTERS M AND E IDENTIFY, RESPECTIVELY, METHODOLOGY AND ENVIRONMENT MODULES.
2. DEPTH. THE CURRICULUM IS DESIGNED TO AVOID THE NEED FOR TRAINING EVERY INDIVIDUAL IN EVERY ASPECT OF THE LANGUAGE, METHODOLOGY OR ENVIRONMENT.
3. VIEWPOINT. ADA AND SOFTWARE ENGINEERING ARE OF INTEREST TO MORE THAN JUST PROGRAMMERS. FOR EXAMPLE, A SYSTEM ADMINISTRATOR HAS LITTLE NEED FOR DETAILED INSTRUCTIONS ON USING MOST OF THE SOFTWARE TOOLS BUT MUST BE FULLY AWARE OF THE DEMANDS THAT SUCH TOOLS MAKE ON SYSTEM RESOURCES, OF ANY BACKUP REQUIREMENTS, AND SO ON. IN TURN, ALL THOSE CONCERNS ARE IRRELEVANT TO A PROGRAMMER, WHO IS ONLY INTERESTED IN USING THE TOOLS.

INSTRUCTOR NOTES

VG 722.1

4-12i

ARMY CURRICULUM (Continued)

TO PROVIDE THE NECESSARY FLEXIBILITY, THE CURRICULUM DOES NOT PRESCRIBE ANY OF THE FOLLOWING:

- a. THE TEACHING METHODOLOGY, SPECIFIC EXAMPLES, PRESENTATION TECHNIQUES, AND MEDIA USED IN EACH MODULE.
- b. THE SPECIFIC CONTENTS OF THOSE COURSES WHICH ARE PARTICULARLY SENSITIVE TO THE CHARACTERISTICS OF DIFFERENT ORGANIZATIONS. MOST NOTICEABLY, METHODOLOGY AND ENVIRONMENT MODULES ARE DEFINED IN GENERAL OUTLINE, AND CAN BE ADAPTED TO ANY METHODOLOGY OR ENVIRONMENT.
- c. THE EXACT TRAINING REQUIRED BY EACH INDIVIDUAL OF AN ORGANIZATION, AND THE TOTAL SET OF SKILLS TO BE TAUGHT IN AN ORGANIZATION.
- d. THE "PACKAGING" OF MODULES INTO COURSES. IT SHOULD BE REMEMBERED THAT A MODULE DEFINES A CAPSULE OF KNOWLEDGE, NOT NECESSARILY A COMPLETE COURSE. THE MOST EFFECTIVE COURSES WILL BE THOSE INTEGRATING SEVERAL MODULES, POSSIBLY FROM DIFFERENT AREAS.

THE CURRICULUM DOES DEFINE A SET OF PRECEDENCES AMONG THE MODULES. THE INTENDED INTERPRETATION OF THE CURRICULUM CHART IS AS FOLLOWS: INPUTS TO THE BOX CORRESPONDING TO A GIVEN MODULE DEFINE THE PREREQUISITES FOR THAT MODULE. IT IS SPECIFICALLY NOT THE INTENT TO RECOMMEND SPECIFIC PATHS THROUGH THE CHART. IN OTHER WORDS, A LINE FROM A BOX B1 TO A BOX B2 MEANS THAT MODULE B2, IF OF INTEREST, SHOULD BE TAKEN AFTER MODULE B1; IT DOES NOT MEAN THAT, AFTER TAKING MODULE B1, AN INDIVIDUAL MUST PROCEED TO MODULE B2.

INSTRUCTOR NOTES

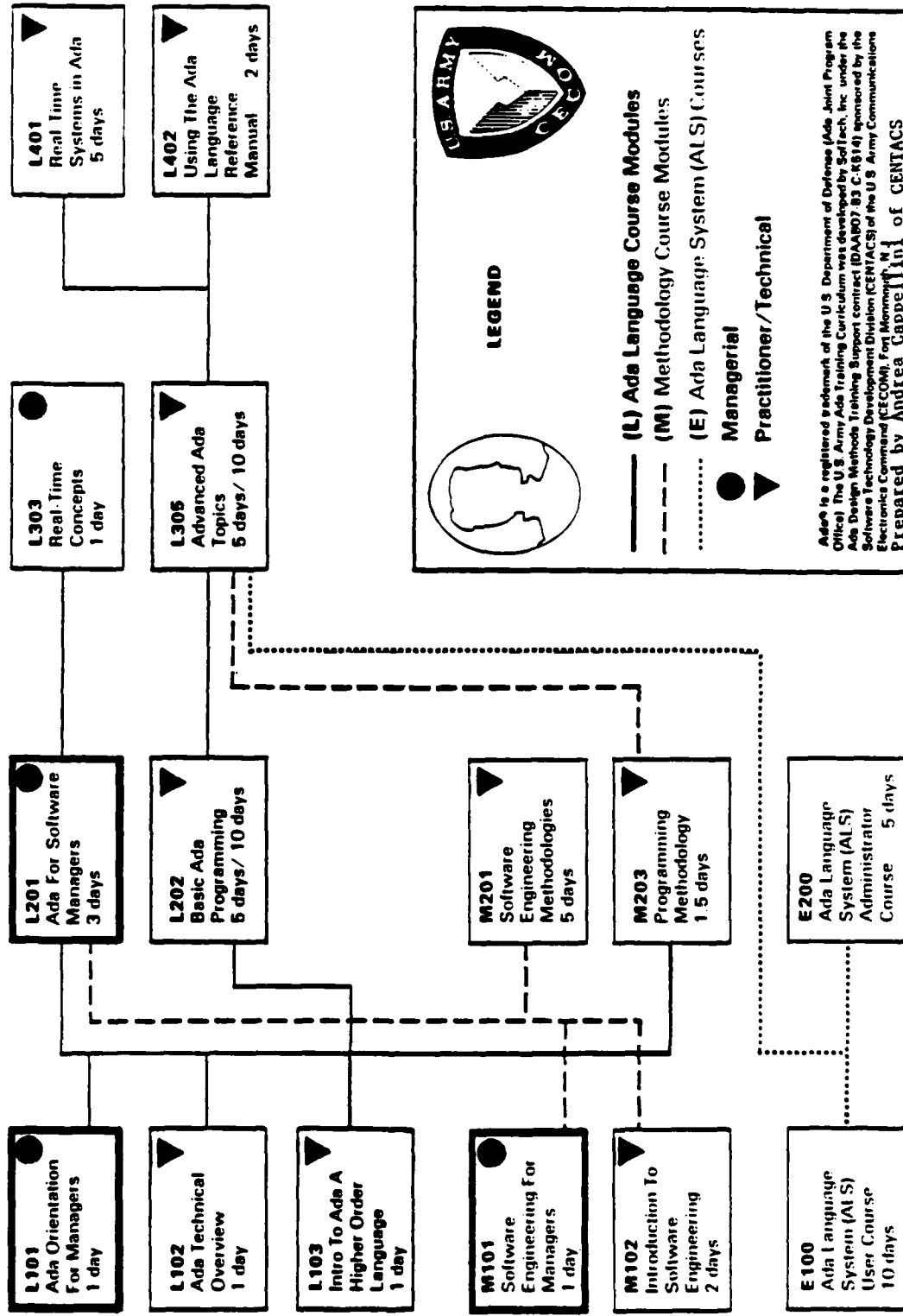
VG 722.1

4-13i

| Year | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025 | 2026 | 2027 | 2028 | 2029 | 2030 | 2031 | 2032 | 2033 | 2034 | 2035 | 2036 | 2037 | 2038 | 2039 | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 | 2096 | 2097 | 2098 | 2099 | 2100 |
|------|
| 1999 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025 | 2026 | 2027 | 2028 | 2029 | 2030 | 2031 | 2032 | 2033 | 2034 | 2035 | 2036 | 2037 | 2038 | 2039 | 2040 | 2041 | 2042 | 2043 | 2044 | 2045 | 2046 | 2047 | 2048 | 2049 | 2050 | 2051 | 2052 | 2053 | 2054 | 2055 | 2056 | 2057 | 2058 | 2059 | 2060 | 2061 | 2062 | 2063 | 2064 | 2065 | 2066 | 2067 | 2068 | 2069 | 2070 | 2071 | 2072 | 2073 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 | 2080 | 2081 | 2082 | 2083 | 2084 | 2085 | 2086 | 2087 | 2088 | 2089 | 2090 | 2091 | 2092 | 2093 | 2094 | 2095 | 2096 | 2097 | 2098 | 2099 | 2100 |

EXAMPLE: SOFTWARE MANAGERS

U.S. ARMY Ada TRAINING CURRICULUM

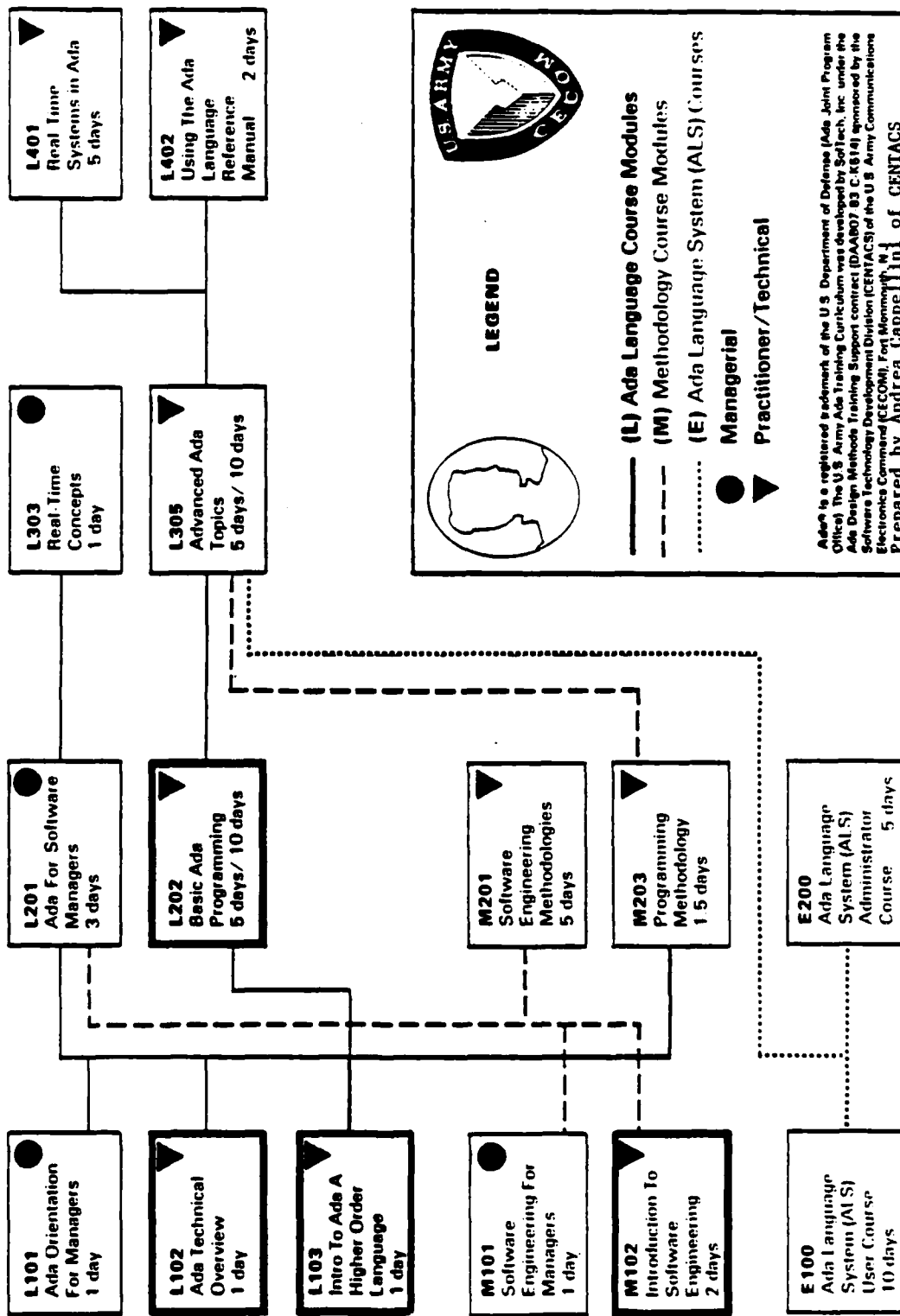


INSTRUCTOR NOTES

IN THIS CONTEXT, A "PROGRAMMER" IS A (TYPICALLY, JUNIOR) PERSON WHO IMPLEMENTS INDIVIDUAL MODULES FOLLOWING SPECIFICATIONS WRITTEN BY A "DESIGNER."

EXAMPLE: JUNIOR PROGRAMMERS

U.S. ARMY Ada TRAINING CURRICULUM



INSTRUCTOR NOTES

A LIST OF CONSIDERATIONS WHEN SELECTING ADA PRODUCTS.

(AN INFORMAL VERSION OF THE SLIDE:

- A COMPILER ALONE AND USE THE EXISTING OPERATING SYSTEM
- YOU ALREADY HAVE AN ENVIRONMENT
- WILL THE OBJECT CODE PRODUCED BY THE COMPILER MEET YOUR APPLICATION DEMANDS
- HOW MUCH ASSEMBLY CODE WILL BE NECESSARY TO MEET YOUR PARTICULAR SPACE/TIME CONSTRAINTS
- WILL YOU NEED SPECIALIZED I/O FACILITIES (OVER AND ABOVE THE LIMITED FUNCTION SUPPLIED WITH ANY ADA COMPILER)
- CONVERSELY, DO YOU WANT A SMALL, BARE-BONES I/O PACKAGE?
- KLUDGING OF I/O AND CONTINUE SUPPORT CAN BE MORE COSTLY IN THE LONG RUN THEN INITIAL CUSTOMIZED SUPPORT
- PERFORMANCE CHARACTERISTICS OF THE RUNTIME SUPPORT SYSTEM. THIS BECOMES VITAL FOR REAL TIME PROCESSING (I.E. TASKS, EXCEPTION, INTERRUPT SCHEDULING)).

WHEN SHOPPING FOR AN IMPLEMENTATION

- COMPILER VS. FULL APSE
- COMPILER VS. OBJECT CODE PERFORMANCE
- I/O PACKAGE*
- RUNTIME SUPPORT SYSTEM*
 - SCHEDULING
 - INTERRUPT HANDLING
 - FAULT TOLERANCE
 - SECURITY
 - DISTRIBUTED TASKING

*HAVE ONE CUSTOMIZED, RATHER THAN RESORTING TO "KLUDGING" THE APPLICATION SOFTWARE (IMPLEMENTATIONS CAN, AND WILL, VARY WIDELY IN THESE AREAS).

INSTRUCTOR NOTES

NEED FOR QUALITY MULTI-PHASE TRAINING CAN'T BE STRESSED ENOUGH.

THE STUDENT NEEDS TO BE ABLE TO POSE QUESTIONS REGARDING THE MATERIAL HE/SHE IS EXPOSED TO AND AS SUCH AN INANIMATE BOOK OR VIDEO-TAPE IS UNABLE TO RESPOND.

ADA CANNOT EFFECTIVELY BE USED WITHOUT AN UNDERSTANDING OF SOFTWARE ENGINEERING PRINCIPLES AND METHODS. FOR EXAMPLE, IF A DESIGNER DOESN'T UNDERSTAND MODULARITY AND INFORMATION HIDING, HOW CAN HE CONSTRUCT A SYSTEM SUCH THAT IT IS RESILIENT TO CHANGE? AND THEN HOW CAN THE PROGRAMMER USE THE ADA FEATURES THAT DIRECTLY SUPPORT THESE CONCEPTS IN SUCH A WAY THAT CHANGES CAN BE MADE TO THE SYSTEM WITH THE EFFECTS OF THAT CHANGE LOCALIZED?

SHOPPING FOR TRAINING

- TEXTBOOKS AND VIDEOTAPES ARE NOT SUFFICIENT
- ONE COURSE IS NOT SUFFICIENT
 - MUST ADDRESS DIFFERENT VIEWPOINTS
 - MUST INTEGRATE WITH METHODOLOGY
- TRUE LEARNING HAPPENS OUTSIDE CLASSROOM
 - PLAN FOR ON-GOING ASSISTANCE
 - IDENTIFY AND PROPERLY TRAIN PROGRAMMERS
 - MAKE IN-HOUSE CONSULTANTS AVAILABLE

INSTRUCTOR NOTES

THE GUIDE PROVIDES ONE POSSIBLE PLAN FOR IMPROVING SOFTWARE DEVELOPMENT, PRODUCTIVITY, AND THE USE OF ADA WITHIN AN ORGANIZATION.

THE ADA EFFORT PROVIDES AN EXCELLENT OPPORTUNITY TO RENEW AND IMPROVE THE SOFTWARE DEVELOPMENT PROCESS AT A PARTICULAR ORGANIZATION -- TO RETAIN THE COMPETITIVE EDGE.

TOP MANAGEMENT'S COMPLETE COMMITMENT CAN'T BE STRESSED ENOUGH. LIP SERVICE IS EASILY SPOTTED BY OTHER PROJECT PERSONNEL AND QUICKLY STOPS ANY BENEFITS THAT MIGHT HAVE STARTED.

THE ADA FOCAL POINT SHOULD BE ABLE TO SEE POSSIBLE IMPLICATIONS OF DoD ACTIVITIES FOR YOUR ORGANIZATION.

INVOLVEMENT OF YOUR BEST PEOPLE EARLY SHOULD BE STRESSED.

GUIDE FOR EFFECTIVE USE OF ADA

- ONLY A SUGGESTION, A STARTING POINT TO YOUR OWN CUSTOMIZED STRATEGY
- TOP MANAGEMENT'S TOTAL COMMITMENT NEEDS TO BE SHOWN IN SPEECH AND ACTION
- APPOINT AN ADA EFFORT FOCAL POINT
KNOWLEDGEABLE IN: ADA
DOD ADA RELATED ACTIVITIES AND STANDARDS
TECHNICAL ASPECTS OF YOUR ORGANIZATION'S PRODUCT
- GET MANY PEOPLE INVOLVED IN THIS EFFORT
THIS INCREASES THE LEVEL AND QUANTITY OF ACCEPTANCE
DEALS WITH PSYCHOLOGICAL FEARS/RESISTANCE

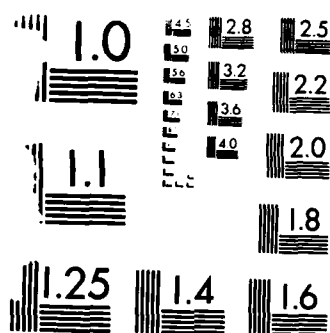
ADA (TRADEMARK) TRAINING CURRICULUM: ADA ORIENTATION
FOR MANAGERS L101 TEACHER'S GUIDE(U) SOFTECH INC
WALTHAM MA 1986 DAAB07-83-C-K514

3/3

F/G 5/9

44

[illegible]



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

GUIDE FOR EFFECTIVE USE OF ADA (Continued)

- LET PERSONNEL KNOW THEY ARE MAKING A REAL CONTRIBUTION TO YOUR ORGANIZATION
 - INCREASES INVOLVEMENT
 - INCREASES ACCEPTANCE
- IDENTIFY WHAT NEEDS TO BE DONE TO IMPROVE YOUR ORGANIZATION'S
 - PRODUCTIVITY
 - USE OF ADA, APSE
 - USE OF MODERN SOFTWARE ENGINEERING METHODS
- DIFFERING AMOUNTS OF RETRAINING WILL BE NECESSARY DEPENDING ON PERSONNEL'S CURRENT LANGUAGES AND METHODOLOGIES -- IDENTIFY FOR YOUR ORGANIZATION

GUIDE FOR EFFECTIVE USE OF ADA (Continued)

- ADOPT OR DEVELOP AN INTEGRATED LIFE-CYCLE METHODOLOGY

WHY?: PROBLEM AND SOLUTION ARE DEALT WITH IN AN ORGANIZED MANNER

WITHOUT: DEVELOPMENT IS MORE SUSCEPTIBLE TO FAILURE
SHOULD BE MORE THAN A REQUIREMENT OR DESIGN LANGUAGE
SHOULD EXPLICITLY ADDRESS OR INCLUDE

- CUSTOMER AND MANAGEMENT REVIEWS DURING EACH LIFE-CYCLE PHASE
- PERSONNEL, TIME, MONEY CONSTRAINTS
- DOCUMENTATION PLANS FOR SOFTWARE SYSTEM AND USER SUPPORT
- DETAILED EXPLANATION OF CONTRIBUTING METHODOLOGIES/CONCEPTS, ADA LANGUAGE, AND DESIGN DECISIONS

WILL INVOLVE RESEARCH, EXPERIMENTS, DISCUSSION

INSTRUCTOR NOTES

AN ORGANIZATION SHOULD BE CONSTANTLY REVIEWING ITS SOFTWARE DEVELOPMENT PROCESS. IN LIGHT OF YOUR EXPERIENCES AND THOSE OF OTHER ORGANIZATIONS, YOU MAY WANT OR NEED TO REVISE AND REFINE PARTS OF YOUR TRANSITION STRATEGY AND METHODOLOGIES. (THIS IS THE ITERATIVE PROCESS.)

GUIDE FOR EFFECTIVE USE OF ADA (Continued)

- DEVELOP AN INCREMENTAL IMPLEMENTATION STRATEGY
 - INVOLVE YOUR BEST PEOPLE EARLY
 - USE EASY-TO-IMPLEMENT ITEMS IN THE EARLY STAGES
 - NEED TO IMPLEMENT THE ENTIRE PLAN (EVEN THE HARD PARTS) OR LITTLE IMPROVEMENT IN SOFTWARE PRODUCTS WILL BE REALIZED
- UPGRADING SOFTWARE DEVELOPMENT PRACTICES/PRODUCTIVITY IS AN ITERATIVE PROCESS

WHAT IS A PROGRAM DESIGN LANGUAGE ?

- DOCUMENTS A DESIGN
- "STRUCTURED ENGLISH"
- FLOW OF CONTROL REPRESENTED BY A LIMITED SET OF STRUCTURED FORMS
- REPLACES FLOW-CHARTS

INSTRUCTOR NOTES

UNFORTUNATELY, "ADA PDL" IS USED BY DIFFERENT PEOPLE WITH DIFFERENT MEANINGS, LET'S TRY TO SORT THINGS OUT.

THE CONCEPT OF PDL GOES BACK TO THE LATE 60'S.

BUT WHAT DOES "DESIGN" MEAN ?

- LATE "60's AND 70's"

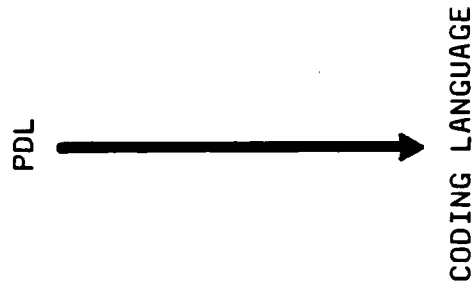
GENERAL STRUCTURE OF ALGORITHMS

- NOW,

MODULAR DECOMPOSITION OF A SYSTEM (THE ARCHITECTURE)

HIGH-LEVEL LANGUAGES AS A PDL

- DESIGN CAN BE CHECKED BEFORE CODING IN A LOWER-LEVEL LANGUAGE
- AUTOMATED DOCUMENTED SUPPORT



INSTRUCTOR NOTES

FIRST AMBIGUITY: WHEN WE TALK ABOUT "ADA PDL" IS ADA

THE PDL

THE CODING LANGUAGE

BOTH

NEITHER

ADA/PDL



ASSEMBLER-FORTRAN-JOVIAL-PL/I

- ADA PROCEDURE/FUNCTION CALLS REPLACE ENGLISH ACTION PHRASES IN ORIGINAL PDL'S
- ADA CAN REPRESENT SYSTEM ARCHITECTURE AND INTERFACES (NOT AVAILABLE IN ORIGINAL PDL'S)
- MODERN METHODOLOGICAL APPROACHES SUPPORTED IN ADA CAN BE UTILIZED
- ADA/PDL CAN BE A LOW RISK TRANSITION STRATEGY

INSTRUCTOR NOTES

PURE ADA, OR AUGMENTED ADA? (AS A PDL)

VG 722.1

4-25i

[illegible]

HOW MUCH ADA IN A PDL ?



- DESIGNERS CAN USE MODERN SOFTWARE ENGINEERING CONCEPTS THAT ADA SUPPORTS • DESIGN IS NOT OBSCURED BY LOW-LEVEL DETAILS
- AUTOMATED DESIGN VALIDITY CHECKS • DESIGN IS NOT EXPRESSED IN A FORMAL LANGUAGE
- MUST KNOW ALL OF ADA AND METHODOLOGICAL TECHNIQUES • NEED ONLY KNOW SOME ADA

ADA/PDL



ADA

???

- DESIGNER MAY TEND TO CODE RATHER THAN DESIGN

- A PDL SHOULD HAVE CONSTRUCTS THAT ARE NOT CONSTRAINED BY THE CODING LANGUAGE

!!!

- ADA HAS THE FEATURES NECESSARY TO EXPRESS DESIGN

- DOCUMENTATION OF SYSTEM IS MORE UNDERSTANDABLE AND MAINTAINABLE

INSTRUCTOR NOTES

WE WON'T GET INTO THE OTHER POSSIBILITY: IF ADA IS TO BE THE IMPLEMENTATION LANGUAGES,

WHAT'S A GOOD PDL?

HERE'S A SUMMARY.

VG 722.1

4-27i

ADA AND PDL

?

- ADA/PDL HAS DIFFERENT MEANINGS
- A PDL FOR ADA MAY/MAY NOT BE NEEDED
- A PDL FOR ADA SHOULD PROBABLY BE A HIGHER-LEVEL LANGUAGE
- USING ADA AS A PDL FOR SOME LOWER-LEVEL LANGUAGE CAN BE AN EFFECTIVE TRAINING TOOL FOR ADA TRANSITION

INSTRUCTOR NOTES

VG 722.1

5-i

201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 117

Section 5

Where is Ada Now and Tomorrow?

TOPICS OUTLINE

WHY ADA?

WHAT ADA IS NOT

WHAT ADA IS

WHAT ARE SOME TRANSITION ISSUES WITH ADA

WHERE IS ADA NOW AND TOMORROW

INSTRUCTOR NOTES

TOPICS COVERED ARE:

WHICH COMPILERS AND WHAT KIND OF TRAINING IS AVAILABLE NOW AND IN THE NEAR FUTURE.

WHAT IS DoD DOING THAT IS RELATED TO ADA AND WHAT IMPACT IT COULD HAVE ON AN ORGANIZATION.

FINALLY, WHERE CAN YOU FIND MORE INFORMATION.

WHERE IS ADA NOW AND TOMORROW

- DoD AND ADA RELATED ACTIVITIES

- FOR MORE INFORMATION

- SUMMARY

DoD AND ADA RELATED ACTIVITIES

ADA JOINT PROGRAM OFFICE (AJPO)

- ESTABLISHED IN DECEMBER 1980
- TRI-SERVICE OFFICE UNDER DUSD
- RESPONSIBILITIES
 - DISTRIBUTION, MAINTENANCE AND CONFIGURATION MANAGEMENT OF DoD USED ADA TOOLS, LIBRARIES, ENVIRONMENTS
 - COORDINATE DoD - FUNDED ADA EFFORTS
 - COORDINATE ADA EDUCATION AND TRAINING PROGRAMS
 - TECHNICAL POINT OF CONTACT FOR ADA
 - ENFORCE DoD POLICY
 - COORDINATE AND MANAGE ADA COMPILER VALIDATION
- IMPORTANCE: DoD OFFICE RESPONSIBLE FOR THE CONTROL AND ENCOURAGEMENT OF THE DEVELOPMENT OF THE ADA LANGUAGE, ENVIRONMENT, METHODOLOGIES AND ITS IMPLEMENTATION IN DoD SYSTEMS

INSTRUCTOR NOTES

THE IMPLEMENTING FORCE OF THE ADA EFFORT FOR DoD.

VHSIC = VERY HIGH SPEED INTEGRATED CIRCUIT

OUSD = OFFICE OF THE UNDERSECRETARY OF DEFENSE

R&AT = RESEARCH AND ADVANCED TECHNOLOGY

VG 722.1

5-5i

SOFTWARE TECHNOLOGY FOR ADAPTABLE RELIABLE SYSTEMS (STARS)

- FORMERLY CALLED SOFTWARE INITIATIVE
- DoD STRATEGY TO AUGMENT THE ADA AND VHSIC PROGRAMS IN AN EFFORT TO MAINTAIN U.S. LEAD IN COMPUTER TECHNOLOGY
- GOAL IS TO IMPROVE PRODUCTIVITY, ACHIEVE GREATER RELIABILITY AND ADAPTABILITY
- SEVEN YEAR PLAN COMMENCING FY82 COORDINATED BY JOINT SERVICE TEAM UNDER OUSD/R&AT TO BE EXECUTED BY DoD ORGANIZATIONS

INSTRUCTOR NOTES

SEI CONTACTS:

JOHN MANLEY, DIRECTOR

MARLO BARBACCI, ASSOCIATE DIRECTOR

SOFTWARE TECHNOLOGY FOR ADAPTABLE RELIABLE SYSTEMS (STARS) (Continued)

- NATIONAL SOFTWARE ENGINEERING INSTITUTE HAS BEEN ESTABLISHED
AT CARNEGIE MELLON UNIVERSITY IN PITTSBURGH TO TEST NEW
TECHNIQUES AND STATE-OF-THE-ART CAPABILITIES FOR APPLICATION
TO REAL PROJECTS
- PLAN IS STILL UNDER REVISION
- IMPORTANCE: A TOOL FOR NATIONAL BUSINESS AND TECHNOLOGY
SURVIVAL WITH ADA AS A VEHICLE

INSTRUCTOR NOTES

FREEMAN AND WASSERMAN ARE TWO PROFESSORS FROM CALIFORNIA.

(IF SOMEBODY ASKS) METHODMAN IS AVAILABLE FROM

NTIS

SPRINGFILED, VA

DOCUMENT #AD A123 449

EDUCATIONMAN/METHODMAN

- STUDIES DONE FOR THE AJPO
- METHODMAN
 - PART OF STUDY BY PETER FREEMAN AND TONY WASSERMAN
 - LISTS CHARACTERISTICS THAT METHODOLOGIES FOR USE WITH ADA SHOULD HAVE
 - CURRENT STUDY COMPLETED; WITH MORE WORK COULD BECOME BASIS FOR FUTURE MIL-STD FOR DEVELOPMENT OF MILITARY SYSTEMS
- EDUCATIONMAN
 - SCOPE OF STUDY WILL INCLUDE METHODS FOR TEACHING ADA AND SOFTWARE METHODOLOGIES

INSTRUCTOR NOTES

STANDARDIZATION IS A THORNY SUBJECT. STANDARDIZE TOO EARLY, THE RISK OF A POORER QUALITY EXISTS. STANDARDIZE TOO LATE, PROLIFERATION HAS ALREADY OCCURRED AND IT BECOMES ALMOST IMPOSSIBLE TO STANDARDIZE (E.G. PASCAL).

STANDARDIZATION

- OF ADA LANGUAGE

- RESTRICTS PROLIFERATION OF ADA DIALECTS TO A MINIMUM

- ANSI/MIL-1815A AS OF 17 FEBRUARY 1983

INSTRUCTOR NOTES

THE CONTRACTOR FOR THE ACVC WAS THE AJPO WHEREAS THE CONTRACTOR FOR THE COMPILER IS THE ARMY.

(IF SOMEBODY ASKS):

ACVC AND ARMY CONTRACTS WERE WON INDEPENDENTLY BY TWO DIFFERENT ORGANIZATIONS WITHIN SOFTECH.

(IF SOMEBODY ASKS) "AVO POLICIES AND PROCEDURES" IS AVAILABLE FROM:

NTIS
SPRINGFIELD, VA

DOCUMENT #PB83 110601

STANDARDIZATION (Continued)

- OF ADA COMPILERS
 - HELP REDUCE DEVELOPMENT/MAINTENANCE COSTS OF SOFTWARE, IMPROVE QUALITY OF SOFTWARE, IMPROVE SUPPORT TOOLS/ENVIRONMENTS
- COMPILER VALIDATION
 - ADA VALIDATION OFFICE (AVO)
 - DEPARTMENT OF AJPO
 - PERFORMS FORMAL VALIDATION
 - ADA COMPILER VALIDATION CAPABILITY (ACVC)
 - TEST SUITE USED IN VALIDATION
 - APPROXIMATELY 1600 TESTS DEVELOPED BY SOFTECH UNDER CONTRACT TO THE AJPO
 - VERSION 1.0 RELEASED 17 FEBRUARY 1983

INSTRUCTOR NOTES

THE PURPOSE OF THE KIT AND KITIA WAS TO DEVELOP A SET OF REQUIREMENTS FOR TOOL PORTABILITY. THIS EFFORT RESULTED IN THE CAIS, OR COMMON APSE INTERFACE SET.

SCFTECH WAS AWARDED THE CAIS CONTRACT IN NOVEMBER 1985 TO FURTHER REFINE THESE REQUIREMENTS INTO A SPECIFICATION. SOME PROTOTYPES MAY ALSO BE INCLUDED IN THIS EFFORT.

STANDARDIZATION (Continued)

- OF ADA ENVIRONMENTS
 - PROMOTES INTEROPERABILITY (DATA) AND TRANSPORTABILITY (TOOLS)
 - KIT/KITIA
 - KAPSE INTERFACE TEAM (KIT)
 - ORGANIZED BY AJPO
 - OVERSEEN BY NAVAL OCEAN SYSTEMS CENTER (NOSC)
 - PERSONNEL ARE MILITARY AND DOD CONTRACTORS
 - ESTABLISHED TO IDENTIFY, EXAMINE, AND SET STANDARDIZATION POLICIES FOR KAPSE INTERFACES
 - STANDARD KERNEL INTERFACE SPECIFICATION BY 1985
 - KAPSE INTERFACE TEAM INDUSTRY/ACADEMIA (KITIA)
 - INDUSTRY AND ACADEMIC COUNTERPART OF KIT
 - CAIS
 - COMMON APSE INTERFACE SET

INSTRUCTOR NOTES

THE AIR FORCE HAS HAD MANY PROBLEMS WITH INTRODUCING JOVIAL J73 AS THE AF STANDARD LANGUAGE. IT HOPES TO AVOID THOSE DIFFICULTIES WITH ADA. AS A RESULT IT IS ATTEMPTING TO CONTROL ADA INTRODUCTION SO THAT USER WILL USE ADA BECAUSE THEY WANT TO NOT BECAUSE THEY HAVE TO. JOVIAL WAS MANDATED FOR ECS USE WITHOUT THE COMPILERS BEING OF SUFFICIENT MATURITY.

ADA INTRODUCTION PLAN

- AIR FORCE PROPOSED METHOD OF INTRODUCING ADA
 - TO BE DONE IN 4 PHASES
 - BEGIN/END OF PHASE DEPENDENT ON "MATURITY" CRITERIA OF COMPILERS, TOOLS, DOCUMENTATION
 - USE IN PARALLEL PROJECT DEVELOPMENT THE KEY
 - GREATER CONTRACTOR ACCEPTANCE
- PLAN IS STILL UNDER REVIEW

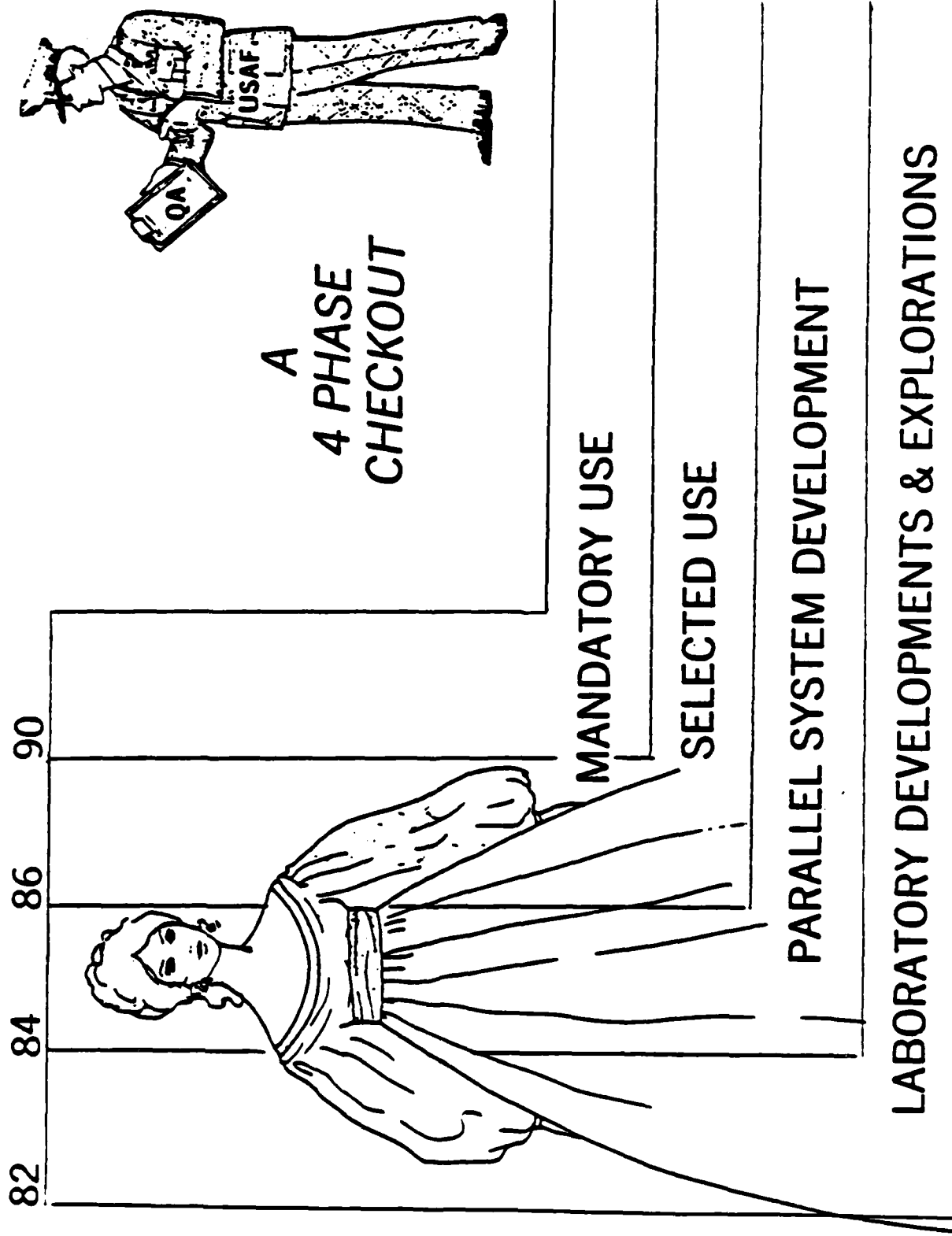
INSTRUCTOR NOTES

NOTE THE DATES ARE NOT FIXED; THIS IS ONLY A GUIDE. THE MATURITY CRITERIA ARE TO BE THE REAL CUT OFF POINTS.

VG 722.1

5-12i

AF ADA ECS INTRODUCTION



INSTRUCTOR NOTES

NOTE FINAL BULLET.

NO RESTRICTIONS ON USE OF ADA MEANS ALL APPLICATION AREAS (EMBEDDED OR DATA PROCESSOR OR...) ARE FREE TO USE IT.

WILL BE REISSUED AS DoD DIRECTIVE 3405.xx.

VG 722.1

5-13i

DoD D 5000.31 REVISION

- DoD PROGRAMMING LANGUAGE POLICY
- WILL APPLY TO ALL DoD COMPUTER APPLICATIONS
- ONLY APPROVED HOLDS TO BE: ADA
CMS-2
FORTRAN
COBOL
ATLAS
JOVIAL
- WAIVERS CONTROLLED BY INDIVIDUAL SERVICES
- NO RESTRICTIONS ON USE OF ADA
- CURRENTLY UNDER DEBATE BY TRI-SERVICES
- IMPORTANCE: MANDATORY USE OF ADA IN FUTURE

INSTRUCTOR NOTES

THIS IS THE EXCITING POSSIBILITY OF THE ADA LANGUAGE AND OTHER LANGUAGES TO COME.
IMAGINE SOFTWARE "CHIPS" AS WE NOW HAVE HARDWARE CHIPS THAT ARE PLUGGED IN AS NEEDED.

WRITING REUSABLE MODULES REQUIRES GREAT ABSTRACTION POWER. YOU MUST THINK OF A MODULE
THAT NOT ONLY SOLVES YOUR PRESENT PROBLEM, BUT ALSO COULD BE USEFUL AGAIN, POSSIBLY IN
OTHER PROJECTS.

SOFTWARE COMPONENTS INDUSTRY

- REUSABLE, OFF-THE-SHELF SOFTWARE WITH ADA
- WILL HELP ACHIEVE DoD GOALS OF
 - INCREASED PRODUCTIVITY
 - INCREASED RELIABILITY
 - DECREASED COST
- WRITING REUSABLE SOFTWARE MAY REQUIRE A CHANGE IN MIND SET

INSTRUCTOR NOTES

ADA/JUG IS A USER-ORIENTED GROUP WITH TOP DEFENSE CONTRACTORS, IMPLEMENTORS, EDUCATORS, GOVERNMENT OFFICIALS (AF, AJPO) MEETING TO EXCHANGE CURRENT STATUS, CONCERNS, NEW IDEAS.

SIGADA HAS MORE OF AN IMPLEMENTORS, RESEARCH BENT.

ADA LETTERS IS A PUBLICATION OF THE SPECIAL ADA INTEREST GROUP OF THE ACM.

ARPA NET IS AN AUTOMATED ADA INFORMATION AND MAIL SYSTEM.

ACCOUNTS AVAILABLE TO GOVERNMENT CONTRACTORS FOR SPECIFIC PROJECTS. SEE YOUR CONTRACTING OFFICER.

VG 722.1

5-15i

FOR MORE INFORMATION

- ADA - JOVIAL USERS GROUP (AdaJUG)
- SIGADA
- ADA LETTERS
- ADA JOINT PROGRAM OFFICE (AJPO)
- ARPANET
- SEMINARS
- BOOKS

AJPO

ADA JOINT PROGRAM OFFICE
1211 SOUTH FERN STREET
ROOM C-107
ARLINGTON, VA 22202

(202) 694-0208 (ADA INFORMATION CLEARINGHOUSE)

SIGADA AND ADA LETTERS

ADATEC: TECHNICAL STUDY GROUP OF THE ACM

ADALETTERS: SIGADA PUBLICATION

FOR MEMBERSHIP IN ACM SIGADA

ACM, INC.

P.O. BOX 12115

CHURCH STREET STATION

NEW YORK, NY 10249

INSTRUCTOR NOTES

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

ADAJUG

LANGUAGE CONTROL FACILITY

CAROLE STEELE

ASD/ADOL

WRIGHT-PATTERSON AFB, OH 45433

(513) 255-4472

INSTRUCTOR NOTES

THIS SECTION PROVIDES A QUICK REVIEW OF THE MODULE.

IN SUMMARY

INSTRUCTOR NOTES

VG 722.1

5-20i

THE UNIVERSITY OF CHICAGO

WE RAISED SOME QUESTIONS ABOUT ADA ...

WHY ADA?

WHAT IS ADA?

HOW DOES ADA HELP?

TRANSITIONING TO ADA?

WHERE IS ADA NOW AND TOMORROW

INSTRUCTOR NOTES

VG 722.1

5-21i

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

WHY ADA ?

THE ADA EFFORT IS CENTRAL TO DoD'S PLANS TO COMBAT THE SOFTWARE CRISIS.

WHAT IS ADA ?

- ADA IS NOT A PANACEA
- ADA IS A COMBINATION OF A LANGUAGE, AN ENVIRONMENT, SOFTWARE ENGINEERING PRINCIPLES AND METHODS
- THE ADA LANGUAGE IS COMPREHENSIBLE AND DESIGNED FOR SPECIFIC GOALS
- THE ADA PROGRAMMING SUPPORT ENVIRONMENT IS EXPANDABLE TO MEET OUR GROWING NEEDS FOR COMPLEXITY MANAGEMENT

HOW DOES ADA HELP ?

THE ADA EFFORT AND THE SOFTWARE CRISIS:

- MODERN SOFTWARE ENGINEERING METHODS
 - INCREASED PRODUCTIVITY
 - INCREASED RELIABILITY, MAINTAINABILITY
- COMMON HIGH ORDER LANGUAGE
 - DESIGNED WITH MODERN SOFTWARE DEVELOPMENT METHODS
 - SUPPORTS THE MANAGEMENT OF COMPLEXITY AND CHANGING REQUIREMENTS
- COMMON SUPPORT ENVIRONMENT
 - REDUCES COST OF WRITING CUSTOMIZED SYSTEMS PROGRAMS
 - PORTABILITY OF SOFTWARE/PROGRAMMERS
 - LIFE CYCLE SUPPORT OF SOFTWARE DEVELOPMENT

TRANSITIONING TO ADA

- ADA WILL BE DIFFERENT
 - MORE INITIAL TRAINING
 - COMPILERS/ENVIRONMENT MUST BE CAREFULLY SELECTED
 - INCREASED DESIGN EMPHASIS
- ADA/PDL CAN HELP

INSTRUCTOR NOTES

SOME OF THESE TOPICS HAVE BEEN ADDRESSED BRIEFLY IN THIS MODULE, BUT THIS IS A LIST OF THE SCOPE OF UNDERSTANDING A MANAGER WILL NEED TO EFFECTIVELY DEAL WITH ADA IN HIS OR HER ORGANIZATION.

HOW MUCH SHOULD A MANAGER LEARN ?

- HOW TO INTRODUCE ADA/METHODOLOGY
- GENERAL CHARACTERISTICS OF LOCAL METHODOLOGY
- WHAT TO LOOK FOR IN DESIGN REVIEWS/DOCUMENTS
- USE OF MANAGEMENT TOOLS (IF/AS APPROPRIATE)
- RECOGNIZING INVALID TECHNICAL OBJECTIONS
- HANDLING GENUINE DIFFICULTIES

ADA IS NOW

001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 054 055 056 057 058 059 060 061 062 063 064 065 066 067 068 069 070 071 072 073 074 075 076 077 078 079 080 081 082 083 084 085 086 087 088 089 090 091 092 093 094 095 096 097 098 099 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 101

Material: Ada Orientation for Managers (L101)

We would appreciate your comments on this material and would like you to complete this brief questionnaire. The completed questionnaire should be forwarded to the address on the back of this page. Thank you in advance for your time and effort.

1. Your name, company or affiliation, address and phone number.

2. Was the material accurate and technically correct?

Yes ☐

No ☐

Comments:

3. Were there any typographical errors?

Yes ☐

No ☐

If yes, on what pages?

4. Was the material organized and presented appropriately for your applications?

Yes ☐

No ☐

Comments:

5. General Comments:

place
stamp
here

COMMANDER
US ARMY MATERIEL COMMAND
ATTN: AMCDE-SB (OGLESBY)
5001 EISENHOWER AVENUE
ALEXANDRIA, VIRGINIA 22233

END
DTIC
FILMED
4-86