

CONTROL OF FLIGHT SOFTWARE DEVELOPMENT COSTS

MARCH 11, 1971

20 pages

WSU SPECIAL COLLECTIONS  
FOR RESEARCH USE ONLY

WSU S2B2FF35

PROBLEM

SOFTWARE IS EXPENSIVE

SCHEDULES ARE DIFFICULT TO PREDICT AND TO MEET

COST AND SCHEDULE OVERRUNS ARE COMMON

FUTURE PROGRAMS WILL BE 10 TIMES LARGER THAN PAST PROGRAMS

HOW CAN SOFTWARE COSTS BE KEPT UNDER CONTROL?

TYPICAL PROGRAMMING PROBLEM AREAS.

COORDINATION OF SOFTWARE EFFORT

SOFTWARE SPECIFICATIONS

SCHEDULES

MACHINE RESOURCES (FIXED AND ERASABLE MEMORY)

CODING DIFFICULTIES (FIXED POINT, LACK OF COMPILER, ETC.)

ERASABLE MANAGEMENT

RESTART PROTECTION

VERIFICATION TESTING

COMMUNICATION OF CRITICAL DATA

DEFINITION OF INTERFACES

DOCUMENTATION

CONFIGURATION CONTROL

e, |

SOLUTIONS

DEVELOP COMPREHENSIVE MANAGEMENT PLAN  
DEFINE MAJOR PROBLEM AREAS  
SPECIFY TOOLS AND TECHNIQUES REQUIRED TO MAINTAIN CONTROL  
OVER SOFTWARE DEVELOPMENT

SOFTWARE MANAGEMENT PLAN

IDENTIFY KEY ELEMENTS OF SOFTWARE DEVELOPMENT  
DEFINE INTER-RELATIONSHIPS AMONG THESE ELEMENTS  
DEFINE MAJOR PROBLEM AREAS  
IDENTIFY REQUIRED TOOLS AND TECHNIQUES

WSU SPECIAL COLLECTIONS  
FOR RESEARCH USE ONLY

KEY ELEMENTS OF SOFTWARE DEVELOPMENT

TRANSLATION OF TOP LEVEL MISSION REQUIREMENTS INTO SOFTWARE

REQUIREMENTS AND ORGANIZATION

SOFTWARE ANALYSIS, DESIGN, AND SPECIFICATION

CODE GENERATION

SOFTWARE VERIFICATION

DOCUMENTATION

CONFIGURATION CONTROL

HARDWARE AND SOFTWARE SPECIFICATION

SPECIFY HARDWARE AND SOFTWARE TO MINIMIZE COST OF CODE

GENERATION AND VERIFICATION

UTILIZE RADICAL SIMPLIFICATION OF HARDWARE AND SOFTWARE

STRUCTURE, IF NECESSARY, TO CONTROL COSTS

SPECTRUM OF SOFTWARE STRUCTURES

MOST COMPLEX

1. GENERAL PURPOSE STRUCTURE  
OPERATING SYSTEM (ACCESS METHODS, RESOURCE ALLOCATION, ETC)  
MULTIPROGRAMMED JOB SCHEDULING  
INTERRUPTS
2. AGC STRUCTURE  
NO OPERATING SYSTEM  
MULTIPROGRAMMED JOB SCHEDULING  
INTERRUPTS
3. PSEUDO BATCH-MODE STRUCTURE (STS TASK 5)  
SHORT JOB SEGMENTS  
BATCH SCHEDULING OF JOBS  
NO INTERRUPTS
4. POINTER DRIVEN STRUCTURE (POSEIDON)  
NO MULTIPROGRAMMING  
DETERMINISTIC PROGRAM SEQUENCING

LEAST COMPLEX

EXAMPLE I: GENERAL PURPOSE STRUCTURE

ADVANTAGES

EXPERIENCE WITH GROUND BASED SYSTEMS  
EXTENSIVE LOGICAL SWITCHING AND SUPERVISORY CAPABILITY

DISADVANTAGES

VERY EXPENSIVE TO GENERATE AND TO VERIFY CODE  
COMPLEX OPERATING SYSTEM IS INAPPROPRIATE TO A WELL DEFINED  
APPLICATION  
DIFFICULT TO SPECIFY  
COST LIKELY TO GET OUT OF CONTROL

WSU SPECIAL COLLECTIONS  
FOR RESEARCH USE ONLY



## EXAMPLE 2: AGC PROGRAM STRUCTURE

### ADVANTAGES

IDEAL FOR MACHINE WITH:

LIMITED RESOURCES (TIME, MEMORY, ETC.)  
SHORT RESPONSE TIME REQUIREMENTS  
NO SUBSYSTEM COMPUTATIONAL CAPABILITY

MODERATE COST FOR CODE GENERATION AND VERIFICATION  
FLEXIBLE PROGRAM STRUCTURE  
MODULAR PROGRAM ELEMENTS  
EASILY SPECIFIABLE

### DISADVANTAGES

INTERRUPTS REQUIRED  
MULTIPROGRAMMED SYSTEM IS EXPENSIVE TO VERIFY  
DIFFICULT TO PREVENT INTERACTION BETWEEN COMPETING JOBS

EXAMPLE 3: PSEUDO BATCH-MODE STRUCTURE (STS TASK 5)

ADVANTAGES

HIGHLY SIMPLIFIED SOFTWARE STRUCTURE  
NO INTERRUPTS OR MULTIPROGRAMMING  
SIMPLIFIES PROGRAM SPECIFICATION  
PROMOTES MODULARITY OF PROGRAMS  
ALLOWS SPECIFICATION OF RESOURCES BY:

CPU TIME  
SYSTEM RESPONSE

JOBS ARE RUN AS DECOUPLED SEGMENTS - MINIMAL INTERACTION  
SIMPLIFIED SUB SYSTEM SELL-OFF PROCEDURE

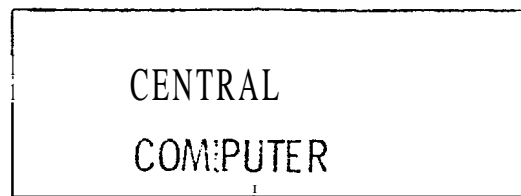
RUN RELEVANT SUBSYSTEM MODULES  
SIMULATE LOAD OF OTHER SUBSYSTEM JOB SEGMENTS

DISADVANTAGES

PREPROCESSING REQUIRED AT SUBSYSTEM LEVEL-  
SYSTEM RESPONSE TIME IS RELATIVELY SLOW  
JOBS MUST BE SEGMENTED

WSU SPECIAL COLLECTIONS  
FOR RESEARCH USE ONLY

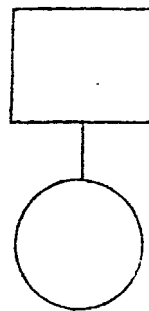
RESPONSE TIME  
AT THIS LEVEL  
 $\approx 10$  MS.



I/O BUS

RESPONSE TIME  
AT THIS LEVEL  
 $\approx 100 \mu$  SEC.

DEDICATED  
LOGIC



SUBSYSTEM

COUNTERS,  
TIMERS, ETC.



SUBSYSTEM

PSEUDO BATCH-MODE STRUCTURE

APOLLO STRUCTURE

LIMITED SUBSYSTEM LOGICAL CAPABILITY  
FLOATING POINT  
HARDWARE RESTART PROTECTION  
MICROPROGRAMMABLE OP-CODES  
TEST-COOPERATIVE HARDWARE  
MINIMAL INTERRUPTS  
COMPILER  
NO MULTIPROGRAMMING  
MINIMAL EXECUTIVE  
MINIMAL PRIORITY STRUCTURE

NO SUBSYSTEM LOGICAL CAPABILITY  
NO FLOATING POINT  
SOFTWARE RESTART PROTECTION  
FIXED OP-CODE REPEATERS  
NON TEST-COOPERATIVE HARDWARE  
EXTENSIVE INTERRUPTS  
ASSEMBLER/INTERPRETER  
MULTIPROGRAMMING  
MODERATE EXECUTIVE  
EXTENSIVE PRIORITY STRUCTURE

Figure 3 Comparison Between Pseudo Batch-Mode Structure and Apollo Structure

HARDWARE SPECIFICATION

FAST CPU  
FLOATING POINT  
SPECIALIZED OP-CODES FOR  
VECTOR-MATRIX MANIPULATION  
MICROPRAMMED OP-CODES  
SINGLE INSTRUCTION RESTART  
ADEQUATE MEMORY  
LONGER WORD LENGTH  
ADEQUATE ERASABLE  
MASS MEMORY  
TEST-COOPERATIVE FEATURES

APOLLO PROBLEM SOLVED

COMPUTATIONAL OVERLOAD REDUCED  
ELIMINATE FIXED POINT OPERATIONS  
ELIMINATE INTERPRETIVE OPERATIONS  
DELAY DETAILED MACHINE SPECIFICATION  
ELIMINATE SOFTWARE RESTART PROBLEMS  
GREATER PRECISION, FEWER DOUBLE AND  
TRIPLE PRECISION COMPUTATIONS  
REDUCE ERASABLE CONFLICTS  
REMOVE LIMIT ON GROWTH OF MEMORY  
REDUCE VERIFICATION COST AND IMPROVE  
CONFIDENCE IN FLIGHT CODE

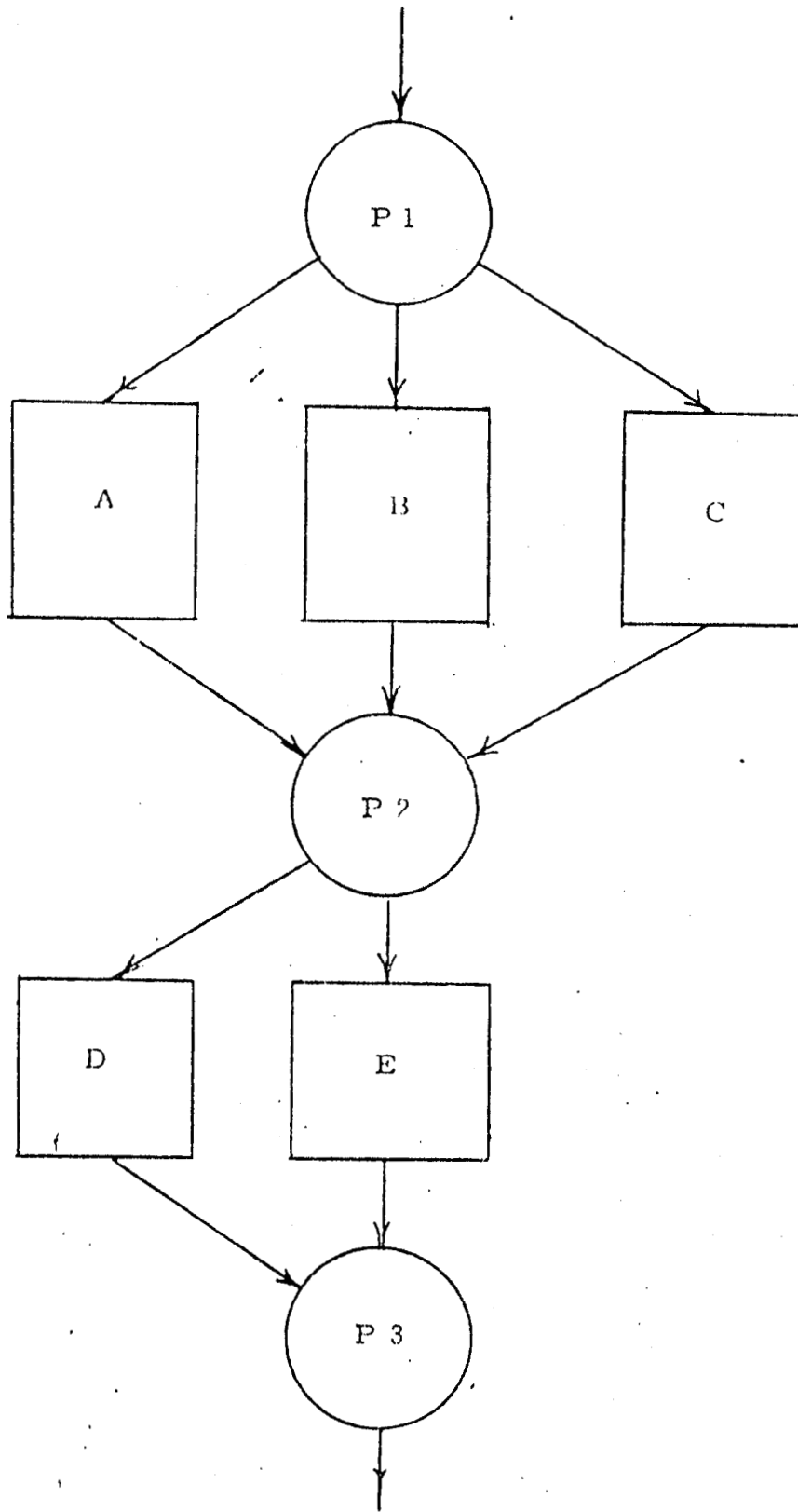
EXAMPLE 4: POINTER DRIVEN SOFTWARE STRUCTURE - POSEIDON COMPUTERADVANTAGES

STATE OF MACHINE DEFINED BY SET OF POINTERS  
NO OPERATING SYSTEM  
NO INTERRUPTS - GREATLY SIMPLIFIES VERIFICATION  
STRUCTURE EMPHASIZES MODULARITY  
EASILY SPECIFIED FUNCTIONS  
STRUCTURE LENDS ITSELF TO STATIC VERIFICATION

DISADVANTAGES

LESS FLEXIBLE THAN APOLLO  
REQUIRES STRUCTURED SEQUENCING  
BEST FOR DETERMINISTIC SYSTEMS (LOCAL PROCESSORS, ACE, OR  
UNMANNED MISSIONS)

WSU SPECIAL COLLECTIONS  
FOR RESEARCH USE ONLY



Pointer 1

Level 1  
Program;

Pointer 2

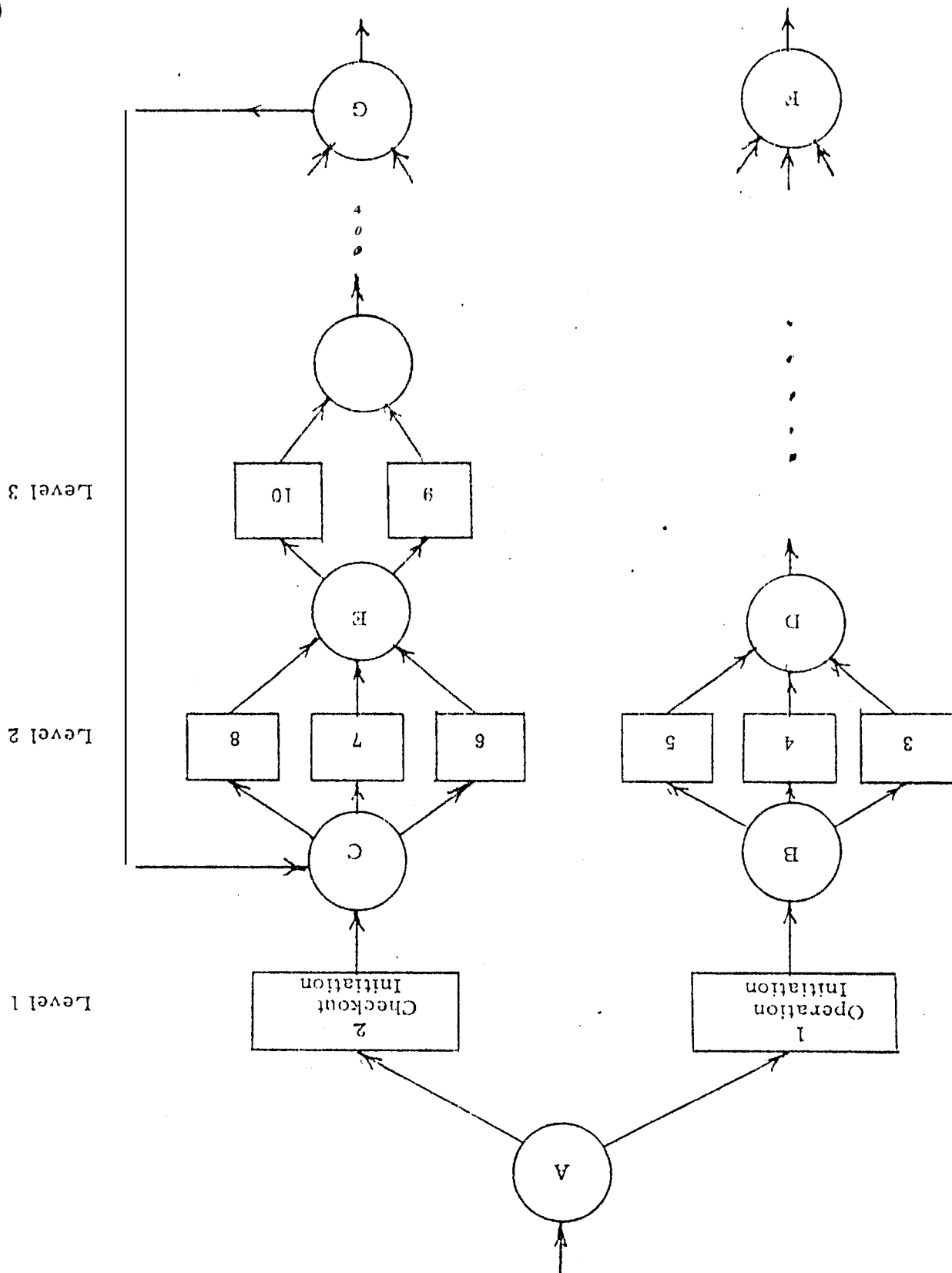
Level 2  
Program:

Pointer 3

Etc.

Figure 4 Pointer Driven Software Structure

Figure 5 Example of Software Structure





RECOMMENDED TOOLS AND TECHNIQUES

COMPILER AND METACOMPILER

AUTOMATED FLIGHT COMPUTER VERIFICATION FACILITY

INTERACTIVE PROGRAM DEVELOPMENT AND VERIFICATION TECHNIQUES

SELF-CONTAINED AUTOMATED CHECKOUT FACILITIES IMBEDDED IN  
FLIGHT SOFTWARE

COMPREHENSIVE STATIC VERIFICATION TECHNIQUES

SELF-DOCUMENTATION

MODELLING TOOL TO EVALUATE SOFTWARE STRUCTURES

WSU SPECIAL COLLECTIONS  
FOR RESEARCH USE ONLY

COMPILER

ESTABLISH CONTINUITY BETWEEN GUIDANCE ANALYSTS AND  
FLIGHT COMPUTER PROGRAMMERS

REDUCE COST OF CODE GENERATION

PERFORM STATIC VERIFICATION

ENFORCE PROGRAMMING CONVENTIONS

PROVIDE SELF-DOCUMENTATION OF CODE

REDUCE DEPENDANCY OF CODE ON MACHINE ARCHITECTURE

AUTOMATED FLIGHT COMPUTER VERIFICATION FACILITY

EST-COOPERATIVE FLIGHT COMPUTER LINKED TO GENERAL PURPOSE  
COMPUTER

COMMON FACILITY FOR PROCEDURAL SIMULATIONS AND DIGITAL  
SIMULATIONS

DETAILED DIAGNOSTIC CAPABILITY

STATE OF MACHINE CONTINUOUSLY RECORDED

RERUN CAPABILITY

INTERACTIVE PROGRAM DEVELOPMENT AND VERIFICATION

INEXPENSIVE TV DISPLAY FACILITY (\$5,700 PER TERMINAL)

CONTROLLED ACCESS TO COMMON DATA BASE

INTERACTIVE TECHNIQUES FOR:

DEVELOPMENT PLAN

GUIDANCE ALGORITHM DEVELOPMENT

FLIGHT PROGRAMMING

SIMULATION INPUT AND EDITED OUTPUT

MAINTAINANCE OF PROGRAMS

FNIS