NATIONAL CONFERENCE
AND WORKSHOP
ON

METHODOLOGIES AND TOOLS FOR REAL-TIME SYSTEMS

Washington, D.C.
March 10-11, 1986

Presented by

The National Institute for Software Quality and Productivity
P.O. Box 70555, Washington, D.C. 20088
(301) 983-3295

The National Institute is chartered to identify and promote important new software development technology. Its objective is to focus industry attention on methods and tools to improve software quality and productivity. The Institute's programs are guided by a National Council of prominent industry leaders. Consistent with the goal of maintaining the competitive position of the U.S. computer industry, the Institute maintains liaison with relevant industry consortia and government R&D programs.

The National Institute for Software Quality & Productivity
P.O. Box 70555 • Washington, D.C. 20088
(301) 983-3295

March 10, 1986

Dear Colleague:

Welcome to the second National Conference on Methodologies and Tools for Real-Time Systems. The program comprises seventeen presentations by some of the leading members of the software industry. The topics represent important new technology with a record of successful use in the development of large scale systems. The conference is followed by two days of workshops on some of the same topics.

An exhibit of tools to support real-time systems development is an important adjunct to the conference. New products in this area will be a principal feature of this and future conferences. These conference proceedings cannot, of course, reflect the exchange of ideas on important issues which occur at such a meeting among the attendees.

This conference is the second in this subject area given by The National Institute, which has been formed to focus industry's attention on software quality and productivity. The Institute's efforts are guided by a National Council of leading innovators in the software industry. The Institute's primary objectives are: (a) the identification and promotion of new software development technology; and (b) the formation of an industrial society to represent the software industry.

The growth in software reliability and productivity has been painfully slow, making this U.S. industry vulnerable to future foreign competition. A coordinated, national, active program to more rapidly promote technology for improving quality and productivity is essential. The National Institute will help bring new, practical, proven technology into general use more rapidly via conferences, workshops, and on-site training.

The next National Conference on Methodologies and Tools is planned for Fall '86. We look forward to seeing you there.

Sincerely,

Paul D. Johnson

Paul D. Johnson
Executive Director

CONFERENCE AGENDA

Raymond W. Wolverton, General Chairman

Monday (March 10, 1986)

08:00	Late Registration (Lobby)	
08:30	Keynote Address "Meeting the International Competitive Challenge"	James M. LeMunyon Senior Staff, U.S. Congress
SESSION I:	Advanced Methodologies	Ray **Wolverton**, Chairperson
09:00	"PAMELA: A Methodology for Ada*-based Real-Time Systems"	G. W. Cherry Thought Tools, Inc.
09:50	"An Integrated Graphical Environment for Software Development"	P. Pircher Interactive Develop. Environment
10:40	"The TAGS Methodology: Automation, Formalism, and Productivity"	M. Carrio Teledyne Brown Engineering Co.
11:30	-- BREAK --	
11:45	Luncheon Speaker -- "A System Environment Inventory"	Richard A. Margo Margo & Associates
13:00	"SREM at Age Eight: Distributed Systems Design"	M. Alford TRW
13:50	"SDE: The ITT System Development Environment"	R. Wolverton ITT
14:40	-- BREAK --	
14:50	"A Programming Environment for CSP"	N. Delisle Tektronix
15:40	"An Implementation of a Structured Methodology for Real-Time Systems in PSL/PSA"	D. Teichrow ISDOS
16:30	Methodology Support Tools Demonstrations --	
19:30	TAGS, PROMOD, Cadre, Index Technology, CASE 2000 IDE (Interactive Development Environment), EPOS, PAMELA, Tektronix, SPQR (Tool Demos continue next day).	

Tuesday (March 11, 1986)

SESSION II:	Tools, Testing, and Management	Paul Clements, Chairperson
08:30	"DSDM: A Comprehensive Software Development Methodology"	T. Clark Computer Sciences Corp.
09:10	"Object Oriented Development for Ada Systems"	D. Firesmith Magnavox
09:50	"Structured Methods for Large Avionics Systems"	D. Hatley Lear-Siegler
10:30	"From Prototype to Effecient Implementation"	E. Schonberg N.Y.U.
11:10	"Modular Design of Real-Time Systems"	T. Scott PROMOD, Inc.
11:50	-- BREAK --	
12:00	-- LUNCHEON --	
	Panel Discussion: "Criteria for Methodology Selection" R. Wolverton (Moderator): ITT P. Clements: NRL — R. Jensen: Hughes Aircraft M. Alford: TRW — J. Baker: Yourdon & Co. M. Carrio: Teledyne, Brown	
13:20	"Structured Real-Time Analysis & Design"	T. McCabe McCabe & Associates
14:00	"EPOS: An Automated, Method Independent Software Development System"	R. Torrick SPS: Software Products & Services
14:50	-- BREAK --	
15:00	"Structured Systems Development for Real-Time Systems"	J. Baker Yourdon & Co.
15:50	The Software Cost Reduction Methodology"	P. Clements Naval Research Laboratory
16:40	-- ADJOURN --	

*Ada is a registered trademark of the U.S. Department of Defense, Ada Joint Program office.

ONE-DAY WORKSHOPS

WEDNESDAY, MARCH 12

OOD: Object Oriented Development

Donald G. Firesmith
Magnavox

OOD is a new approach for design and implementation of real-time embedded Ada software. It specifically supports Information Hiding, Levels of Abstraction, and Abstract Data Types as well as modern software engineering principles.

EPOS: Engineering and Project Management Oriented Support System

R. Torick
SPS: Software Products and Services

An automated requirement to code generation system, EPOS is methodology independent. EPOS supports a variety of design methods, documentation, graphics, QA, CM, etc. and has been extensively applied in Europe.

THURSDAY, MARCH 13

PAMELA: Process Abstraction for Embedded Systems

George W. Cherry,
Thought** Tools, Inc.

PAMELA supports all of Ada's advanced features (generics, packages, tasks, exceptions, and both forms of separate compilation). It recognizes abstract data types as well as abstract processes. PAM specifically supports modularity, reusability, modifiability, maintainability, clarity, and reliability.

TAGS: Technology for Automated Generation of Systems

D. Brown, Ph.D.
Auburn University

TAGS is a computer-aided tool to support development of large embedded systems. It supports the design, implementation, and maintenance phases of the software life cycle, including automated generation of procedural code (e.g. Ada) and documentation.

Magnavox
ELECTRONIC SYSTEMS COMPANY

OBJECT-ORIENTED DEVELOPMENT

by

Donald G. Firesmith
Software Methodologist

Magnavox Electronic Systems Company
Advanced Software Systems Division
1313 Production Road
Fort Wayne, IN 46808
(219) 429-4327

MX 92-173-1
(MX92173-1 Text 55)
UNCLASSIFIED

Magnavox
ELECTRONIC SYSTEMS COMPANY

WHY IS OBJECT-ORIENTED DEVELOPMENT (OOD) IMPORTANT

OOD is one of the extremely few software development methods actually designed for modern:

1) Ada (*) applications and

2) Real-time, embedded software.

OOD is a significant improvement over more traditional functional decomposition and modeling methods in that OOD:

1) Better manages the size, complexity, and concurrancy of today's systems.

2) Better addresses important software engineering principles such as abstract data types, levels of abstraction, and information hiding.

3) Produces a better design that more closely matches reality and is less hardware dependent.

4) Produces more maintainable software by better localizing data and thus limiting the impact of requirements changes.

5) Specifically exploits the power of Ada.

(*) Ada is a registered trademark of the U.S. Government (AJPO).

MX-92-173-2
UNCLASSIFIED

DONALD FIRESMITH/MAGNAVOX 03-03-86

OOD IS:

* A software development method
* Object-oriented
* Ada-oriented
* Based upon modern software engineering
* Globally top-down
* Recursive
* Supports extensive parallel development
* Revolutionary in approach
* A 'grab and go' method
* Relatively easy to learn
* Being successfully used by several companies
* Still evolving

OOD IS A SOFTWARE DEVELOPMENT METHOD

OOD is a step-by-step systematic approach.

OOD permits the management of complexity.

OOD is supported by standards and procedures.

OOD covers all, or a major portion, of the software life-cycle.

OOD has an optimal domain of application -- the development of modern Ada applications (e.g., real-time, embedded software).

OOD requires training to be effective.

OOD IS OBJECT-ORIENTED

An OBJECT is an entity that:

1) Has a value (e.g., data) or state (e.g., Ada task).
2) Suffers and/or causes operations.

OOD produces Ada objects that correspond to objects in the real world.

OOD produces templates for Ada OBJECTS and operations that operate on these Ada OBJECTS.

OOD emphasizes the implementation of objects in terms of ABSTRACT DATA TYPES. OOD groups, in the same package:

1) A type (i.e., an object template) and
2) All operations that operate upon such objects.

OOD produces a substantially different software architecture than traditional functional decomposition methods such as Structured Design which generate units, each of which implements some FUNCTION of the requirements specification.

OOD IS ADA-ORIENTED

Ada is an object-oriented high-level language.

The design produced by OOD is top-down in terms of Ada:

1) Nesting,
2) Visibility (i.e., the Ada "with" statement), and
3) Abstract Data Types.

Packages, which are the main building blocks of properly designed Ada software, are also the main building blocks produced by OOD.

OOD separately develops Ada specifications and bodies.

OOD's unit testing naturally accounts for Ada compilation order constraints.

OOD IS ADA-ORIENTED (CONT)

OOD Diagrams clearly identify the various Ada programming units.

Ada PDL is an integral part of OOD's design and coding steps.

The objects produced by OOD are implemented in Ada as:

1) Constants
2) Abstract data types
3) Tasks

The operations produced by OOD are implemented in Ada as:

1) Subprograms
2) Task entries

OOD IS BASED UPON MODERN SOFTWARE ENGINEERING

OOD specifically addresses each of the following software engineering principles or concepts:

a) Abstract Data Types.

b) Abstraction Levels.

c) Cohesion.

d) Concurrency.

e) Coupling.

f) Information Hiding.

g) Localization.

h) Maintainability.

i) Modularity.

j) Organizational Independence.

k) Readability.

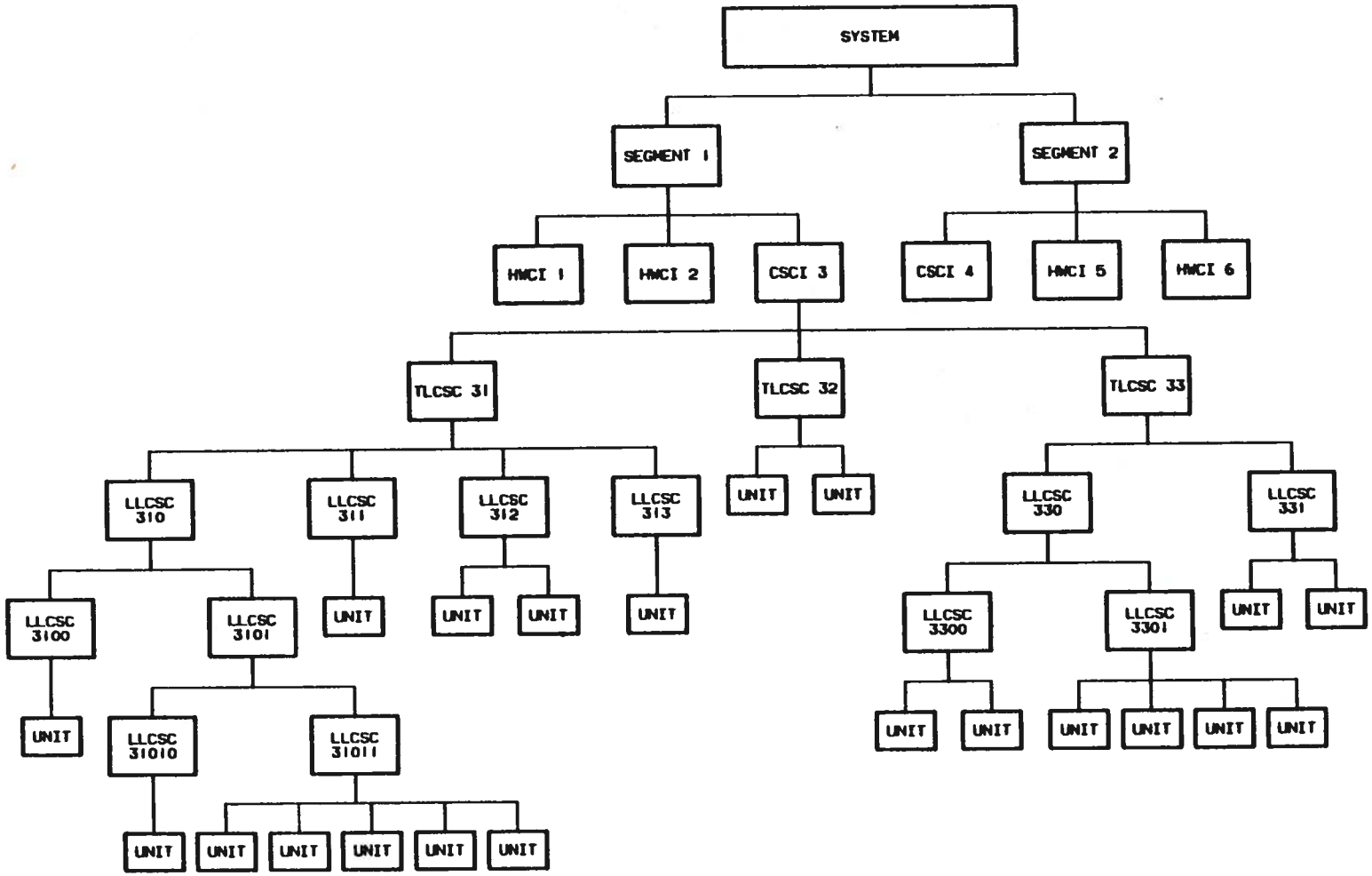
l) Reusability.

m) Structure.

n) Testability

o) Verifiability.

TRADITIONAL (DOD-STD-2167) SOFTWARE HIERARCHY



THE CONCEPT OF BOOCH AND SUBBOOCH

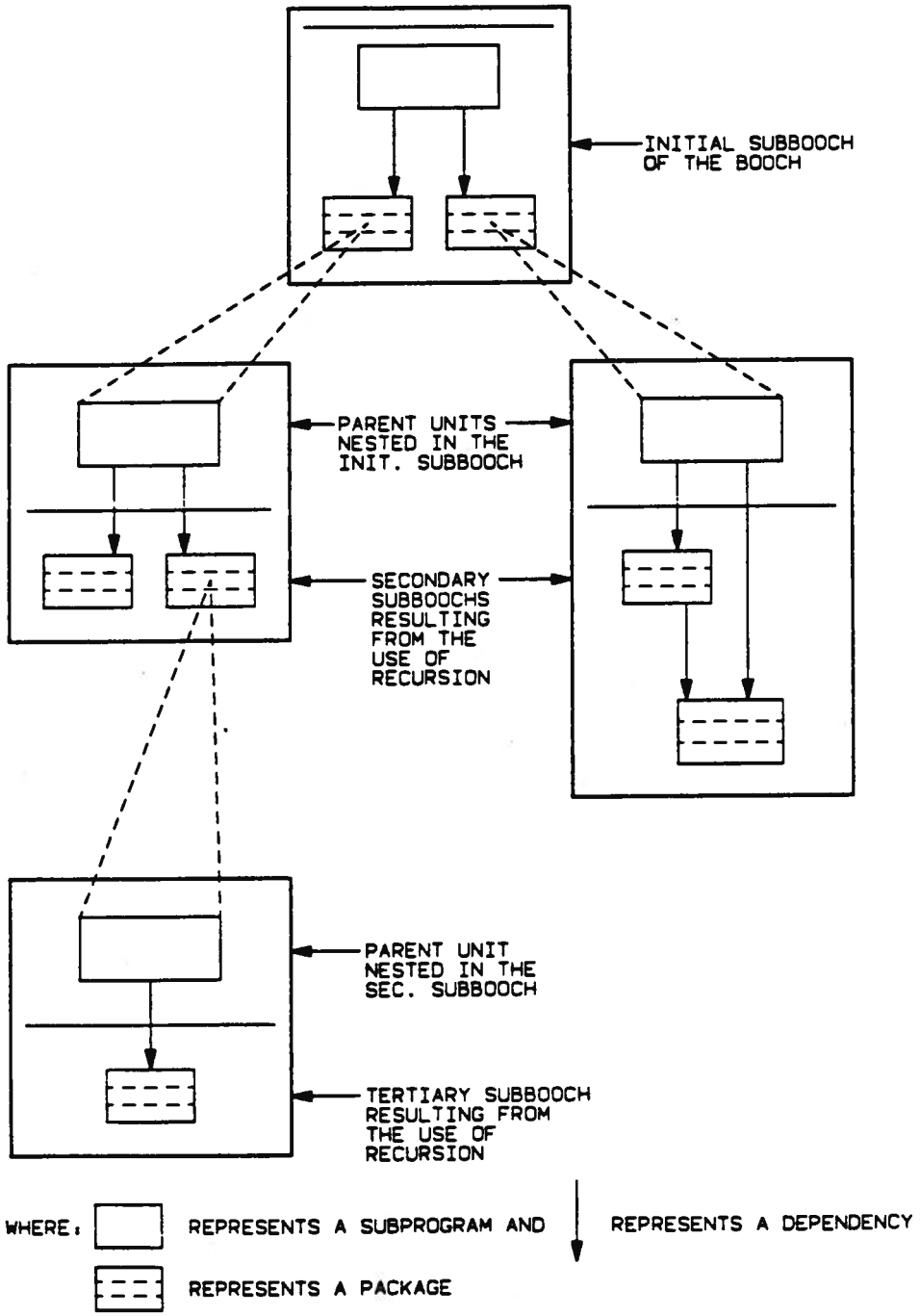
Two concepts, unique to OOD, are the Booch and Subbooch.

A BOOCH is the collection of all software resulting from the recursive application of OOD to a specific set of coherent software requirements -- requirements that specify a single well-defined problem.

A SUBBOOCH is a subset of a booch that is identified and developed during a single recursion of OOD.

Note that these two concepts have no obvious natural relationship to the hierarchical decomposition entities of CSCI, TLCSC, and LLCSC.

SAMPLE BOOCH STRUCTURE



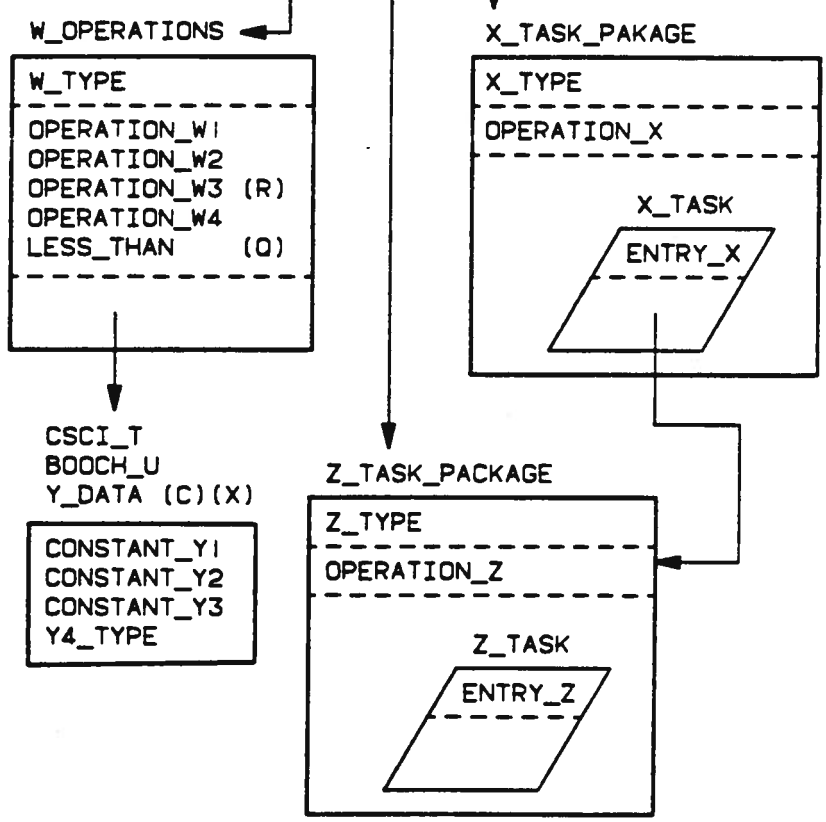
SAMPLE SUBBOOCH STRUCTURE

PROJECT X	MAGNAVOX ELECTRONIC SYSTEMS COMPANY OOD DIAGRAM FORM (SEF13)	RELEASE 1
SUBBOOCH: CSCI_X.BOOCH_Y.EXAMPLE		
PARENT SUBBOOCH: CSCI_X.BOOCH_Y.EXAMPLES_PARENT		

PARENT UNIT:

V_OPERATIONS.OPERATION_V3

SUBBOOCH UNITS:



(C) = common or reusable	(D) = test driver	(O) = overloaded
(R) = requires recursion	(S) = stub	(X) = Booch external
Certification:		Date:

SEF13 Version 02 03/03/86

DONALD FIRESMITH/MAGNAVOX 03-03-86

MX-92-173-13
UNCLASSIFIED

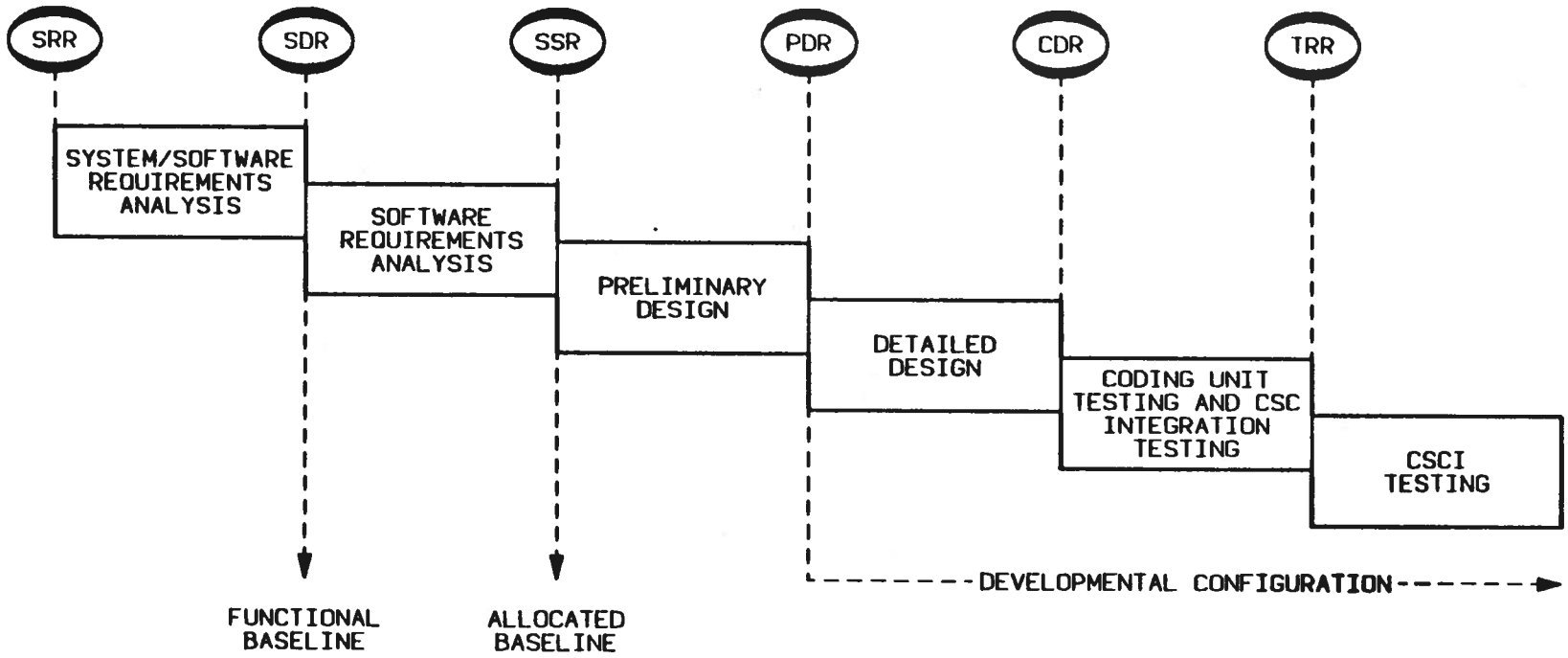
TRADITIONAL METHODS ARE LOCALLY TOP-DOWN

Traditional software development methods are restricted to the classic "waterfall" life-cycle in which:

1) The software requirements are analyzed first.
2) The preliminary design is developed second.
3) The detailed design follows.
4) And so on.

Because the software is developed in a top-down manner only within the boundries of each life-cycle phase, these methods are at best only locally top-down.

TRADITIONAL (DOD-STD-2167) SOFTWARE LIFE-CYCLE



DONALD FIRESMITH/MAGNAVOX 03-03-86

MX-92-173-15A
UNCLASSIFIED

OOD IS GLOBALLY TOP-DOWN

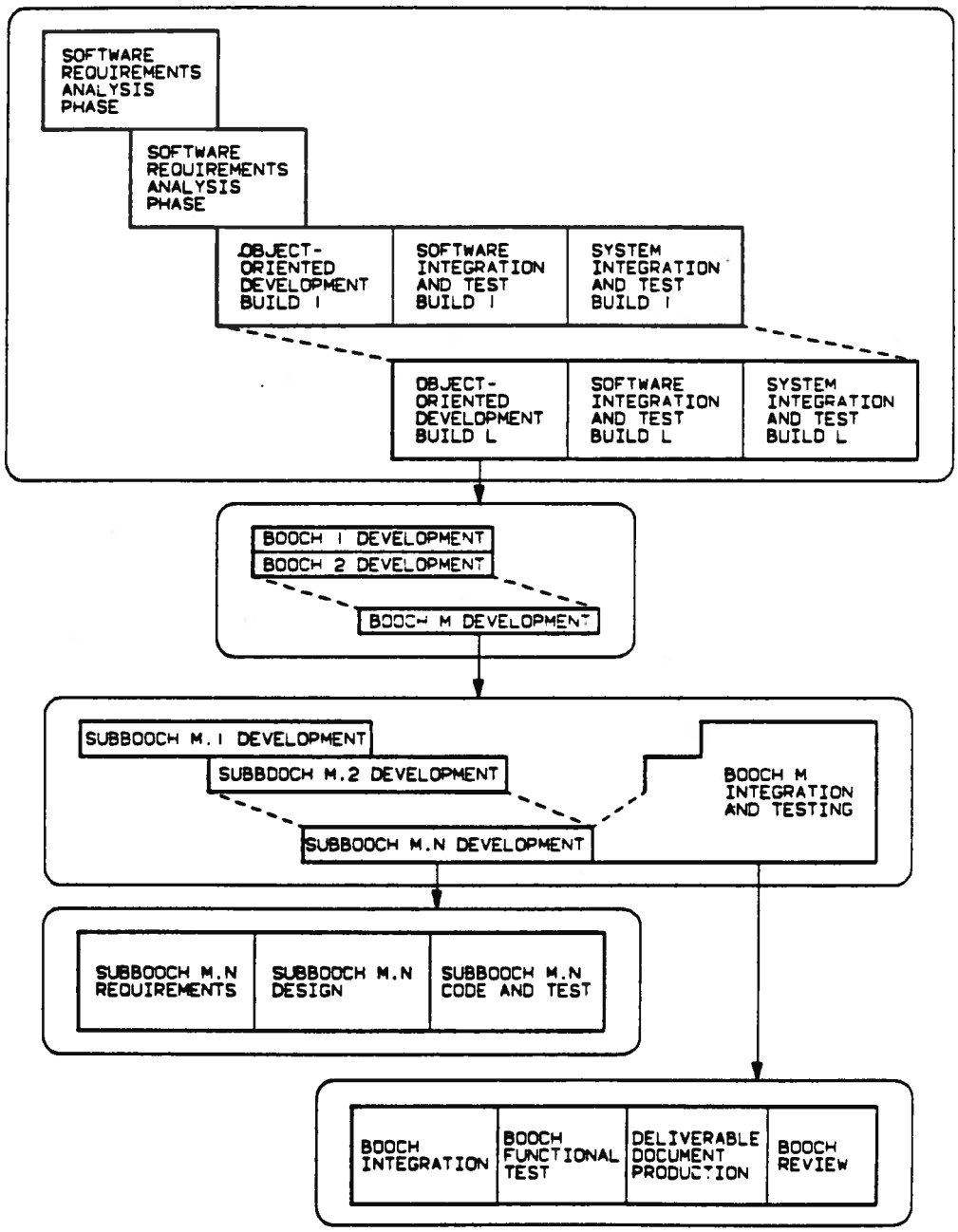
Beginning with the highest abstraction level and progressing steadily down the tree of unit dependencies implemented by the Ada "with" statement, the software is designed, coded, and tested in small manageable subbooches.

This allows very significant parallel development based upon the "Design a little, code a little, test a little" concept.

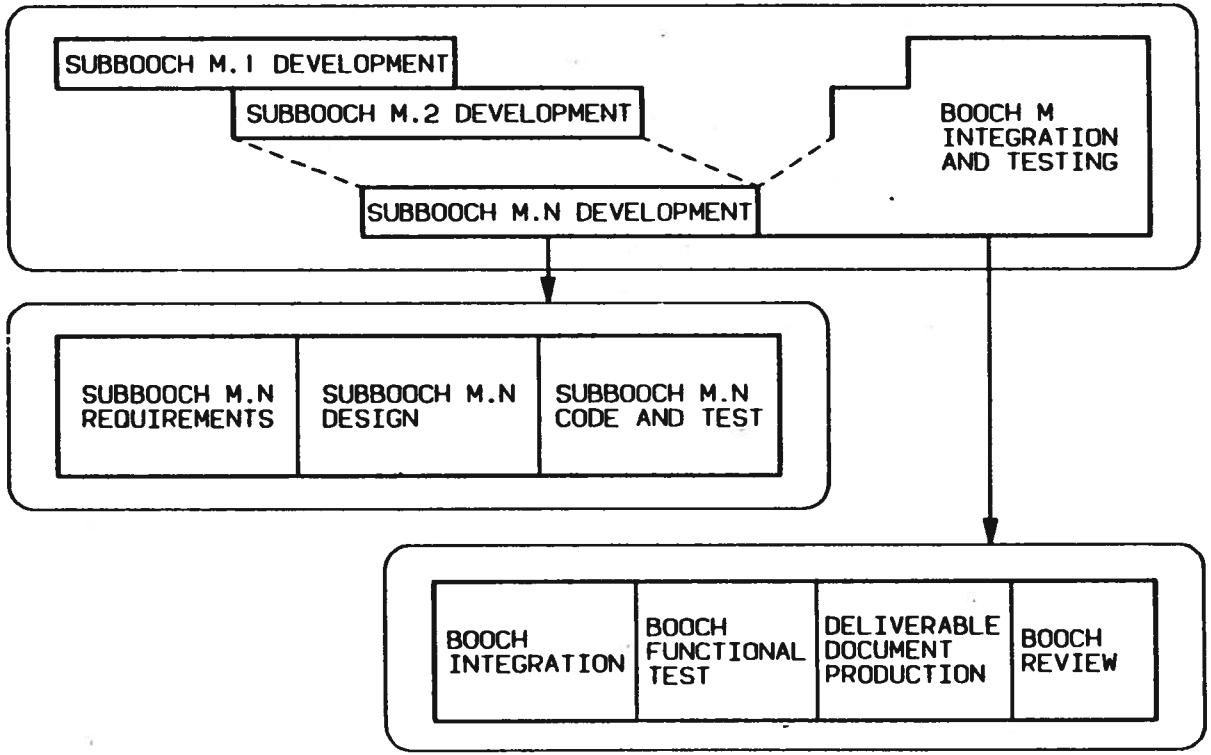
Thus, the booch grows top-down, subbooch by subbooch, via the recursive application of OOD until the entire software tree is completed.

Locally, however, OOD employs the appropriate technique (top-down or bottom-up) depending upon the specific requirements of each individual development activity.

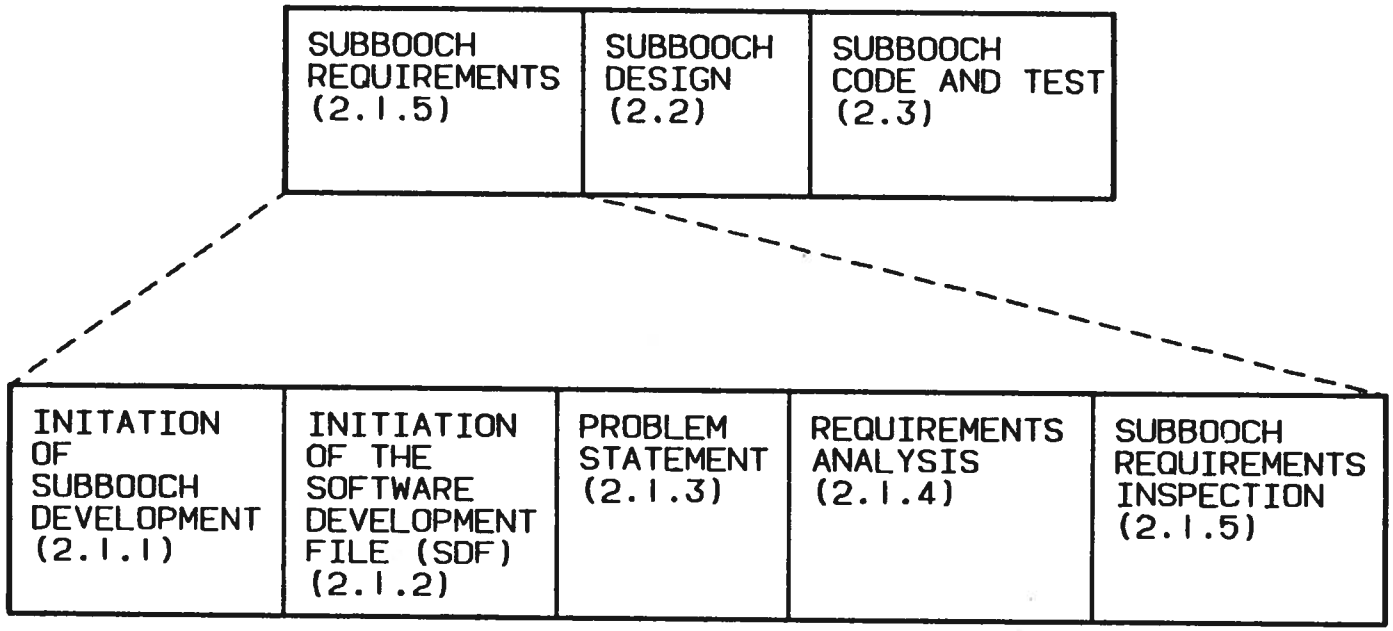
OOD SOFTWARE LIFE-CYCLE



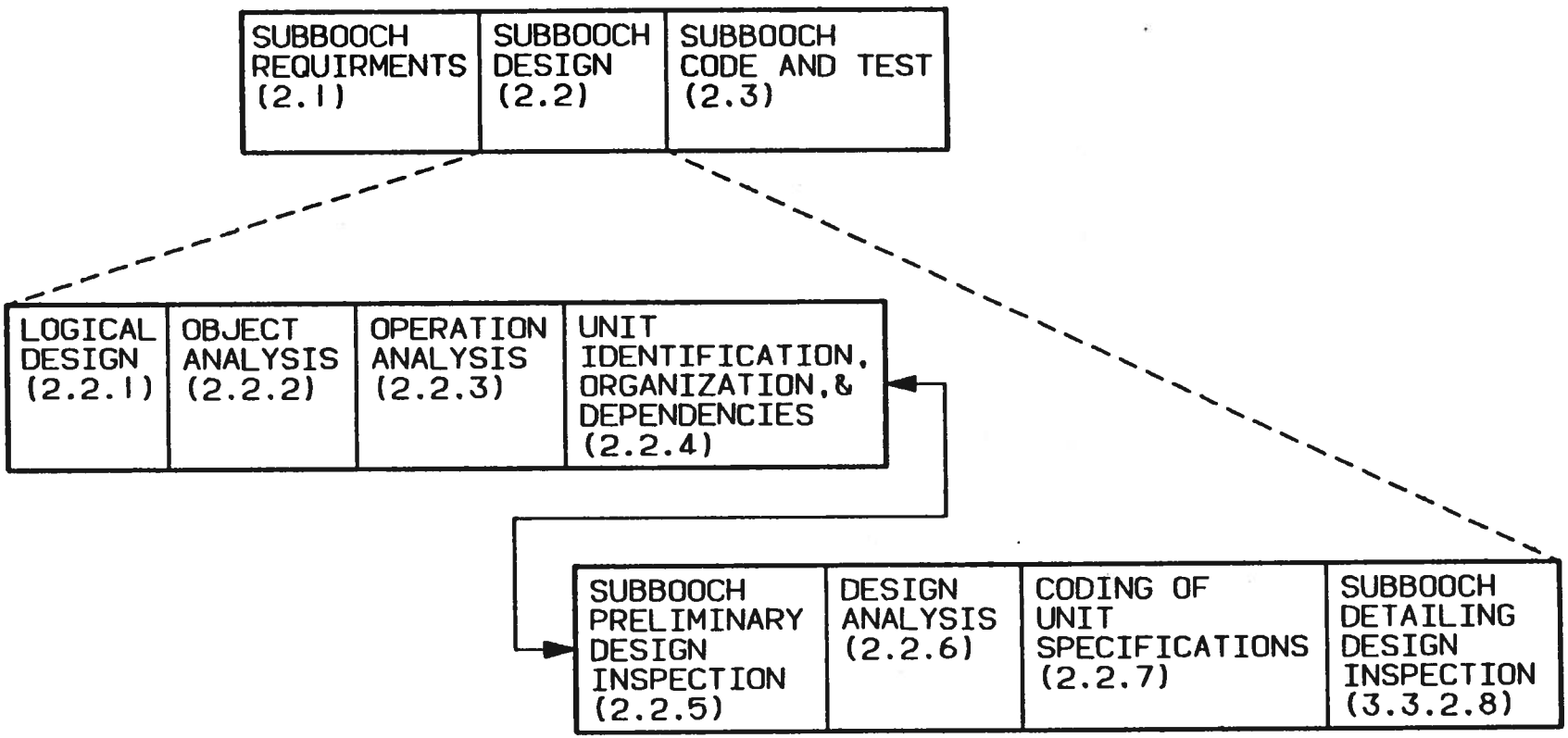
BOOCH DEVELOPMENT



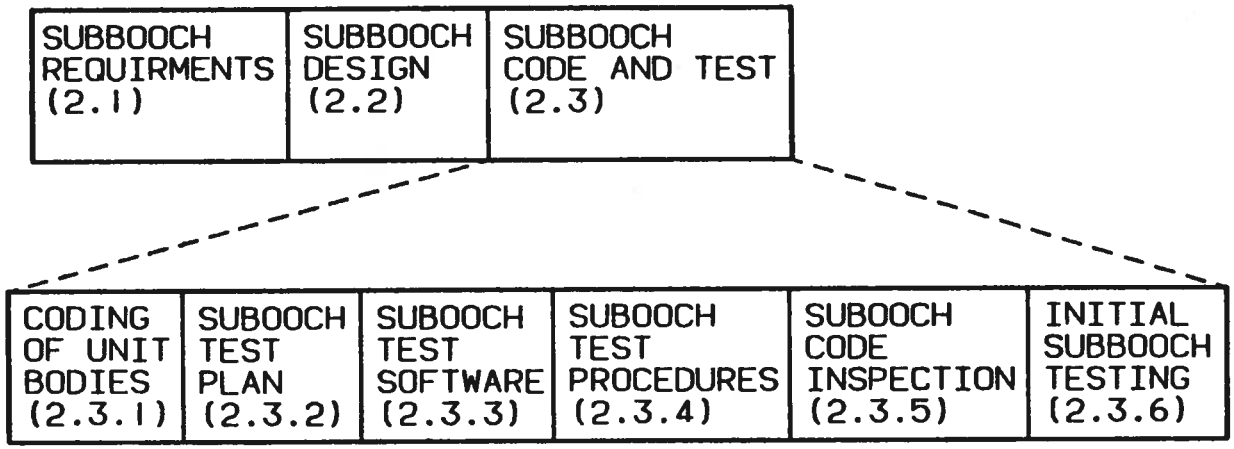
SUBBOOCH REQUIREMENTS



SUBBOOCH DESIGN



SUBBOOCH CODE AND TEST



RESPONSIBILITY MATRIX

Object-Oriented Development Process		MGMT	Software Dev. Team			MC	SQE
Step	Title		D	P	T		
1	INITIATION OF BOOCH DEVELOPMENT	1					4
2	SUBBOOCH DEVELOPMENT						
2.1	SUBBOOCH REQUIREMENTS SUBPHASE						
2.1.1	Initiation of Subbooch Development	1					4
2.1.2	Initiation of the SDF	3	1				4
2.1.3	Problem Statement	3	1	2	2		4
2.1.4	Requirements Analysis	3	1	2	2		4
2.1.5	Subbooch Requirements Inspection	1	2	2	2		4
2.2	SUBBOOCH DESIGN SUBPHASE						
2.2.1	Logical Design	3	1	2	2		4
2.2.2	Object Analysis	3	1	2	2		4
2.2.3	Operation Analysis	3	1	2	2		4
2.2.4	Unit Id., Org., and Dependencies	3	1	2	2		4
2.2.5	Subbooch Preliminary Design Inspection	3	2	1	1		4
2.2.6	Design Analysis	3	1	2	2		4
2.2.7	Coding of Unit Specifications	3	1	2	2		4
2.2.8	Subbooch Detailed Design Inspection	3	2	1	2	1	4
2.3	SUBBOOCH CODE AND TEST SUBPHASE						
2.3.1	Coding of Unit Bodies	3	2	1	2		4
2.3.2	Subbooch Test Plan	3	2	2	1		4
2.3.3	Subbooch Test Software	3	2	2	1		4
2.3.4	Subbooch Test Procedures	3	2	2	1		4
2.3.5	Subbooch Code Inspection	3	1	2	2	1	4
2.3.6	Initial Subbooch Testing	3	2	2	1		4
3	BOOCH INTEGRATION AND TESTING						
3.1	BOOCH INTEGRATION	3			1		4
3.2	BOOCH FUNCTIONAL TESTING	3			1		4
3.3	BOOCH DELIVERABLE DOCUMENTATION	2	1	1	1		4
3.4	BOOCH REVIEW	1	2	2	2	1	1

MGMT = Management
D = Designer(s)
P = Programmer(s)
T = Tester(s)
MC = Metrics Collector(s)
SQE = Software Quality Evaluation

1 = Primary or major responsibility
2 = Secondary responsibility
3 = Managerial responsibility
4 = Independent audit responsibility

DONALD FIRESMITH/MAGNAVOX 03-03-86

PRACTICAL EXPERIENCE

PROBLEMS

* Requirements overspecified with design information
* Replacing previous mindset often very difficult
* Ada-oriented test training lacking
* Recursion difficult to master
* Method training must continue beyond classroom
* Further method development needed

BENEFITS

* Improved designs
* Parallel development enhanced
* Code quickly written
* Design/Code easily modified