Using MIL-STD-498 and ISO/IEC 12207 for OOD and RAD

Lewis Gray, Ph.D.

Software Technology Conference
April 24, 1996

API (Ada PROS, Inc.)
12224 Grassy Hill Court
Fairfax, Virginia 22033-2819 USA
703.591.5247 FAX: 703.591.5005
adapros@aol.com

Abelia Corporation offers Lewis Gray's authoritative, on-site courses
on the standards that are mentioned in this presentation.
For detailed information about our courses, contact lewis@abelia.com.

This Presentation is Elaborated in the STC '96 Proceedings

- TITLE: "Using MIL-STD-498 and ISO/IEC 12207 with OOD and RAD" --39 pages, 34 figures
- OUTLINE:

Object-Oriented Development (OOD)
(Booch's model, OOA, O-O design)
Rapid Application Development (RAD)
(Martin's model, require's, design, coding)
MIL-STD-498 "Software Development and Documentation"
(What it does and why, "waterfall" bias, reviews, non-hier design, require's, CASE)
ISO/IEC 12207 "Software Life Cycle Processes"
(What it does, "waterfall" bias, reviews, non-hier design, CASE, U.S. adoption)

This Presentation is About OOD

Booch Macro Process -- Booch Micro Process - MIL-STD-498 - ISO/IEC 12207

My role

OOD meta-model
-all OOD methods-

Your role

- Your past experience with OOD
- Your software development goals

- Your OOD method
- Your tailored project standard

Major Topics

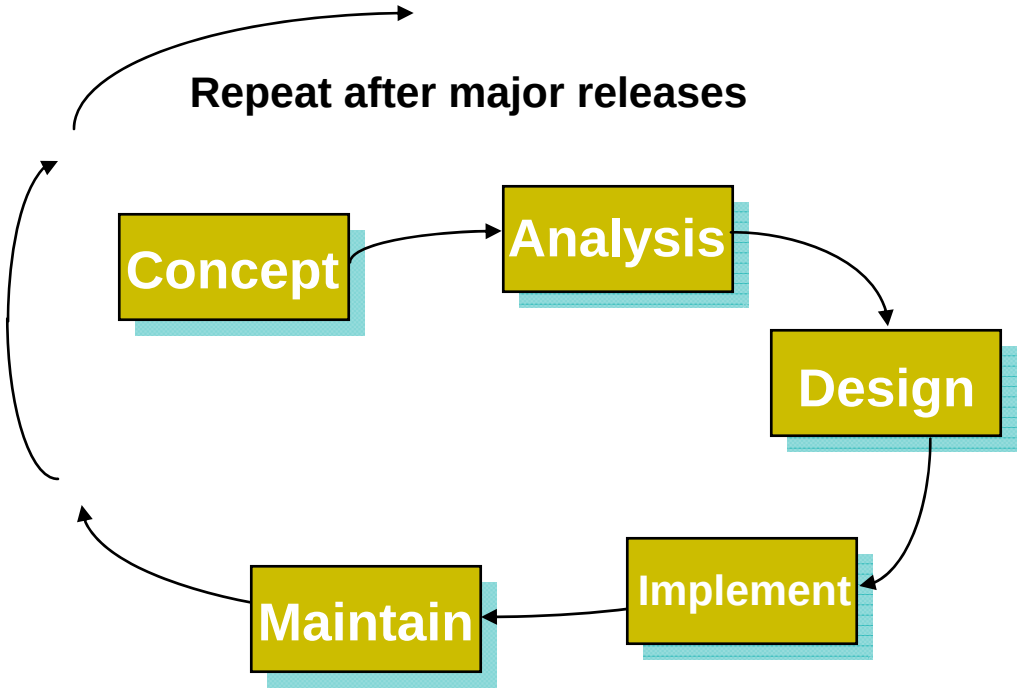
Booch O-O life cycle

Booch O-O practices

Using MIL-STD-498 with O-O

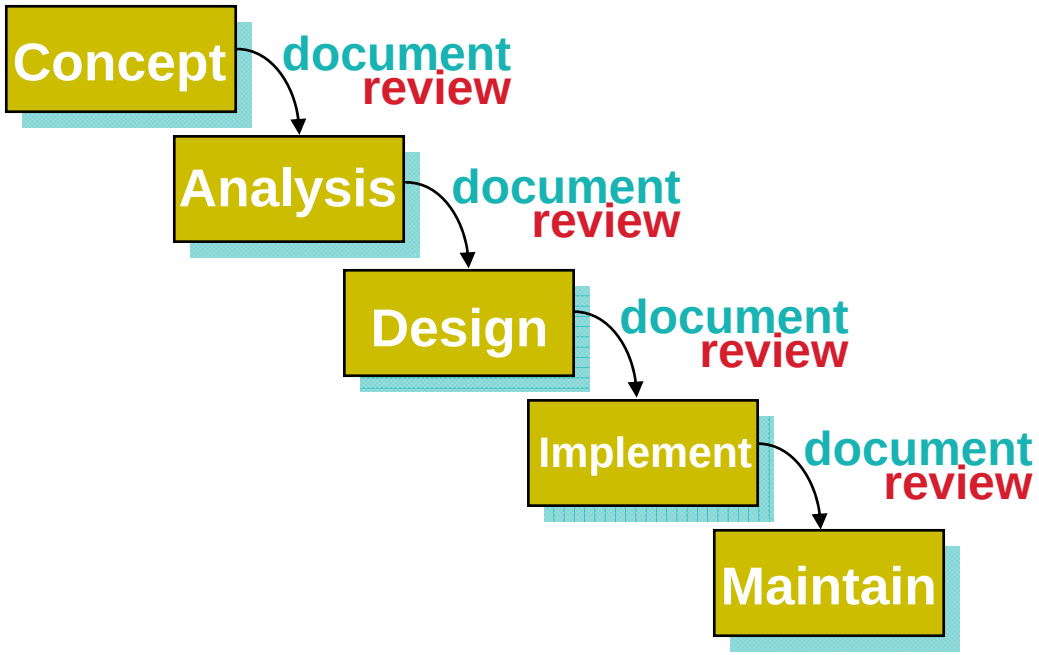
Using ISO/IEC 12207 with O-O

Booch Macro Process
For Each Software Release



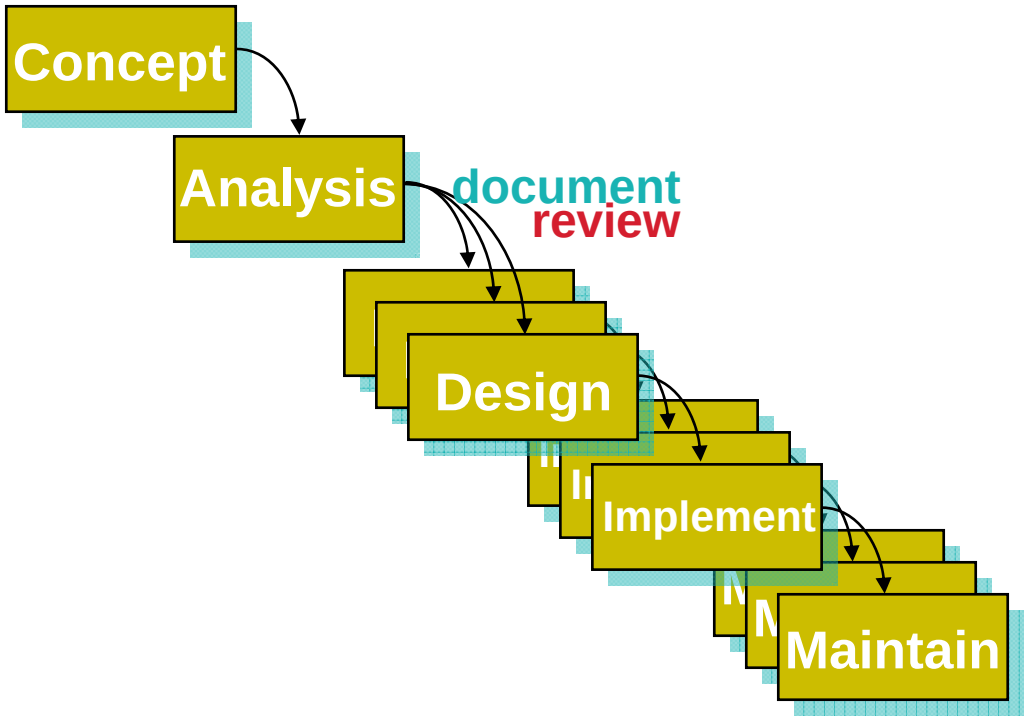
- Technical managers plan it, not developers
- Includes planning, risk management, tools, engineering, reviews, documentation, process and product evals., process improvement...
- Incremental: successive enhancements

“Waterfall” Model



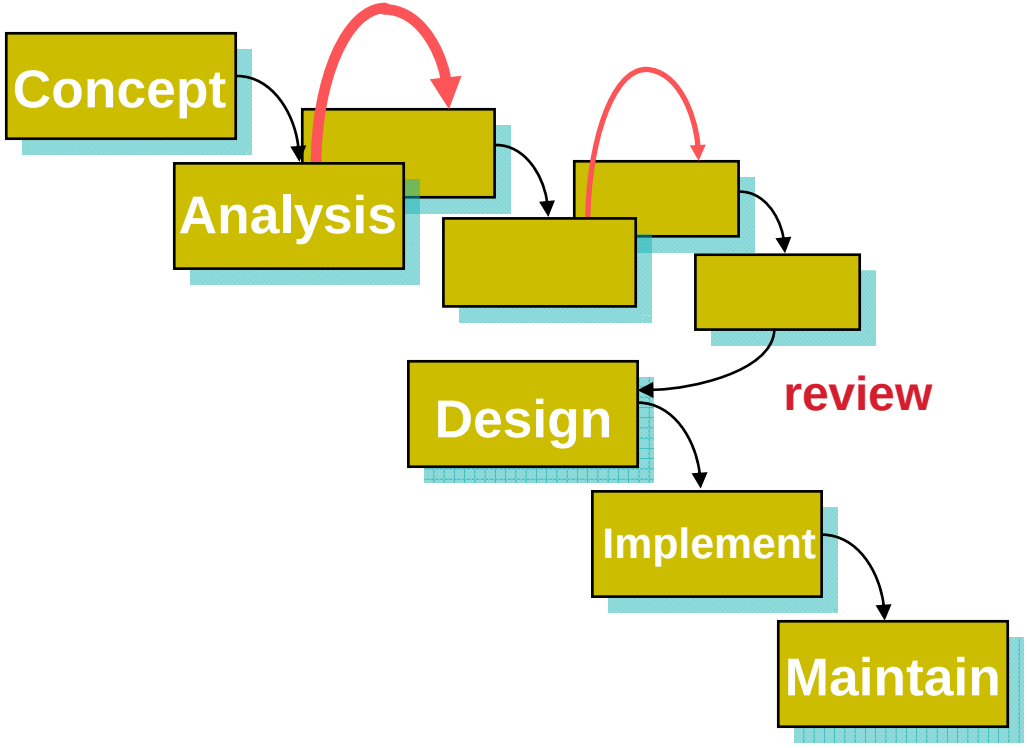
- MIL-STD-498 “Grand Design”
- One release
- All requirements are defined first
- Design is carried out before coding, usually in two stages
- Exit criteria from each activity are usually successful review of traditional document

Incremental Model



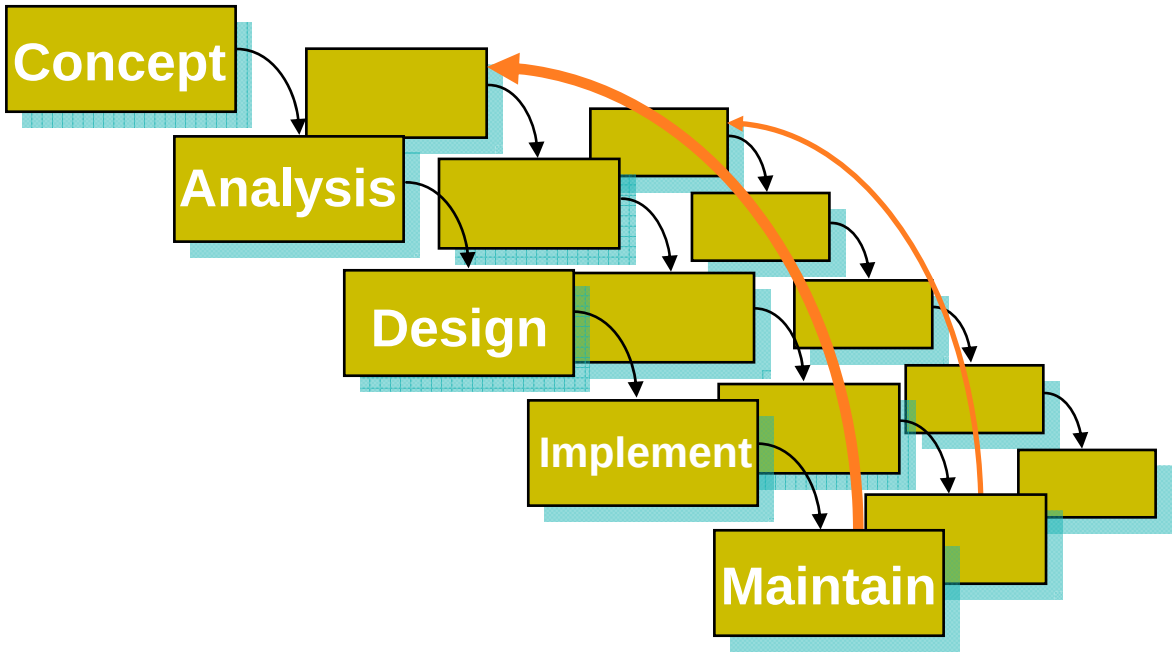
- MIL-STD-498 “Incremental”
- Many releases
- All requirements are defined first
- Within a release, design is carried out before coding
- Exit criteria from requirements activities are usually successful review of document

Spiral Model: Each Release



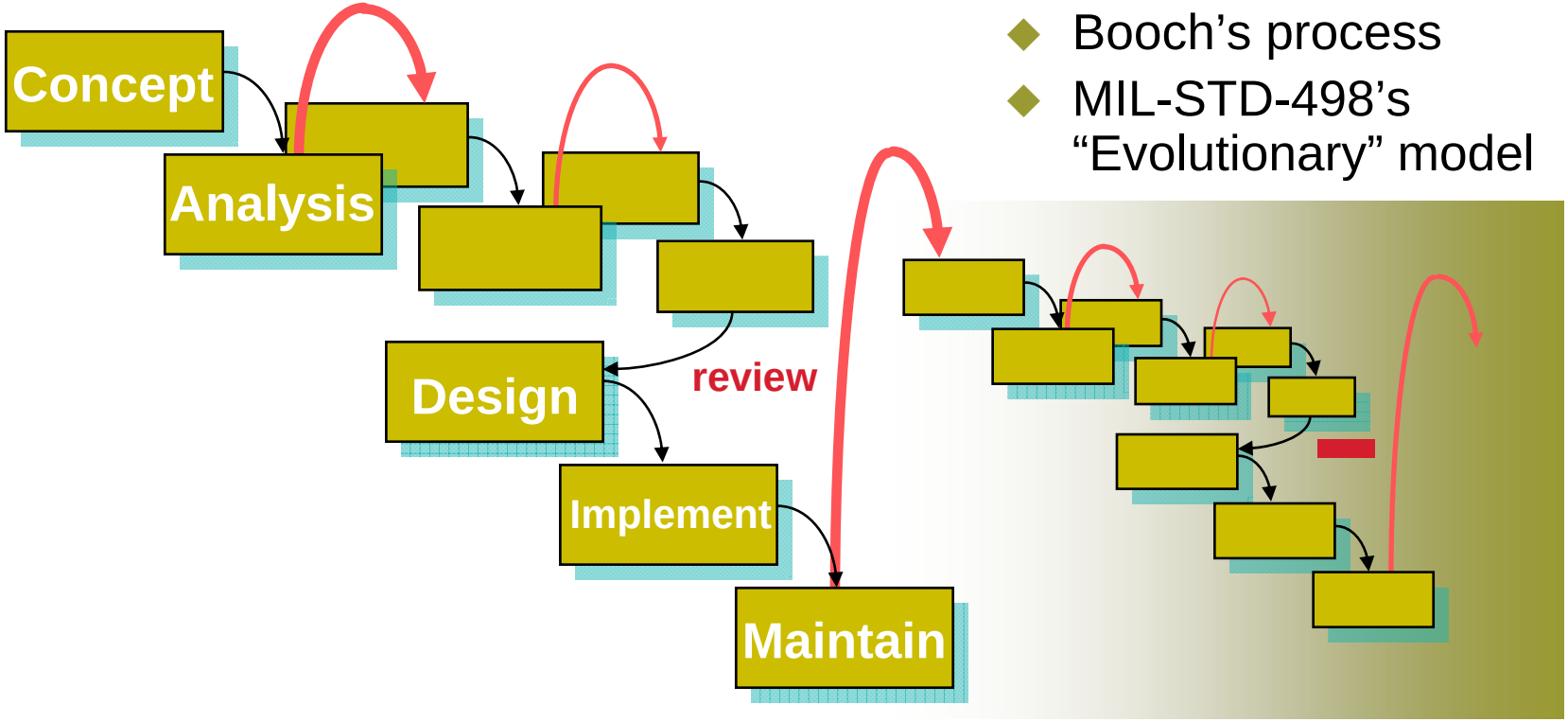
- One or many releases
- Risk reduction emphasis
- Requirements are defined with prototypes
- Exit criterion from requirements activities is usually successful review
- Design, code, test, release, maintain can be done like “waterfall” model

Evolutionary Model A (OOD)



- Booch's process
- MIL-STD-498's "Evolutionary" model
- Many releases
- Each release may include a requirements - design - code - test - release - maintain sequence
- Each release contributes to defining requirements for later releases

Evolutionary Model B (OOD)



- Booch's process
- MIL-STD-498's "Evolutionary" model

Major Topics

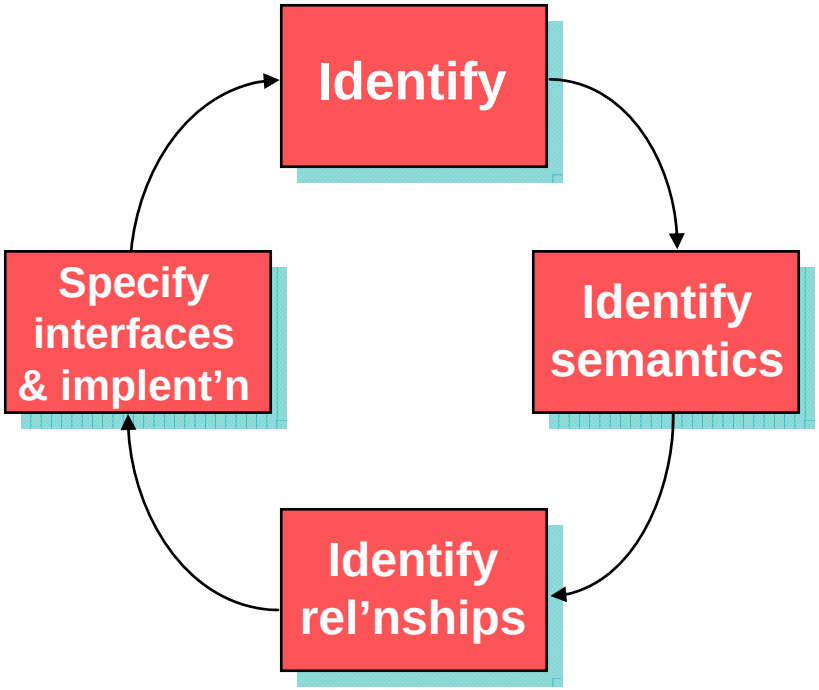
Booch O-O practices

Using MIL-STD-498 with O-O

Booch O-O life cycle

Using ISO/IEC 12207 with O-O

Booch Micro Process for Classes and Objects



- Developers plan it
- Represents daily activities of individuals or small teams
- Occurs in Analysis (OOA), Design (OOD), Implementation (OOP)
- Outputs are class diagrams, object diagrams, interaction diagrams, state transition diagrams, module diagrams… specifications, software

Coad and Yourdon's OOA ('90)

	Booch's micro process
Identify Objects	Identify classes and objects
Identify Structures	Identify class and object relationships
Define Structures	Identify class and object relationships
Define Attributes (and instance connections)	Identify class and object relationships
Define Services (and message connections)	• Identify class and object semantics • Specify class and object interfaces and implementation

Booch's Object-Oriented Design ('94)

	Booch's micro process
Refine project data dictionary as repository of system abstractions	Identify classes and objects
Develop specifications for each abstraction, write interface for each class, early object diagrams and interaction diagrams, look for opportunities for reuse	Identify class and object semantics
Produce class diagrams, object diagrams, and module diagrams, organize models into subsystems, map classes and objects to modules	Identify class and object relationships
Finalize class specifications, module diagrams, produce executable model of system	Specify class and object interfaces and implementation

Major Topics

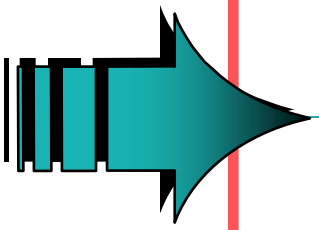
Booch O-O practices

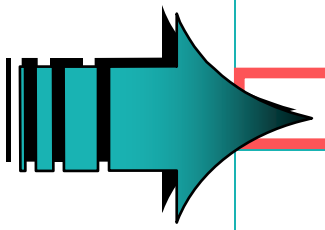
Using MIL-STD-498 with O-O

Booch O-O life cycle

Using ISO/IEC 12207 with O-O

Seven Issues with DOD-STD-2167A Related to OOD: In the Proceedings





1	Perceived preference for "waterfall" development model
2	Compatibility with incremental / evolutionary development models
3	Dependence on formal reviews and audits
4	Compatibility with Ada / O-O methods
5	Distinction between requirements and design
6	Emphasis on preparing documents
7	Use of CASE tools

MIL-STD-498
Software Development and Documentation
5 December 94

MIL-STD-498 solved the problems with DOD-STD-2167A without creating any new ones.

DOD-STD-2167A General Requirements

4.6
Transitioning to Support

4.1
Software Development Management

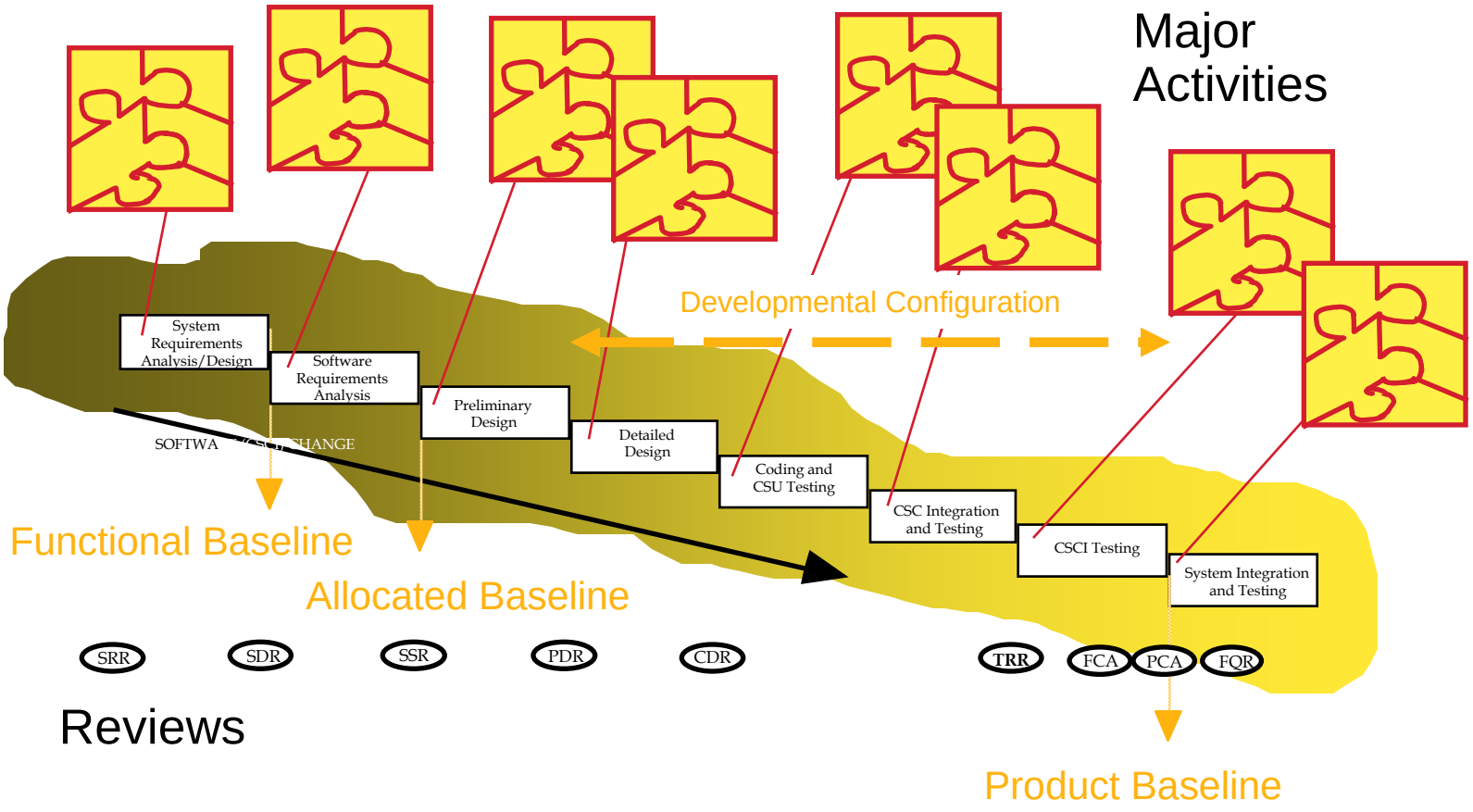
4.5
Software Configuration Management

4.2
Software Engineering

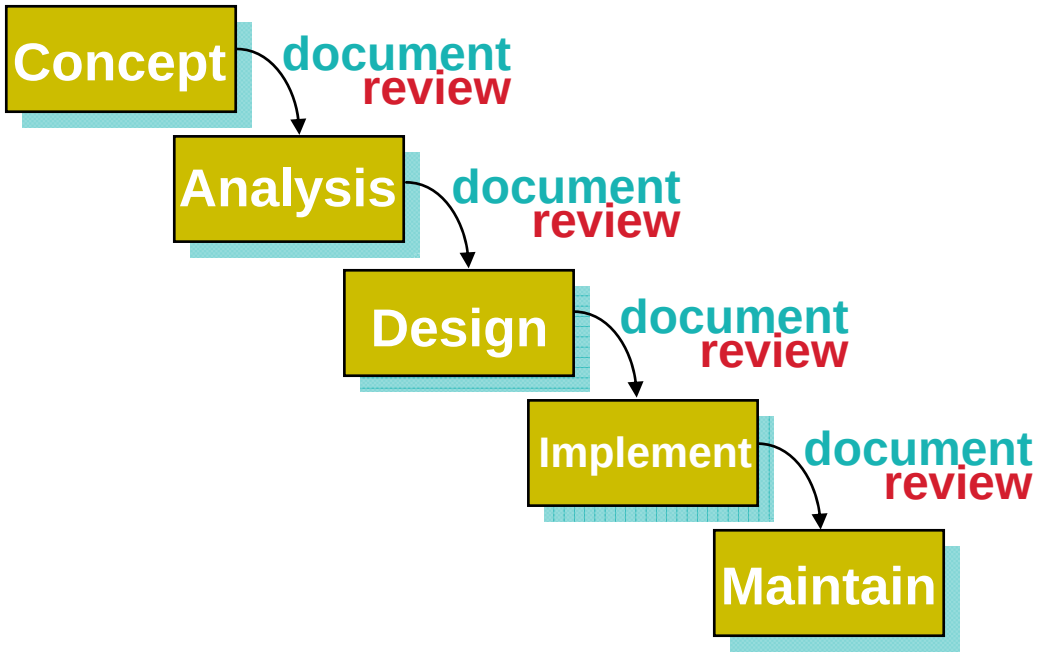
4.4
Software Product Evaluations

4.3
Formal Qualification Testing

DOD-STD-2167A



DOD-STD-2167A's "Waterfall" Bias



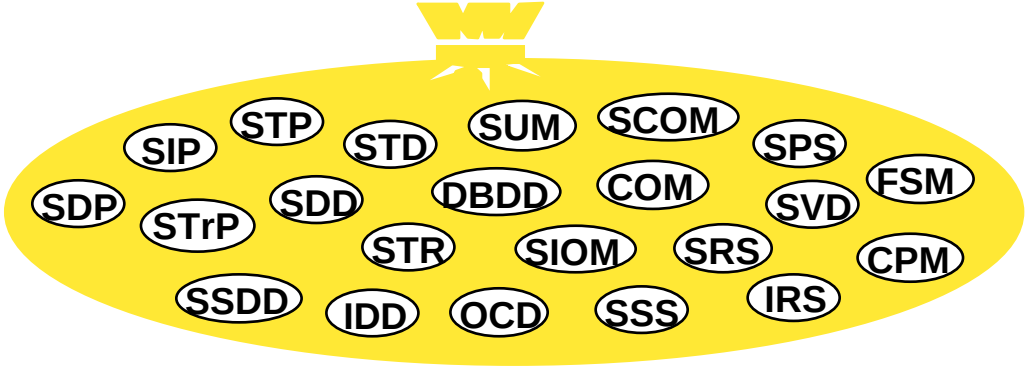
- One release
- All requirements are defined first
- Design is carried out before coding, usually in two stages
- Exit criteria from each activity are usually successful review of traditional document

Elements of MIL-STD-498 Development

MAJOR ACTIVITIES:
No sequence for the 25 activities, no linking

REVIEWS: No schedule for the 2 kinds of joint reviews

DELIVERABLE DATA:
Alternatives to 22 traditional documents are recommended

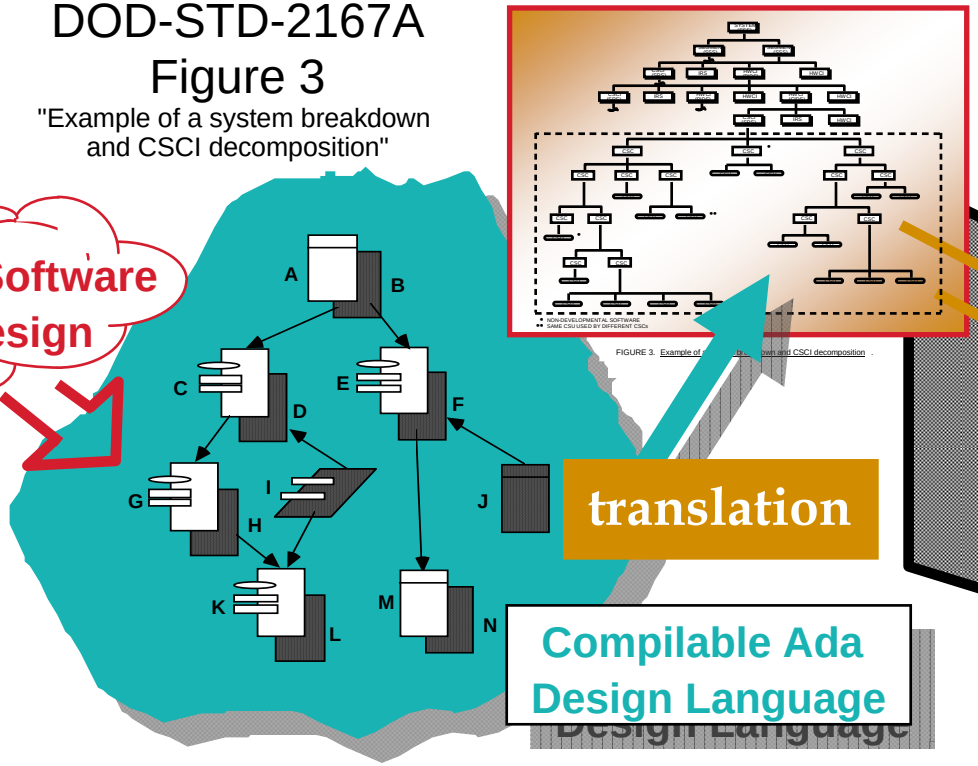


O-O Software Design

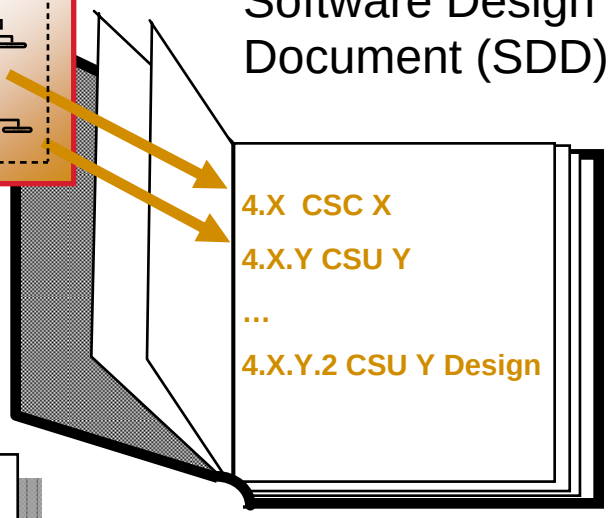
- DOD-STD-2167A's Translation Problem
- DOD-STD-2167A's Linking Problem
- MIL-STD-498 Compatibility with Non-Hierarchical Designs
- MIL-STD-498's Software Unit
- New Concepts in MIL-STD-498

DOD-STD-2167A's Translation Problem

DOD-STD-2167A
Figure 3
"Example of a system breakdown and CSCI decomposition"



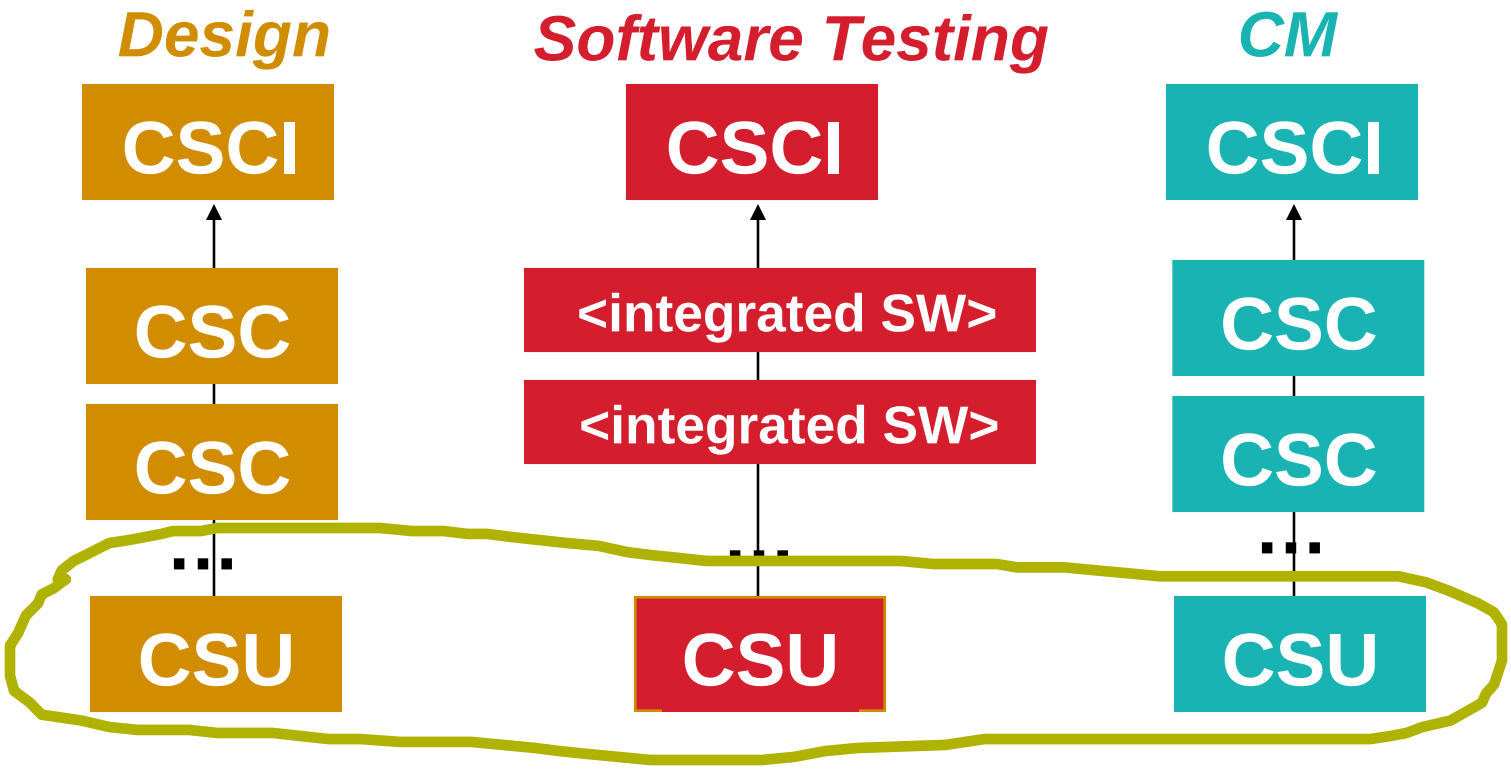
Software Design Document (SDD)



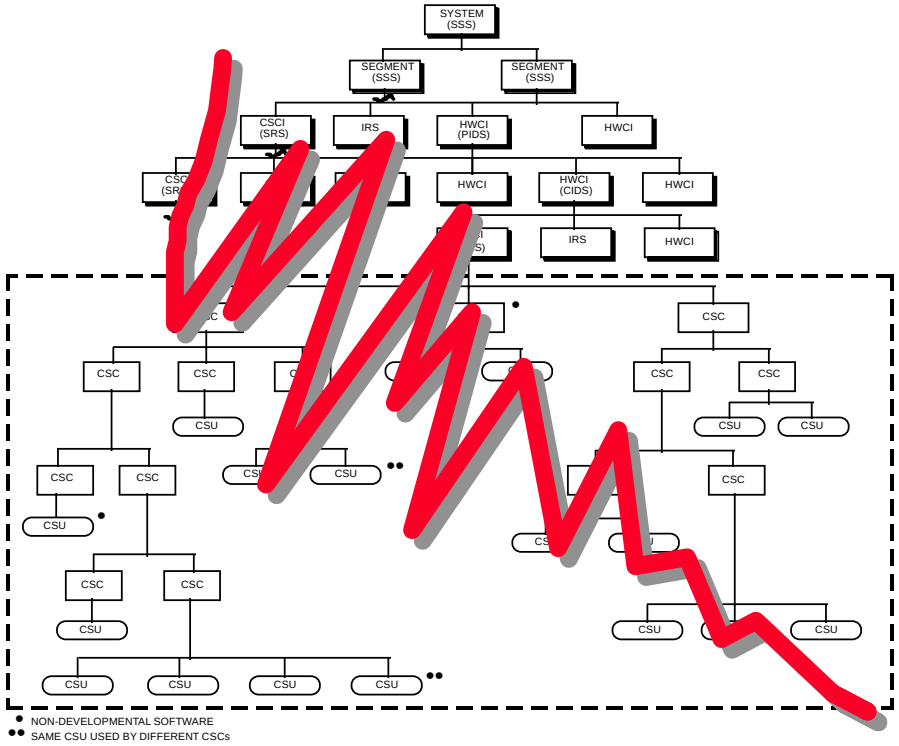
the technical product of the software designers

used by the Government to evaluate the technical product of the software designers

DOD-STD-2167A's Linking Problem



MIL-STD-498 Compatibility with Non-Hierarchical Designs



- Encourages “documenting” engineering decisions with actual engineering records, e.g., class, object, and module diagrams, class specifications, state transition diagrams …
- Avoids forcing hierarchical software designs

FIGURE 3. Example of a system breakdown and CSCI decomposition.

MIL-STD-498's Software Unit is Not Software

Software Units: Design

3.45 Software unit. An element in the design of a CSCI…

5.6.2 "The developer shall define and record the architectural design of each CSCI (identifying the software units comprising the CSCI, their interfaces, and a concept of execution among them) and the traceability between the software units and the CSCI requirements…"

5.6.3 "The developer shall develop and record a description of each software unit…"

Software Unit Software

5.7.1 "The developer shall *develop and record software corresponding to each software unit* in the CSCI design…"

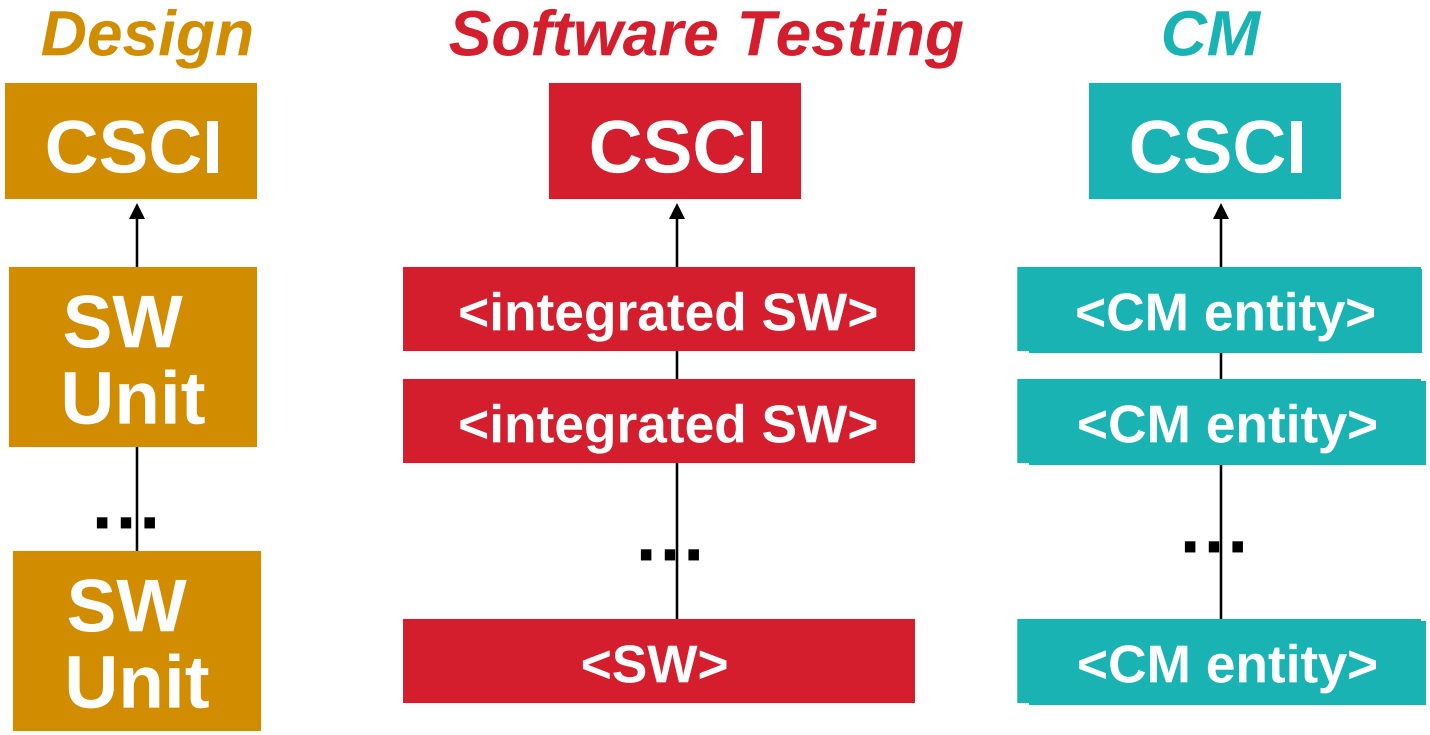
5.7.2 "The developer shall establish test cases…test procedures, and test data for *testing the software* corresponding to each software unit…"

5.7.3 "The developer shall *test the software* corresponding to each software unit…"

Software or Software

5.14.1 "The developer shall…***identify the entities*** to be placed under configuration control…The identification scheme shall be at the level at which entities will actually be controlled, ***for example, computer files, electronic media, documents, software units, configuration items.***"

New Concepts in MIL-STD-498



Major Topics

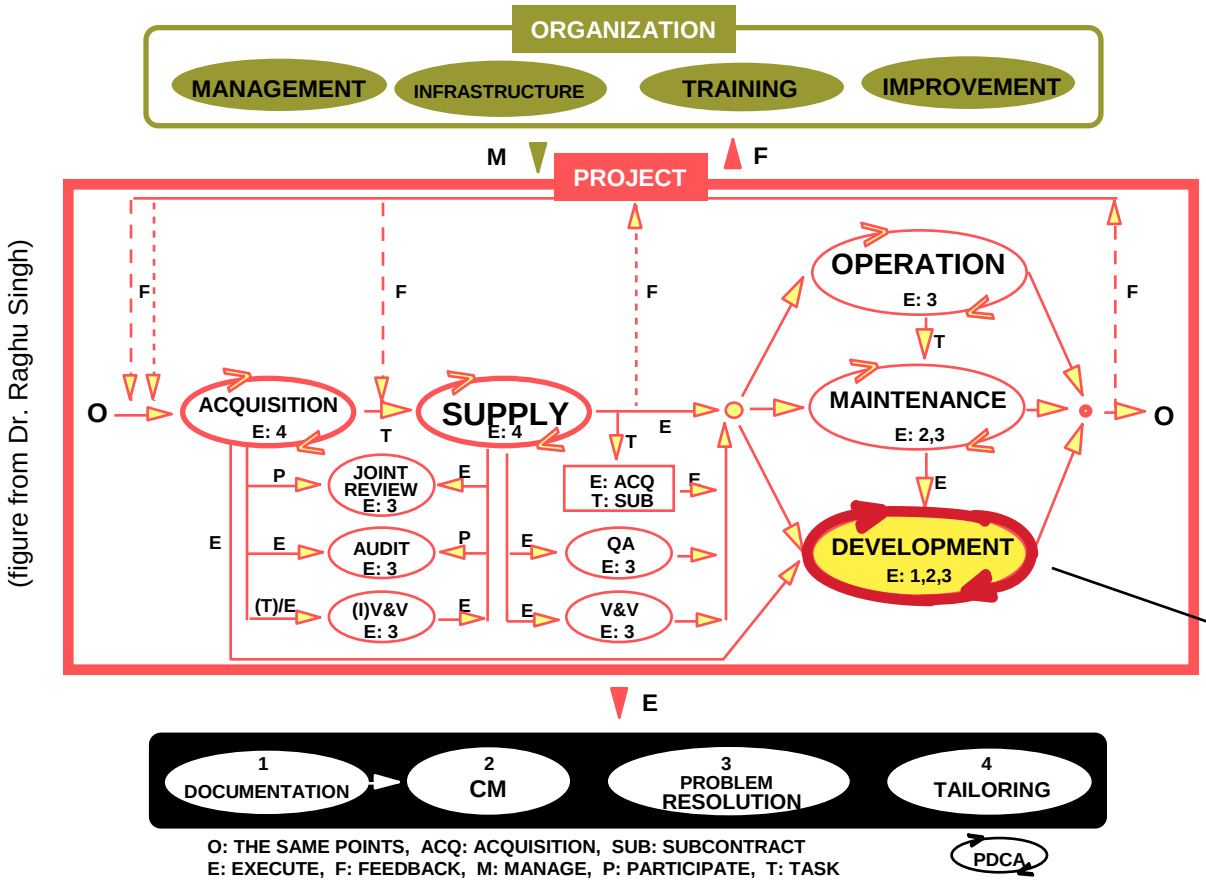
Booch O-O life cycle

Booch O-O practices

Using MIL-STD-498 with O-O

Using ISO/IEC 12207 with O-O

ISO/IEC 12207's Key Processes



(figure from Dr. Raghu Singh)

- Organizations acquire software through projects
- Projects participate in contracts
- MIL-STD-498 corresponds to Development process of ISO/IEC 12207

Compatibility with O-O Life Cycle Models

- "This International Standard does not prescribe a specific life cycle model or software development method." (par. 1.5)
- "The developer shall define or select a software life cycle model appropriate to the scope, magnitude, and complexity of the project. The activities and tasks of the development process shall be selected and mapped onto the life cycle model…Activities and tasks may overlap or interact and may be performed iteratively or recursively." (par. 5.3.1.1)

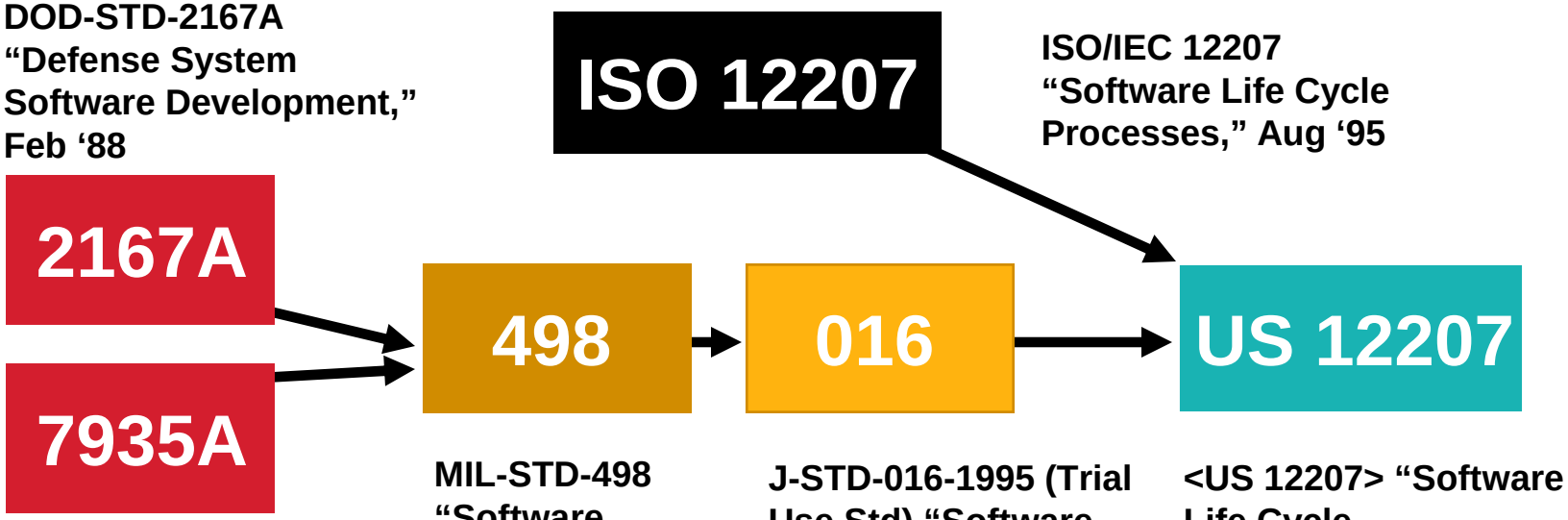
Software Design with ISO/IEC 12207

- ISO/IEC 12207 also avoids the translation and linking problems with DOD-STD-2167A.

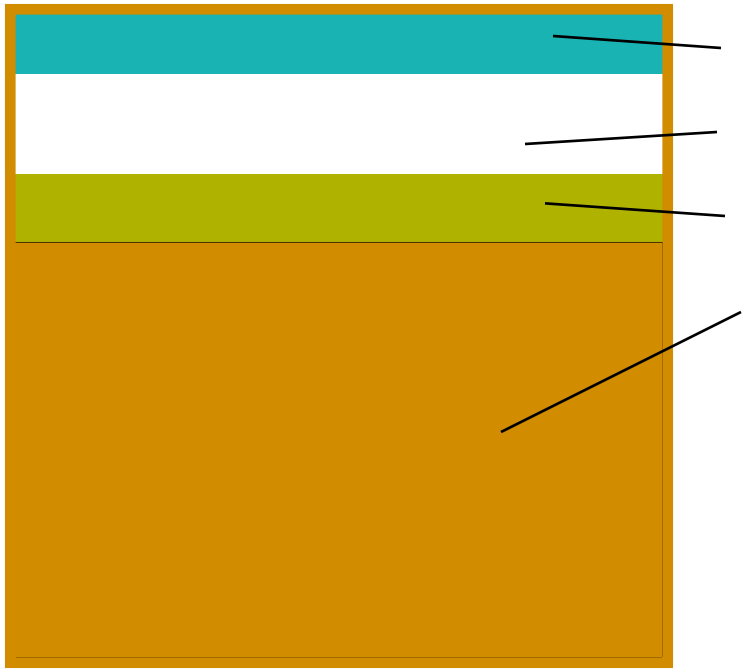
Topic all by itself!

- ISO/IEC 12207 requires developers to create software components and to record their design.
- The standard is compatible with any design method and any relationship between design elements, as the requirements in MIL-STD-498 are.

Looking Ahead to US 12207



Planned Logical Structure of US 12207



- Forward: U.S. Introduction
- ISO/IEC 12207 as is (no changes)
- ISO/IEC 12207 Annexes
- J-STD-016 Annexes (including new technical annotations and J-016 product descriptions [i.e., content of former 498 DIDs])
- ***Physical structure of US 12207 could be "facing pages" of ISO 12207 and US annotations***

Bringing it Home

Booch Macro Process -- Booch Micro Process - MIL-STD-498 - ISO/IEC 12207

Your role

- Your OOD method
- Your tailored project standard

Using MIL-STD-498 with OOD

- MIL-STD-498 is a checklist of activities to consider when planning a software development project. It's intended for experienced, skilled readers. It's not a textbook, manual, or handbook on how to develop software.
- MIL-STD-498 defaults to requiring all activities. Don't let this happen!
- First, understand your software process. Then tailor MIL-STD-498 to apply your process.
- Use the project SOW to make as much of the software process self documenting as is appropriate. Establish an early acquisition or supply milestone to decide what is appropriate.

Preparing for US 12207 by Investing in MIL-STD-498 Competence

- ISO/IEC 12207 is more abstract, more general, than MIL-STD-498. MIL-STD-498 is more abstract than DOD-STD-2167A and earlier military software development standards. Standards assume more today.
- US 12207 will bridge between MIL-STD-498 and ISO/IEC 12207.
- Learn how to perform your “one software process” with MIL-STD-498, establish practices to do it.
- Your practices will be compatible with US 12207, and as a result with ISO/IEC 12207 also.