THE MANAGEMENT IMPLICATIONS OF THE RECURSIVE NATURE OF OBJECT-ORIENTED DEVELOPMENT

presented by

DONALD G. FIRESMITH

Magnavox Electronic Systems Company
M/S 10-C-3 Dept. 566
1313 Production Road
Fort Wayne, IN 46808
(219) 429-4327

THE IMPORTANCE OF THIS TOPIC

- Object-Oriented Development (OOD) is the most popular class of Ada-oriented software development methods.

- OOD methods are RECURSIVE, globally top-down, "hierarchical" composition software development methods that modularize according to object abstraction and information hiding.

- The recursive nature of OOD is often overlooked, underemphasized, or misunderstood.

- The recursive nature of OOD requires major, often unrecognized, changes in management.

THE RECURSIVE NATURE OF OOD

- THE BASIC METHOD
- ASSEMBLIES
- SUBASSEMBLIES
- THE OOD LIFE-CYCLE

THE BASIC METHOD

OOD methods share the following basic steps. Recursively, by abstraction level:

- Identify the relevant abstract objects.

- For each abstract object:

  When too large for a single package =>

  Identify the subsystem that encapsulates its component packages. $\} \rightarrow$

  When small enough for a single package =>

  Identify the associated Ada objects and operations on Ada objects of that type. Note that:

  - Ada objects and operations are only considered within the context of the abstract object package in which they will be encapsulated.

  - Operations are only considered within the context of the type of Ada object on which they operate.

Identify the associated Abstract State Machine (ASM) or Abstract Data Type (ADT) package that encapsulates its Ada objects and operations. Note: one package per abstract object.

Code and partially test the packages.

- Recurse the method on any stubbed operations (~~i.e.~~ eg., create a new abstraction level of abstract object packages).

ASSEMBLY

The total set of all Ada programming units developed during all recursions of the OOD process applied to a specific set of coherent software requirements (i.e., requirements that specify a single well-defined problem).

SUBASSEMBLY

The set of those Ada programming units developed during only a single NON-RECURSIVE pass through the OOD process. This is the amount of software documented on a standard OOD diagram. A subassembly is a small (roughly 1-2 KSLOC), manageable subset of an assembly.

The OOD Life-Cycle

THE MANAGEMENT IMPLICATIONS

- PRODUCTS
- STAFFING
- SCHEDULES
- RISKS
- BENEFITS

PRODUCTS

- Early compilable designs:

  Because package specifications are naturally produced on a subassembly by subassembly basis (i.e., design a little, code a little, test a little), OOD produces compilable, rather than paper, designs very early in the software development life-cycle. This allows the contractor to use the compiler to check interface consistencies and to easily ensure design and code consistency.

- Early software:

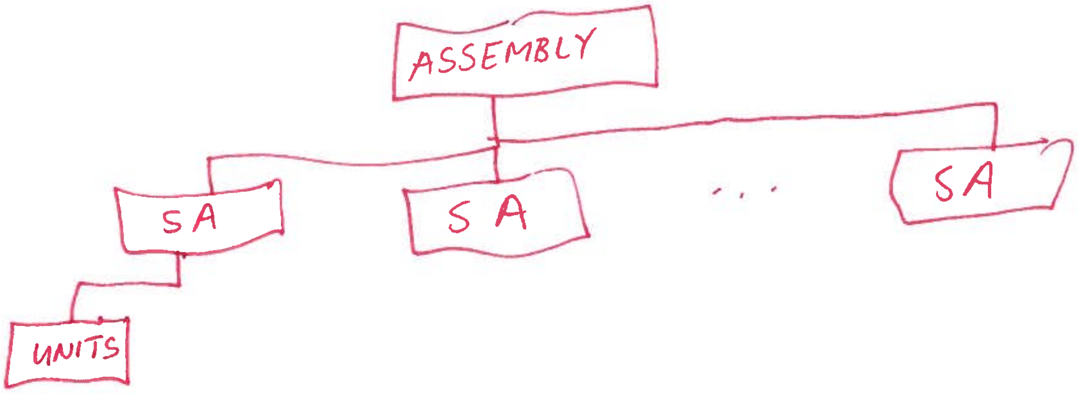
  Because package bodies are also naturally produced on a subassembly by subassembly basis (i.e., design a little, code a little, test a little), OOD also produces executable code very early in the software development life-cycle. This allows the contractor to detect bugs earlier by testing (i.e., package and subassembly testing) the software earlier. This lowers both development risks and costs.

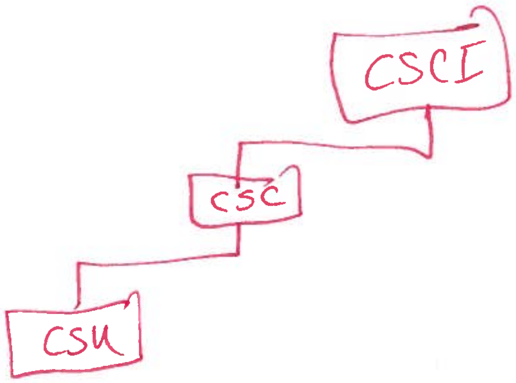
- Software Development Files (SDFs):

  The optimum use of SDFs is on a subassembly basis. SDFs should be OOD oriented and document the natural intermediate products of the process.

- New products at design reviews (PDR and CDR) fixed

- Mapping to CSCIs, CSCs, and CSUs





Depending upon when OOD is initiated, an assembly may be either a DOD-STD-2167 system, subsystem/segment, CSCI, TLCSC, or LLCSC.

STAFFING

- Software Development Teams

  A successful approach is to assign a software development team (consisting of a designer, coder, and tester with one or more optional peer inspectors from other teams) to develop each subassembly.

- Parallel development

  Different software development teams simultaneously work in parallel on different subassemblies.

- Mongolian horde myth

  Because the number of subassemblies tends to grow exponentially, and because object abstraction, info. hiding, and the proper use of Ada minimize coupling between subassemblies, OOD allows the manager to increase productivity by increasing the staff.

SCHEDULE

- New life-cycle model
- Incompatibility with DOD-STD-2167 and MIL-STD-1521
- New intermediate milestones
- New heuristics

RISKS

- Training problems
- Problems with standards
- Lack of metrics
- Management expectations
- Customer and IV&V expectations
- Incompatibility with military standards
- Deferring coding and testing until after design
- "What if you find a major design problem late in the process?"

BENEFITS

- Parallel development leads to high productivity
- Validate design as you develop
- Early customer review of compilable design actual code possible
- Early discovery of errors
- Documentation mirrors development

CONCLUSION

- TBD