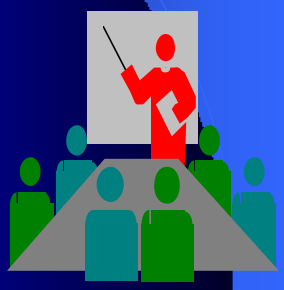
FAA National Software Conference
Object Oriented Technology

Object-Oriented Technology in Aviation
~Certification Concerns~

Leanna Rierson
Federal Aviation Administration
Leanna.Rierson@faa.gov
August, 2000

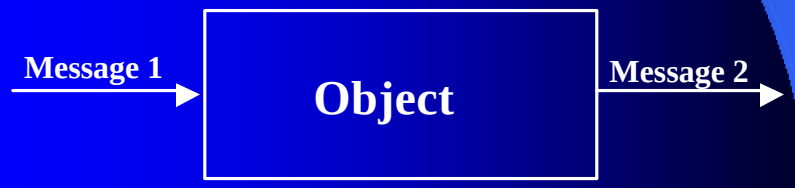
Presentation Outline

- Overview of Object-Oriented Technology (OOT)
- Certification Concerns of OOT
- Summary



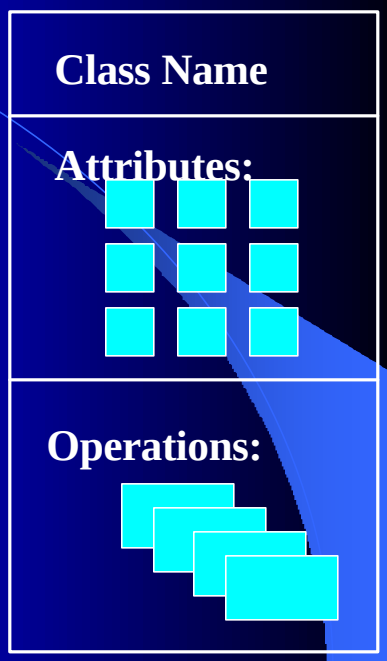
OOT Overview

- IEEE Definition of OOT: "A software development technique in which a system or component is expressed in terms of objects and connections between those objects"
- Centered around "objects" and "classes"



OOT Overview (2/7)

- Definition of Class: "a set of objects that share a common structure and a common behavior" (Booch)



OOT Overview (3/7)

7 Major Principles

Typical	Unique to OOT
– abstraction	– **encapsulation
– modularity	– **hierarchy
– concurrency	– **typing
– persistence	

OOT Overview (4/7)

- Abstraction: *Helps to address complexity by providing crisply defined boundaries.*
- Modularity: *The process of partitioning a program into logically separated and defined components that possess defined interactions and limited access to data.*
- Concurrency: *Process of carrying out several events simultaneously.*
- Persistence: *Property of an object through which its existence transcends time and/or space.*

OOT Overview (5/7)

- Encapsulation:
  - The mechanism that binds together code and the data it manipulates
  - Keeps code and data safe from outside interference and misuse
  - Generally achieved through *information hiding*

OOT Overview (6/7)

- Hierarchy: The ordering of abstractions.
  - Examples of hierarchy: *single inheritance* and *multiple inheritance*
  - Sub-class "inherits" all of the existing attributes and operations of the original class, called the "parent" or "superclass"

OOT Overview (7/7)

- Typing: Enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted ways
  - *Polymorphism* is a concept closely related to typing.
  - *Polymorphism* comes from the Greek meaning "many forms."

OOT Methodology

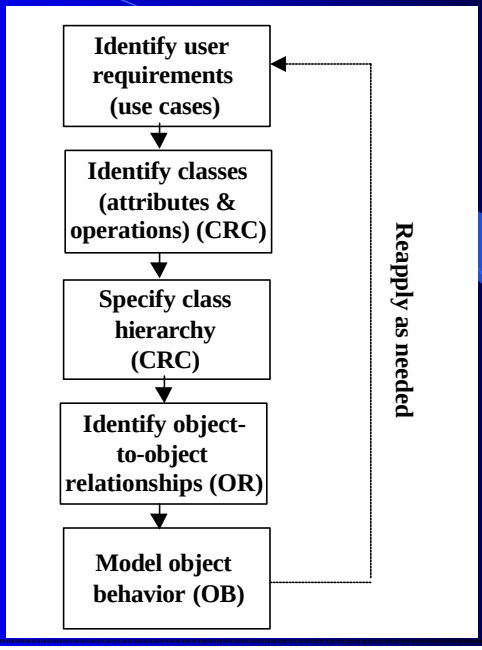
OOA = Object-Oriented Analysis

OOD = Object-Oriented Design

OOP = Object-Oriented Programming

OOV/T = Object-Oriented Testing

OOA



OOD

- Blueprint for software construction.
- Four layers of design are usually defined:
  - subsystem layer,
  - class and object layer,
  - message layer, and
  - responsibilities layer.

OOP

- Examples: SmallTalk, Java, C++, Ada 95
- C++ starting to be used in airborne avionics
- Some concerns: dynamic memory allocation, , multiple inheritance, virtual base classes, run-time identification, templates, exceptions, and namespaces are deleted

OOV/T

- Process of detecting errors and verifying correctness of the OOA, OOD, and OOP. OOV/T
- Includes reviews, analyses, and tests of the software design and implementation

OOV/T

- OOV/T requires slightly different strategies and tactics than the traditional structured approach.
  - Because of inheritance, encapsulation, and polymorphism.
- Most developers use a "design for testability" approach to begin addressing any verification/test issues early in the program.

Certification Concerns of OOT

OOT in Airborne Software - Certification Concerns

- Planning -
  - Address OOT issues up front
- Traceability -
  - Multiple inheritance makes traceability more complex
  - Tools for multiple life cycle phases are limited
- Target Compatibility
  - Dynamic allocation can be problematic

OOT in Airborne Software - Certification Concerns (cont)

- Structural Coverage
  - Inheritance and polymorphism make structural coverage more difficult
- Dead/Deactivated Code
  - Sub-classes over-riding superclasses might introduce dead code
- Verification/Test
  - Requires different test methods

OOT in Airborne Software - Certification Concerns (cont)

- Overuse of Inheritance
  - Could make data & control coupling objectives difficult to meet
- Ambiguity
  - Polymorphism & overloaded functions may lead to ambiguity
- Library Dependence
  - Libraries should be used cautiously

OOT in Airborne Software - Certification Concerns (cont)

- Coding Issues (language sub-set)
  - Example of C++ restrictions:
    - Minimize dynamic binding
    - Minimize operator overloading
    - Minimize control flow complexity
    - Use “new” only at initialization
    - Avoid using “delete”
    - Avoid use of exception handling
    - Avoid multiple inheritance
    - Avoid type-cast pointers

Future Efforts

- FAA is Striving to Carry Out Further Research in this Area

Your Thoughts

- What kinds of experience have you had?
- What are your concerns?

Summary

- Many manufacturers desire to use OOT in their airborne aviation products
- OOT should be used carefully in order to meet the objectives of DO-178B
- FAA is continuing to research this area further
- FAA welcomes input