

Shlaer-Mellor Object-Oriented Analysis

Introductory Class

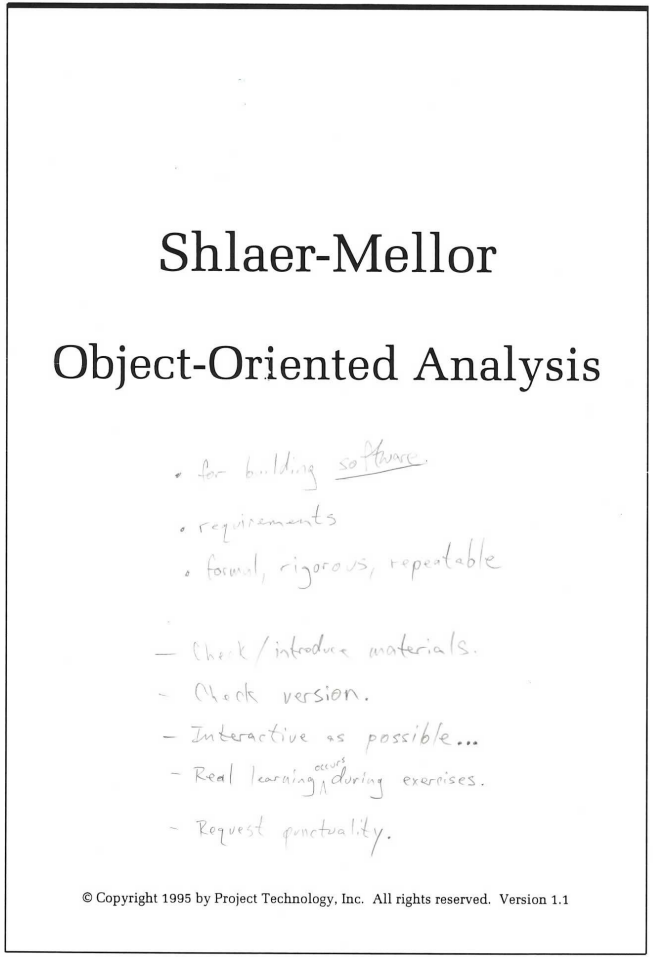




Shlaer-Mellor

Object-Oriented Analysis

© Copyright 1995 by Project Technology, Inc. All rights reserved. Version 1.1





- For building software
- Requirements
- Formal, rigorous, repeatable

- Check/introduce materials
- Check version
- Interactive as possible...
- Real learning occurs during exercises
- Request punctuality



Table of Contents

1. Introduction
   - 1.1 The Shlaer-Mellor Method
   - 1.2 Object-Oriented Software Development
   - 1.3 Planning and Object-Oriented Analysis
   - 1.4 Objectives

2. Domains
   - 2.1 Concept of a Domain
   - 2.2 Types of Domains
   - 2.3 Bridges, Clients and Servers
   - 2.4 Domains and Recursive Design
   - 2.5 Identifying Domains

3. Information Modeling Concepts
   - 3.1 Information in the Real World
   - 3.2 Modeling Real World Things
   - 3.3 Modeling Associations Between Things
   - 3.4 Graphic Representation of the Information Model

4. Objects
   - 4.1 Finding Objects
   - 4.2 Documenting Objects
   - 4.3 Choosing Names for Objects
   - 4.4 Testing Objects

© Copyright 1994 by Project Technology, Inc. All rights reserved.



5. Attributes
   - 5.1 Finding Attributes
   - 5.2 Identifiers
   - 5.3 Attribute and Domain Descriptions

6. Modeling Relationships
   - 6.1 Reprise of Relationship Modeling
   - 6.2 Relationship Multiplicity
   - 6.3 Relationship Conditionality
   - 6.4 Associative Objects
   - 6.5 Subtype and Supertype Objects
   - 6.6 Relationship Descriptions

7. State Models
   - 7.1 Concept of a Lifecycle
   - 7.2 Formalization of State Models
   - 7.3 Building State Models
   - 7.4 State Machines
   - 7.5 Lifecycle Forms
   - 7.6 When to Formalize Lifecycles

8. Dynamics of Relationships
   - 8.1 Relationship Lifecycles
   - 8.2 Competitive Relationships

© Copyright 1994 by Project Technology, Inc. All rights reserved.



9. Analysis of Large Systems
   - 9.1 Object Communication Model and Layering State Models
   - 9.2 Patterns of Communication
   - 9.3 Simulation
   - 9.4 Timers
   - 9.5 Repetitive Control

10. Process Modeling
   - 10.1 Process Models in Object-Oriented Analysis
   - 10.2 Action Data Flow Diagrams
   - 10.3 Labeling Data Flows
   - 10.4 Executing Processes
   - 10.5 Forming, Naming, and Assigning Processes
   - 10.6 Managing Process Modeling

11. Subsystems
   - 11.1 Concept of a Subsystem
   - 11.2 Initial Definition of a Subsystem
   - 11.3 Numbering and Labeling Conventions
   - 11.4 Defining Subsystem Boundaries

12. Wrap-Up

Appendix A: Project Matrix

Appendix B: The Process of Analysis



Chapter 1: Introduction

Section 1: The Shlaer-Mellor Method



The Shlaer-Mellor Method

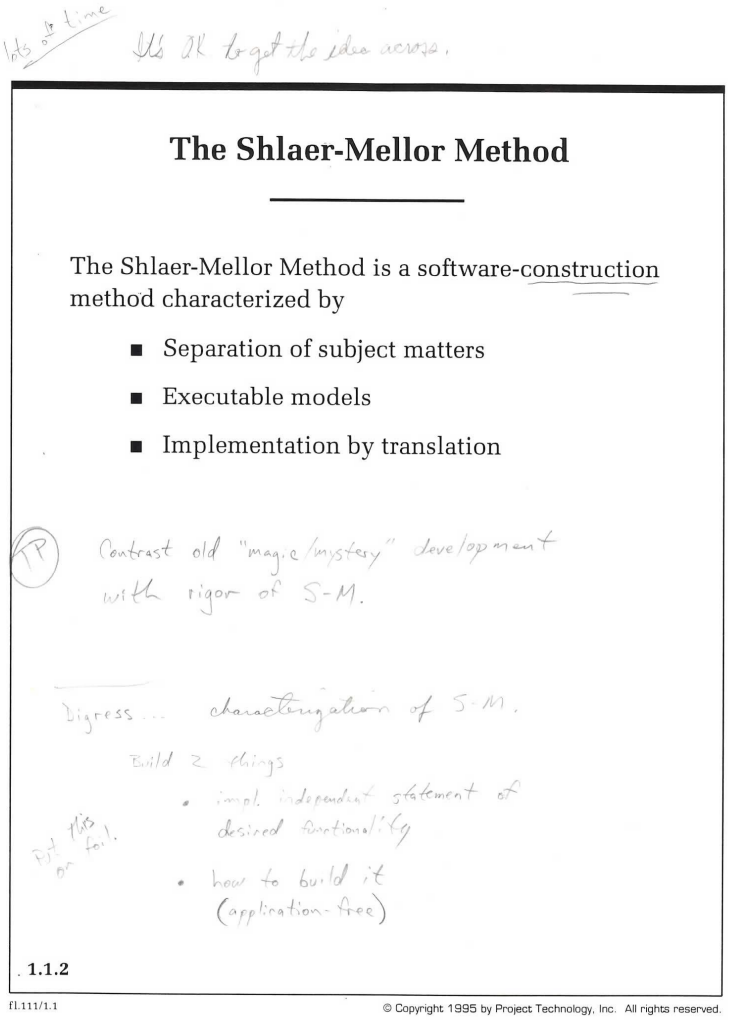
The Shlaer-Mellor Method is a software-construction method characterized by

- Separation of subject matters
- Executable models
- Implementation by translation

1.1.2

f1.111/1.1

© Copyright 1995 by Project Technology, Inc. All rights reserved.



The Shlaer-Mellor Method

The Shlaer-Mellor Method is a software-construction method characterized by

- Separation of subject matters
- Executable models
- Implementation by translation

1.1.2

f1.111/1.1 © Copyright 1995 by Project Technology, Inc. All rights reserved.



- Lots of time
- It's OK to get the ideas across
- TP: Contrast old "magic/mystery" development with rigor of S-M

- Digress… characterization of S-M

- Build 2 things (next slide)
  - Implementation independent statement of desired functionality
  - How to build it (application free)



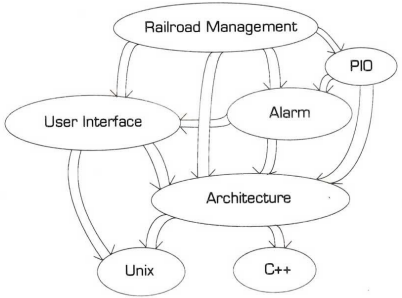
Build 2 things:

- implementation independent statement of desired functionality
- "how to build it" application free



Separation of Subject Matters

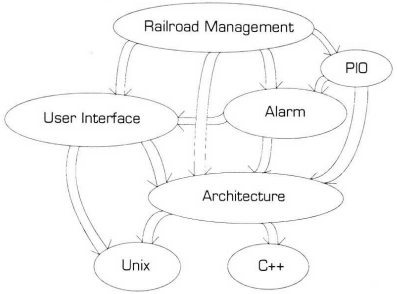
Divide the problem into distinct, independent subject matters called domains.



© Copyright 1995 by Project Technology, Inc. All rights reserved.

Separation of Subject Matters

Divide the problem into distinct, independent subject matters called domains.



Subject matters: "What types of experts do I need?"

1.1.3

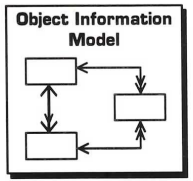
© Copyright 1995 by Project Technology, Inc. All rights reserved.

- TP: Subject matters. "What types of experts do I need?"

Executable Models

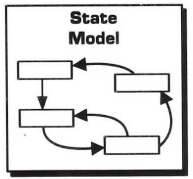
- For each domain

  build an object information model



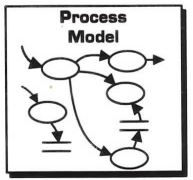
- For objects in the information model

  build a state model



- then for each state in each state model,

  build a process model



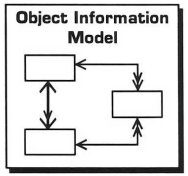
This collection of models constitute a Shlaer-Mellor Object-Oriented Analysis.

1.1.4

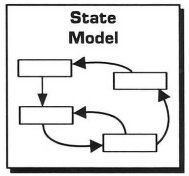
fl.111/1.1 © Copyright 1995 by Project Technology, Inc. All rights reserved.

Executable Models

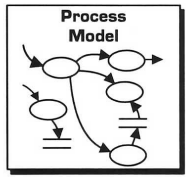
- For each domain
  build an object information model



- For objects in the information model
  build a state model



- then for each state in each state model,
  build a process model



This collection of models constitute a Shlaer-Mellor Object-Oriented Analysis.

1.1.4

TP Overview of models and order.
A+B+C = S-M Analysis

© Copyright 1995 by Project Technology, Inc. All rights reserved.

- "Specifications"
- TP: Overview of models and order.
- A + B + C = S-M analysis



Executable Models Verifying the Analysis

OOA models are

- formal
- precise
- complete.

formal, rigorous "religious"

OOA models can be executed to

- verify the models satisfy requirements
- identify missing or incomplete requirements
- identify performance characteristics of the domain

before any code is written.

1.1.5

fl.111/1.1

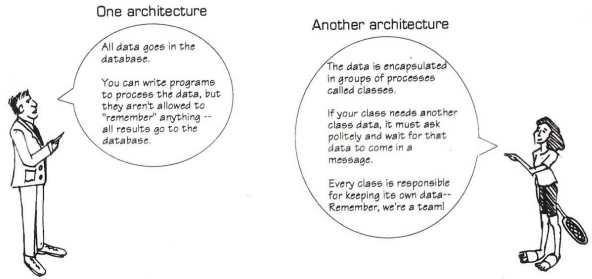
© Copyright 1995 by Project Technology, Inc. All rights reserved.



Software Architecture Domain

The software architecture domain proclaims and enforces systemwide policies regarding

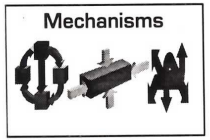
- the organization and access of data
- the overall control (when things happen) of the system as a whole
- the organization of processing.



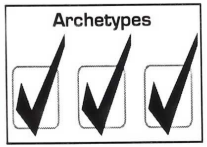
The software architecture comprises a set of highly reusable, uniform rules for constructing the code.

© Copyright 1995 by Project Technology, Inc. All rights reserved.

Components of the Software Architecture



- Compiled code that executes the rules of OOA in real-time.



- Source code fragments for each component in an OOA model that must be coded separately but uniformly.

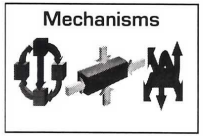
AND

- Rules for translating the OOA models and populating the code fragments.

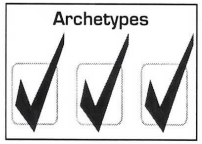
The architecture (mechanisms plus archetypes) serves as an application-free statement of the software design.

© Copyright 1995 by Project Technology, Inc. All rights reserved.

Components of the Software Architecture



- Compiled code that executes the rules of OOA in real-time.



- Source code fragments for each component in an OOA model that must be coded separately but uniformly.

AND

- Rules for translating the OOA models and populating the code fragments.

The architecture (mechanisms plus archetypes) serves as an application-free statement of the software design.

1.1.7

f1.111/1.1

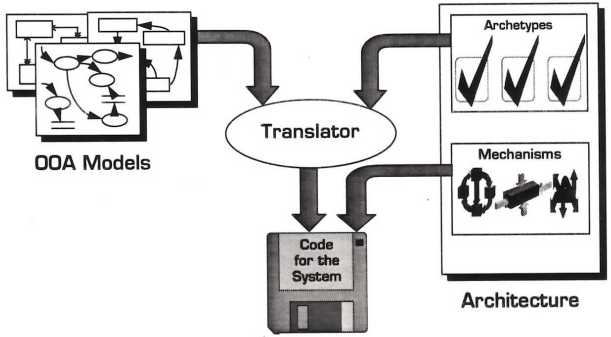
© Copyright 1995 by Project Technology, Inc. All rights reserved.

- TP: Try to put some meat on the idea of "architecture"
- "Model Compilers"



Implementation by Translation

Code for the system is constructed by translating the OOA models using the software architecture.

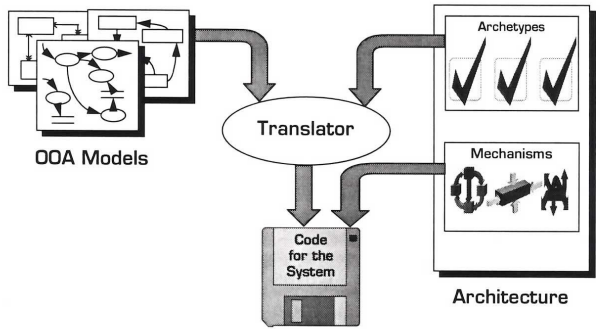


1.1.8

© Copyright 1995 by Project Technology, Inc. All rights reserved.

Implementation by Translation

Code for the system is constructed by translating the OOA models using the software architecture.



Wow! Really!?

additional analogy: source + compiler = code

1.1.8

f1.111/1.1

© Copyright 1995 by Project Technology, Inc. All rights reserved.

- TP: Wow! Really!?
- Additional analogy: source + compiler = code



Implementation by Translation

Because a translation approach to software construction separates application information from implementation strategies, it is easy to:

- accommodate application requirement changes ← Modify the application OOA models
- tune for performance
- port the system to different implementation environments

Modify the translation rules



1.1.9

fl.111/1.1

© Copyright 1995 by Project Technology, Inc. All rights reserved.



Applying the Shlaer-Mellor Method

- Partition the system into domains
- Analyze each domain
- Verify the analysis through simulation
- Specify the translation of the analysis models
- Build the translation components
- Translate the models of each domain.

1.1.10

fl.111/1.1

© Copyright 1995 by Project Technology, Inc. All rights reserved.

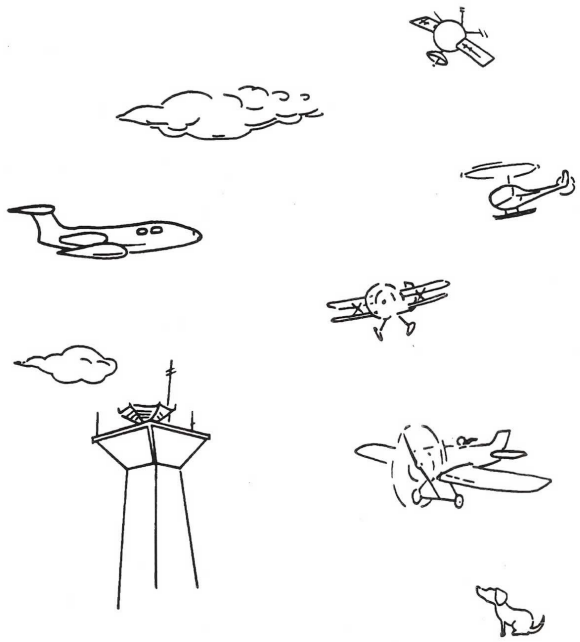


Chapter 3: Information Modeling Concepts

Section 1: Information in the Real World



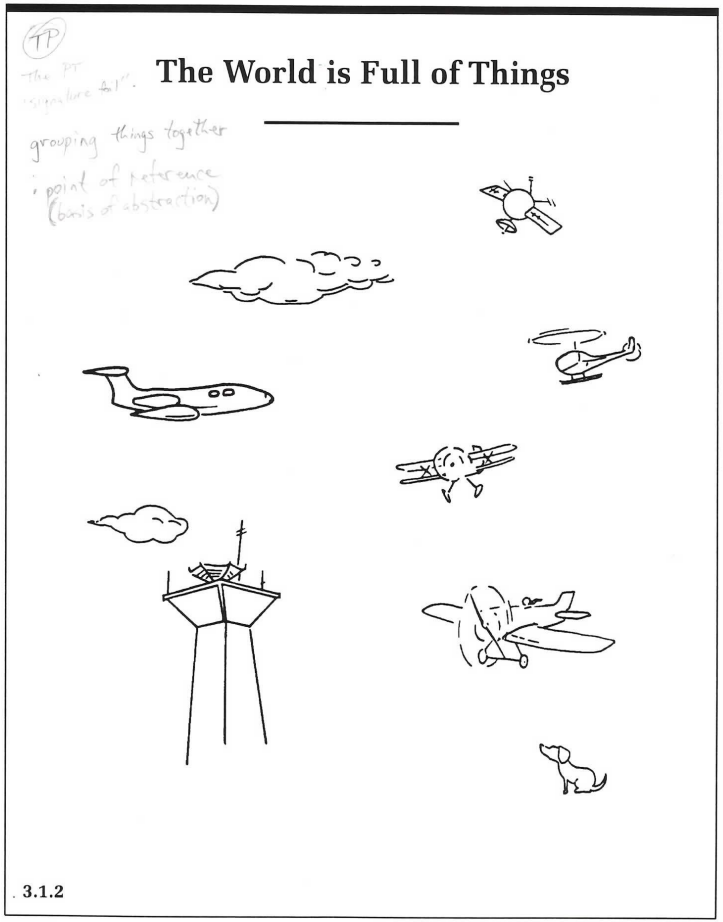
The World is Full of Things



3.1.2

fl.033/1.1

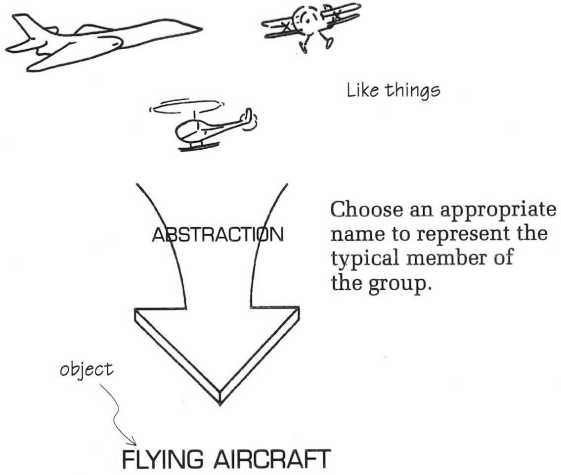
© Copyright 1993 by Project Technology, Inc. All rights reserved.



- TP: The PT “signature foil”
- Grouping things together
- Point of reference (basis of abstraction)

Objects

Things that behave similarly and have the same characteristics can be denoted by a common name.



The name is the name of an object—an abstraction we hold in our minds.

3.1.8

fl.033/1.1

© Copyright 1993 by Project Technology, Inc. All rights reserved.