Software Development Methodologies

Lecturer: **Raman Ramsin**

Lecture 3

Seminal Methodologies: Fusion

Fusion

- First introduced in 1992 by a team of practitioners at Hewlett-Packard Laboratories
- A revised and detailed version was released in 1994
- Result of the integration, unification and extension of older methodologies, mainly OMT, Booch, OOSE and RDD
- Described as a full-lifecycle method, but analysis starts when a preliminary requirements document is already available
- Provides consistency and completeness checks between phases, and criteria for determining when to move from one phase to the next

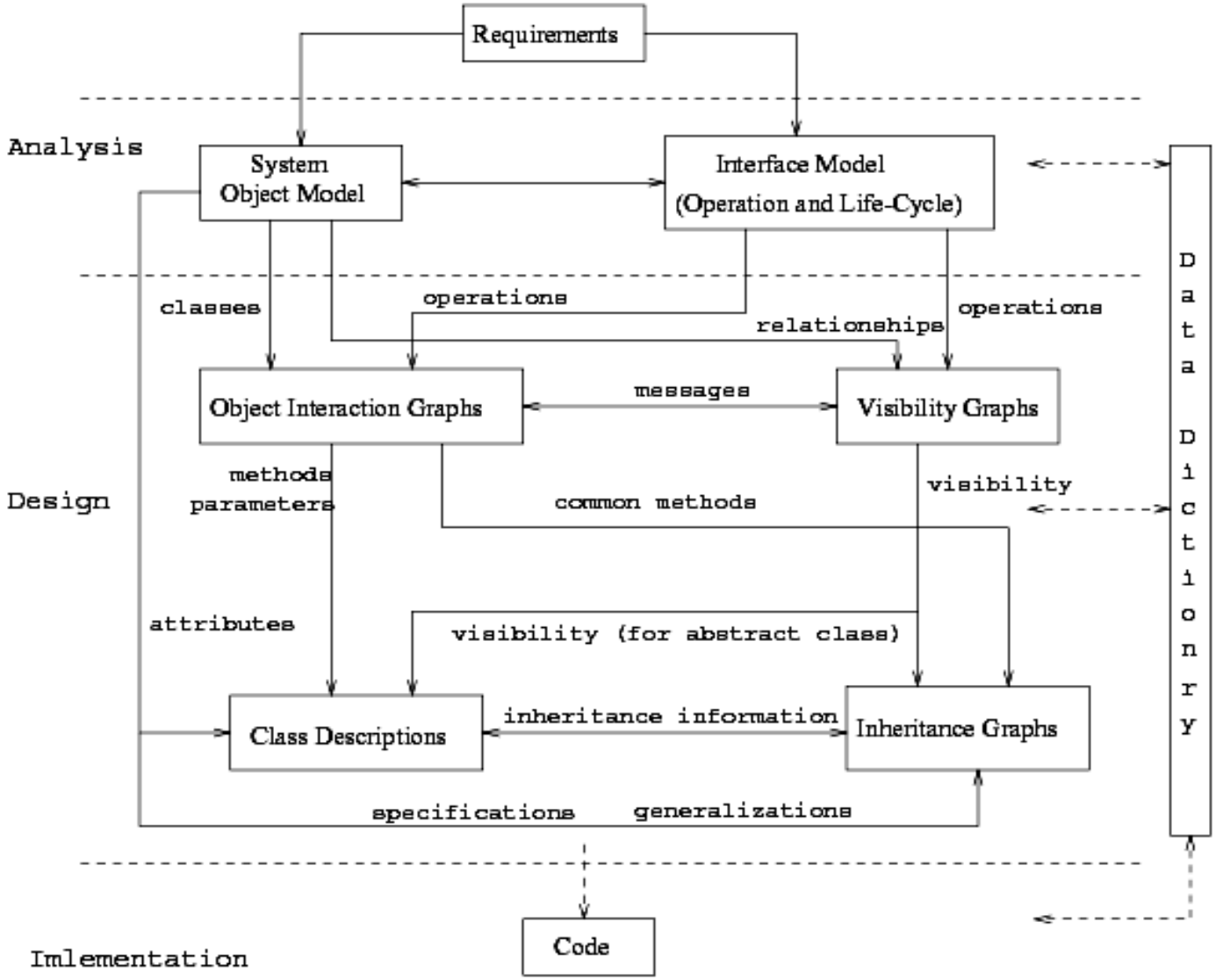
Fusion: Process

- ***Analysis***: the focus is on what the system does. The system is described from the standpoint of the user. Requirements are mapped to the *System Specification.* Analysis models describe:
  - classes and objects of interest found in the application domain, and their relationships,
  - the operations which are to be performed by the system, and
  - the proper ordering of these operations.

- ***Design***: the focus is on how the system is to do what has been defined during analysis. The specification of the system is mapped to a blueprint for the implementation of the system. Design models describe:
  - realization of system operations in terms of cooperating objects,
  - how these objects are linked together,
  - how classes, to which the objects belong, are specialized and refined (the inheritance structure of the classes), and
  - the detailed particulars of each class's attributes and methods.

- ***Implementation***: the focus is on the actual coding of the system. The system design is mapped to a particular programming environment.

Fusion: Process



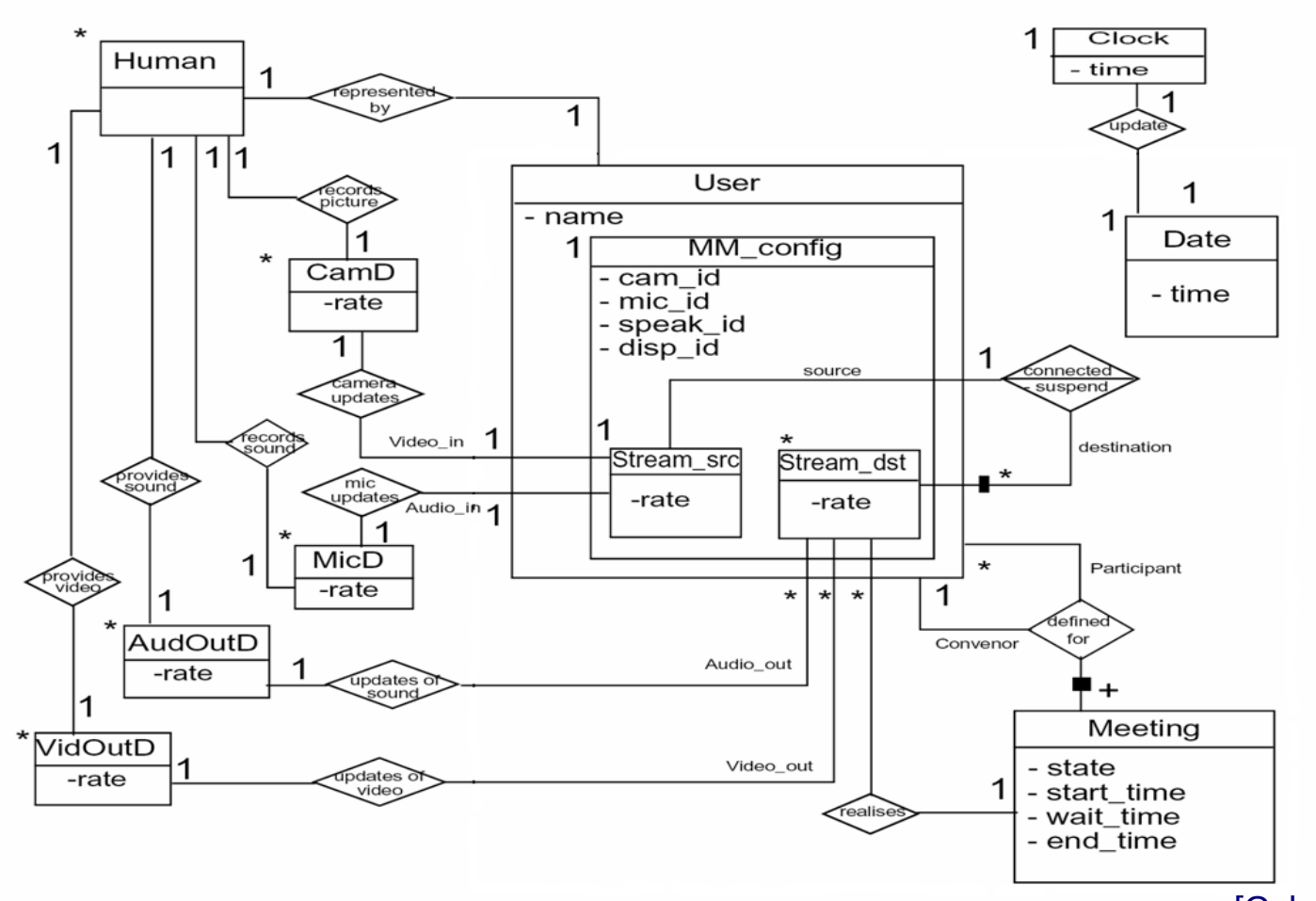
[Coleman et al. 1994]

Fusion Process: Analysis Phase

- Concerned with capturing the requirements of the system
- The requirements specification document is the standard input to the analysis phase
- Two models are produced in this phase:
  - **System Object Model**
  - **System Interface Model**, further divided into two models:
    - **LifeCycle Model**
    - **Operation Model**
- Steps:
  1. Develop an overall *Object Model* encompassing the system and its environment
  2. Develop the *System Object Model* by explicitly defining the system boundary
  3. Develop the *System Interface Model* denoting system operations
  4. Check the analysis models using detailed checklists

Fusion Process: Analysis Phase

- Step 1: Develop an overall *Object Model*, through
  - Grammatical Parsing, resulting in objects, classes, relationships and attributes
  - Model perfection through interviews and observation



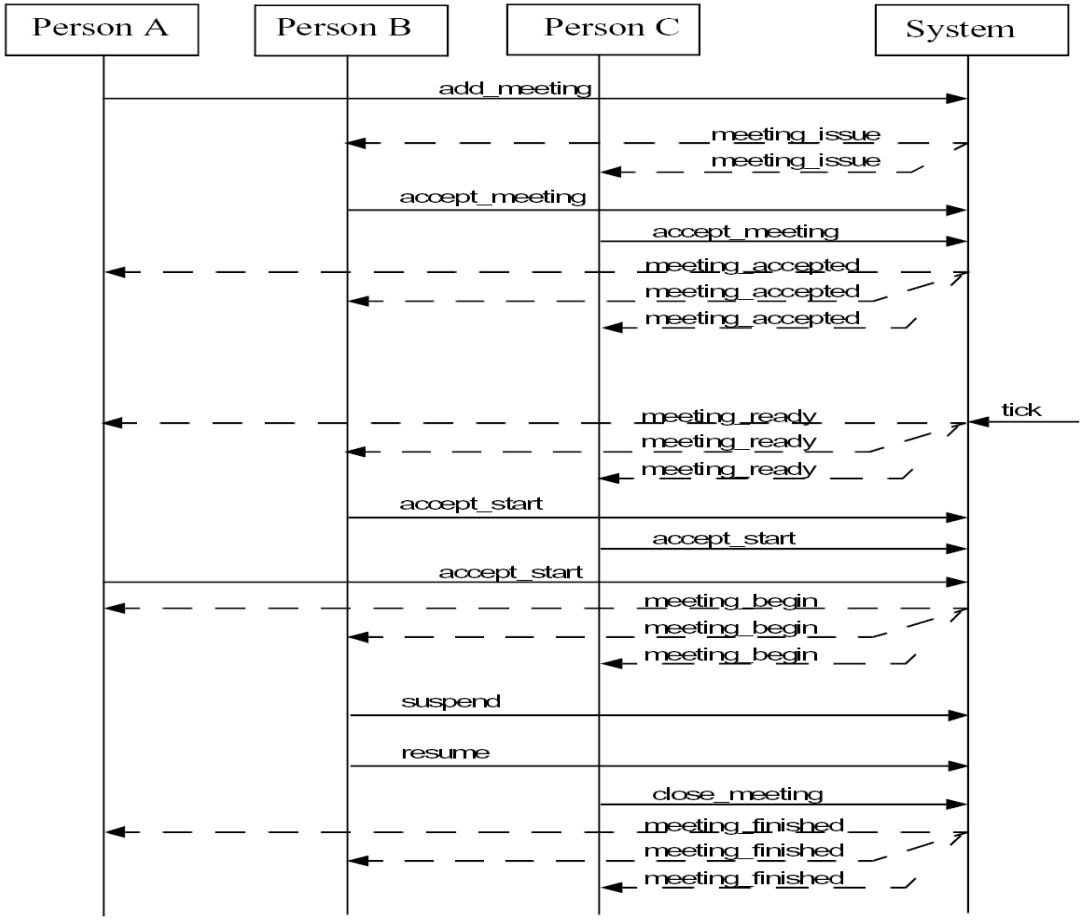
[Coleman et al. 1994]

Fusion Process: Analysis Phase

- Step 2: Develop the *System Object Model*, through:
  - Determining interaction patterns between the system and outside *agents* (users, devices or other systems); typical interactions (input and output *events*) are modeled as *Transaction Scenarios* accentuating the time order of events. An input event and its effect on the system are collectively called a system *operation*.
  - Specification of the *System Interface Diagram* by integrating all *transaction scenarios*
  - Producing the *System Object Model* by adding a boundary to the overall *object model*, based on the agents and system operations identified

Fusion Process: Analysis Phase

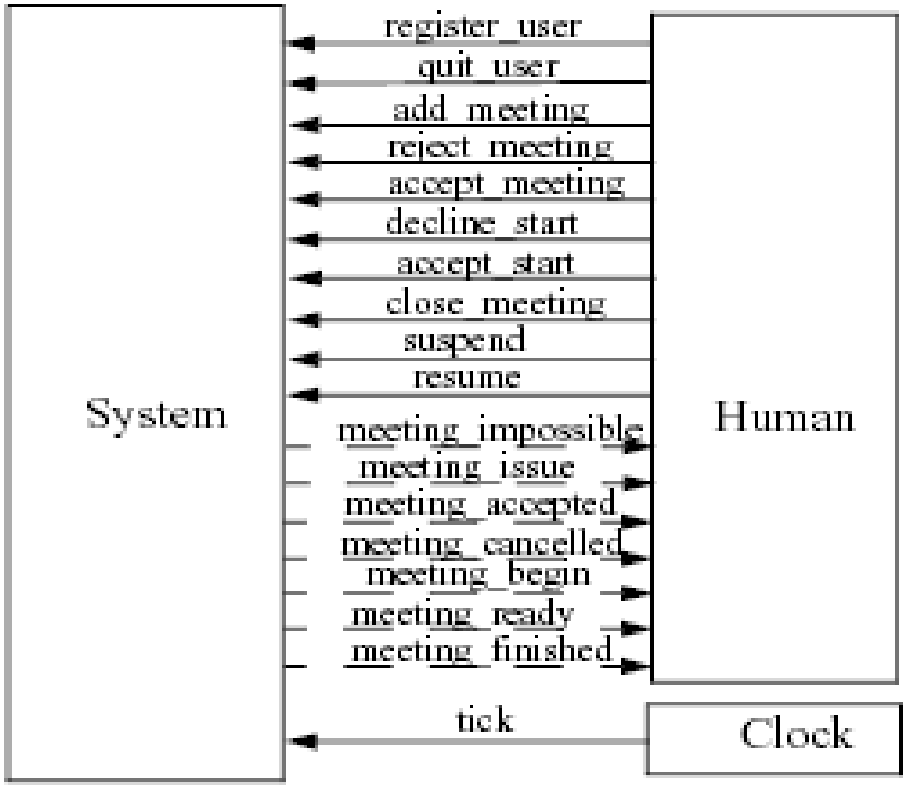
Transaction Scenario



[Coleman et al. 1994]

Fusion Process: Analysis Phase

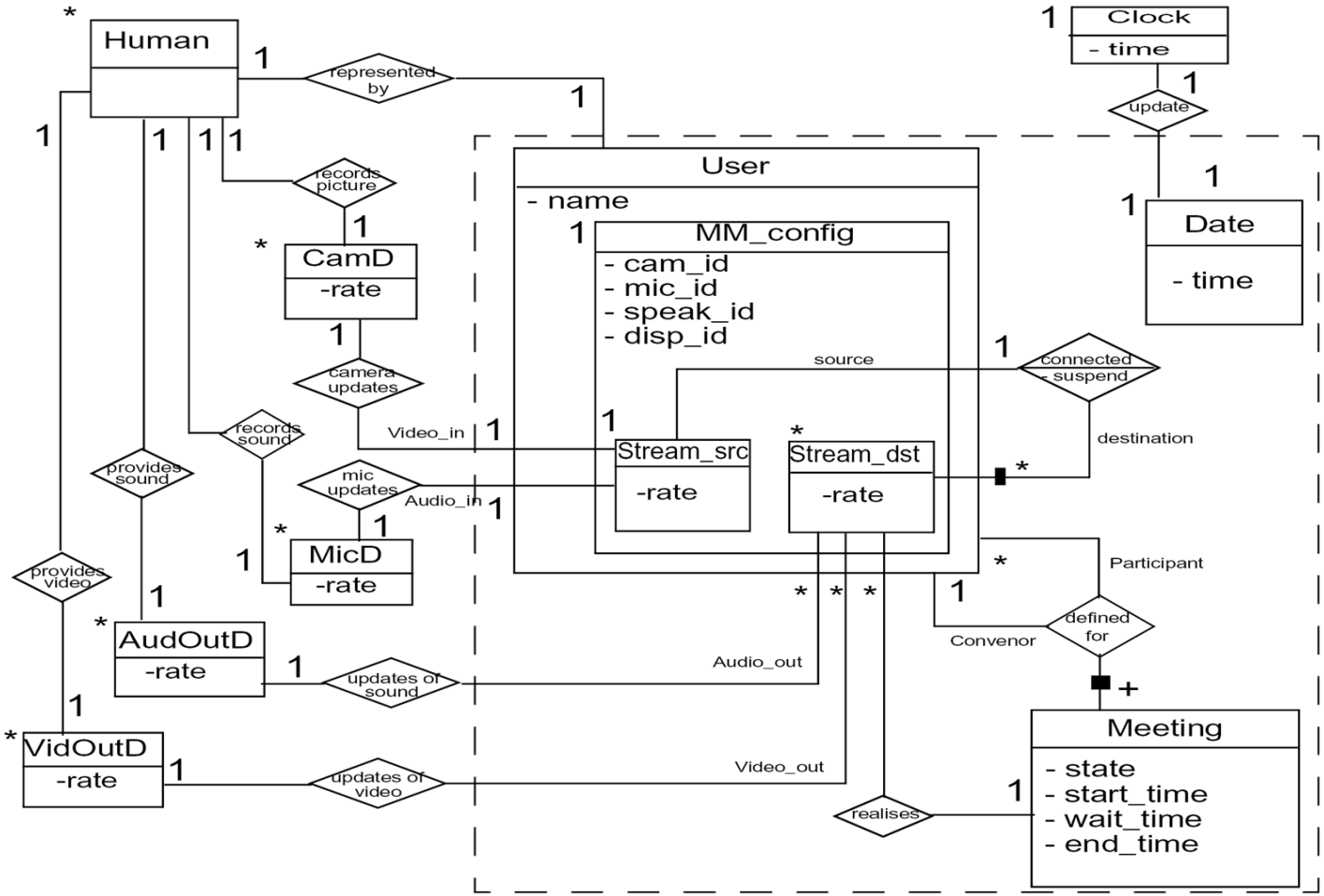
System Interface Diagram



[Coleman et al. 1994]

Fusion Process: Analysis Phase

System Object Model



[Coleman et al. 1994]

Fusion Process: Analysis Phase

- Step 3: Develop the *System Interface Model*, through:
  - Developing the *Life-Cycle Model*, showing the allowable sequences of system operation invocations throughout the lifetime of the system
  - Developing the *Operation Model*, capturing the details of all system operations in *Operation Schemata*

Fusion Process: Analysis Phase

Life-Cycle Model

```
Bank: ((Initialization | Transaction | Enquiry | Finalization)* || Statements)
Initialization = open_account . #account_number
Transaction = withdraw_money . (#withdraw_receipt | #insufficient_funds)
              | deposit_money . #deposit_receipt
              | transfer_money . #transfer_receipt
Enquiry = check_balance . #current_balance
Statements = account_statements . #monthly_statement*
Finalization = close_account . #final_report
```

Life-Cycle Model

life cycle *[Name :] Regular_Expression*

*(LocalName = Regular_Expression)**

Regular_Expressions:Name	Any event name (operation), local name, or output event
Concatenation	x.y
Alternation	x | y
Repetition	
Zero or more	x*
One or more	x+
Optional	
Interleaving	||
Grouping	(x)

[Coleman et al. 1994]

Fusion Process: Analysis Phase

Operation Schema

Operation: reject_meeting
Description: deletes the meeting (in defined state); sent by a Human agent to the system.
Reads: *supplied* name : Name
supplied m : Meeting
current_user : User with name = name
Changes: *destroy* m : Meeting
defined_for
Sends: *multicast* Human :{meeting_cancelled}
Assumes: current_user *in* members(m) *and*
m.state = defined
Result: *forall* u : members(m) @ *multicasts* to Human(u.name) meeting_cancelled(m)

Operation: operation name
Description: description

Reads: values operation can access without modification
Changes: values the operation may access and modify
Sends: agent communication
Assume: preconditions
Result: postconditions

[Coleman et al. 1994]

Fusion Process: Design Phase

- Concerned with finding a strategy for implementing the system specification developed during the analysis phase

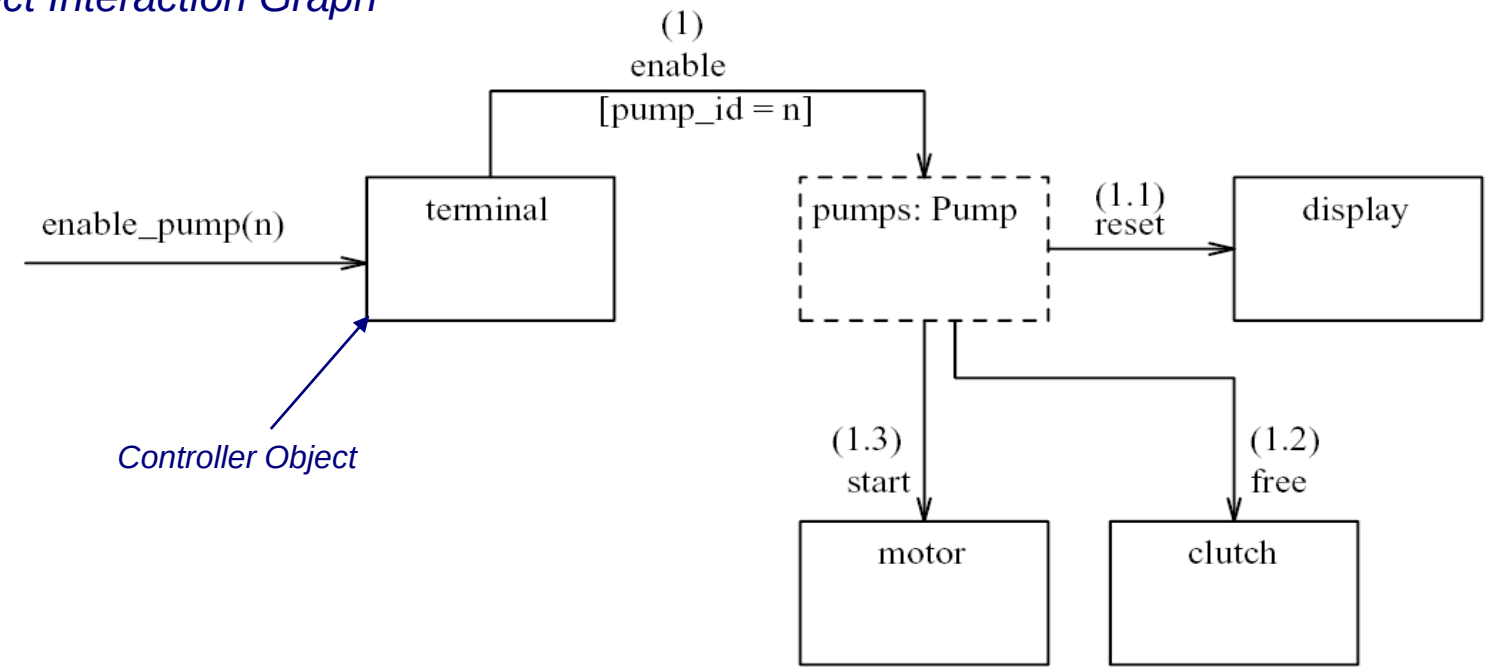
- The output of the design phase consists of four parts:
  - a set of ***Object Interaction Graphs*** describing how objects interact for implementing system operations,
  - a set of ***Visibility Graphs*** describing object communication paths,
  - a set of ***Class Descriptions*** providing detailed descriptions of class interfaces, and
  - a set of ***Inheritance Graphs*** elaborating the inheritance relationships between classes.

- Steps:
  1. Develop *Object Interaction Graphs,* which describe how system operations are implemented through object interactions and message passing
  2. Develop *Visibility Graphs,* which describe *server* objects that *client* objects need to reference, and specify the kind of references that are needed
  3. Specify *Class Descriptions,* which store detailed information about classes
  4. Develop *Inheritance Graphs,* which depict generalization-specialization hierarchies enhanced through factoring out common structure and behaviour

Fusion Process: Design Phase

- Step 1: Develop *Object Interaction Graphs*; each system operation should be realized by an object interaction graph, which describes how the system operation is implemented through object interactions and message passing. One of the objects, termed the *controller*, initiates the message sequence in response to an input event.

Object Interaction Graph

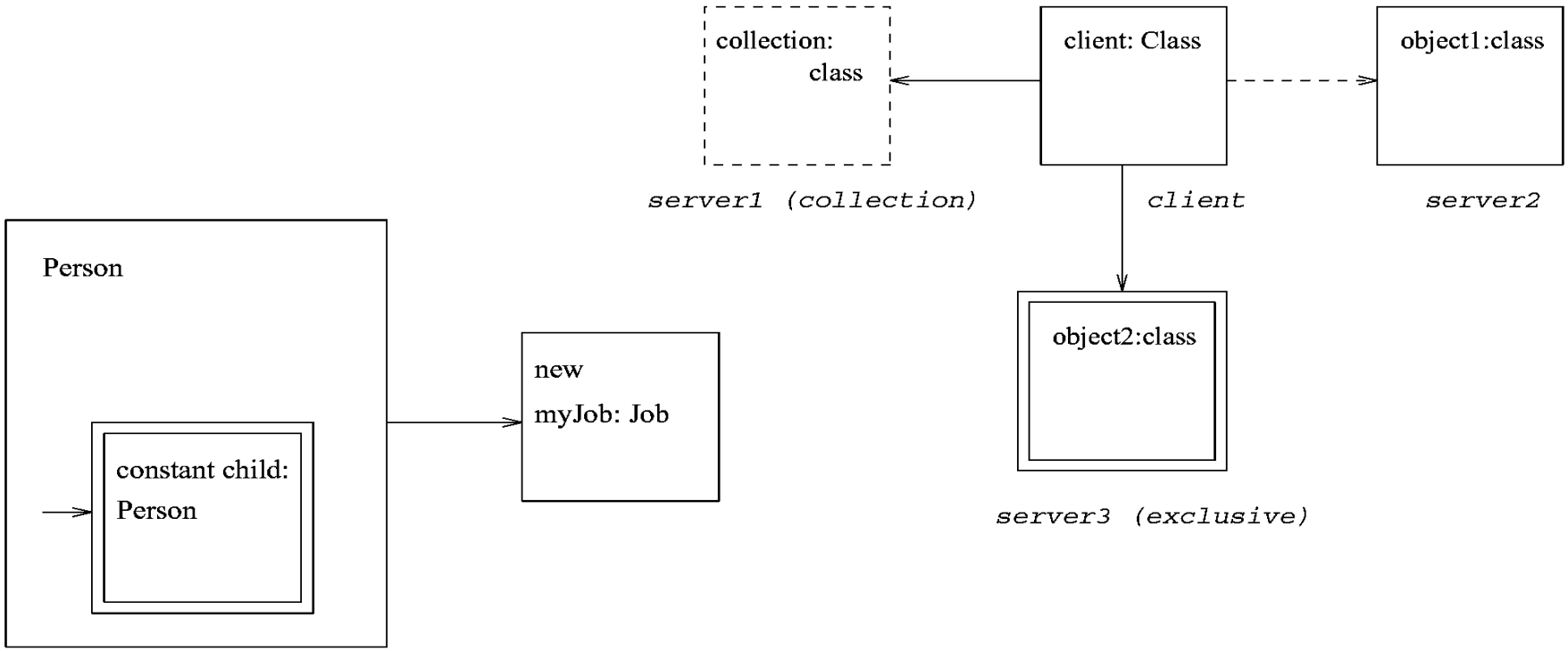


[Coleman et al. 1994]

Fusion Process: Design Phase

- Step 2: Develop *Visibility Graphs*; the visibility of objects is described using the following characteristics: Reference Lifetime (temporary or permanent), Server Visibility (exclusive or shared), Server Binding (the degree of lifetime-dependency between the client and the server), and Reference Mutability (whether a server can be changed).

Visibility Graph



[Coleman et al. 1994]

Fusion Process: Design Phase

- Step 3: Specify *Class Descriptions*; class descriptions store detailed information about classes, including class name, immediate superclasses, attributes, and methods.

Class Description

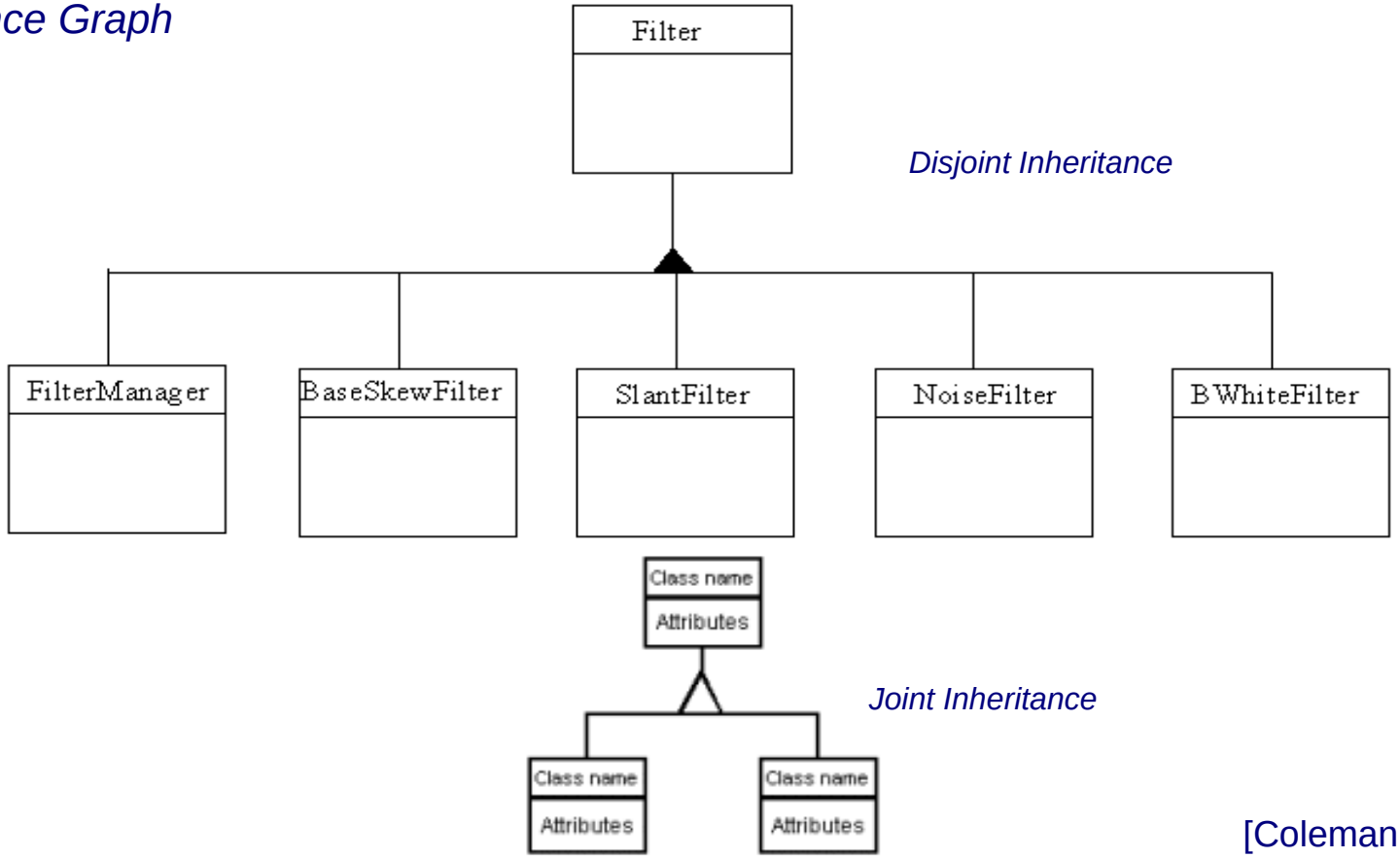
```
class Pump
  attribute pump-id: integer
  attribute status; pump-status
  attribute constant terminal: Terminal
  attribute constant clutch: Clutch
  attribute constant motor: exclusive bound Motor
  attribute constant timer: Timer
  method enable
  method disable
  method is-enabled: Boolean
  method delivery-complete
endclass
```

[Coleman et al. 1994]

Fusion Process: Design Phase

- Step 4: Develop *Inheritance Graphs*; generalization-specialization hierarchies previously identified among analysis classes are enhanced by factoring out common structure and behaviour in order to increase reusability and maintainability.

Inheritance Graph



[Coleman et al. 1994]

Fusion: Strengths and Weaknesses

- ***Strengths***
  - Based on functional, behavioural and structural modeling of the problem-domain and the system
  - Smooth transition from task to task and from phase to phase
  - Traceability to requirements via scenarios of system usage
  - Rich models (structural, functional, and behavioural)
  - Support for formalism
  - Strong functional and behavioural modeling at the system level through identifying detailed scenarios of interaction with the system at the system boundary
  - Extra attention to details of inter-object visibility and the references that objects need to make to each other
  - Detailed inter-object/inter-class models produced during design

Fusion: Strengths and Weaknesses

- ***Weaknesses***
  - Partial coverage of the generic analysis phase: the process starts when a preliminary informal requirements document is already available.
  - Structural model identified during analysis is discontinued in the design phase, with its information broken down and then perfected, thus damaging seamlessness
  - The number of diagrams and other deliverables is prohibitive
  - No modeling of intra-class behaviour
  - No intra-system behavioural and functional modeling during the analysis stage; this has been done intentionally, but nevertheless damages the comprehensiveness of analysis
  - No modeling of physical configuration

Reference

- Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., Jeremaes. P., *Object-Oriented Development: The Fusion Method*. Prentice-Hall, 1994.