Structural Modeling

Gregor v. Bochmann, University of Ottawa

Based on Powerpoint slides by Gunter Mussbacher
with material from:
K.E. Wiegers, D. Leffingwell & D. Widrig, M. Jackson, I.K. Bray, B. Selic,
Volere, Telelogic, D. Damian, S. Somé 2008, and D. Amyot 2008-2009

Table of Content

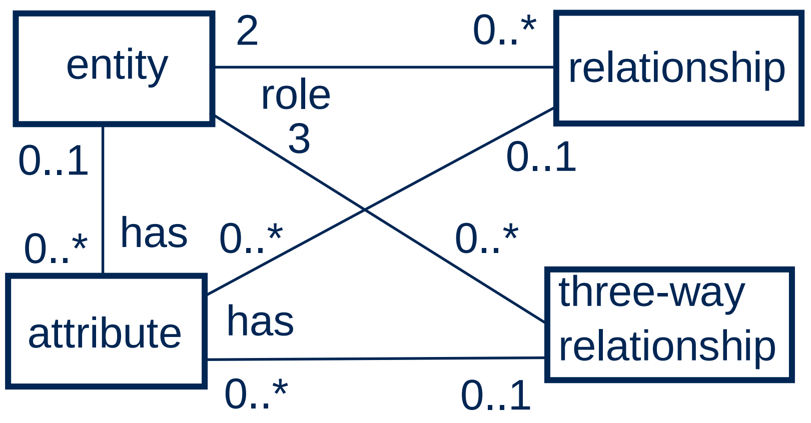
Entity-relationship modeling (ERM)

Entity-Relationship modeling (originally proposed by Peter Chen in 1976)

- Concepts:
  - Entity: represents a type of entity instances, defines the properties that hold for all such instances.
  - Relationship: represents relationship instances that hold between certain pairs of entity instances.
    - The related entity types are also called roles.
    - Multiplicity information indicate how many instances of the "other" side may be related to a given instance of "this" side.
  - Attribute: An entity or a relationship may have one or several attributes. Each attribute is identified by a name and its type, where such a type is usually some simple data type such as integer or character string. Note: An entity type is normally not used as the type of an attribute, because such a situation is rather represented by a relationship between the given entity and the attribute type.

ERM – Meta-model

- Meta-Model (using UML Class Diagram notation)

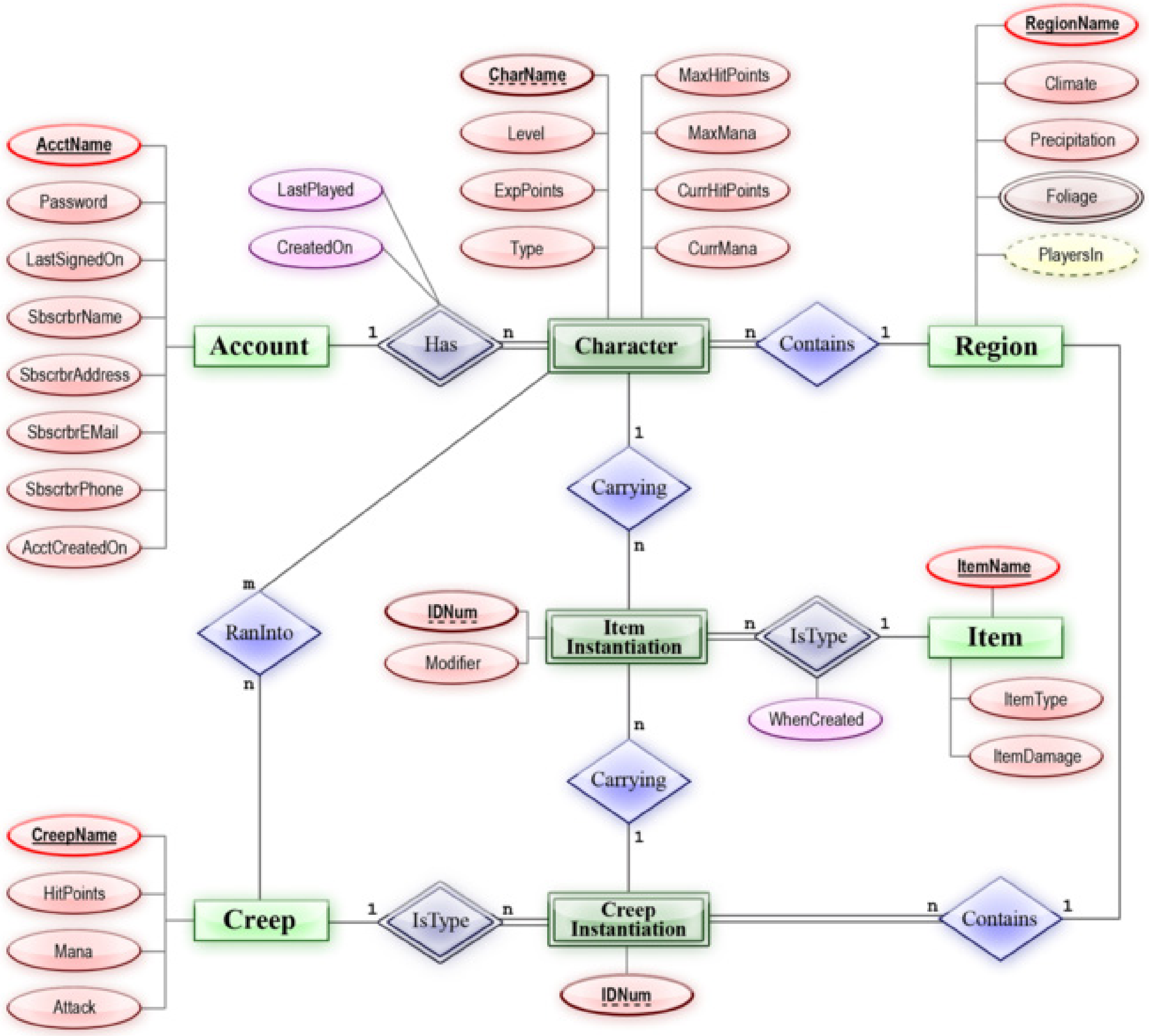


Constraint: each attribute instance is related exactly to one entity, relationship or three-way relationship. (I could not express this fact by the multiplicity notation)

has

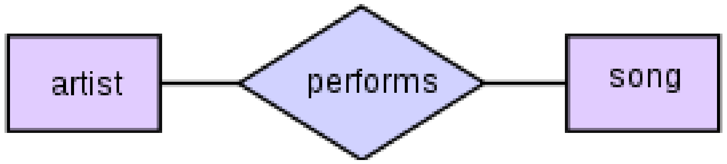
ERM – Notations (1)

- There are a variety of notations that have been used for ERM
- Chen's notation

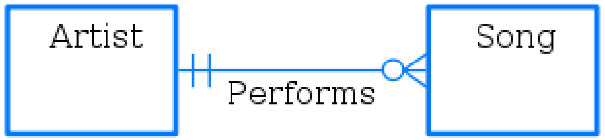


ERM – Notations (2)

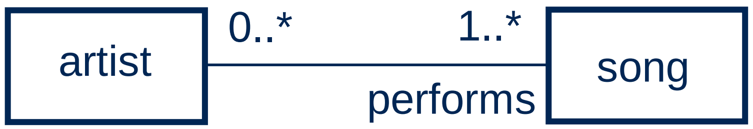
- “normal” ERM notation



- So-called Crow’s foot notation (showing multiplicity)



- UML Class Diagram notation
  (note: the notation for multiplicities in UML are more powerful than other notations)



- Note: the notations are different, but the concepts are the same.

Object-oriented modeling (OOM) – (1)

Object-oriented modeling is essentially ERM with certain additional concepts. An Entity is called a Class

- Again, various notations have been introduced. We will use in the following the notation of UML Class Diagrams.
- Additional Concepts:
  - Inheritance: This is the idea that some entity B inherits all properties that are defined for another entity A. This means that B is a specialization of A. One also says that B is a refinement of A or B extends A.
    - Important note: Inheritance is not a relationship as defined above, since it does not define relationship instances between the instances of the two entities.
  - Internal state of entity instances and methods for interacting with it: An important issue of object-orientation is information hiding. In particular, certain internal attributes are not directly accessible from outside the entity instance. An entity is characterized by its interface which includes the list of accessible attributes and the list of methods that can be called for manipulating the internal state of the instance. Note: The methods have behavior – the rest is structure.

Object-oriented modeling (OOM) – (2)

- Refinement of ERM Concepts:
  - Composite structure: A subtype of Relationship (with special notation) is used to show that one entity is part of another (composed) entity.
  - Calling relationship: Another subtype of Relationship has the meaning that an entity instance playing one role can access attributes and call methods on an instance playing the other role to which it is related.
- Disadvantages of OOM
  - Difficulties of modeling two-way communication : an interface defines only one direction of communication. For event-based systems, one often needs two-way communication relationships. Note: one can use two one-way relationships.
  - The encapsulation of the behavior inside the objects (the information hiding approach) is often not suitable for modeling the problem domain during requirements engineering (see examples below).

Object-oriented modeling (OOM) – History

- Proposed approaches for ”OO Analysis”
  - Shlaer and Mellor (1988)
  - Colbert (1989)
  - Coad and Yourdon (1989)
  - Wirf-Brock (1990)
  - Rumbaugh (1991)
  - Jacobson (1992)
  - Martin-Odell (1992)
  - Rational Unified Process (RUP) (1998)

- UML unifies many of these notations

Methodology for Object-Oriented Analysis (OOA)

- Five main steps
  - Identify core classes within problem domain
  - Model relationships between classes
    - Class diagram
  - Define the attributes associated with each class
  - Determine relevant operations for each class
  - Define the messages that may be passed between objects
    - Interaction diagram, state machine diagram

OOA Methodology – Library Example (1)

- A library system is intended to provide its users with the ability to automate the process of:
  - Acquiring library items
  - Cataloguing library items
  - Browsing library items
  - Loaning library items
- Library items comprise published and recorded material
- The system will be administered by a member of the library staff
- Users must register with the system administrator before they can borrow library items

Source: Sommerville & Kotonya

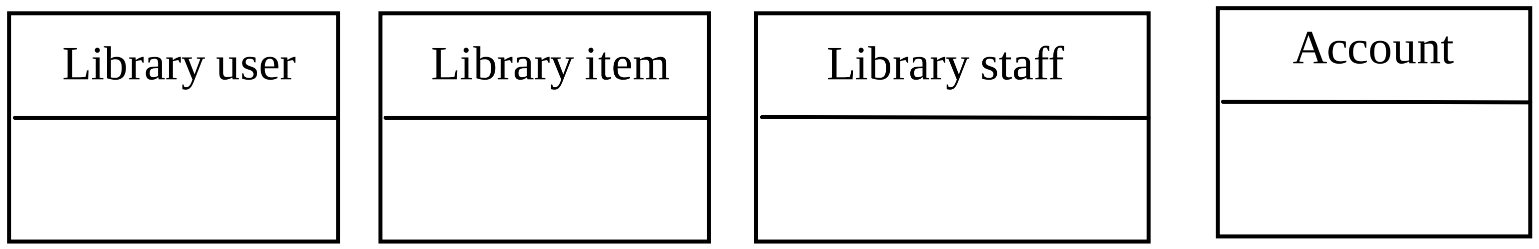
OOA Methodology – Library Example (2)

- Library users are drawn from three primary groups
  - Students, Members of staff, and External users
- All library users have as part of their registration
  - Name, Library number, Address, Account
- In addition the following information is also required for registration
  - Students – *Degree programme and admission number*
  - Staff – *Staff number*
  - External users – *Employer details*

Source: Sommerville & Kotonya

OOA Methodology – Library Example – Step 1

- Identify initial classes

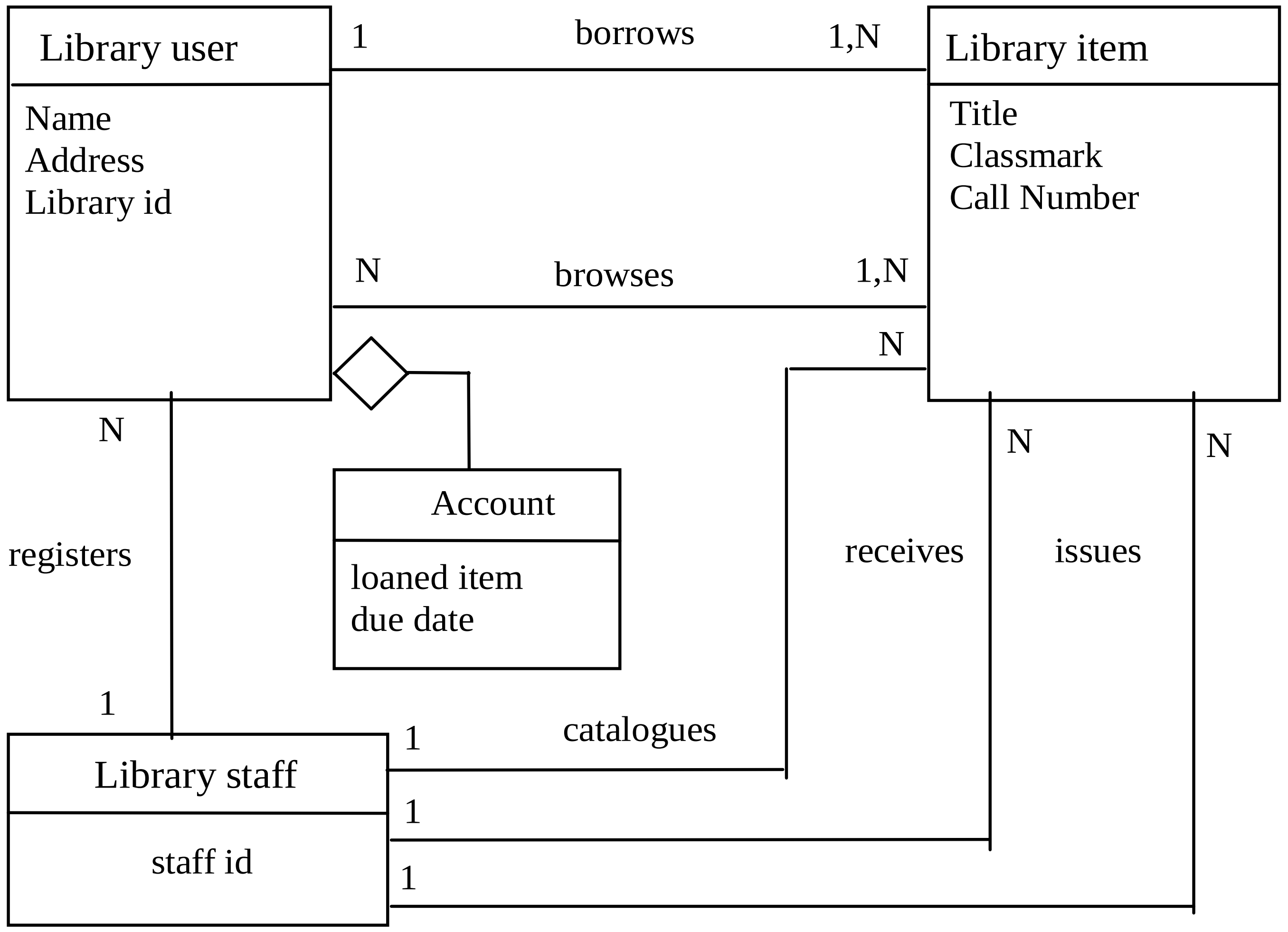


OOA Methodology – Library Example – Step 2

- Identify relationships between classes from the partial requirements
  - (i) A library user borrows a library item
  - (ii) A library item is recorded or published
  - (iii) The system administrator registers the library user
  - (iv) Library users are students, staff, and external users
  - (v) The system administrator catalogues the library items
  - (vi) The library assistant issues the library items

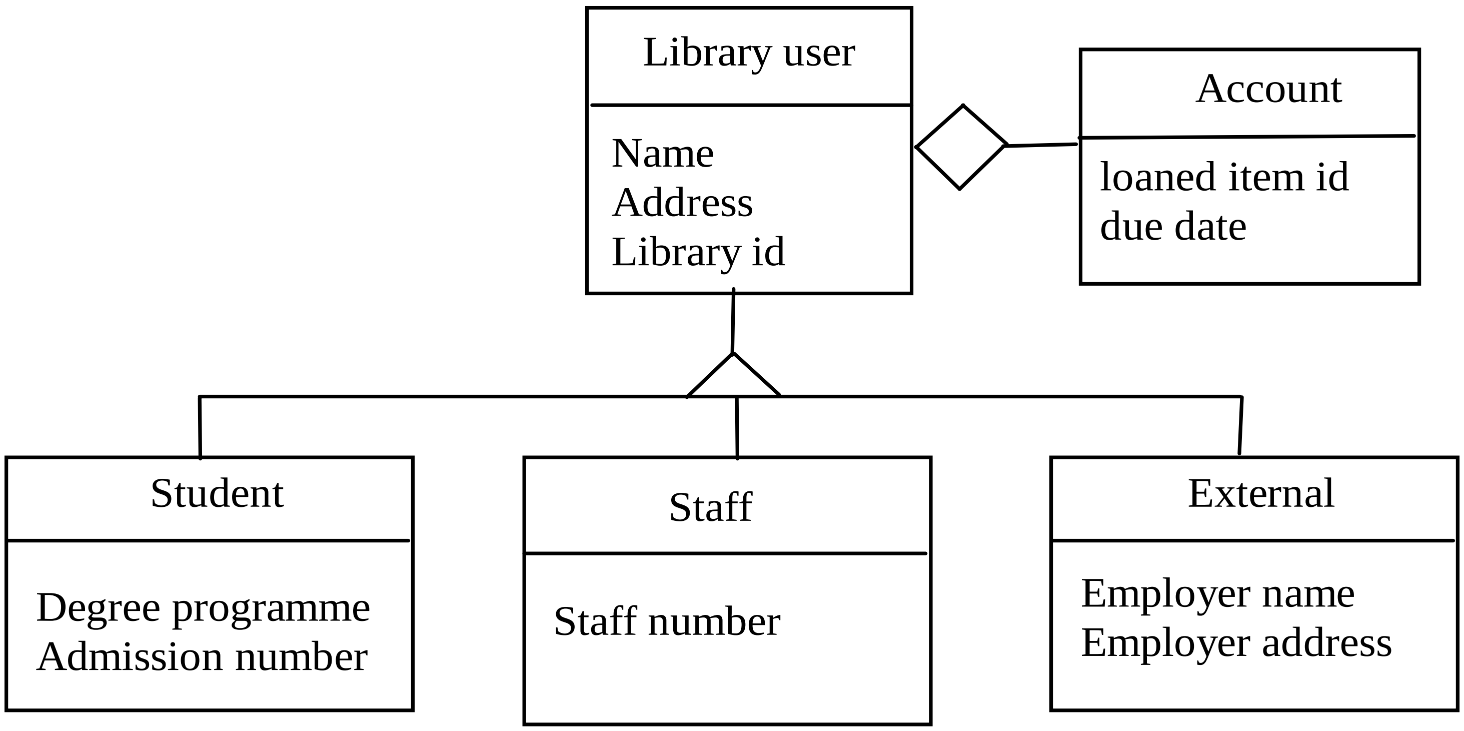
OOA Methodology – Library Example – Step 2 (2)

- Show attributes and relationships in basic model



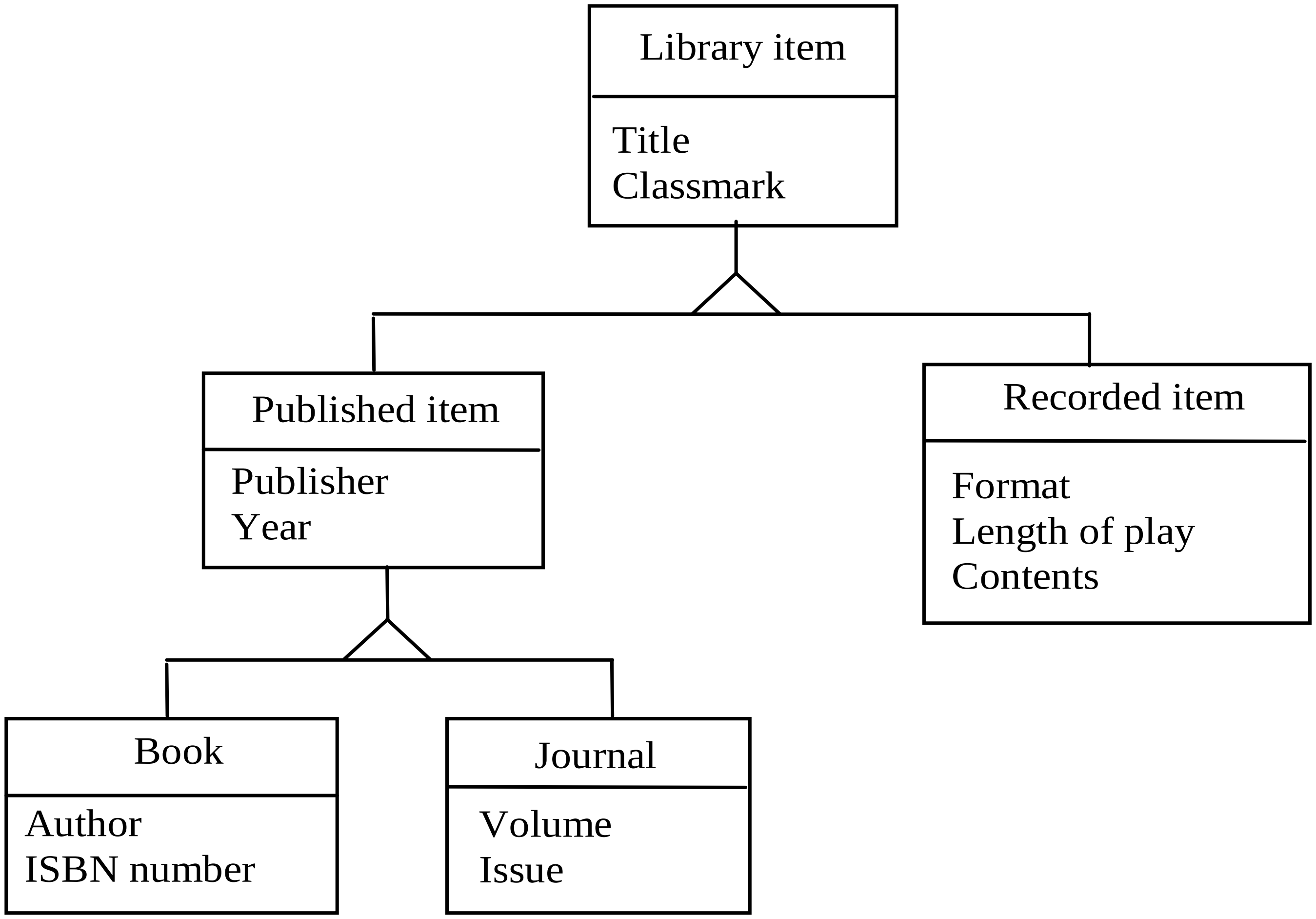
OOA Methodology – Library Example – Step 2 (3)

- Identify inheritance relationships for library user



OOA Methodology – Library Example – Step 2 (4)

- Identify inheritance relationships for library item



OOA Methodology – Library Example – Step 3

- Identify attributes and populate model with them

- Attributes can be revealed by the analysis of the system requirements
- For example, it is a requirement that all library users must be registered before they can use the library
  - This means that we need to keep registration data about library users
  - Library users are also provided with an account to keep track of the items loaned to them
- Library items may have the attributes title, description, and classmark
- Library users may have the attributes name, address, and library id

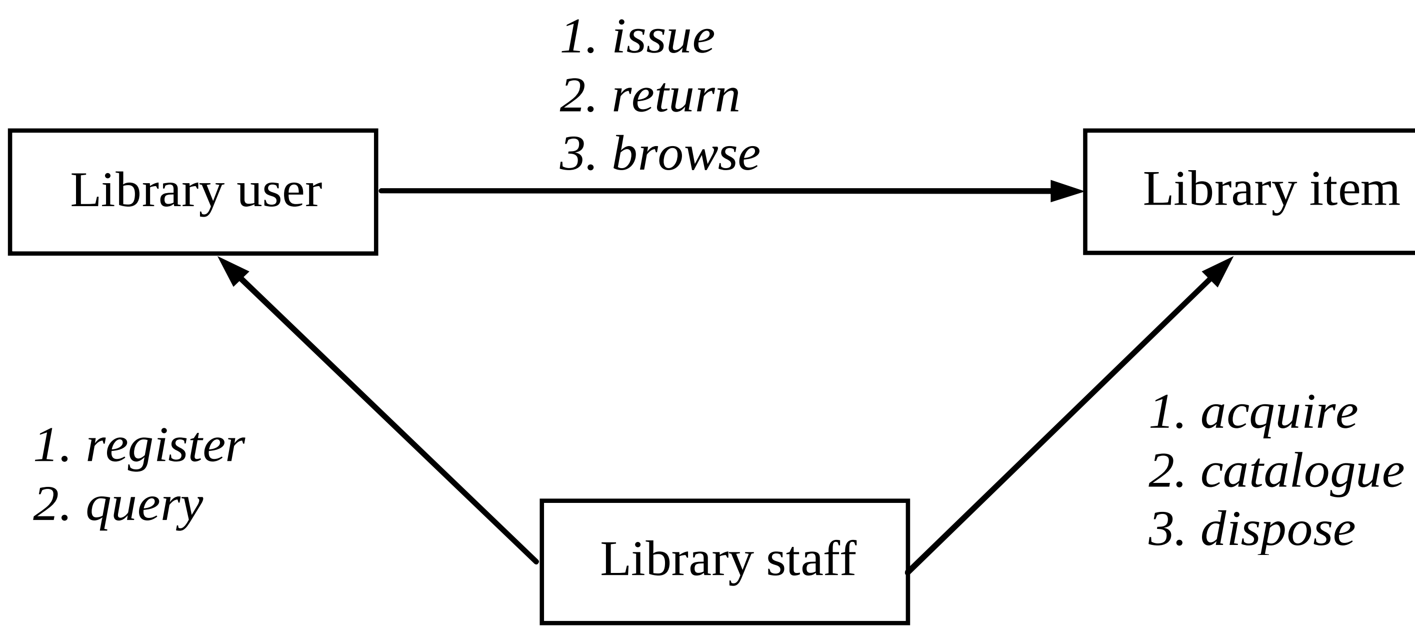
OOA Methodology – Library Example – Step 4

- Identify object operations

- This step is intended to describe operations to be performed on the objects
- Certain operations are implicit from the object structure
  - CRUD operations (create – read – update – delete)
  - Operations for accessing and modifying the attribute values (getters and setters)
  - These operations are assumed and we need not show them explicitly in the model
- One way of identifying operations is by modeling the messages that may be passed between the objects

OOA Methodology – Library Example – Step 4 (2)

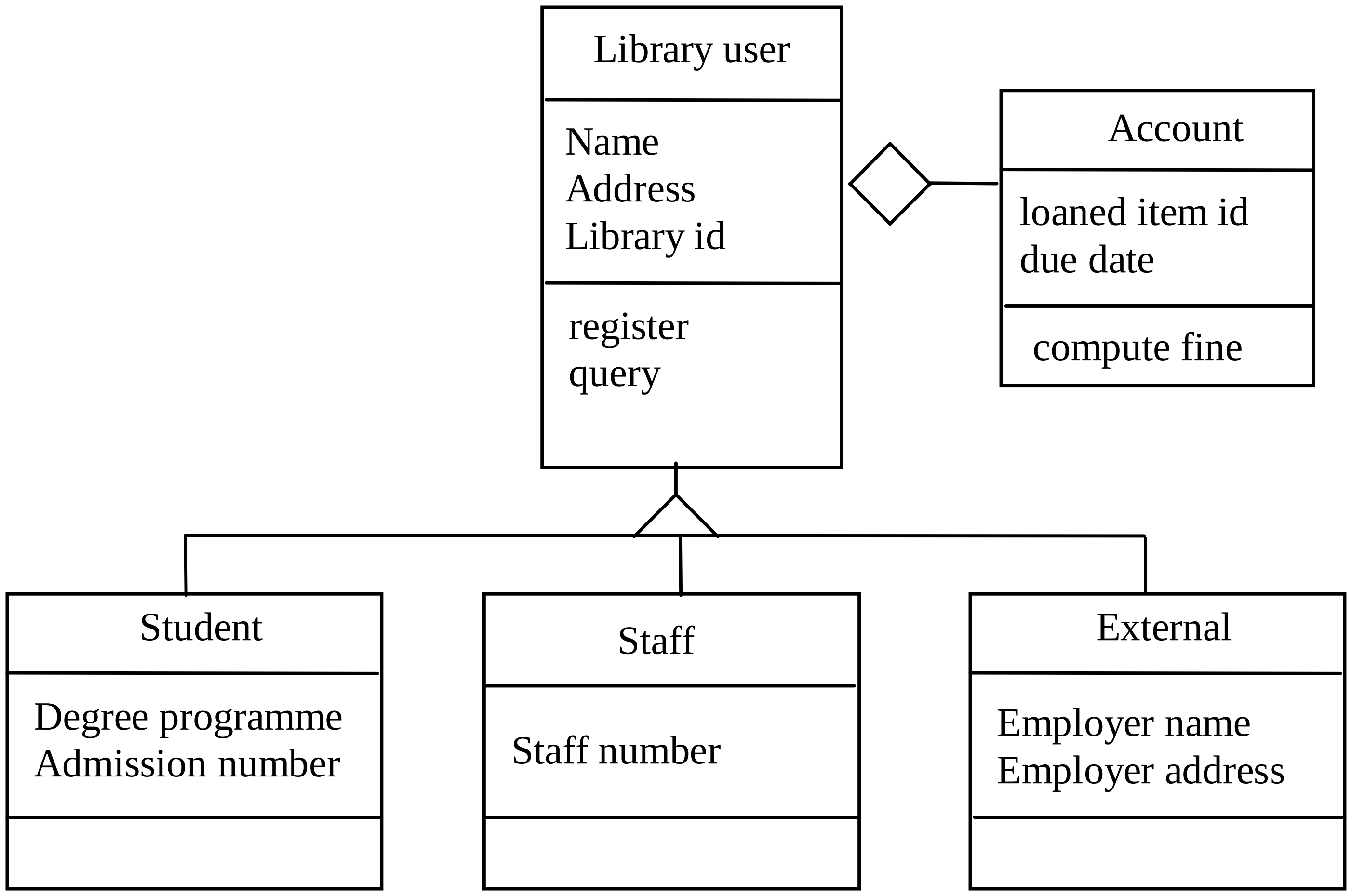
- Identify messages between objects



- Find required messages for each scenario (play out the scenario), then take union of all messages

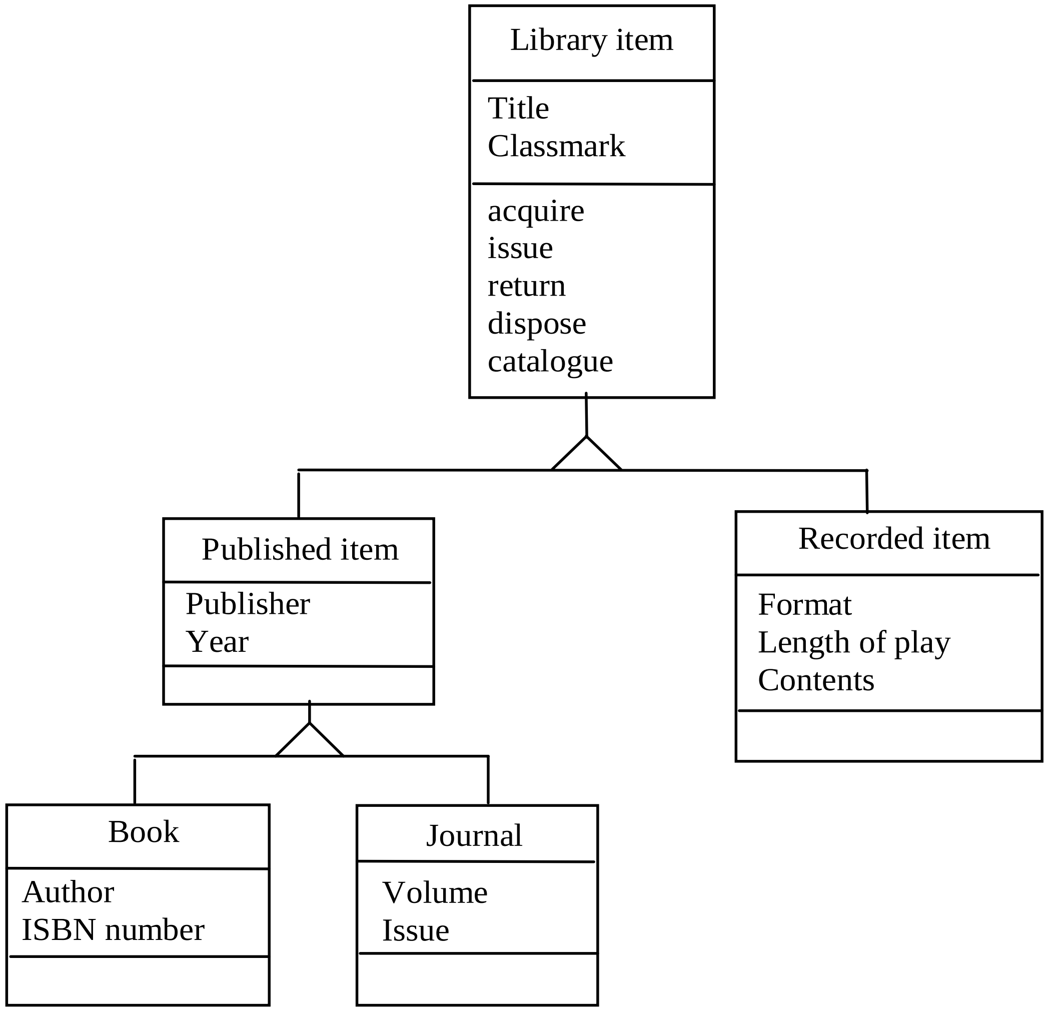
OOA Methodology – Library Example – Step 4 (3)

- Populate model of library user with discovered operations



OOA Methodology – Library Example – Step 4 (4)

- Populate model of library item with discovered operations

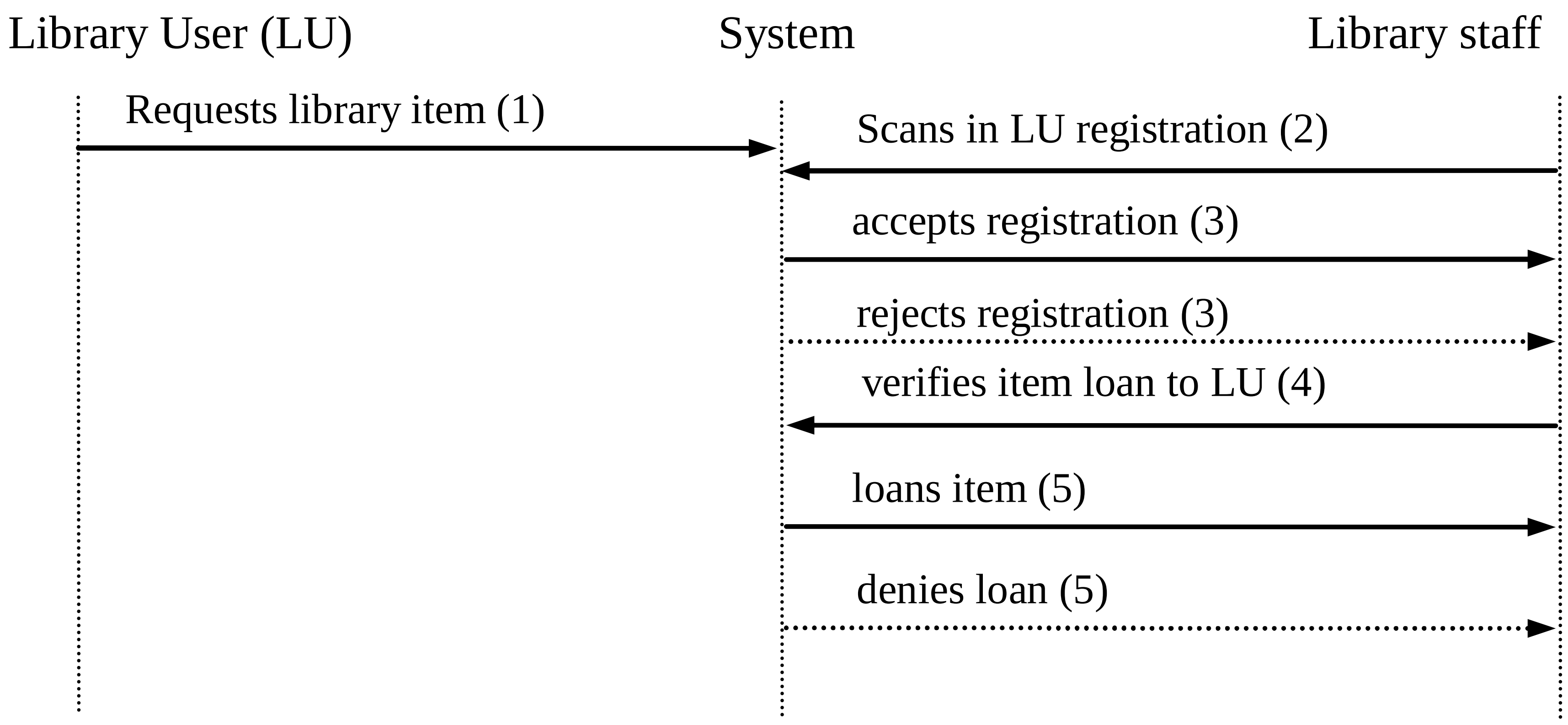


Note: This makes no sense as a model of the problem domain. How can a book (library item) perform the method acquire or return ?

It may, however, make sense as the internal design of the system-to-be. In this case the objects are instances within the computer system that should reflect the objects in the real world.

OOA Methodology – Library Example – Step 5

- Define the messages that may be passed between objects

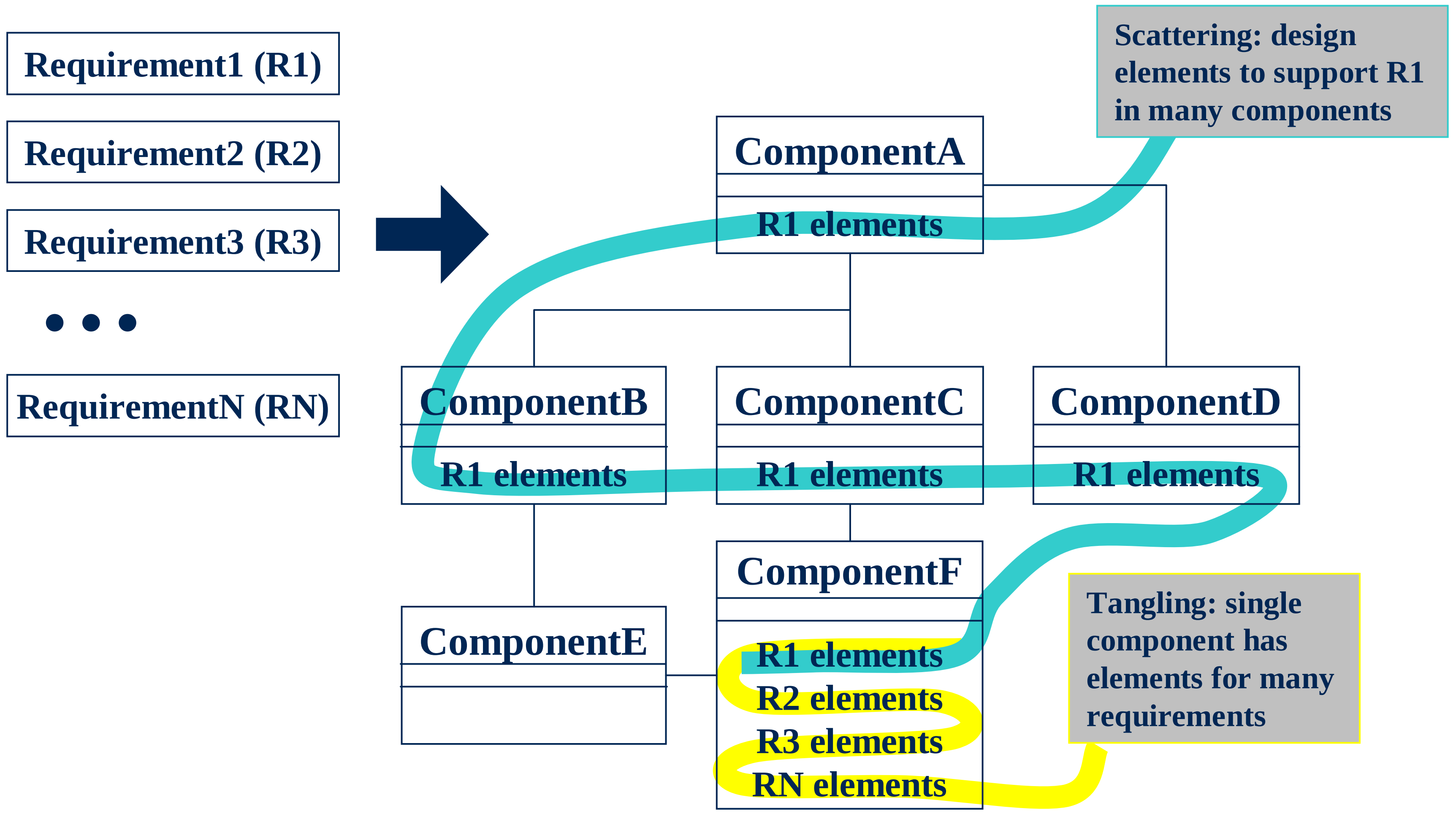


OO Analysis – Problems (1)

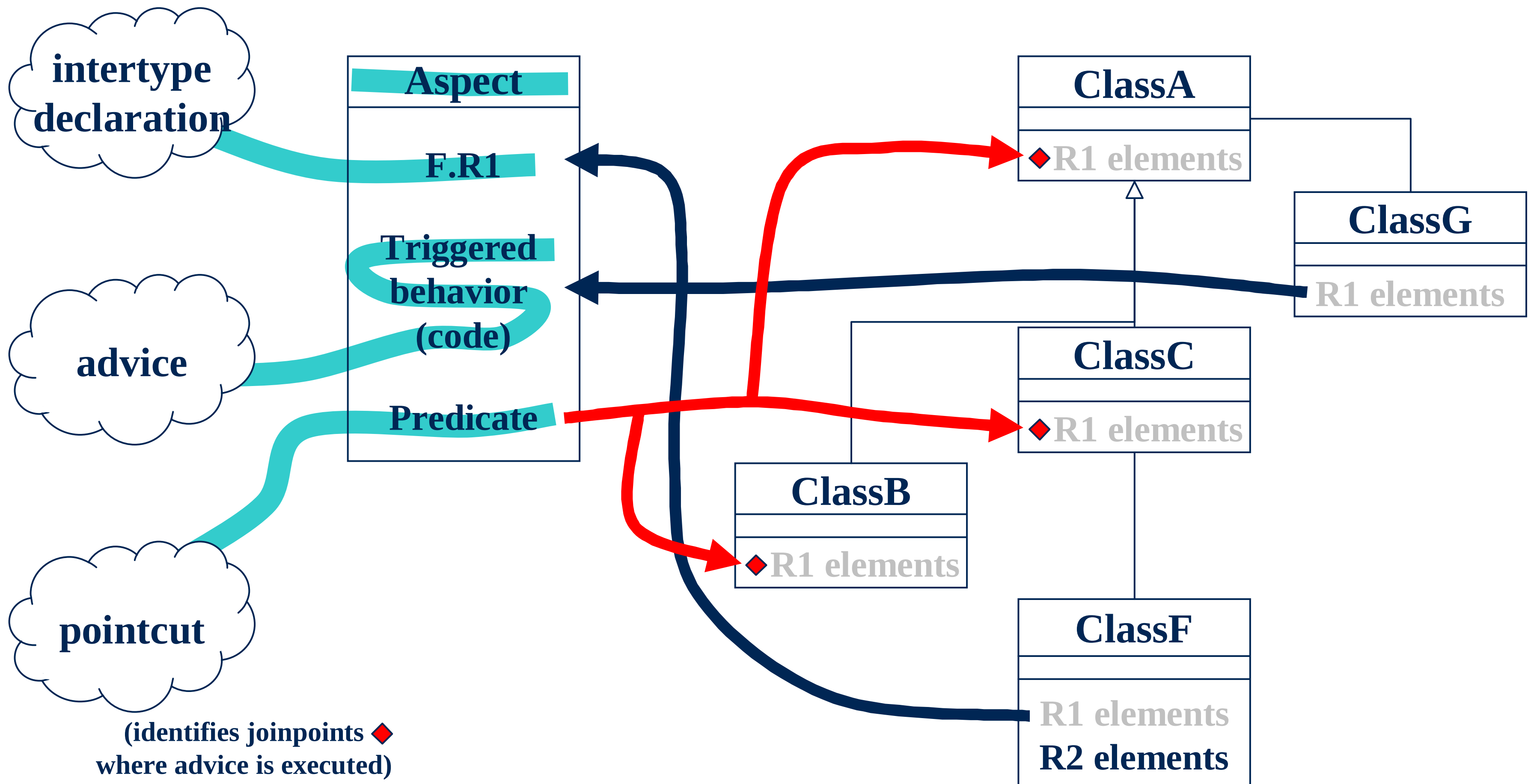
- Caution: Not really analysis
  - Most OOA approaches actually address high-level design
  - Assume a pre-existing requirements document
  - Class diagrams can however be used for analysis, especially for the description of domain concepts
- Use case analysis supplements OOA, filling in some gaps
- Further composition and decomposition problems
  - Related requirements cannot all be assigned to a single component or a single class
  - One scenario may affect several classes at once
  - OO modularization is not perfect either...

    ➔ Scattering and tangling effects - Motivation for aspect-oriented analysis and design

OO Analysis – Problems (2)



A partial solution – Aspects



Terminology based on AspectJ: www.eclipse.org/aspectj