

Modeling with UML (1.4)

Class Model Diagrams

- UML class model diagrams are commonly used to represent the structural aspects of system design problems
- A class diagram consists of a collection of object classes and the relationships among them

Classes

- A *class* is a distinct object type participating in the system being built
- A common noun in English often indicates an object type
- A class is represented by a rectangular box, possibly partitioned into three parts vertically (i.e. separated with horizontal lines)
 - Class name
 - Attributes
 - Operations

Class Features

- Classes have *features* (attributes and operations)
- While objects and classes exist in the real world or in the mind of the designer, features exist inside of the computer
- An *attribute* is a property of a class
 - Typically, attributes have types that correspond to primitive or composite data types available on the computer
- An *operation* (method) is a service provided by an object. It may take typed parameters and possibly return a typed value

Relationships

- Relationships exist among classes, and they are represented by lines connecting the related classes
- UML class model diagrams have three kinds of relationships
 - Generalization
 - Dependency
 - Association

Library Information System

- This exercise asks you to create a UML class diagram that models the problem of managing the information resources for a library
 - That is, devise an analysis model
- Assume that somebody else will be designing the program from your analysis
 - Include classes, their attributes and operations and the relationships among them
 - Indicate attribute types, cardinality of associations, generalization and aggregation relationships

Library Problem Requirements

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction—they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot go higher than the value of the overdue item.
9. The library also has reference books and magazines, which cannot be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item exactly once—unless there is an outstanding request for the item, in which case the patron must return it.

Linguistic Analysis

- One approach to dealing with a problem such as this is to linguistically analyze the text of the problem description
 - This approach was developed independently by Abbott and by Booch
- In particular, nouns will map to classes, active verbs will map to operations, certain stative verbs will map to relationships, and adjectives and adverbs will map to attributes

Nouns in Requirements

- One approach to beginning the modeling process is to review the requirements looking for nouns as candidate classes.
- When we do that for the library problem, we get the following possibilities

patron, library card, system, library, name, address, phone number, time, item, fine, libraryCardNumber, child, restriction, book, AVMaterial, week, bestSeller, limit, day, cent, value, reference book, magazine, request, age

Candidate Classes

- From these nouns, with can identify some that are good candidates for classes.
 - We may have to merge some synonyms in doing this

Patron, Child, Item, Book, LibraryCard,
System, Library, AVMaterial, ReferenceBook,
Magazine, BestSeller, Fine, Request

Remaining Nouns to be Considered

name, address, phone number, time, fine,
libraryCardNumber, restriction, week, limit, day,
cent, value, age

Attributes

- Some of the remaining nouns map naturally to attributes of the identified classes
 - name, address, phoneNumber (Patron)
 - age (Child)
 - libraryCardNumber (LibraryCard)

Remaining Nouns to be Considered

time, fine, restriction, week, limit, day, cent, value

Utility Classes

- Some nouns belong to classes, but the classes are more general than just this one problem
 - We can imagine them belong to a library

Date: time, week, day, limit

Money: fine, cent, value

Remaining Noun to be Considered

restriction – deferred until we have a better understanding of the problem

Inferred Attributes

- Sometimes we can identify missing elements of our model and introduce model elements for them
- For example, we can infer some missing attributes from the requirements:
 - `dueDate : Date (Item)`
- It is a natural part of an analyst's job to point out missing, confusing or incorrect parts of an initial requirements document

Operations

- We can perform a similar analysis on verbs
- Active verbs often indicate operations that we can assign to classes
 - query of the `itemsCurrentlyCheckedOut`
 - query of `whenDue`
 - query to `computeOverdueFines`
 - `checkOut`
 - `request`
 - `renew`
 - `return`

Stative Verb

- Another category of verb is called
- Verbs of this type descriptive rather than indicating actions
- For example, the stative verb phrase *is a kind of* or just *is a* indicates a generalization relationship
- Likewise, *is a part of*, *comprises*, or *has a* are indicative of aggregation associations

Associations

- Other stative verbs may indicate general associations
- For example, the idea of a patron checking out an item from a library suggests an association between `Patron` and `Item`
- Hence, it makes sense to treat it as an association (`CheckedOut`)
- Likewise for `Request`

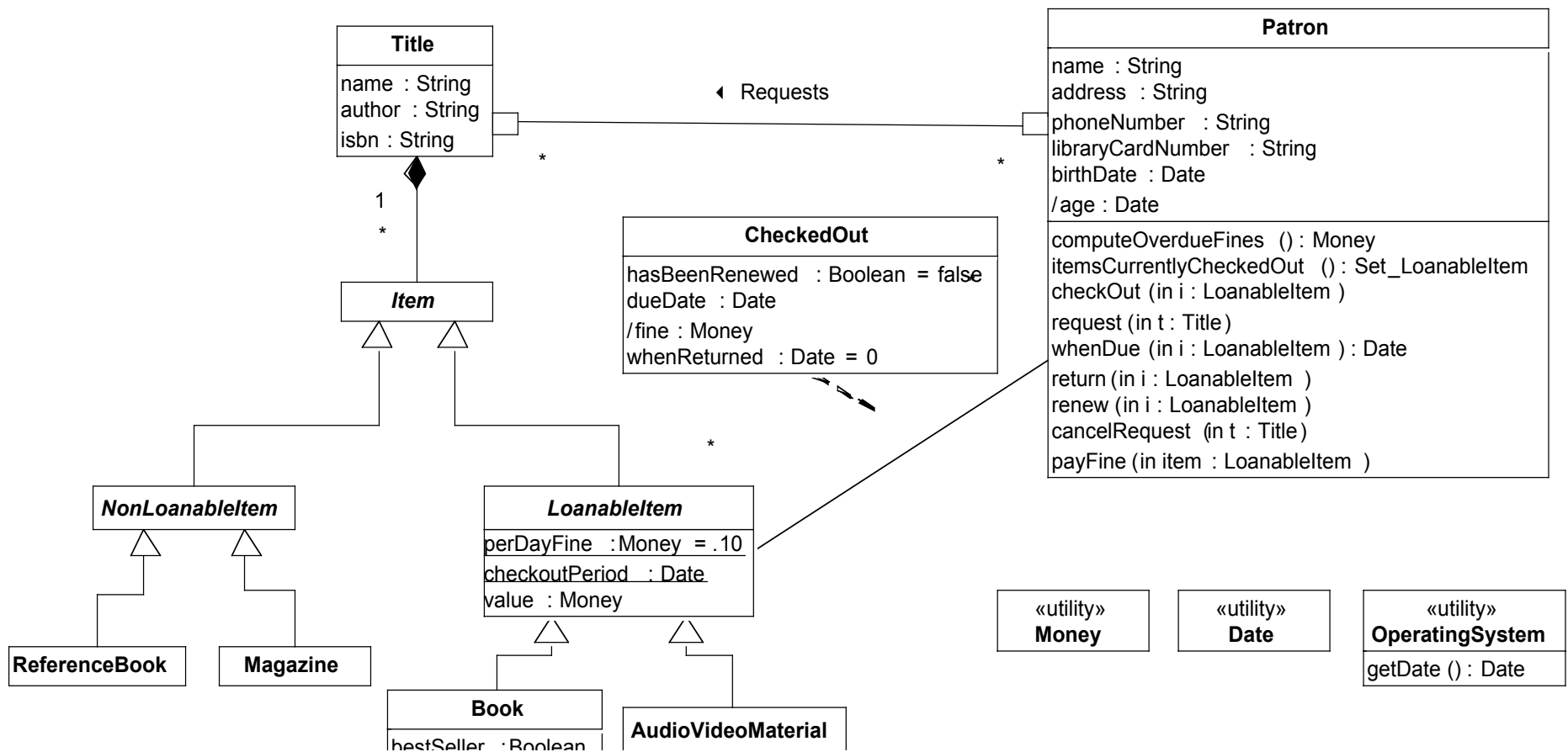
Aside: Scenarios

- A different approach than linguistic analysis is to write use *usage scenarios (use cases, user stories)*
- The scenarios can then be used to generate CRC (class-responsibility-collaborator) cards
 - E.g. an actor in a scenario becomes a candidate class
- Sample library scenarios include the following
 - A patron checks out 3 books and 1 a/v material
 - A patron attempts to renew a book
 - A librarian runs a report of all patrons with outstanding overdue fines
 - A patron attempts to request a book
 - A book is returned
 - A child attempts to check out 7 books

Missing Requirements

- In performing this analysis, other, unstated requirements may surface that should be confirmed with the customer
 - The system should provide operations for the `Patron` to make or cancel a request
 - The system should provide an operation for a `Patron` to renew an `Item`
 - The system should allow a `Patron` to pay a fine
 - The system should be able to determine whether or not an `Item` can be checked out
 - The system should be able to determine whether an `Item` has been renewed

Possible Class Model Diagram



Notes on the Diagram

- Diamonds on association ends
 - Composition (filled-in diamond): if a `Title` goes away, so does its `Items`
 - Aggregation (open diamond): a grouping relationship that doesn't imply linked lifetimes
- Class attributes (underlined): properties of the class as a whole
- Qualified associations (rectangular association end): indexes instances
- Cardinality: number of instances at one end that correspond to a single instance at another end
- Derived attributes (slash): computed rather than stored
- Stereotypes (`<<name>>`): metamodel types
- Default values of attributes (assignment)
- Reading direction (`<|` on `Requests`)

Issues Raised

1. When does a `CheckedOut` link get deleted?

Issues Raised

1. When does a `CheckedOut` link get deleted?
 - That is, a `Patron` can return an overdue `Item` but delay paying the fine. Then, the link has to remain, but no additional fine need accrue
 - This implies one or more attributes on `CheckedOut`
 - Whether the `Item` has been returned (`boolean`) and the date of the return (`Date`)
 - Alternatively, a special `Date` (0) could indicate that the `Item` has not yet been returned
 - Associations with features are called *association classes*

Issue

2. The stated requirements really describe a *library information system* and not a library
 - Hence, there is likely no `Library` class
 - But there may be a `LibraryInformationSystem` class that contains the other classes
 - Implicit in the diagram
 - It could also be modeled as the name of a package that contains the classes in the diagram
 - Such an approach would be better expressed by replacing `Patron` with `PatronRecord`, etc.
 - This conversion is part of the design process, not the analysis process

Issue

3. `LibraryCard` has no independent role
 - **Just include** `LibraryCardNumber` as an attribute of `Patron`

Issue

4. Should `Child` be a subclass of `Patron` or be determined computationally?

Issue

4. Should `Child` be a subclass of `Patron` or be determined computationally?
 - If a separate class is used, then a given instance would have to change its class when the corresponding `Child` reached its thirteenth birthday
 - If *childness* is computed, then we have to have a operation for determining what the current `Date` is
 - A `System` class with a `getDate()` operation
 - This means that `age` becomes a derived attribute, and we need a `birthDate` attribute for `Patron`

Issue

- 5. ReferenceBooks and Magazines have no features other than that they cannot be checked out
 - This raises the question of whether they couldn't be modeled with a `boolean canBeCheckedOut` attribute of `Item`
 - Leaving them as classes allows the customer to specify more properties and enables future growth

Issue

6. Whether a `Book` is a best seller, however, has been modeled with an attribute. It could have instead been another subclass
 - Its only current use is in computing the length of the checkout period
 - So an attribute was used instead of a subclass

Issue

7. Need operations for creating
Items **and** Patrons
 - Constructors need to be added to
the diagram

Issue

8. Is it necessary to have separate **classes** for `AVMaterial`, `ReferenceBook`, `Magazine`, `BestSeller`?

Issue

9. It would appear that *time*, *week*, *limit* and *day* are related. In a situation such as this, it is the analyst's job to invent a class (`Date`) that is not explicit in the supporting documents
 - Likewise for `Money` subsuming *cent*, *value*, and *fine*

Issue

10. What is it that a Patron is requesting when they make a Request?

Issue

10. What is it that a Patron is requesting when they make a Request?

- Patrons don't request Items! They really request any one of a set of Items with the same title
- Hence, there is a missing concept in the diagrams, a Title
- That is, a Title is an aggregation of Items that share certain properties

Issue

11. As modeled, a request for a `Book` (e.g. *A Perfect Storm*) can be satisfied with `AudioVideoMaterial` (the corresponding movie). This is likely not what is intended
 - This could be handled by making `Request` an association class and adding an enumeration attribute to distinguish the book from the movie

Issue

12. As modeled, a user can request a

`NonLoanableItem`

- Title **should contain only**
`LoanableItems`

Other UML Issues

- UML supports multiple inheritance, which has not been illustrated in this diagram
- UML encourages the use of role names for the ends of association, which don't appear in the diagram

OCL and Constraints

- If we look back at the original requirements, we note that many of them are not satisfactorily addressed by the class model diagram
- For example, the various rules for the length of time which an `Item` can be checked out are not specified
- UML has a textual notation called OCL for expressing these kinds of constraints

Requirement 6

Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks

- This property can be expressed with an OCL constraint (called an *invariant*) on instances of class `Book`
- A constraint can refer to the values of `Book`'s attributes
- It states a rule that must hold for all instances

```
context Book inv:  
  if bestSeller then  
    checkoutPeriod = 2 -- weeks  
  else  
    checkoutPeriod = 3  
  endif
```

Conclusions

- Requirements analysis is not easy. The process of modeling caused us to think through many issues that the requirements did not make clear
- Some new concepts, like `Title`, may have to be introduced
- New operations may need to be invented
- A class model diagram alone is not sufficient for modeling
- There may be more than one correct answer