1. Case Study: Making an Object Event Model of a Public Library

We consider the case of a public library as an example of a business system, for which we make both an OES model and an IS model for investigating the research questions posed in the Introduction.

1.1. Making an IS Model

As a result of requirements and domain analysis, we have obtained the conceptual information model shown in Figure 6, describing 9 entity types with attributes and associations.

The 9 entity types described by this model include 3 potential event types: book pickup notifications, book lendings and book returns. However, from an IS modeling point of view, these entity types may also be viewed as types of objects that represent documents recording events. This interpretation (of events as document objects) is, in fact, supported by the presence of date/time attributes in these entity types and by the *many* multiplicity in their participation associations with object types.

Observation: Since, ontologically, ***objects participate in events***, the associations between object types and event types are *participation associations*.

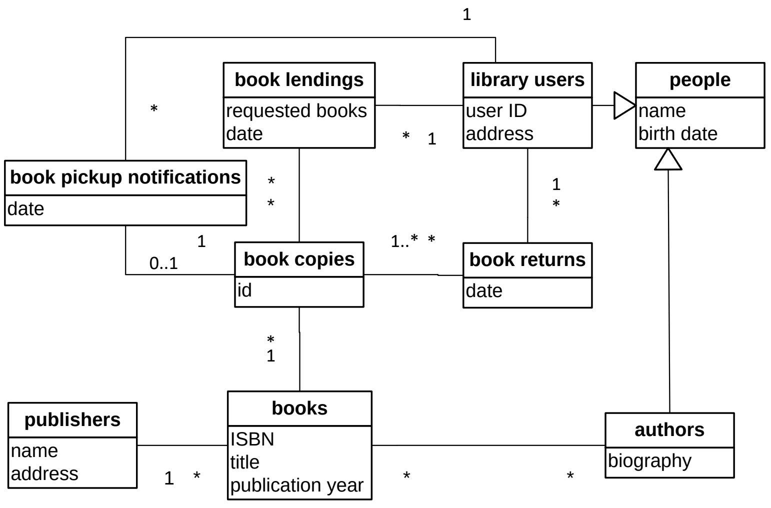

Figure 1: A conceptual information model for an IS for the public library.

An information design model for an IS for the public library based on the conceptual information model of Figure 6 is shown in Figure 7.

Notice that while historic lending events are stored in the population of *BookLending*, current lendings, which still need to be completed, are recorded in the multivalued *Book* attribute *lendings*. This allows the IS to periodically check if users have to be reminded to return lended books. The multivalued *Book* attribute *reservations* allows to check if a returned book copy has to get the status RESERVED and the corresponding user has to be notified to pick up the reserved book.

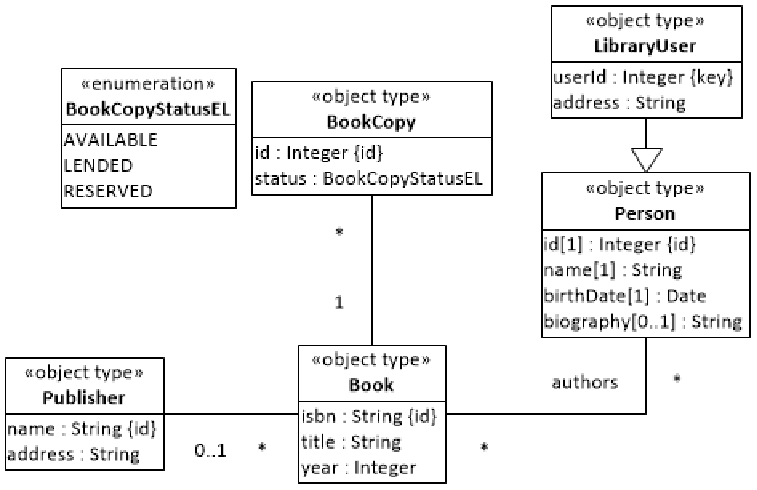

Figure 2: A basic information design model for a public library IS including only object types.

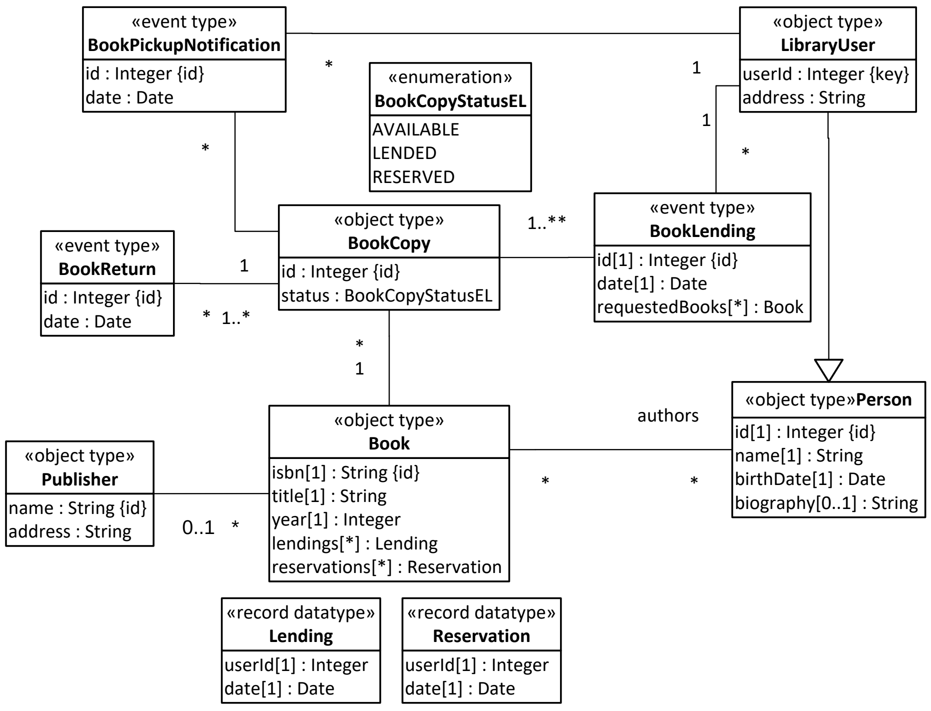

Figure 8: An extended information design model for a public library IS.

1.2. Making an OES Model

The conceptual information model shown in Figure 8 describes the relevant entity types of an OES model, where the date/time attributes of the event types described in the IS model of Figure 6 have been dropped and the *many* multiplicities of the participation associations described in the IS model of Figure 6 have been replaced with 0..1 multiplicities.

Notice that in addition to the three event types described by the IS model of Figure 6, the OES model in Figure 8 includes two further event types, which are essential for capturing the dynamics of a library as a business system: *user arrivals* and *user departures*. These two event types, which are important for capturing user behavior in the simulation model, would normally only be included in an IS model, if the IS is connected to member card scanners for automatically recording the entry and exit of users.

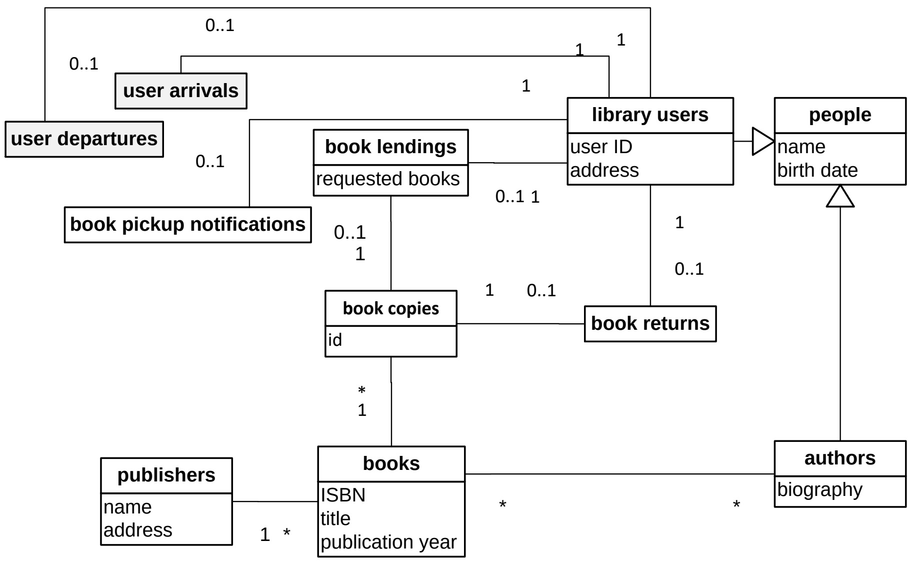

Figure 3: A conceptual information model for an OES model of the public library.

While user arrivals and book pickup notifications are instantaneous events, book lendings and book returns may be modeled either as instantaneous events or as activities. This consideration explains that we have a choice to make an OE Graph or an Activity Network as a process model for the public library. We start with an OE Graph in Section 5.2.1 followed by an Activity Network in Section 5.2.2.

1.2.1. Modeling the Public Library Process as an OE Graph

Based on the conceptual information model for OES shown in Figure 8, we can now make a conceptual process model for an OE simulation of the public library in the form of an OE Graph as shown in Figure 9.

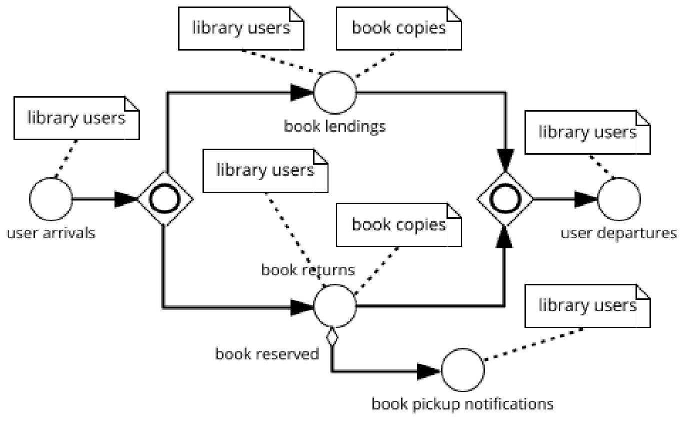

Figure 4: An OE Graph as a conceptual process model for an OE simulation of the public library.

This model describes the possible flows of events, assuming that book lendings and book returns are instantaneous events. When users arrive, two events may happen in any order, as indicated by the *Inclusive OR Gateway*: they may return any number of books in a *book return* event, or they may lend any number of books in a *book lending* event. When a book is returned for which a reservation has been made, then a *book pickup notification* event happens.

1.2.2. Modeling the Public Library Process as an Activity Network

Modeling book lendings and book returns as *instantaneous events* abstracts away from the duration these events have in reality and does not allow modeling the library's service desk involved in these events as a *resource* that can be busy and thus create waiting lines. In a more realistic model, these events should be modeled as *activities* depending on an available service desk as a resource.

The two event types book lendings and book returns can also be modeled as activity types, as in Figure 10.

Compared to the model of Figure 9, turning *book lendings* and *book returns* into activities implies that they have some *duration* as the time between their (implicit) start and end events. Typically, in an OES model, the duration of activities of a certain type is defined in the form of a random variable (with an underlying probability distribution) in an OE class model.

This model describes the possible flows of events and activities. When users arrive, they can return any number of books with a *book return* activity, or they can lend any number of books with a *book lending* activity.

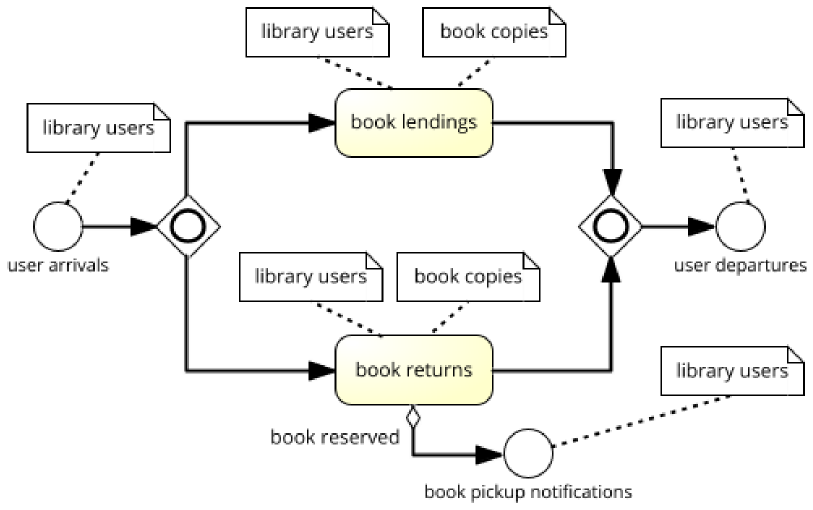

Figure 5: A conceptual process model for the public library in the form of a DPMN Activity Network.

While we can use the same information design modeling language both for the IS project and for the OES project, we cannot use the same information design model for both projects. Rather, based on Observation 2, we can derive the OES class design model from the IS class design model by

1. Dropping the event types' ID and date/time attributes, since these attributes are implicit in OE class models.
2. Changing the event multiplicities of participation associations from * to 0..1.
3. Adding those further event types that are essential for capturing the dynamics of the system under investigation.
4. Designating the exogenous event types, for which a recurrence function has to be specified (typically representing a random variable).
5. Defining a random variable duration function for all activity types.

The result is the OE class design model shown in Figure 11.

Notice that, like in the IS model of Figure 7, the *Book* attribute *reservations* allows checking if a returned book copy has to get the status RESERVED and a *BookPickupNotification* event has to be created for notifying the corresponding user. The attribute *lendings* has been moved from *Book* to *LibraryUser* where it allows the OES to periodically check if users have to go to the library for returning lended books.

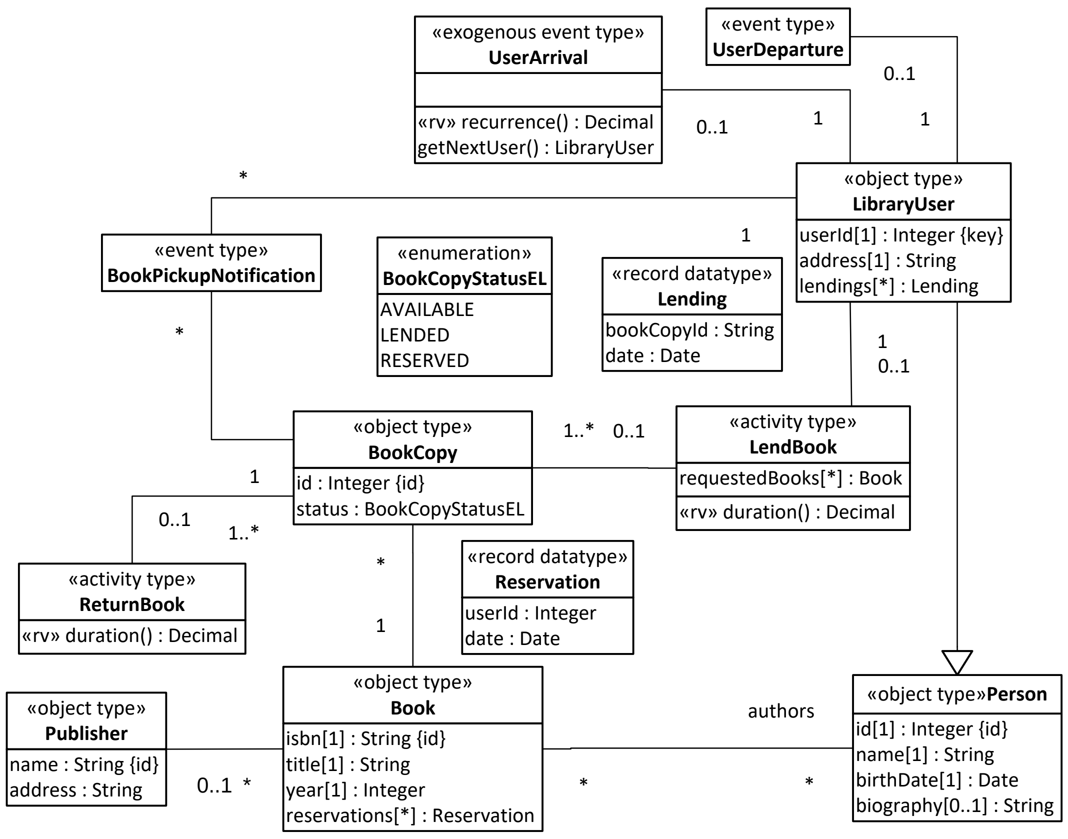

Figure 6: An OE class design model as the basis of an Activity Network design model.

While in the library IS, the periodic checks if book copies need to be returned are performed by the library, which sends out reminder messages to the users concerned, the library OES takes care of this by modeling users that periodically check their lending records for making sure that they don't miss a due date. In the *UserArrival* event class, a *getNextUser* method has been added for selecting a user, either on the basis of a current due date or at random. This method is also in charge to create a list of requested books (at random) to be passed to the follow-up *LendBook* activity.

Finally, the Activity Network design model shown in Figure 12 is derived from the conceptual process model of Figure 10 on the basis of the OE class design model of Figure 11 by expressing the event rules for *UserArrival* events and for *LendBook* and *ReturnBook* activity end events, such that these rules are triggered on completion of these activities. For instance, the *LendBook* activity end event rule is specified by (1) the *l:LendBook* activity rectangle declaring the activity variable *l* as representing the current *LendBook* activity; (2) the *u:LibraryUser* object rectangle introducing the activity variable *u* as bound to the user object referenced by the expression *l.user*, which denotes the user involved in the current *LendBook* activity; (3) the state change code of the annotation attached to the object rectangle; and (4) a follow-up event of type *UserDeparture*.

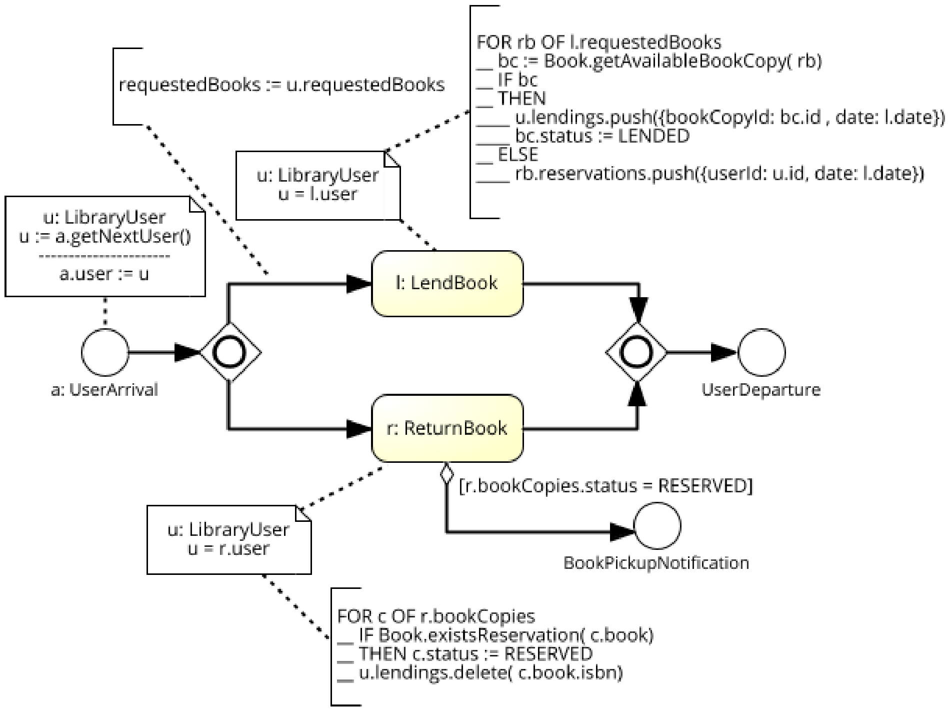
FOR rb OF l.requestedBooks
__ bc := Book.getAvailableBookCopy(rb)
__ IF bc
__ THEN
____ u.lendings.push({bookCopyId: bc.id , date: l.date})
____ bc.status := LENDED
__ ELSE
____ rb.reservations.push({userId: u.id, date: l.date})
requestedBooks := u.requestedBooks
u: LibraryUser
u = l.user
u: LibraryUser
u := a.getNextUser()
a.user := u
l: LendBook
a: UserArrival
UserDeparture
r: ReturnBook
[r.bookCopies.status = RESERVED]
u: LibraryUser
u = r.user
BookPickupNotification
FOR c OF r.bookCopies
__ IF Book.existsReservation(c.book)
__ THEN c.status := RESERVED
__ u.lendings.delete(c.book.isbn)

Figure 7: A DPMN process design model in the form of an Activity Network.