FIT5138 Assignment 3

Group 6

Jialin Liu - 28840402
Jianyun Bi - 28284224
Mingzhe Liu - 28571479

02.06.2019

1 System design

The library system contains 1 context including all sets and constants which will be used in the system, and 3 machines including every event and function in the system.

There are 7 basic events in the BasicLibrary Machine, which are:

- AddBook:

The event happens when library need add a new book title and realated copies under this book title wiil be add at the same time, and at least one, at most 10 copies will be added.

- UserRegister:

The event happens when some new user try to register as a formal user of the library.

- UserDeregister:

The event happens when specific formal user want to deregister the user of the library and the user has borrowed copies cannot deregister, they have to retuen book then deregister.

- Borrow:

The event happens when a registered user wants to borrow a copy from the library. Then the copy is borrowed by the user, set on loan and remore from library shelf.

- Return:

The event happens when a registered user return a copy that he borrowed from the library. Then the copy will no longer on loan and back to library shelf.

- RemoveCopies:

Remove exsited copies from library, copies cannot be borrowed.

- AddCopies:

Add copies under one book titles to library. Book should be already added in library.

There are 1 basic event in the LibararyHoldSystem Machine, which is:

- Hold:

The event happens when a registered user try to borrow a copy which is already been borrowed. Although the user wants to hold on a copy, the library wiil check that all copies under this copy's book title is on loan and not on library shelf. Then the actual thing happens is that the user will be set on hold to this copy's related book title and all the copies under this book title. User cannot hold a book that he has already hold and one user can hold at most 3 books.

There are 3 refine events in the LibraryHoldSystem Machine, which are:

- HoldPickup:

The event happens when a book copy has been return to library with a user holding relationship. If there is a hold, just the user on hold to this copy can pich up this copy.

- UserDeregisteredWithHold:

The event prevents users who hold on books to deregistered.

- RemoveHoldCopies:

Preventing remove copies that are on hold.

There are 1 basic event in the RequestLibrarySystem, which is:

- RequestNewBook:

The event happens when a registered user want to borrow a book that not include in the library now. Then this user will be set request relation to the new book. One user can at most request for 3 books.

There are 4 refine event in the RequestLibrarySystem, which are:

- AddRequestBook:

The event happens when a new book which is requested by a user add in the library, the related copies will be kept for users who request this book (use total bijection function) and send notifications to the users who request this book.

- RequestPickup:

The event happens when a user try to pick up the book that has requested before.

- UserDeregisteredWithRequest:

This event prevents users who has request on new book to deregister from the library.

- RemoveRequestCopies:

Preventing remove copies that are on hold for request.

2 Assumptions in system

1. Only the registered user can borrow/request/hold a book.
2. The maximum number of book that each user can borrow from the library at same time is 5.
3. The maximum number of book that each user can request from the library at same time is 3.
4. The maximum number of book that each user can hold from other user at same time is 3.
5. Every user who want to deregister from library have to return all the books that borrow from the library and have no request and hold relations.
6. All new books that user trying to request can only be request once.
7. After the book that held by user is returned to library, system will send notification to the user who held this book, and need pick up by user.

3 The answer for Task 3.1

1. **What is the relationship between a book title and its copies?**
   COPY and BOOK(title) are two sets in our project, connected by a constants named book_sets. The book_set includes all relations of book titles and the copies under or connected to these book titles.Basically, the book title and copy relation are many to one relation.

2. **When a held book copy is returned to the library, what prevents a random user from borrowing it?**
   In the LibraryHoldSystem Machine, there is a validation on borrow event that the system will check the if the copy and the user is in a hold relation. If yes, then this system can start the pickup progress and prevent a random user from borrowing the book copy which is on hold.

3. **Should a user be allowed to borrow/hold multiple copies of the same book title at the same time? Why or why not?**
   User can not hold or borrow too many copies at the same time because if user borrow too many books, the resourse in library will be limitted. If the user hold on too many copie, the borrow function will be limited because too many holds means if some copies are under too many holds, user who wants to borrow it need to wait for a long time, until all holds is no longer avaliable.

4. **How do we make sure a user cannot borrow/hold too many book copies?**
   In our project, user can only borrow one copy at one time. Because for each book, we have to check if the book is on hold or on hold for request. The max number of hold copies of one user can hold at the same time is 3.
   For hold, user can ask to hold one copy at one time, however, to be more efficiency, we will check is the all copies that are under the book title of that copy is on loan, then if yes, we will set all these copies on hold to save the time for waiting on hold for user. Therefore, the actual limitation is on book. Each user cannot request for more than 3 books.

5. **What is actually being held, a book title or a book copy?**
   In this event-B model, we divide the book title and copy into two parts. When user wants to hold a copy, it actually hold a book title. User is actually choose a book title to hold instead of holding one current copy which is borrowed by other user.

6. **What is actually being requested, a book title or a book copy?**
   For request, user only can request a new book title. Then the library will add this new book title and copies under this book title to library.

7. **What happens when multiple users place a hold on the same book?**
   For the system, every user is a individual unit, when users come into the system for holding book, each event is individual. Thus, system will store all these hold pairs, for next time, if any of the copies under the book is avaliable, any of the users hold on the book have priority to pickup the book.

8. **How should a user be notied when a book she requested/held is ready?**
   We have a set called notification. Basically, it a relation between books and users. System will search the information of request book and corresponed user from the request. Then, add this relationship in the notification set. Based on this, the notification will be sent to the corresponed user and inform the user that the copy is ready for take.

4 Source Code of Rodin

All source code of Rodin are shown below.

CONTEXT LibraryContext ▷

Context for library

SETS

USER ▷ all users in the world

BOOK ▷ book set including all book (titles) in the world

COPY ▷ copy set including all copies in the world

NOTIFICATION ▷ all possible notifications that can generate

CONSTANTS

book_sets ▷ shows the relationship between BOOK and COPY

maxuser ▷ the max user number

notification ▷ the notification is based the relationship between book and user

AXIOMS

axm1: $book\_sets \in COPY \rightarrow BOOK$ ▷

the relationship of BOOK set and COPY set is total function between COPY and BOOK, because all copies must related to book.

axm2: $maxuser \in \mathbb{N}_1$ ▷

number of max user is positive number.

axm3: $finite(USER)$ ▷

USER set is finite.

axm4: $card(USER) = maxuser$ ▷

the cardinality of USER is equals to the number of maxuser, in this case it means it is a positive number.

axm5: $notification \in BOOK \leftrightarrow USER$ ▷

the notification is based the relationship between book and user, and one new book can trigger notification to many users, one user can wait for many notification of new books.

END

MACHINE BasicLibrary ▷

Basic function of the library

SEES LibraryContext

VARIABLES

registered_users ▷ The users registered in library system.

books_in_library ▷ The books (title) in the library.

copies_in_library ▷ Copies that the library has.

copies_on_shelf ▷ Copies that are in the library(On bookshelf).

copies_on_loan ▷ Copies that is already on loan.(Borrow pairs)

library_book_sets ▷ The pairs of the book (title) and the copies in library.

INVARIANTS

inv1: $registered\_users \subseteq USER$ ▷

The registered users need to be the users in the world.

inv2: $books\_in\_library \subseteq BOOK$ ▷

The books that the library has need to be the books in the world.

inv3: $copies\_in\_library \subseteq COPY$ ▷

The copies of the library has to be the copies in the world.

inv4: $copies\_on\_loan \in copies\_in\_library \nrightarrow registered\_users$ ▷

The copies on loan are pairs in particial function because the user can borrow no copy and copy can have no user to borrow.

inv5: $copies\_on\_shelf = copies\_in\_library \setminus dom(copies\_on\_loan)$ ▷

Copies on shelf is the all copies of the library minus the copies that have been borrowed.

inv6: $library\_book\_sets \in copies\_in\_library \rightarrow books\_in\_library$ ▷

library book sets are pairs of total function of copies in library and books in library, because all books in library have copies and all copies in library must have a related book.

inv7: $library\_book\_sets \subseteq book\_sets$ ▷

The book and copy relation in library is a subset of the book and copy relation in whole world.

EVENTS

Initialisation ▷

initialisation event

begin

act1: $registered\_users := \varnothing$

act2: $books\_in\_library, library\_book\_sets := \varnothing, \varnothing$

act3: $copies\_in\_library, copies\_on\_shelf, copies\_on\_loan := \varnothing, \varnothing, \varnothing$

end

Event AddBook ⟨ordinary⟩ $\widehat{=}$ ▷

Add book event

any

book ▷ The book which is going to be added.

copies ▷ The related copies.

book_set ▷ The pairs between book and the copies.(To explain book and corresponding copies.)

where

grd1: $book \in BOOK$ ▷
The book needs to be one of the BOOK in the world.
grd2: $book \notin books\_in\_library$ ▷
This book is not in the library.
grd3: $copies \subseteq COPY$ ▷
Copies are copies in the world.
grd4: $book\_set \subseteq book\_sets$ ▷
The pairs of book and copies are subset of all books and copies pairs in the world.(Copies are under the book title. However, not all copies under the book title are added to library.)
grd5: $book\_set \in copies \rightarrow \{book\}$ ▷
Book set is the pairs of copies and book.

then

act1: $books\_in\_library := books\_in\_library \cup \{book\}$ ▷
Add this book to library.
act2: $copies\_in\_library := copies\_in\_library \cup copies$ ▷
Add all related copies to library.
act3: $copies\_on\_shelf := copies\_on\_shelf \cup copies$ ▷
Put all copies on shelf.
act4: $library\_book\_sets := library\_book\_sets \cup book\_set$ ▷
Add pairs to library ralation.

end

Event UserRegister ⟨ordinary⟩ $\widehat{=}$ ▷
User Register

any

new_user ▷ Use who want to be a member of the library.

where

grd1: $new\_user \in USER$ ▷
New user needs to be one of the USER.
grd2: $new\_user \notin registered\_users$ ▷
New user cannot be a registered_users.
grd3: $registered\_users \neq USER$ ▷
Registered users cannot be more than users in the world.

then

act1: $registered\_users := registered\_users \cup \{new\_user\}$ ▷
Add user to registered_users.

end

Event UserDeregister ⟨ordinary⟩ $\widehat{=}$ ▷
User Deregister

any

user ▷ Use who wants to deregister.

where

grd1: $user \in registered\_users$ ▷
The user who wants to deregister needs to be a registered user first.
grd2: $copies\_on\_loan \nrightarrow \{user\} = \varnothing$ ▷
The user does not have any book that are not returned.

then

act1: $registered\_users := registered\_users \setminus \{user\}$ ▷
Exclude the user from registered users.

end

Event Borrow ⟨ordinary⟩ $\widehat{=}$ ▷

Borrow copy event.

any

user ▷ User wants to borrow copy.

copy ▷ The copy that the user wants to borrow.

where

grd1: $user \in registered\_users$ ▷

The user needs to be a registered user.

grd2: $copy \in copies\_on\_shelf$ ▷

Copy is not borrowed by other users.

grd3: $card(copies\_on\_loan \rhd \{user\}) \in 0 \mathinner{..} 4$ ▷

The user cannot hold more than 4 books.

then

act1: $copies\_on\_shelf := copies\_on\_shelf \setminus \{copy\}$ ▷

Remove copy from shelf.

act2: $copies\_on\_loan := copies\_on\_loan \cup \{copy \mapsto user\}$ ▷

Give user the copy and add relation.

end

Event Return ⟨ordinary⟩ $\widehat{=}$ ▷

Return copy.

any

copy ▷ The copy will be returned.

where

grd1: $copy \in dom(copies\_on\_loan)$ ▷

Copy needs to be a copy has a relation of borrow.

then

act1: $copies\_on\_shelf := copies\_on\_shelf \cup \{copy\}$ ▷

Add this copy to shelf.

act2: $copies\_on\_loan := \{copy\} \lhd copies\_on\_loan$ ▷

Exclude the borrow relation from the library borrow relations.

end

Event RemoveCopies ⟨ordinary⟩ $\widehat{=}$

any

copies ▷ copies that the library wants to remove

where

grd1: $copies \subseteq copies\_in\_library$ ▷

copies are in the library

grd2: $copies \not\subseteq dom(copies\_on\_loan)$ ▷

copies are not on loan

then

act1: $copies\_on\_shelf := copies\_on\_shelf \setminus copies$ ▷

remove copies from library shelf

act2: $copies\_in\_library := copies\_in\_library \setminus copies$ ▷

remove copies from library

act3: $library\_book\_sets := library\_book\_sets \setminus (copies \lhd library\_book\_sets)$ ▷

exclude these book and copy relations from library_book_sets

end

Event AddCopies ⟨ordinary⟩ $\widehat{=}$

any

copies
book
set

where

grd1: $copies \not\subseteq copies\_in\_library$
grd2: $book \in books\_in\_library$
grd3: ⟨theorem⟩ $set \in (copies \rightarrow \{book\})$
grd4: $set \subseteq book\_sets$

then

act1: $copies\_in\_library := copies\_in\_library \cup copies$ ▷
add copies to library
act2: $copies\_on\_shelf := copies\_on\_shelf \cup copies$ ▷
add copies to library shelf
act3: $library\_book\_sets := library\_book\_sets \cup set$ ▷
add copies and book pairs to library_book_sets

end

END

MACHINE LibraryHoldSystem
REFINES BasicLibrary
SEES LibraryContext
VARIABLES

holds ▷ The hold pairs.
hold_books ▷ The books in hold pairs.
hold_related_copies ▷ The pairs gives the information of the relation between users and hold copies.
registered_users ▷ The users registered in library system.
books_in_library ▷ The books (title) in the library.
copies_in_library ▷ Copies that the library has.
copies_on_shelf ▷ Copies that are in the library(On bookshelf).
copies_on_loan ▷ Copies that is already on loan.(Pairs of copies and users)
library_book_sets ▷ The pairs of the book (title) and the copies in library.

INVARIANTS

inv1: $holds \in books\_in\_library \leftrightarrow registered\_users$ ▷
The relation here represents one user can hold many books, one book can be hold by many users
inv2: $hold\_books \subseteq books\_in\_library$ ▷
The hold book must be a book that has existed in library.
inv3: $hold\_books = dom(holds)$ ▷
The domain of hold are the books on hold.
inv4: $hold\_related\_copies \in (dom(library\_book\_sets \rhd hold\_books) \leftrightarrow registered\_users)$ ▷
one user can hold many related copies, one related copy can be hold by many users
inv5: $hold\_related\_copies \cap copies\_on\_loan = \varnothing$ ▷
hold related copies have no coinside with copies on loan pairs.

EVENTS

Initialisation ▷
Initialisation
begin
act1: $registered\_users := \varnothing$
act2: $books\_in\_library, library\_book\_sets := \varnothing, \varnothing$
act3: $copies\_in\_library, copies\_on\_shelf, copies\_on\_loan := \varnothing, \varnothing, \varnothing$
act4: $holds, hold\_books, hold\_related\_copies := \varnothing, \varnothing, \varnothing$
end

Event Hold ⟨ordinary⟩ $\widehat{=}$ ▷
hold event
any
book ▷ The book user wants to hold.
user ▷ User who wants to hold a book.
related_copies ▷ The pairs of the copies related to the book and the user.
where
grd1: $book \in books\_in\_library$ ▷
book needs to be a book in library

grd2: $user \in registered\_users$ ▷
user is a registered user.
grd3: $book \mapsto user \notin holds$ ▷
the relation of book and user is no a hold relation.
grd4: $related\_copies \in (dom(library\_book\_sets \rhd \{book\}) \rightarrow \{user\})$ ▷
ralated copies is the pairs of the copies related to the hold book and the user.
grd5: $related\_copies \cap copies\_on\_loan = \varnothing$ ▷
copies on hold has no coinside with copies on loan.
grd6: $dom(library\_book\_sets \rhd \{book\}) \subseteq dom(copies\_on\_loan)$ ▷
All the copies related to this book are on loan.
grd7: $card(holds \rhd \{user\}) \in 0 \mathinner{..} 2$ ▷
user cannot hold more than 2 books

then

act1: $holds := holds \cup \{book \mapsto user\}$ ▷
Add this book and user pair to hold.
act2: $hold\_related\_copies := hold\_related\_copies \cup related\_copies$ ▷
Add all related copies pairs to hold_related_copies
act3: $copies\_on\_loan := copies\_on\_loan$
act4: $copies\_on\_shelf := copies\_on\_shelf$

end

Event HoldPickup ⟨ordinary⟩ $\widehat{=}$ ▷
Check if use is on hold of this book, if yes borrow this use one copy and delete book-user; copies-user pairs.

extends Borrow

any

user ▷ User wants to borrow copy.
copy ▷ The copy that the user wants to borrow.
related_book ▷ book related the this copy
related_copies ▷ pairs shows relation of all copies realted to the book with the user.

where

grd1: $user \in registered\_users$ ▷
The user needs to be a registered user.
grd2: $copy \in copies\_on\_shelf$ ▷
Copy is not borrowed by other users.
grd3: $card(copies\_on\_loan \rhd \{user\}) \in 0 \mathinner{..} 4$ ▷
The user cannot hold more than 4 books.
grd4: $copy \mapsto user \in hold\_related\_copies$ ▷
pair of this copy and user belongs to hold_related_copies
grd5: $related\_book \in (ran(\{copy\} \lhd library\_book\_sets))$ ▷
find the related book(just one book element in this set)
grd6: $related\_copies \in (dom(library\_book\_sets \rhd \{related\_book\}) \rightarrow \{user\})$ ▷
related_copies is the pairs of the copies and the suer.

then

act1: $copies\_on\_shelf := copies\_on\_shelf \setminus \{copy\}$ ▷
Remove copy from shelf.
act2: $copies\_on\_loan := copies\_on\_loan \cup \{copy \mapsto user\}$ ▷
Give user the copy and add relation.

act3: $holds := holds \setminus \{related\_book \mapsto user\}$ ▷
exclude related book and user pair.
act4: $hold\_related\_copies := hold\_related\_copies \setminus related\_copies$ ▷
exclude related copies from hold_related_copies.

end

Event UserDeregisteredWithHold ⟨ordinary⟩ $\widehat{=}$

extends UserDeregister

any

$user$ ▷ Use who wants to deregister.

where

grd1: $user \in registered\_users$ ▷
The user who wants to deregister needs to be a registered user first.
grd2: $copies\_on\_loan \twoheadrightarrow \{user\} = \varnothing$ ▷
The user does not have any book that are not returned.
grd3: $user \notin ran(holds)$ ▷
user who wants to deregistered cannot hold any book, which means this user is not in any pair of holds.

then

act1: $registered\_users := registered\_users \setminus \{user\}$ ▷
Exclude the user from registered users.

end

Event RemoveHoldCopies ⟨ordinary⟩ $\widehat{=}$

extends RemoveCopies

any

$copies$ ▷ copies that the library wants to remove

where

grd1: $copies \subseteq copies\_in\_library$ ▷
copies are in the library
grd2: $copies \nsubseteq dom(copies\_on\_loan)$ ▷
copies are not on loan
grd3: $copies \nsubseteq dom(hold\_related\_copies)$ ▷
copies cannot be on hold

then

act1: $copies\_on\_shelf := copies\_on\_shelf \setminus copies$ ▷
remove copies from library shelf
act2: $copies\_in\_library := copies\_in\_library \setminus copies$ ▷
remove copies from library
act3: $library\_book\_sets := library\_book\_sets \setminus (copies \lhd library\_book\_sets)$ ▷
exclude these book and copy relations from library_book_sets

end

END