Advanced Software Engineering

CSE608

A Case study of a Library System

Dr. Islam El-Maddah

Problem description

students can go to the library front office with the book they need to borrow scan it and scan their IDs then the system add the book to them and show them message when to return the book

The student when returning the book simply scan the book copy and the system set the copy as free again and someone will manually move it back to its normal shelf

Problem description

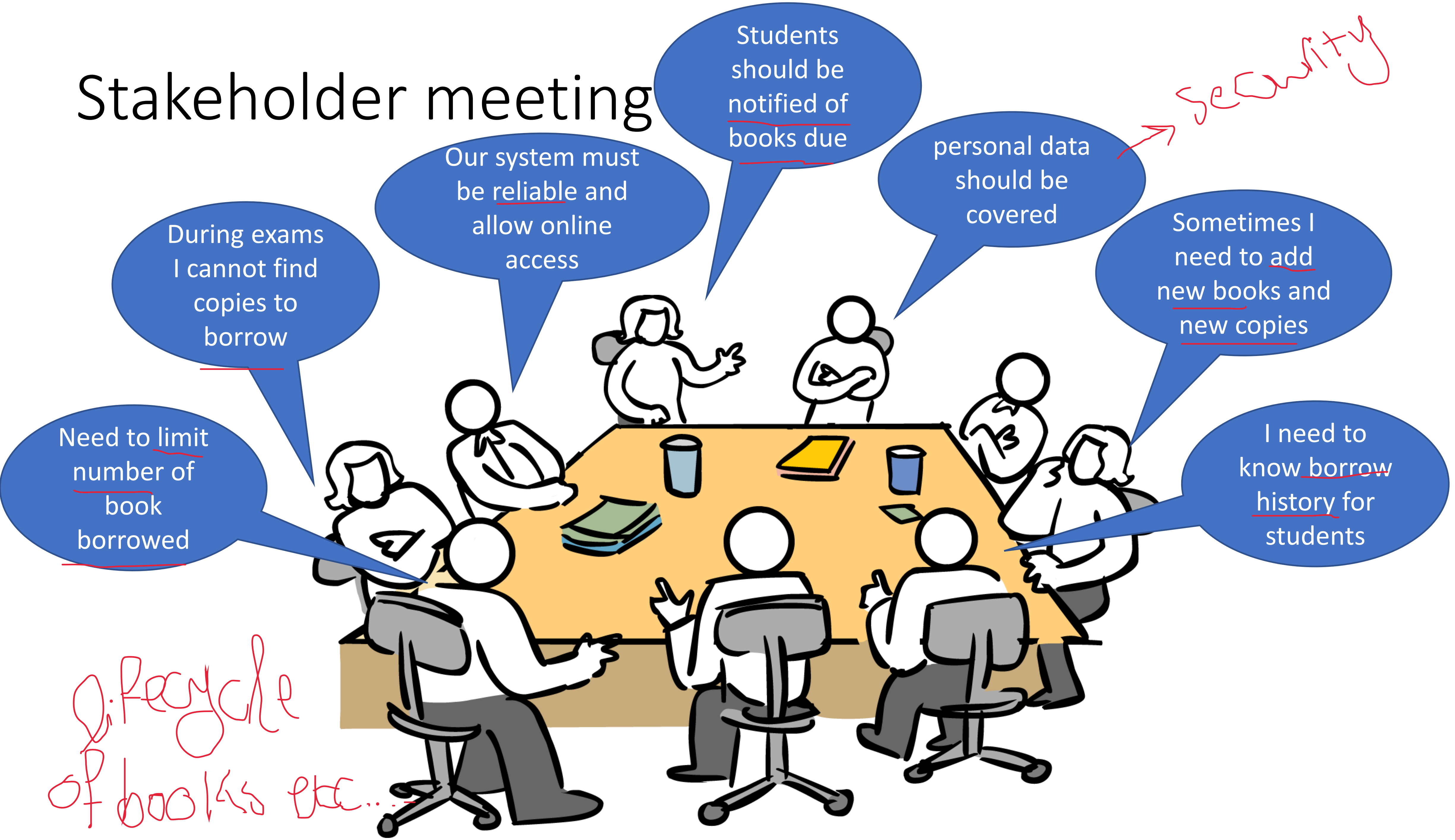
The student can also search for books by title or by category and the system list all related books.

If some book is not available for borrowing the student can **reserve** it and when any copy of it return the library system notify the student of the book arrival so he can borrow it.

The admin can add or remove books as well as add and remove book copies

The history of student borrowed books as well as the history of the book copy need to be stored until last three years

Stakeholder meeting

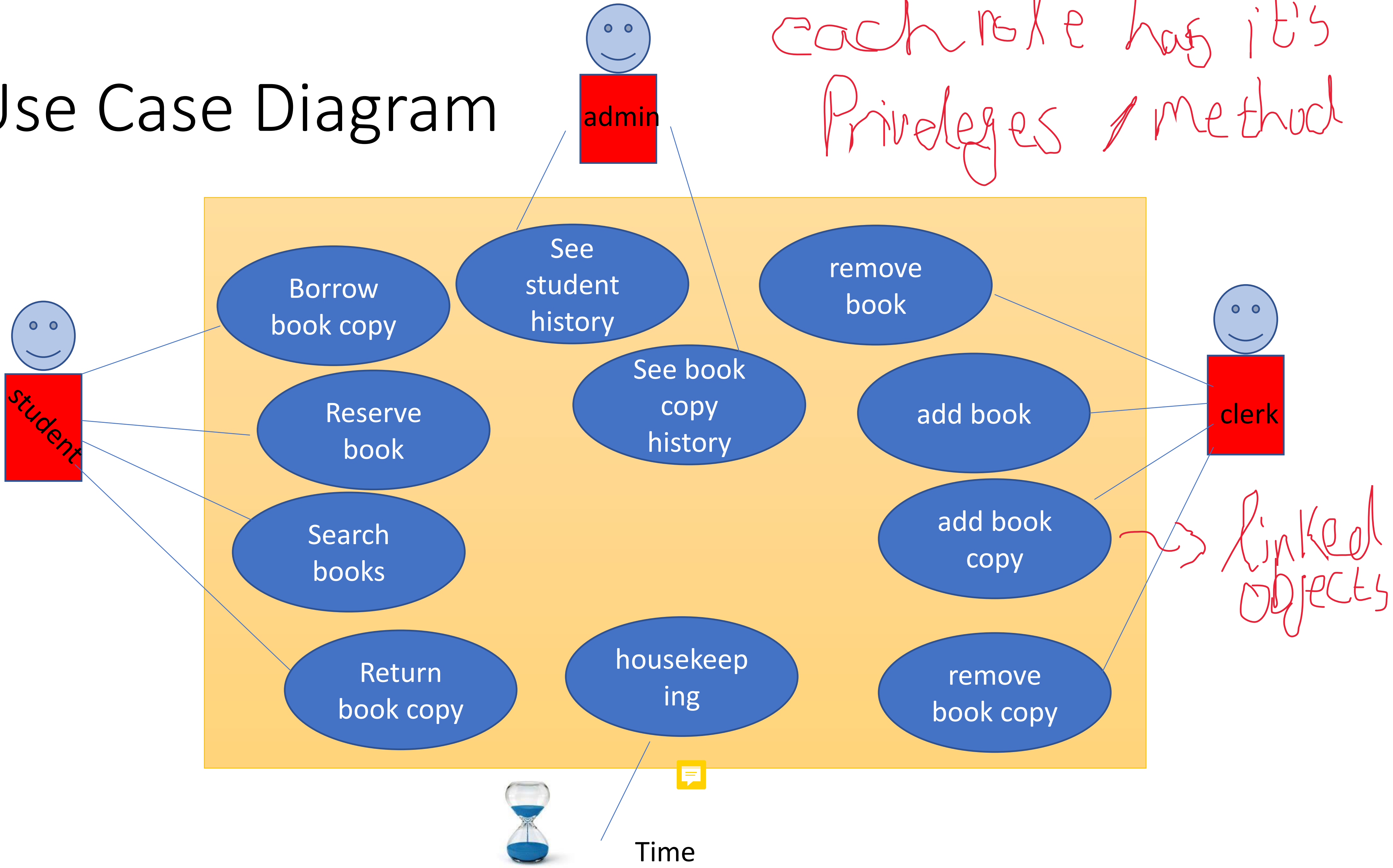

During exams I cannot find copies to borrow
Our system must be reliable and allow online access
Students should be notified of books due
personal data should be covered
Sometimes I need to add new books and new copies
Need to limit number of book borrowed
I need to know borrow history for students

→ Security

Q: Recycle of books etc...

Use Case Diagram

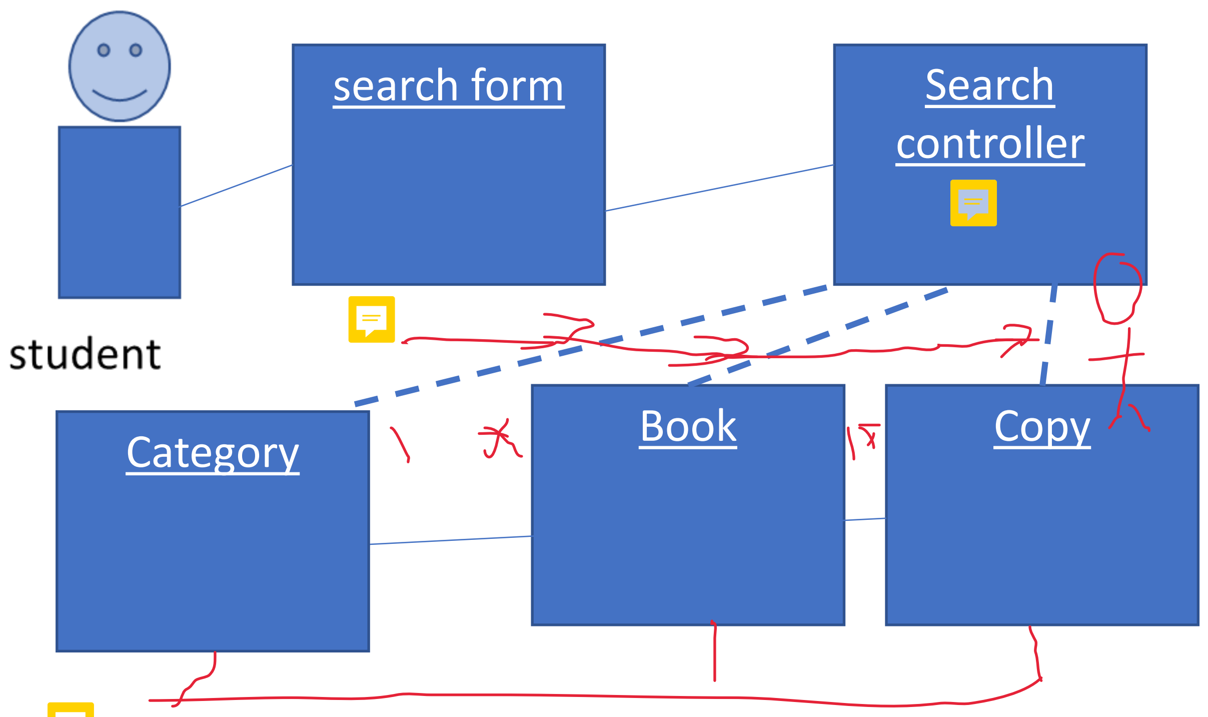
each role has it's Priveleges / method

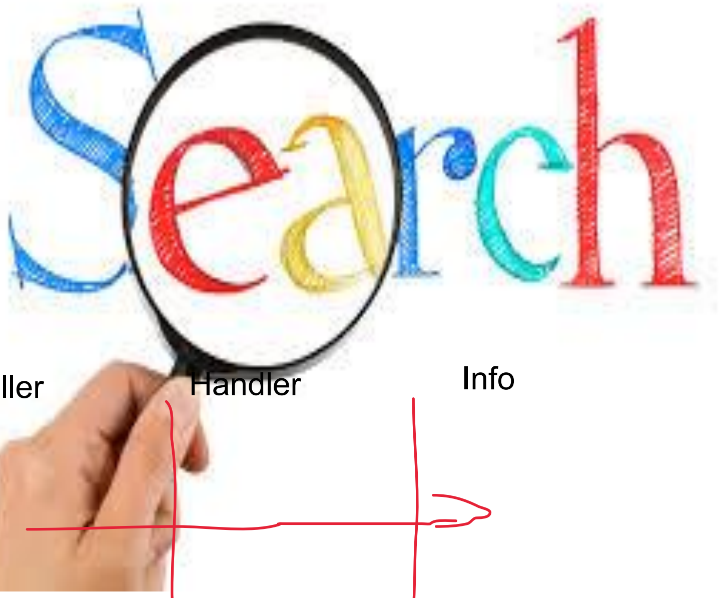


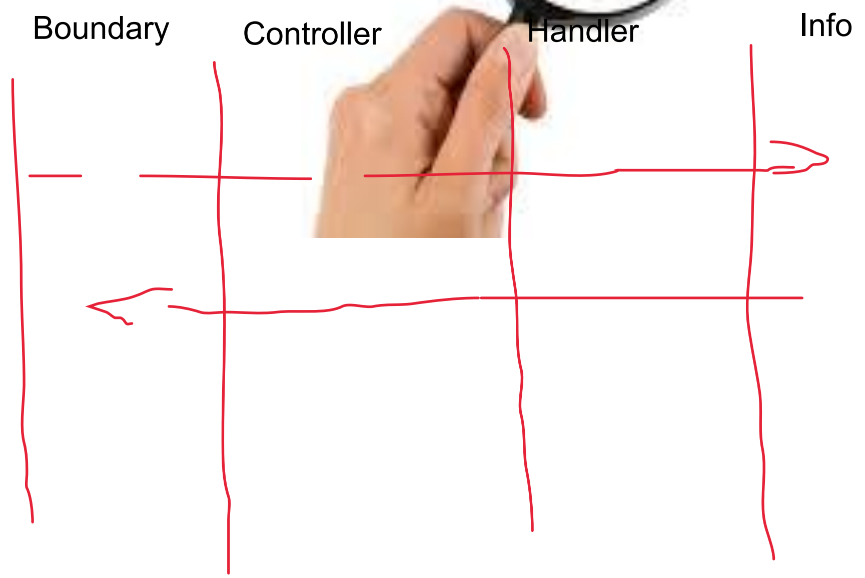
Search Use Case description

Use case	Search books	
Actor	student	
trigger	Student needs to look for books	
Pre condition	System is running student is logged in	
Post condition	Student see the selected book list	
input	Category name	
output	none	
Main course	1 Actor	Enters the category name
	2 System	Accept the category name
	3 Actor	Press search
	4 System	Looks for books with this category and display their details and number of copies available
	5 Actor	Observe the result and terminate the search form (OK button)
	6 System	Ends the dialog and return control to the actor again
alternative	1.1 category name is wrong or empty book list, the system send s a message and ask the	

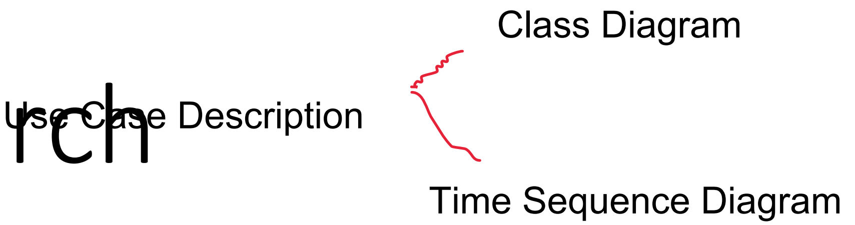
Analysis class for search

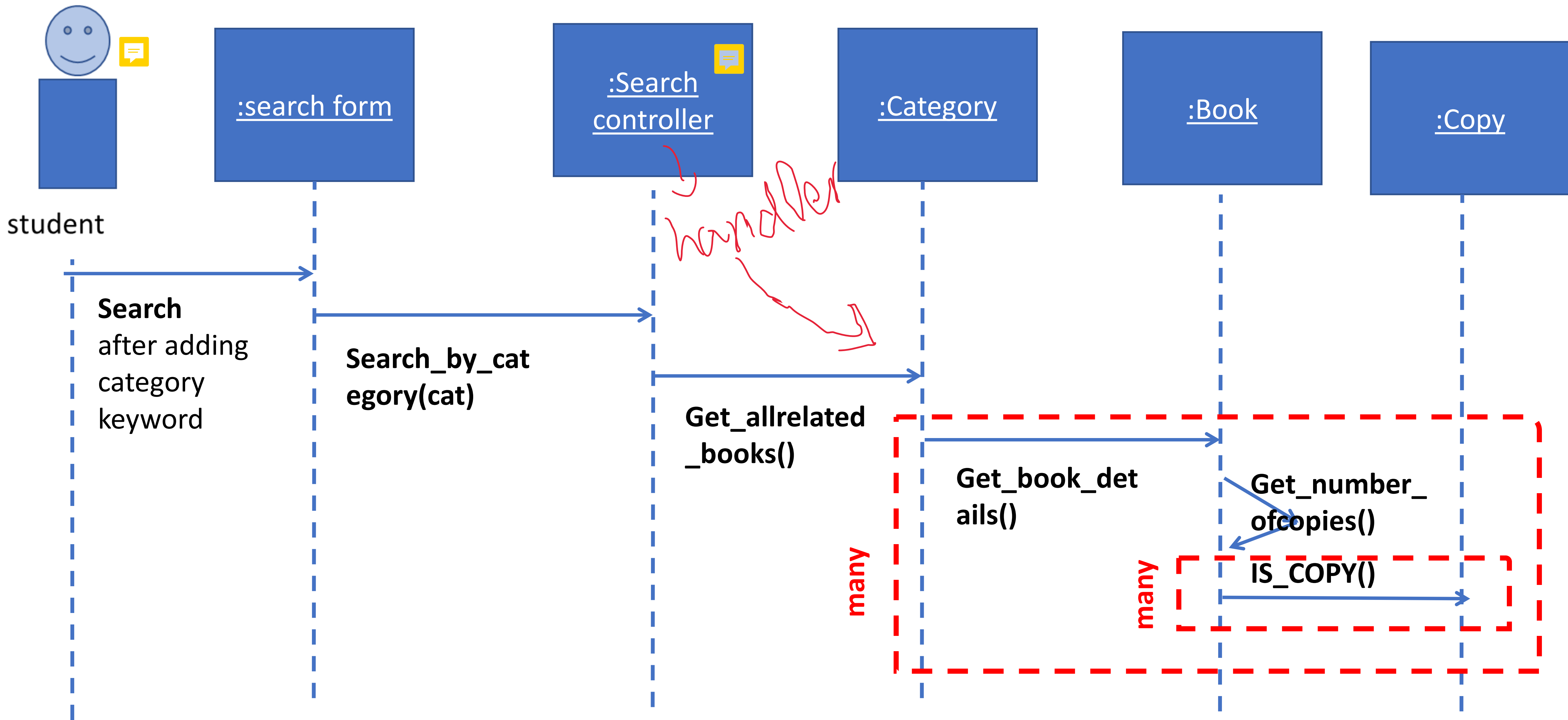




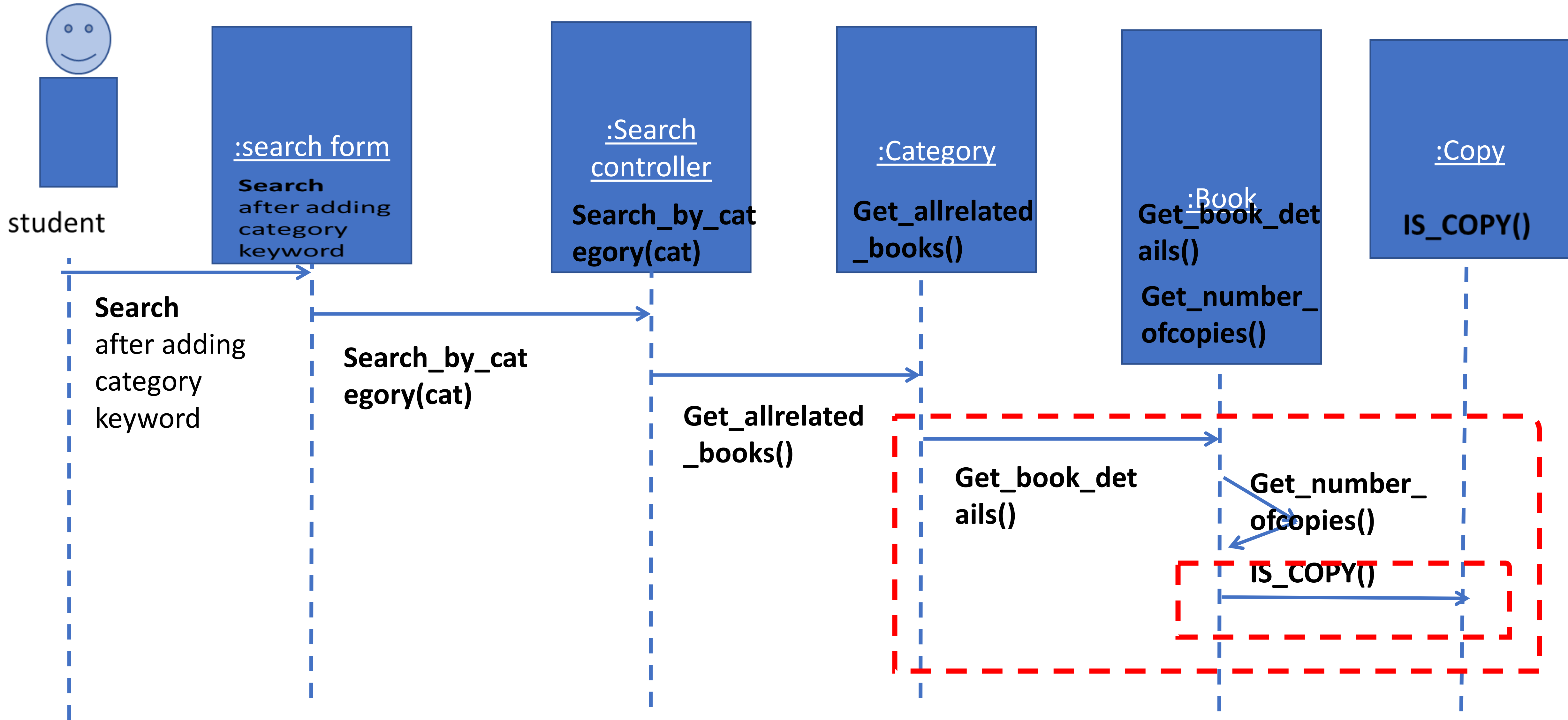


Time Sequence Diagram for search

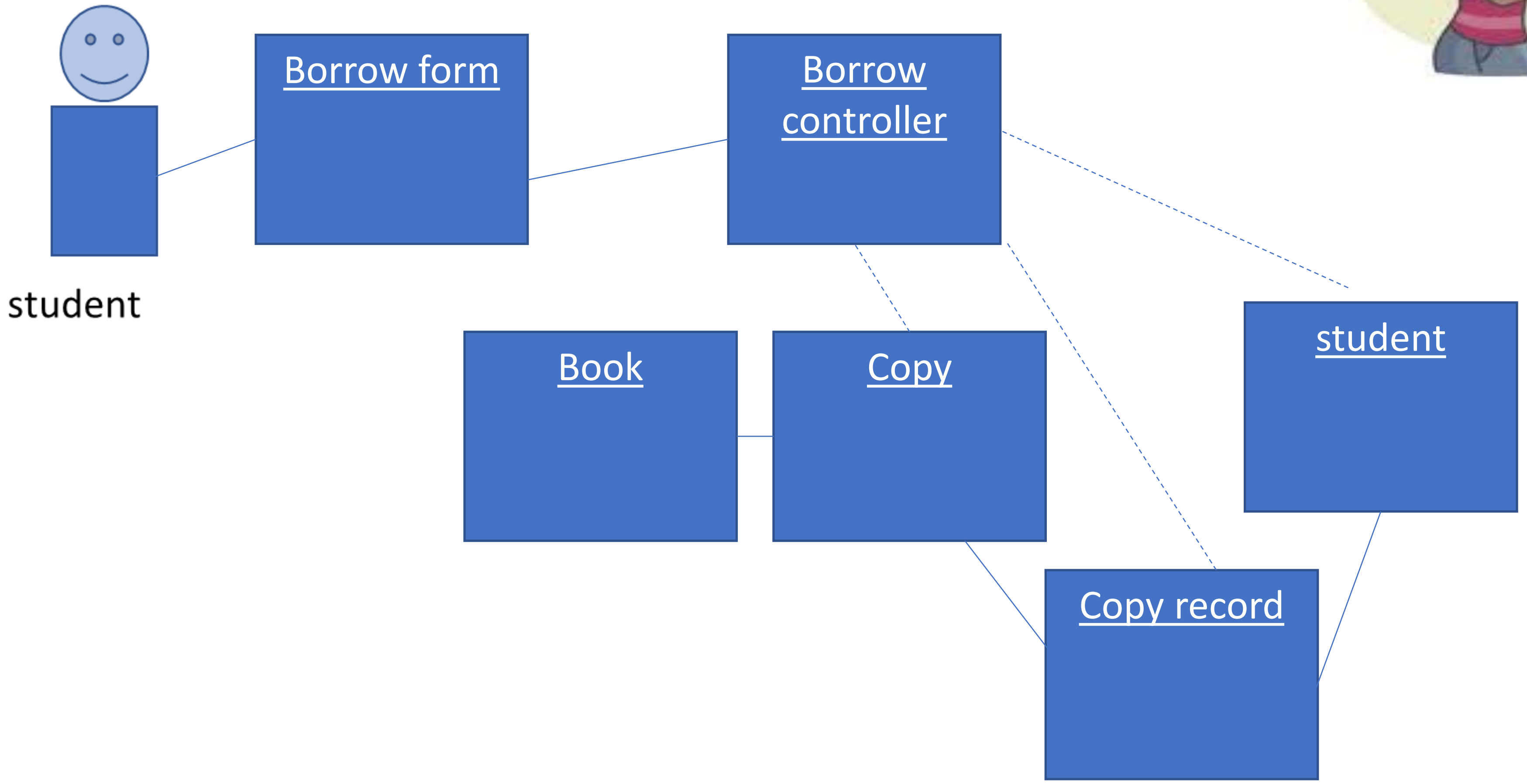




Accommodating messages from the sequence diagram

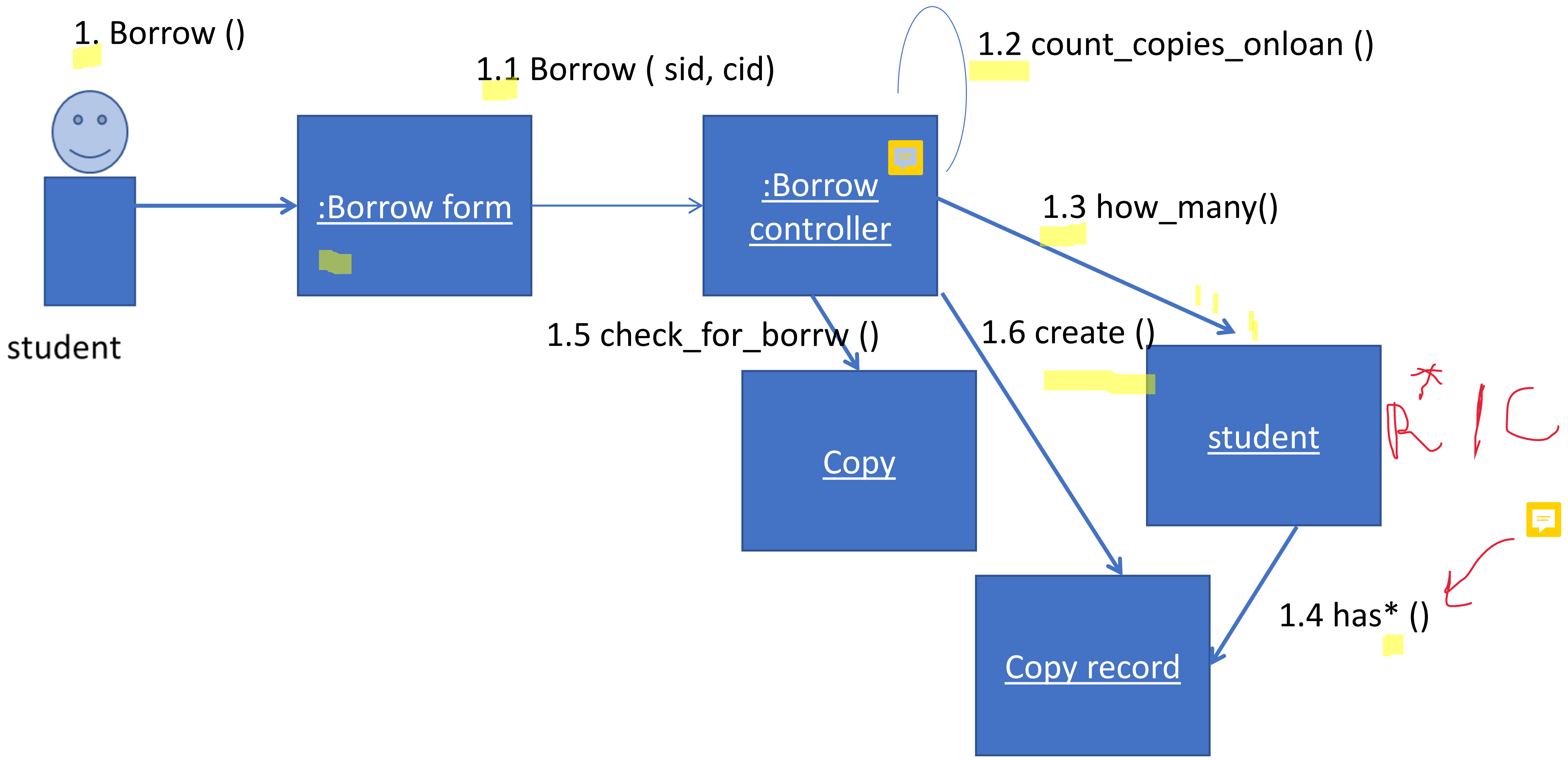


Analysis classes for borrow

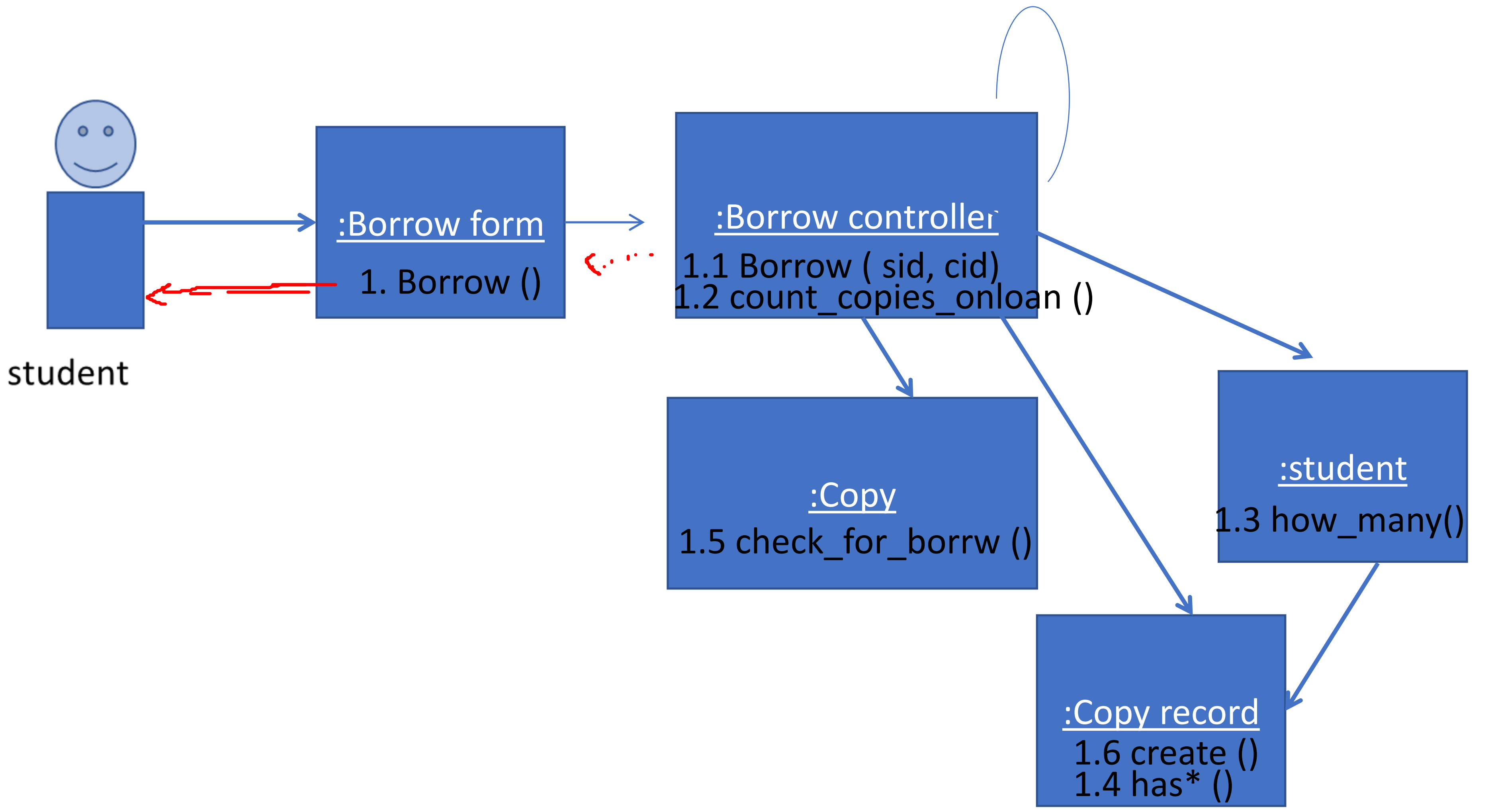


To keep time of borr
return, history copy
record is needed si
reserve record

Collaboration diagram borrow



Accommodating the methods/messages in the analysis classes



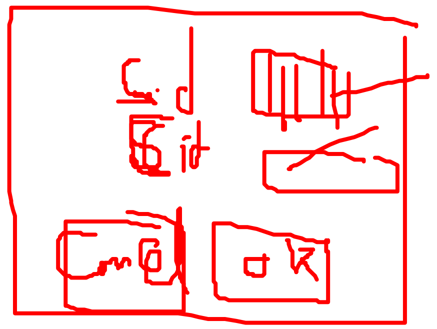
Unify classes !!
Unify class methods

- Book_copy and copy are the same class etc.

Borrow scenario may end up either happy ☺ or unhappy if the student has many books on loan ☹ or the copy is not correctly understood as free ☹

Create() is the simple new in C++ or java

Delete is delete



Borrow in the form means it is a button or a menu item that is accessed /clicked by the actor notes

The CTRL takes its parameter from the dialog box thus it knows which object (student and copy) it will access so the parameter journey stops at the CTRL notes

The CTRL takes the computed values to decide whether we can do the borrow or not according to the copy type and number of copies on loan with the student

CRUD Matrix

- A double check between the use case model (what needs to be done) and the structure model/classes (how it will be done)
- It may reveal missing use cases or missing classes for use cases which updates the system state

CRUD Matrix

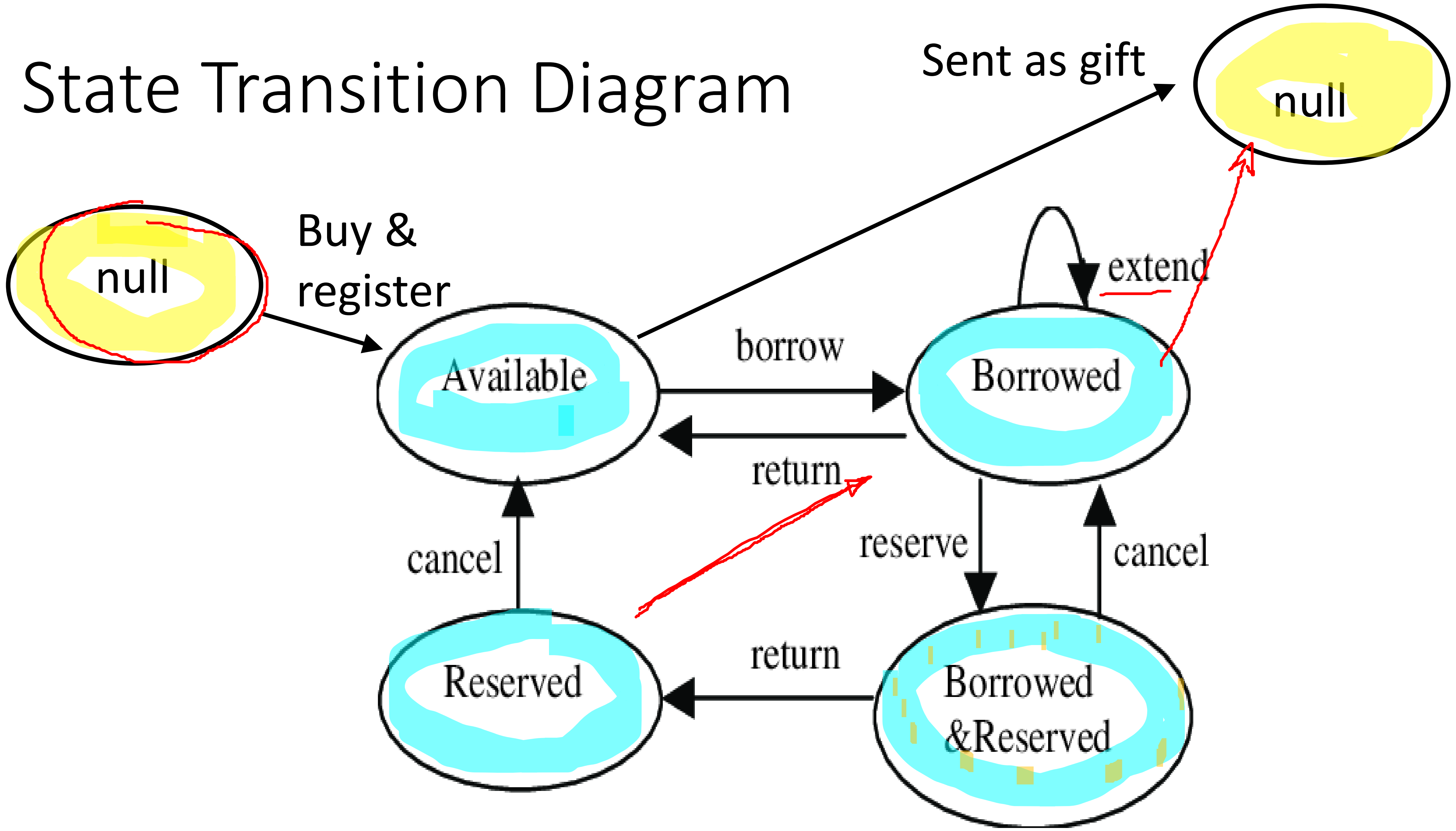
- It can also help grouping related classes and use cases, for example use cases that access class students and book or copy and borrow record, etc. this helps while developing the use cases
- We will keep an eye on classes with no D, C or U/R
- Also will be looking for updating use cases who never issue D,C,U !
- And will be looking for reading use cases who does D,C, U !

CRUD (Create/ Read/ Update/ Delete)

V class Use case →	borrow	search	return	reserve	housekeeping
Student	R			R	
Book		R*		R	
Category		R			
Book_copy	R	R*	R		
Borrow_record	R*/C	R*	U		D*
Search_form		C/R/U/D			
Search_ctrl		C/R/U/D			
Borrow_form	C/R/U/D				
Borrow_Ctrl	C/R/U/D				
Return_form			C/R/U/D		
Return_ctrl			C/R/U/D		
Reserve_form				C/R/U/D	
Reserve_ctrl				C/R/U/D	
INT_Handler (boundary)					C/R/U/D

- Need to unify class after these collaboration or sequence diagrams
- Missing class for reserve could be reserve record between classes student and book
- Need to be linked with return use case so when any one return a reserved book we can send notification to people who stands in a queue (FIFO) most of the time

State Transition Diagram



Non-functional Requirements

number	Category of non-functional requirement	Non-functional requirement	description	Affected use cases	priority	Possible solutions
1	(1) quality	security	Information will not be revealed to others	All those who read/access student data	High	Add login process and access control/ use cipher to hide info
2	(1) quality	Performance	All system functions will be done in limited time	All use cases	High	Add indexes when possible and enhance search algorithms
3	(1) quality	reliability	System will not failed more than once a month	All use cases	medium	System will be replicated in another server
4		usability				
5						