

# COMP1216 - CW2 - Online Library System using Event B modelling

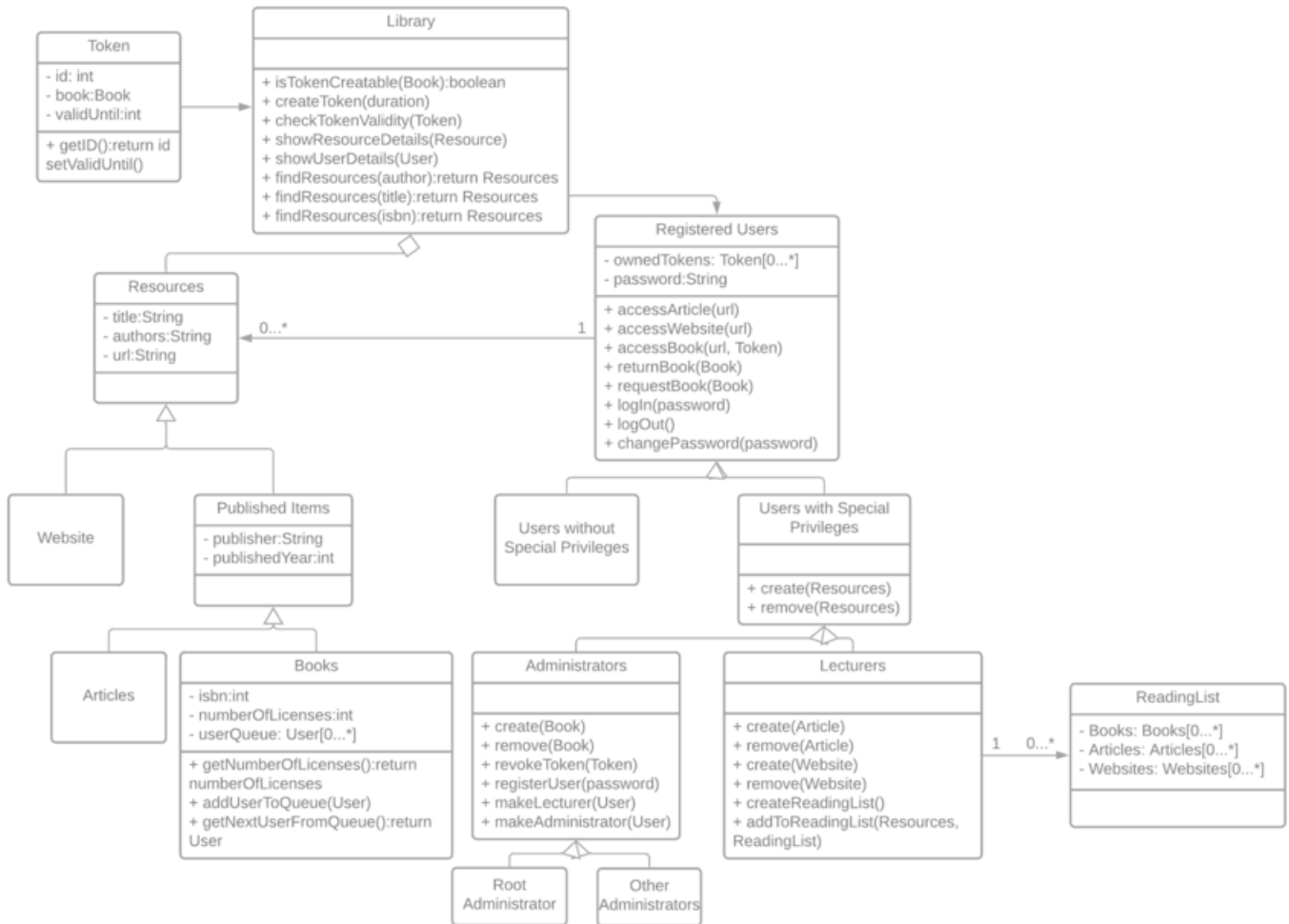
Patrik-Tibor Csanyi, Sandor Kovacs, Charlie Williams,  
Zachariah Ridzuan-Allen  
ptc1g20(ID: 31574602), sk10g20(ID: 31773478), cw2g19(ID: 431023274),  
zra1u19(ID: 30870186)

Group 20

## 1 Introduction

The purpose of this report is to present the formal modelling of an online library from Coursework 1. The tasks were split among us based on the difficulty and effort required but we helped each other to reduce the repetition of invariants and variables. The first 6 requirements were done by Patrik-Tibor Csanyi(ptc1g20) and Sandor Kovacs(sk10g20) as well as the last 6 requirements. The remaining part, requirements from 7 to 16 were done by Zachariah Ridzuan-Allen(zra1u19), Charles Williams(cw2g19), Patrik-Tibor Csanyi(ptc1g20) and Sandor Kovacs(sk10g20) as these parts required the most effort. Lastly the class diagram was done by Patrik-Tibor Csanyi, and the report was written by Sandor Kovacs.

## 2 Diagram



## 3 Users

### 3.1 Context

```

1 context OnlineLibrarySystemContext
2 sets
3 USER
4 PASSWORD
5 constants
6 rootUser
7 defaultPassword

```

```

8  axioms
9  @rootUser: rootUser ∈ USER
10 @DEFAULT_PASSWORD—def: defaultPassword ∈ PASSWORD
11 end

```

In the context we determine the sets we are going to use as well as we set some axioms. These axioms help us to set a rootUser and a defaultPassword.

### 3.2 Code

```

1  machine OnlineLibrarySystem
2  sees OnlineLibrarySystemContext
3
4  /*Declaring the variables we are going to use */
5  variables
6  registeredUsers //Set of registered users
7  administrators //Set of administrators
8  lecturers
9  loggedInUsers
10 loggingPassword
11
12 /*Handling the invariants */
13 invariants
14 @inv1: registeredUsers ⊆ USER
15 @inv2: administrators ⊆ registeredUsers
16 @inv3: lecturers ⊆ registeredUsers
17 @inv4: administrators ∩ lecturers = ∅
18 @inv5: rootUser ∈ administrators
19 @inv6: loggingPassword ∈ registeredUsers → PASSWORD
20 @inv7: loggedInUsers ⊆ registeredUsers
21 events
22
23 /*Initializing our sets */
24 event INITIALISATION
25 begin
26 @beg1:registeredUsers := ∅ ∪ {rootUser}
27 @beg2:administrators := ∅ ∪ {rootUser}
28 @beg3:lecturers := ∅
29 @beg4:loggedInUsers := ∅
30 @beg5:loggingPassword := ∅ ∪ {rootUser ↦ defaultPassword}
31 end
32
33 /*The event to log in a user */
34 event LogIn
35 any u p where
36 @grd1:u ∈ registeredUsers //Check if u is a registered user
37 @grd2: p ∈ PASSWORD //Check if p is the set of PASSWORD
38 @grd3:loggingPassword(u) = p //Check if the user and its password are correct
39 then
40 @act1:loggedInUsers := loggedInUsers ∪ {u} //Add the user to the logged in user set
41 end
42
43 /*The event to log out */
44 event LogOut

```

```

45  any u where
46    @grd1: u ∈ loggedInUsers // Check if u is logged in
47  then
48    @act1: loggedInUsers := loggedInUsers \ {u} // Remove u from the logged in users
49  end
50
51  /*The event to register a new user */
52  event RegisterUser
53  any a u p where
54    @grd1: u ∉ registeredUsers // Check if u is not part of the registeredUsers
55    @grd2: u ∈ USER // Check if u is part of USER
56    @grd3: a ∈ administrators // Check if a is administrator to allow registering a user
57    @grd4: p ∈ PASSWORD // Check if p is part of PASSWORD
58  then
59    /*Register the new user u and the password to this user */
60    @act1: registeredUsers := registeredUsers ∪ {u}
61    @act3: loggingPassword(u) := p
62  end
63
64  /*The event to change the password */
65  event ChangePassword
66  any u p where
67    @grd1: u ∈ loggedInUsers // Check if u is a logged in user
68    @grd2: p ∈ PASSWORD // Check if p is part of PASSWORD
69  then
70    @act1: loggingPassword(u) := p // Change the password to p
71  end
72
73  /*The event to change to administrator */
74  event ChangeToAdministrator
75  any a u where
76    @grd1: a ∈ administrators // Check if a is and administrator
77    @grd2: u ∈ registeredUsers \ (lecturers ∪ administrators) // Check if u is part of
    registered users who are not lecturers or administrators
78  then
79    @act1: administrators := administrators ∪ {u} // Add u to the administrators
80  end
81
82  /*The event to change to lecturer */
83  event ChangeToLecturer
84  any a u where
85    @grd1: a ∈ administrators // Check if a is an administrator
86    @grd2: u ∈ registeredUsers \ (administrators ∪ lecturers) // Check if u is a registered
    user but not an administrator and not a lecturer
87  then
88    @act1: lecturers := lecturers ∪ {u} // Add u to the lecturers
89  end
90
91  end

```

### 3.3 Examining the code

Firstly, we declare the variables we are going to use :

```

1 variables
2   registeredUsers //Set of registered users
3   administrators //Set of administrators
4   lecturers
5   loggedInUsers
6   loggingPassword

```

After this we are going to handle the invariants:

```

1 /*Handling the invariants */
2 invariants
3   @inv1: registeredUsers  $\subseteq$  USER
4   @inv2: administrators  $\subseteq$  registeredUsers
5   @inv3: lecturers  $\subseteq$  registeredUsers
6   @inv4: administrators  $\cap$  lecturers =  $\emptyset$ 
7   @inv5: rootUser  $\in$  administrators
8   @inv6: loggingPassword  $\in$  registeredUsers  $\rightarrow$  PASSWORD
9   @inv7: loggedInUsers  $\subseteq$  registeredUsers

```

After declaring the variables and invariants we are going to begin our events. Firstly, we write the INITIALISATION event which will let us initialise all of our variables/

```

1 events
2
3 /*Initializing our sets */
4 event INITIALISATION
5 begin
6   @beg1:registeredUsers :=  $\emptyset \cup \{\text{rootUser}\}$ 
7   @beg2:administrators :=  $\emptyset \cup \{\text{rootUser}\}$ 
8   @beg3:lecturers :=  $\emptyset$ 
9   @beg4:loggedInUsers :=  $\emptyset$ 
10  @beg5:loggingPassword :=  $\emptyset \cup \{\text{rootUser} \mapsto \text{defaultPassword}\}$ 
11 end

```

After the initialisation we do the log in event to allow a user to log in to the the online library. In order to log in there has to be a registered user and to enter his password correctly. If all of these were met than we add the user to the logged in users set.

```

1 /*The event to log in a user */
2 event LogIn
3 any u p where
4   @grd1:u  $\in$  registeredUsers //Check if u is a registered user
5   @grd2: p  $\in$  PASSWORD //Check if p is the set of PASSWORD
6   @grd3:loggingPassword(u) = p //Check if the user and its password are correct
7 then
8   @act1:loggedInUsers := loggedInUsers  $\cup \{u\}$  //Add the user to the logged in user set
9 end
10

```

```
11 /*The event to log out */
```

We similarly write the log out method which allows a logged in user to log out of the system.

```
1 /*The event to log out */
2 event Logout
3 any u where
4   @grd1: u ∈ loggedInUsers //Check if u is logged in
5 then
6   @act1: loggedInUsers := loggedInUsers \ {u} //Remove u from the logged in users
7 end
```

The following event is to allow an administrator to register a new user, only administrators can register new users with a new password.

```
1 /*The even to register a new user */
2 event RegisterUser
3 any a u p where
4   @grd1: u ∉ registeredUsers //Check if u is not part of the registeredUsers
5   @grd2: u ∈ USER //Check if u is part of USER
6   @grd3: a ∈ administrators //Check if a is administrator to allow registering a user
7   @grd4: p ∈ PASSWORD //Check if p is part of PASSWORD
8 then
9   /*Register the new user u and the password to this user */
10  @act1: registeredUsers := registeredUsers ∪ {u}
11  @act3: loggingPassword(u) := p
12 end
```

A user can change its password if he is logged in to the system.

```
1 /*The event to change the password */
2 event ChangePassword
3 any u p where
4   @grd1: u ∈ loggedInUsers //Check if u is a logged in user
5   @grd2: p ∈ PASSWORD //Check if p is part of PASSWORD
6 then
7   @act1: loggingPassword(u) := p //Change the password to p
8 end
```

An administrator can change a users status to administrator as well as to lecturer. A user is not allowed to be an administrator and lecturer at the same time.

```
1 /*The event to change to administrator */
2 event ChangeToAdministrator
3 any a u where
4   @grd1: a ∈ administrators //Check if a is and administrator
5   @grd2: u ∈ registeredUsers \ (lecturers ∪ administrators) //Check if u is part of
      registered users who are not lecturers or administrators
```

```

6  then
7    @act1: administrators := administrators  $\cup$  {u} // Add u to the administrators
8  end
9
10 /*The event to change to lecturer */
11 event ChangeToLecturer
12   any a u where
13     @grd1: a  $\in$  administrators // Check if a is an administrator
14     @grd2: u  $\in$  registeredUsers \ (administrators  $\cup$  lecturers) // Check if u is a registered
        user but not an administrator and not a lecturer
15   then
16     @act1: lecturers := lecturers  $\cup$  {u} // Add u to the lecturers
17   end
18
19 end

```

## 4 Resources

### 4.1 Context

```

1  context ResourceContext
2  sets
3  RESOURCE
4  TITLES
5  URLS
6  AUTHORS
7  ISBN
8  PUBLISHER
9  PUBLISHED_YEAR
10 end

```

In this context we determine the sets that we are going to use.

### 4.2 The code

```

1  machine OnlineLibrarySystem1
2  refines OnlineLibrarySystem
3  sees OnlineLibrarySystemContext ResourceContext
4
5  /*Declaring the variables we are going to use */
6  variables
7    registeredUsers
8    administrators
9    lecturers
10   loggedInUsers
11   loggingPassword
12   resource
13   book
14   article
15   website
16   authors

```

```

17  urls
18  titles
19  getAuthor
20  getURL
21  getTitle
22  publisher
23  publishedYear
24  isbn
25  getPublisher
26  getPublishedYear
27  getISBN
28
29  /*Declaring the invariants we are going to use */
30  invariants
31    @inv8:resource  $\subseteq$  RESOURCE
32    @inv9:partition(resource, book, website, article)
33    @inv10:authors  $\subseteq$  AUTHORS
34    @inv11:titles  $\subseteq$  TITLES
35    @inv12:urls  $\subseteq$  URLS
36    @inv13:getAuthor  $\in$  (resource  $\cup$  book  $\cup$  website  $\cup$  article)  $\leftrightarrow$  authors // we need to
        change this one
37    @inv14:getURL  $\in$  (resource)  $\mapsto$  urls
38    @inv15:getTitle  $\in$  (resource)  $\mapsto$  titles
39    @inv16:isbn  $\subseteq$  ISBN
40    @inv17:publisher  $\subseteq$  PUBLISHER
41    @inv18:publishedYear  $\subseteq$  PUBLISHED_YEAR
42    @inv19:getPublisher  $\in$  ((resource)  $\setminus$  (website))  $\rightarrow$  publisher
43    @inv20:getISBN  $\in$  book  $\mapsto$  isbn
44    @inv21:getPublishedYear  $\in$  ((resource)  $\setminus$  (website))  $\rightarrow$  publishedYear
45
46  events
47  /* We initialize everything what we are going to use (our variables) */
48  event INITIALISATION extends INITIALISATION
49  begin
50    @beg6:resource :=  $\emptyset$ 
51    @beg7:book :=  $\emptyset$ 
52    @beg8:article :=  $\emptyset$ 
53    @beg9:website :=  $\emptyset$ 
54    @beg10:authors :=  $\emptyset$ 
55    @beg11:urls :=  $\emptyset$ 
56    @beg12:titles :=  $\emptyset$ 
57    @beg13:getAuthor :=  $\emptyset$ 
58    @beg14:getURL :=  $\emptyset$ 
59    @beg15:getTitle :=  $\emptyset$ 
60    @beg16:publisher :=  $\emptyset$ 
61    @beg17:publishedYear :=  $\emptyset$ 
62    @beg18:isbn :=  $\emptyset$ 
63    @beg19:getPublisher :=  $\emptyset$ 
64    @beg20:getPublishedYear :=  $\emptyset$ 
65    @beg21:getISBN :=  $\emptyset$ 
66  end
67
68  event LogIn extends LogIn
69  end
70
71  event LogOut extends LogOut

```



```

72 end
73
74 event RegisterUser extends RegisterUser
75 end
76
77 event ChangePassword extends ChangePassword
78 end
79
80 event ChangeToAdministrator extends ChangeToAdministrator
81 end
82
83 event ChangeToLecturer extends ChangeToLecturer
84 end
85
86 /* The event to create a new book, this event can only be done by a logged in
   administrator */
87 event CreateBook
88   any a b publish publishedY tit ur is auth where
89     @grd1:a ∈ administrators // Check if a is an administrator
90     @grd2:a ∈ loggedInUsers // Check if a is logged in
91     @grd3:b ∈ RESOURCE // Check if b is part of the RESOURCE set
92     @grd4:b ∉ (book ∪ resource) // Check if b is not part of the resource ∪ book
93     /* We check if is, publishedY, publish, auth, ur, tit is part or not part of the
       appropriate sets */
94     @grd5:is ∈ ISBN
95     @grd6:is ∉ isbn
96     @grd7:publishedY ∈ PUBLISHED_YEAR
97     @grd8:publish ∈ PUBLISHER
98     @grd9:auth ∈ AUTHORS
99     @grd10:ur ∈ URLS
100    @grd11:ur ∉ urls
101    @grd12:tit ∈ TITLES
102    @grd13:tit ∉ titles
103  then
104    /* We add b to book and resources */
105    @act1:resource := resource ∪ {b}
106    @act2:book := book ∪ {b}
107    @act3:isbn := isbn ∪ {is}
108    @act4:publishedYear := publishedYear ∪ {publishedY}
109    @act5:publisher := publisher ∪ {publish}
110    @act6:authors := authors ∪ {auth}
111    @act7:urls := urls ∪ {ur}
112    @act8:titles := titles ∪ {tit}
113    @act9:getISBN(b) := is
114    @act10:getPublishedYear(b) := publishedY
115    @act11:getPublisher(b) := publish
116    @act12:getAuthor(b) := auth
117    @act13:getURL(b) := ur
118    @act14:getTitle(b) := tit
119  end
120
121 /* The event to remove a book, can only be done by a logged in administrator */
122 event RemoveBook
123   any b a where
124     @grd1:a ∈ administrators // Check if a is administrator
125     @grd2:a ∈ loggedInUsers // Check if a is logged in

```

```

126   @grd3: b ∈ (resource ∩ book) // Check if the b (book) is not part of the resource ∩
      book
127   then
128   /* We remove everything that is connected to b from everywhere */
129   @act1: book := book \ {b}
130   @act2: resource := resource \ {b}
131   @act3: isbn := isbn \ {getISBN(b)}
132   @act4: titles := titles \ {getTitle(b)}
133   @act5: urls := urls \ {getURL(b)}
134   @act6: getISBN := {b} ≪ getISBN
135   @act7: getTitle := {b} ≪ getTitle
136   @act8: getAuthor := {b} ≪ getAuthor
137   @act9: getPublisher := {b} ≪ getPublisher
138   @act10: getPublishedYear := {b} ≪ getPublishedYear
139   @act11: getURL := {b} ≪ getURL
140   end
141
142   /*The event to create a new website which can only be done by a logged in lecturer*/
143   event CreateWebsite
144   any l w auth ur tit where
145   @grd1: l ∈ (lecturers ∩ loggedInUsers) // Check if the l is a lecturer and if it is logged
      in
146   @grd2: w ∉ (resource ∪ website) // Check if w is not part of our resource ∪ website
147   @grd3: w ∈ RESOURCE // Check if w is part of RESOURCE
148   @grd4: auth ∈ AUTHORS // Check if auth is part of the AUTHORS
149   @grd5: ur ∈ URLS // Check if ur is part of URLS
150   @grd6: ur ∉ urls // Check if ur is not part of our urls
151   @grd7: tit ∈ TITLES // Check if tit is part of TITLE
152   @grd8: tit ∉ titles // Check if tit is not part of our titles
153   then
154   /*If all the grds are true we add the website with the author title and url */
155   @act1: resource := resource ∪ {w}
156   @act2: website := website ∪ {w}
157   @act3: authors := authors ∪ {auth}
158   @act4: urls := urls ∪ {ur}
159   @act5: titles := titles ∪ {tit}
160   @act6: getAuthor(w) := auth
161   @act7: getURL(w) := ur
162   @act8: getTitle(w) := tit
163   end
164
165   /*The event to create a new article which can only be done by a logged in lecturer */
166   event CreateArticle
167   any a l auth ur tit publishedY where
168   @grd1: l ∈ (lecturers ∩ loggedInUsers) // Check if l is a lecturer and if it is logged in
169   @grd2: a ∉ (resource ∪ article) // Check if a is not part of the resource ∪ article
170   @grd3: a ∈ RESOURCE // Check if a is part of RESOURCE
171   @grd4: auth ∈ AUTHORS // Check if auth is part of AUTHORS
172   @grd5: ur ∈ URLS // Check if ur is part of the URLS
173   @grd6: ur ∉ urls // Check if ur is not part of our urls
174   @grd7: tit ∈ TITLES // Check if tit is part of all TITLES
175   @grd8: tit ∉ titles // Check if tit is not part of our titles
176   @grd9: publishedY ∈ PUBLISHED_YEAR // Check if publishedY is part of
      PUBLISHED_YEAR
177   @grd10: publish ∈ PUBLISHER // Check if publish is part of PUBLISHER
178   then

```

```

179  /*We are going to add the article to the resource and article and we add all properties
      related to this article */
180  @act1: resource := resource  $\cup$  {a}
181  @act2: article := article  $\cup$  {a}
182  @act3: authors := authors  $\cup$  {auth}
183  @act4: urls := urls  $\cup$  {ur}
184  @act5: titles := titles  $\cup$  {tit}
185  @act6: getAuthor(a) := auth
186  @act7: getURL(a) := ur
187  @act8: getTitle(a) := tit
188  @act9: publishedYear := publishedYear  $\cup$  {publishedY}
189  @act10: publisher := publisher  $\cup$  {publish}
190  @act11: getPublishedYear(a) := publishedY
191  @act12: getPublisher(a) := publish
192  end
193
194  /*The event to remove an existing website */
195  event RemoveWebsite
196  any w l where
197    @grd1: l  $\in$  (lecturers  $\cap$  loggedInUsers) //Check if the user who wants to remove it is
      logged in and a lecturer as well
198    @grd2: w  $\in$  (resource  $\cap$  website) //Check if w is part of resource  $\cap$  website
199  then
200    /*We remove everything that is related to website w */
201    @act1: website := website  $\setminus$  {w}
202    @act2: resource := resource  $\setminus$  {w}
203    @act3: titles := titles  $\setminus$  {getTitle(w)}
204    @act4: urls := urls  $\setminus$  {getURL(w)}
205    @act5: getTitle := {w}  $\triangleleft$  getTitle
206    @act6: getAuthor := {w}  $\triangleleft$  getAuthor
207    @act7: getURL := {w}  $\triangleleft$  getURL
208  end
209
210  /*The event to remove an existing article */
211  event RemoveArticle
212  any a l where
213    @grd1: l  $\in$  (lecturers  $\cap$  loggedInUsers) //Check if the user l trying to remove the
      article is a lecturer and if it is logged in
214    @grd2: a  $\in$  (resource  $\cap$  article) //Check if a is part of resource  $\cap$  article
215  then
216    /*We remove everything related to article a */
217    @act1: article := article  $\setminus$  {a}
218    @act2: resource := resource  $\setminus$  {a}
219    @act3: titles := titles  $\setminus$  {getTitle(a)}
220    @act4: urls := urls  $\setminus$  {getURL(a)}
221    @act10: getTitle := {a}  $\triangleleft$  getTitle
222    @act6: getAuthor := {a}  $\triangleleft$  getAuthor
223    @act7: getURL := {a}  $\triangleleft$  getURL
224    @act8: getPublisher := {a}  $\triangleleft$  getPublisher
225    @act9: getPublishedYear := {a}  $\triangleleft$  getPublishedYear
226  end
227
228  /*The event to search by a title */
229  event SearchByTitle
230  any r t u where
231    @grd1: r  $\in$  resource

```

```

232     @grd2:t ∈ titles
233     @grd3: u ∈ loggedInUsers
234     @grd4:getTitle(r) = t
235 end
236
237 /*The event to search by ISBN */
238 event SearchByISBN
239 any b i u where
240     @grd1:b ∈ book
241     @grd2:i ∈ isbn
242     @grd3: u ∈ loggedInUsers
243     @grd4:getISBN(b) = i
244 end
245
246 end

```

### 4.3 Examining the code

This machine is refined from the OnlineLibrarySystem. First we determine the variables that we are going to use and we handle the invariants as well, and we allow the machine to see the context.

```

1 machine OnlineLibrarySystem1
2 refines OnlineLibrarySystem
3 sees OnlineLibrarySystemContext ResourceContext
4
5 /*Declaring the variables we are going to use */
6 variables
7     registeredUsers
8     administrators
9     lecturers
10    loggedInUsers
11    loggingPassword
12    resource
13    book
14    article
15    website
16    authors
17    urls
18    titles
19    getAuthor
20    getURL
21    getTitle
22    publisher
23    publishedYear
24    isbn
25    getPublisher
26    getPublishedYear
27    getISBN
28
29 /*Declaring the invariants we are going to use */
30 invariants
31     @inv8:resource ⊆ RESOURCE

```

```

32 @inv9:partition(resource, book, website, article)
33 @inv10:authors  $\subseteq$  AUTHORS
34 @inv11:titles  $\subseteq$  TITLES
35 @inv12:urls  $\subseteq$  URLS
36 @inv13:getAuthor  $\in$  (resource  $\cup$  book  $\cup$  website  $\cup$  article)  $\leftrightarrow$  authors // we need to
    change this one
37 @inv14:getURL  $\in$  (resource)  $\mapsto$  urls
38 @inv15:getTitle  $\in$  (resource)  $\mapsto$  titles
39 @inv16:isbn  $\subseteq$  ISBN
40 @inv17:publisher  $\subseteq$  PUBLISHER
41 @inv18:publishedYear  $\subseteq$  PUBLISHED_YEAR
42 @inv19:getPublisher  $\in$  ((resource)  $\setminus$  (website))  $\rightarrow$  publisher
43 @inv20:getISBN  $\in$  book  $\mapsto$  isbn
44 @inv21:getPublishedYear  $\in$  ((resource)  $\setminus$  (website))  $\rightarrow$  publishedYear
45

```

After handling the invariants we begin our events. The first one is the INITIALISATION, we initialise every set and variables that we are going to use.

```

1
2 events
3 /* We initialize everything what we are going to use (our variables) */
4 event INITIALISATION extends INITIALISATION
5 begin
6 @beg6:resource :=  $\emptyset$ 
7 @beg7:book :=  $\emptyset$ 
8 @beg8:article :=  $\emptyset$ 
9 @beg9:website :=  $\emptyset$ 
10 @beg10:authors :=  $\emptyset$ 
11 @beg11:urls :=  $\emptyset$ 
12 @beg12:titles :=  $\emptyset$ 
13 @beg13:getAuthor :=  $\emptyset$ 
14 @beg14:getURL :=  $\emptyset$ 
15 @beg15:getTitle :=  $\emptyset$ 
16 @beg16:publisher :=  $\emptyset$ 
17 @beg17:publishedYear :=  $\emptyset$ 
18 @beg18:isbn :=  $\emptyset$ 
19 @beg19:getPublisher :=  $\emptyset$ 
20 @beg20:getPublishedYear :=  $\emptyset$ 
21 @beg21:getISBN :=  $\emptyset$ 
22 end

```

After the initialisation we extend the already written events from the OnlineLibrarySystem.

```

1
2 event LogIn extends LogIn
3 end
4
5 event LogOut extends LogOut
6 end
7

```

```

8  event RegisterUser extends RegisterUser
9  end
10
11 event ChangePassword extends ChangePassword
12 end
13
14 event ChangeToAdministrator extends ChangeToAdministrator
15 end
16
17 event ChangeToLecturer extends ChangeToLecturer
18 end
19

```

The first event we write in this machine is the CreateBook event which allows an administrator to create a new book. We need to check if the title, authors published year, publisher are valid. Then we add them to the relevant functions and sets.

```

1  /* The event to create a new book, this event can only be done by a logged in
   administrator */
2  event CreateBook
3  any a b publish publishedY tit ur is auth where
4    @grd1:a ∈ administrators // Check if a is an administrator
5    @grd2:a ∈ loggedInUsers // Check if a is logged in
6    @grd3:b ∈ RESOURCE // Check if b is part of the RESOURCE set
7    @grd4:b ∉ (book ∪ resource) // Check if b is not part of the resource ∪ book
8    /* We check if is, publishedY, publish, auth, ur, tit is part or not part of the
       appropriate sets */
9    @grd5:is ∈ ISBN
10   @grd6: is ∉ isbn
11   @grd7:publishedY ∈ PUBLISHED_YEAR
12   @grd8:publish ∈ PUBLISHER
13   @grd9:auth ∈ AUTHORS
14   @grd10:ur ∈ URLS
15   @grd11: ur ∉ urls
16   @grd12:tit ∈ TITLES
17   @grd13: tit ∉ titles
18  then
19    /* We add b to book and resources */
20    @act1:resource := resource ∪ {b}
21    @act2:book := book ∪ {b}
22    @act3:isbn := isbn ∪ {is}
23    @act4:publishedYear := publishedYear ∪ {publishedY}
24    @act5:publisher := publisher ∪ {publish}
25    @act6:authors := authors ∪ {auth}
26    @act7:urls := urls ∪ {ur}
27    @act8:titles := titles ∪ {tit}
28    @act9: getISBN(b) := is
29    @act10: getPublishedYear(b) := publishedY
30    @act11: getPublisher(b) := publish
31    @act12: getAuthor(b) := auth
32    @act13: getURL(b) := ur
33    @act14: getTitle(b) := tit
34  end

```

As we have written the creation of a book we need to write the removal of one as well. Only an administrator can remove a book and he needs to be logged in. We remove every function and relation that is related to this book.

```

1
2  /* The event to remove a book, can only be done by a logged in administrator */
3  event RemoveBook
4    any b a where
5      @grd1: a ∈ administrators // Check if a is administrator
6      @grd2: a ∈ loggedInUsers // Check if a is logged in
7      @grd3: b ∈ (resource ∩ book) // Check if the b (book) is not part of the resource ∩
        book
8    then
9      /* We remove everything that is connected to b from everywhere */
10     @act1: book := book \ {b}
11     @act2: resource := resource \ {b}
12     @act3: isbn := isbn \ {getISBN(b)}
13     @act4: titles := titles \ {getTitle(b)}
14     @act5: urls := urls \ {getURL(b)}
15     @act6: getISBN := {b} ≪ getISBN
16     @act7: getTitle := {b} ≪ getTitle
17     @act8: getAuthor := {b} ≪ getAuthor
18     @act9: getPublisher := {b} ≪ getPublisher
19     @act10: getPublishedYear := {b} ≪ getPublishedYear
20     @act11: getURL := {b} ≪ getURL
21  end

```

The next event is creating a website. Lecturers can create websites and a website needs a valid author url and title.

```

1  /*The event to create a new website which can only be done by a logged in lecturer*/
2  event CreateWebsite
3    any l w auth ur tit where
4      @grd1: l ∈ (lecturers ∩ loggedInUsers) // Check if the l is a lecturer and if it is logged
        in
5      @grd2: w ∉ (resource ∪ website) // Check if w is not part of our resource ∪ website
6      @grd3: w ∈ RESOURCE // Check if w is part of RESOURCE
7      @grd4: auth ∈ AUTHORS // Check if auth is part of the AUTHORS
8      @grd5: ur ∈ URLS // Check if ur is part of URLS
9      @grd6: ur ∉ urls // Check if ur is not part of our urls
10     @grd7: tit ∈ TITLES // Check if tit is part of TITLE
11     @grd8: tit ∉ titles // Check if tit is not part of our titles
12  then
13    /*If all the grds are true we add the website with the author title and url */
14    @act1: resource := resource ∪ {w}
15    @act2: website := website ∪ {w}
16    @act3: authors := authors ∪ {auth}
17    @act4: urls := urls ∪ {ur}
18    @act5: titles := titles ∪ {tit}
19    @act6: getAuthor(w) := auth
20    @act7: getURL(w) := ur
21    @act8: getTitle(w) := tit

```

22     **end**

Now we need to create an article as well. Just in the case of creating a website only lecturers are allowed to create and besides the valid author, url and title it also needs a valid publisher and published year.

```
1  /*The event to create a new article which can only be done by a logged in lecturer */
2  event CreateArticle
3  any a l auth ur tit publish publishedY where
4      @grd1: l ∈ (lecturers ∩ loggedInUsers) //Check if l is a lecturer and if it is logged in
5      @grd2: a ∉ (resource ∪ article) // Check if a is not part of the resource ∪ article
6      @grd3: a ∈ RESOURCE // Check if a is part of RESOURCE
7      @grd4: auth ∈ AUTHORS // Check is auth is part of AUTHORS
8      @grd5: ur ∈ URLS //Check if ur is part of the URLS
9      @grd6: ur ∉ urls // Check if ur is not part of our urls
10     @grd7: tit ∈ TITLES //Check if tit is part of all TITLES
11     @grd8: tit ∉ titles // Check if tit is not part of our titles
12     @grd9: publishedY ∈ PUBLISHED_YEAR //Check if publishedY is part of
        PUBLISHED_YEAR
13     @grd10: publish ∈ PUBLISHER //Check if publish is part of PUBLISHER
14 then
15     /*We are going to add the article to the resource and article and we add all properties
        related to this article */
16     @act1: resource := resource ∪ {a}
17     @act2: article := article ∪ {a}
18     @act3: authors := authors ∪ {auth}
19     @act4: urls := urls ∪ {ur}
20     @act5: titles := titles ∪ {tit}
21     @act6: getAuthor(a) := auth
22     @act7: getURL(a) := ur
23     @act8: getTitle(a) := tit
24     @act9: publishedYear := publishedYear ∪ {publishedY}
25     @act10: publisher := publisher ∪ {publish}
26     @act11: getPublishedYear(a) := publishedY
27     @act12: getPublisher(a) := publish
28 end
```

The next event is responsible for removing a website. Logged in lecturers can remove websites, and it deletes all of the related information to it.

```
1  /*The event to remove an existing website */
2  event RemoveWebsite
3  any w l where
4      @grd1: l ∈ (lecturers ∩ loggedInUsers) //Check if the user who wants to remove it is
        logged in and a lecturer as well
5      @grd2: w ∈ (resource ∩ website) //Check if w is part or resource ∩ website
6  then
7      /*We remove everything that is related to website w */
8      @act1: website := website \ {w}
9      @act2: resource := resource \ {w}
10     @act3: titles := titles \ {getTitle(w)}
11     @act4: urls := urls \ {getURL(w)}
```



```

12    @act5:getTitle := {w}  $\triangleleft$  getTitle
13    @act6:getAuthor := {w}  $\triangleleft$  getAuthor
14    @act7:getURL := {w}  $\triangleleft$  getURL
15    end

```

Same as in the case of removing a website only logged in lecturers can remove an article.

```

1  /*The event to remove an existing article */
2  event RemoveArticle
3  any a l where
4    @grd1: l  $\in$  (lecturers  $\cap$  loggedInUsers) //Check if the user l trying to remove the
      article is a lecturer and if it is logged in
5    @grd2: a  $\in$  (resource  $\cap$  article) //Check if a is part of resource  $\cap$  article
6  then
7    /*We remove everything related to article a */
8    @act1: article := article \ {a}
9    @act2: resource := resource \ {a}
10   @act3: titles := titles \ {getTitle(a)}
11   @act4: urls := urls \ {getURL(a)}
12   @act10:getTitle := {a}  $\triangleleft$  getTitle
13   @act6:getAuthor := {a}  $\triangleleft$  getAuthor
14   @act7:getURL := {a}  $\triangleleft$  getURL
15   @act8:getPublisher := {a}  $\triangleleft$  getPublisher
16   @act9:getPublishedYear := {a}  $\triangleleft$  getPublishedYear
17   end
18

```

The last 2 events of this part is responsible for searching a resource by title or in the case of a book by ISBN.

```

1  /*The event to search by a title */
2  event SearchByTitle
3  any r t u where
4    @grd1: r  $\in$  resource
5    @grd2: t  $\in$  titles
6    @grd3: u  $\in$  loggedInUsers
7    @grd4: getTitle(r) = t
8  end
9
10 /*The event to search by ISBN */
11 event SearchByISBN
12 any b i u where
13   @grd1: b  $\in$  book
14   @grd2: i  $\in$  isbn
15   @grd3: u  $\in$  loggedInUsers
16   @grd4: getISBN(b) = i
17 end

```

## 5 Reading Lists

### 5.1 Context

```
1 context ReadingListContext
2 sets
3 LIST
4 end
```

### 5.2 The code

```
1 machine OnlineLibrarySystem2
2 refines OnlineLibrarySystem1
3 sees OnlineLibrarySystemContext ResourceContext ReadingListContext
4 variables
5   registeredUsers
6   administrators
7   lecturers
8   loggedInUsers
9   loggingPassword
10
11 resource
12   book
13   article
14   website
15   authors
16   urls
17   titles
18   getAuthor
19   getURL
20   getTitle
21   publisher
22   publishedYear
23   isbn
24   getPublisher
25   getPublishedYear
26   getISBN
27
28   lists
29   getResources
30   getCreator
31
32 /*Handling the invariants */
33 invariants
34   @inv22:lists  $\subseteq$  LIST //Making lists part of LIST
35   @inv23:getResources  $\in$  lists  $\leftrightarrow$  resource //Creating getResources which is a relation
        between lists and resource
36   @inv24:getCreator  $\in$  lists  $\rightarrow$  lecturers //getCreator is a total function between lists and
        lecturers
37
38 /*The events that the machine is going to do */
39 events
40   /*Initialising the invariants*/
```

```

41 event INITIALISATION extends INITIALISATION
42 begin
43   @beg22:lists := ∅
44   @beg23:getResources := ∅
45   @beg24:getCreator := ∅
46 end
47
48 event LogIn extends LogIn
49 end
50
51 event LogOut extends LogOut
52 end
53
54 event RegisterUser extends RegisterUser
55 end
56
57 event ChangePassword extends ChangePassword
58 end
59
60 event ChangeToAdministrator extends ChangeToAdministrator
61 end
62
63 event ChangeToLecturer extends ChangeToLecturer
64 end
65
66 event CreateBook extends CreateBook
67 end
68
69 event RemoveBook extends RemoveBook
70 then
71   @act12:getResources := getResources ▷ {b} //We remove b from the getResources
       when we remove a book
72 end
73
74 event CreateWebsite extends CreateWebsite
75 end
76
77 event CreateArticle extends CreateArticle
78 end
79
80 event RemoveWebsite extends RemoveWebsite
81 then
82   @act8:getResources := getResources ▷ {w} //We remove w from getResources when
       we remove a website
83 end
84
85 event RemoveArticle extends RemoveArticle
86 then
87   @act10:getResources := getResources ▷ {a} //We remove a from getResources when
       we remove an article
88 end
89
90 event SearchByTitle extends SearchByTitle
91 end
92
93 event SearchByISBN extends SearchByISBN

```

```

94  end
95
96  /*The event to create a new reading list*/
97  event CreateReadingList
98  any lect lis where
99    @grd1:lect ∈ lecturers //Checking if lect is a lecturer
100   @grd2:lect ∈ loggedInUsers //Checking if lect is logged in
101   @grd3:lis ∈ LIST //Checking if lis is part of LIST
102   @grd4:lis ∉ lists //Checking if lis is not part of lists
103  then
104    @act1:lists := lists ∪ {lis}
105    @act2:getCreator(lis) := lect
106  end
107
108  /*The event to remove a reading list*/
109  event RemoveReadingList
110  any lect lis where
111    @grd1:lect ∈ lecturers //Checking if lect is a lecturer
112    @grd2:lect ∈ loggedInUsers //Checking if lect is logged in
113    @grd3:lis ∈ lists //Checking if lis is part of lists
114    @grd4:getCreator(lis) = lect //Cheking if the creator of the lis is lect
115  then
116    @act1:getCreator := {lis} ⋈ getCreator
117    @act2:getResources := {lis} ⋈ getResources
118    @act3:lists := lists \ {lis}
119  end
120
121  /*The event to add a resource to the list */
122  event AddResourceToList
123  any lect lis res where
124    @grd1:lect ∈ lecturers //Checking if lect is a lecturer
125    @grd2:lect ∈ loggedInUsers //Cheking if lect is logged in
126    @grd3:lis ∈ lists //Checking if lis is part of lists
127    @grd4:getCreator(lis) = lect //Cheking if the creator of this list is lect
128    @grd5:res ∈ resource //Cheking if res is part of resource
129  then
130    @act1:getResources(lis) := res
131  end
132
133  /*The event to remove a resource from the list */
134  event RemoveResourceFromList
135  any lect lis res where
136    @grd1:lect ∈ lecturers //Cheking if lect is a lecturer
137    @grd2:lect ∈ loggedInUsers //Cheking if lect is logged in
138    @grd3:lis ∈ lists //Checking if lis is part of lists
139    @grd4:getCreator(lis) = lect //Checking if the creator of lis is the lecturer lect
140    @grd5:res ∈ resource //Checking if res is part of resource
141  then
142    @act1:getResources := getResources \ {lis ↦ res}
143  end
144
145  end

```

### 5.3 Examining the code

Firstly, we refine the OnlineLibrarySystem1 and then we declare the variables that we are going to use.

```
1 machine OnlineLibrarySystem2
2 refines OnlineLibrarySystem1
3 sees OnlineLibrarySystemContext ResourceContext ReadingListContext
4 variables
5   registeredUsers
6   administrators
7   lecturers
8   loggedInUsers
9   loggingPassword
10
11 resource
12 book
13 article
14 website
15 authors
16 urls
17 titles
18 getAuthor
19 getURL
20 getTitle
21 publisher
22 publishedYear
23 isbn
24 getPublisher
25 getPublishedYear
26 getISBN
27
28 lists
29 getResources
30 getCreator
31
```

Then we handle the invariants with appropriate notations.

```
1 /*Handling the invariants */
2 invariants
3 @inv22:lists  $\subseteq$  LIST //Making lists part of LIST
4 @inv23:getResources  $\in$  lists  $\leftrightarrow$  resource //Creating getResources which is a relation
   between lists and resource
5 @inv24:getCreator  $\in$  lists  $\rightarrow$  lecturers //getCreator is a total function between lists and
   lecturers
```

The first event we are going to do is the INITIALISATION in which we initialise all our invariants.

```
1 /*The events that the machine is going to do */
2 events
```

```

3  /*Initialising the invariants*/
4  event INITIALISATION extends INITIALISATION
5  begin
6    @beg22:lists := ∅
7    @beg23:getResources := ∅
8    @beg24:getCreator := ∅
9  end

```

Due to refining of the OnlineLibrarySystem1 we get the following events extended. We add another act to removeWebsite, removeBook and removeArticle which allows us to remove the website, book or article from getResources if it was removed.

```

1
2  event Login extends Login
3  end
4
5  event Logout extends Logout
6  end
7
8  event RegisterUser extends RegisterUser
9  end
10
11 event ChangePassword extends ChangePassword
12 end
13
14 event ChangeToAdministrator extends ChangeToAdministrator
15 end
16
17 event ChangeToLecturer extends ChangeToLecturer
18 end
19
20 event CreateBook extends CreateBook
21 end
22
23 event RemoveBook extends RemoveBook
24 then
25   @act12:getResources := getResources ▷ {b} //We remove b from the getResources
   when we remove a book
26 end
27
28 event CreateWebsite extends CreateWebsite
29 end
30
31 event CreateArticle extends CreateArticle
32 end
33
34 event RemoveWebsite extends RemoveWebsite
35 then
36   @act8:getResources := getResources ▷ {w} //We remove w from getResources when
   we remove a website
37 end
38
39 event RemoveArticle extends RemoveArticle

```

```

40 then
41   @act10:getResources := getResources  $\triangleright$  {a} //We remove a from getResources when
      we remove an article
42 end
43
44 event SearchByTitle extends SearchByTitle
45 end
46
47 event SearchByISBN extends SearchByISBN
48 end
49

```

The CreateReadingList event is responsible for creating a new reading list. Only lecturers can create reading list and by creating one they become its owner.

```

1  /*The event to create a new reading list*/
2  event CreateReadingList
3  any lect lis where
4    @grd1:lect  $\in$  lecturers //Checking if lect is a lecturer
5    @grd2:lect  $\in$  loggedInUsers //Checking if lect is logged in
6    @grd3:lis  $\in$  LIST //Checking if lis is part of LIST
7    @grd4:lis  $\notin$  lists //Checking if lis is not part of lists
8  then
9    @act1:lists := lists  $\cup$  {lis}
10   @act2:getCreator(lis) := lect
11 end

```

We also need to remove a reading list. Only the lecturer who created this reading list can delete it. It deletes every relation connected to this reading list.

```

1  /*The event to remove a reading list*/
2  event RemoveReadingList
3  any lect lis where
4    @grd1:lect  $\in$  lecturers //Checking if lect is a lecturer
5    @grd2:lect  $\in$  loggedInUsers //Checking if lect is logged in
6    @grd3:lis  $\in$  lists //Checking if lis is part of lists
7    @grd4:getCreator(lis) = lect //Checking if the creator of the lis is lect
8  then
9    @act1:getCreator := {lis}  $\triangleleft$  getCreator
10   @act2:getResources := {lis}  $\triangleleft$  getResources
11   @act3:lists := lists  $\setminus$  {lis}
12 end

```

We need to be able to add resources to a reading list. Only the creator lecturer can add to this reading list.

```

1  /*The event to add a resource to the list */
2  event AddResourceToList
3  any lect lis res where

```

```

4  @grd1:lect ∈ lecturers//Checking if lect is a lecturer
5  @grd2:lect ∈ loggedInUsers//Cheking if lect is logged in
6  @grd3:lis ∈ lists//Checking if lis is part of lists
7  @grd4:getCreator(lis) = lect//Cheking if the creator of this list is lect
8  @grd5:res ∈ resource//Cheking if res is part of resource
9  then
10 @act1:getResources(lis) := res
11 end
12

```

The RemoveResourceFromList event is responsible for removing a resource from a reading list. Only the creator can remove resources from a reading list.

```

1  /*The event to remove a resource from the list */
2  event RemoveResourceFromList
3  any lect lis res where
4  @grd1:lect ∈ lecturers//Cheking if lect is a lecturer
5  @grd2:lect ∈ loggedInUsers//Cheking if lect is logged in
6  @grd3:lis ∈ lists//Checking if lis is part of lists
7  @grd4:getCreator(lis) = lect//Checking if the creator of lis is the lecturer lect
8  @grd5:res ∈ resource//Checking if res is part of resource
9  then
10 @act1:getResources := getResources \ {lis ↦ res}
11 end

```

## 6 Borrowing books

### 6.1 Context

```

1  context BorrowingContext
2  sets
3  QUEUE
4  end

```

### 6.2 The code

```

1  machine OnlineLibrarySystem3
2  refines OnlineLibrarySystem2
3  sees OnlineLibrarySystemContext ResourceContext ReadingListContext
   BorrowingContext
4
5  /*We declare the variables that we are going to use */
6  variables
7  registeredUsers
8  administrators
9  lecturers
10 loggedInUsers
11 loggingPassword
12

```



```

13 resource
14 book
15 article
16 website
17 authors
18 urls
19 titles
20 getAuthor
21 getURL
22 getTitle
23 publisher
24 publishedYear
25 isbn
26 getPublisher
27 getPublishedYear
28 getISBN
29 lists
30 getResources
31 getCreator
32
33 getTotalTokens
34 getCurrentTokens
35 getBooks
36
37 getReserves
38 queue
39 bookQueues
40 getStudent
41
42 /*Handling the invariants */
43 invariants
44 @inv25:getTotalTokens ∈ book → ℕ
45 @inv26:getCurrentTokens ∈ book → ℕ
46 @inv27:getBooks ∈ registeredUsers ↔ book
47 @inv28:getReserves ∈ book → ℕ
48 @inv29:getStudent ∈ ℕ ↔ registeredUsers
49 @inv30:queue ⊆ QUEUE
50 @inv31:bookQueues ∈ book → queue
51 events
52 event INITIALISATION extends INITIALISATION
53 begin
54 /*Setting every invariant initially to an empty set
55 */
56 @beg25:getTotalTokens := ∅
57 @beg26:getCurrentTokens := ∅
58 @beg27:getBooks := ∅
59 @beg28:getReserves := ∅
60 @beg29:getStudent := ∅
61 @beg30:queue := ∅
62 @beg31:bookQueues := ∅
63 end
64
65 event LogIn extends LogIn
66 end
67
68 event LogOut extends LogOut

```

```

69  end
70
71  event RegisterUser extends RegisterUser
72  end
73
74  event ChangePassword extends ChangePassword
75  end
76
77  event ChangeToAdministrator extends ChangeToAdministrator
78  end
79
80  event ChangeToLecturer extends ChangeToLecturer
81  end
82
83  event CreateBook extends CreateBook
84  end
85
86  event RemoveBook extends RemoveBook
87  end
88
89  event CreateWebsite extends CreateWebsite
90  end
91
92  event CreateArticle extends CreateArticle
93  end
94
95  event RemoveWebsite extends RemoveWebsite
96  end
97
98  event RemoveArticle extends RemoveArticle
99  end
100
101  event SearchByTitle extends SearchByTitle
102  end
103
104  event SearchByISBN extends SearchByISBN
105  end
106
107  event CreateReadingList extends CreateReadingList
108  end
109
110  event RemoveReadingList extends RemoveReadingList
111  end
112
113  event AddResourceToList extends AddResourceToList
114  end
115
116  event RemoveResourceFromList extends RemoveResourceFromList
117  end
118
119  /*The event to borrow a book onlu a registered and logged in user can borrow a book if
    there is still license
120  * for that book
121  */
122  event BorrowBook
123  any u b where

```

```

124 @grd1:b ∈ book //Checking if b is part of book
125 @grd2:getTotalTokens(b) > getCurrentTokens(b) // Checking if there are still tokens
    available to borrow a book
126 @grd3:u ∈ registeredUsers // Checking if u is a registered user
127 @grd4:u ∈ loggedInUsers //Checking if u is logged in
128 then
129 /*We increase the curretnTokens for this book by one and assign the book */
130 @act1:getCurrentTokens(b) := getCurrentTokens(b) + 1
131 @act2:getBooks(u) := b
132 end
133
134 /*The event to reserve a book only a logged in user can reserve a book */
135 event ReserveBook
136 any u b where
137 @grd1:b ∈ book//Checking if b is part of book
138 @grd2:getTotalTokens(b) ≤ getCurrentTokens(b)//Ckecking if there are no more
    tokens available
139 @grd3:u ∈ registeredUsers // Checking if u is a registered user
140 @grd4:u ∈ loggedInUsers//Checking if u is logged in
141 then
142 @act1:getReserves(b) := getReserves(b) + 1
143 @act2:getStudent(getReserves(b)) := u
144 end
145
146 /*The event to return a book that can be done by a user */
147 event ReturnBook
148 any u b where
149 @grd1:b ∈ book//Checking if b is part of book
150 @grd2:u ∈ registeredUsers//Checking if u is a registered user
151 @grd3:u ∈ loggedInUsers//Checking if u is a logged in user
152 then
153 /*Returning the book and decreasing the used tokens*/
154 @act1:getBooks := getBooks \ {u ↦ b}
155 @act2:getCurrentTokens(b) := getCurrentTokens(b) - 1
156 end
157
158 /*The event to revoke a book and it can be done by the administrator */
159 event RevokeBook
160 any a u b where
161 @grd1:b ∈ book//Checking if b is part of book
162 @grd2:a ∈ administrators//Checking if a is an administrator
163 @grd3:u ∈ registeredUsers//Checking if u is a registered user
164 @grd4:a ∈ loggedInUsers//Checking if a is logged in
165 then
166 /*We revoke the book */
167 @act1:getReserves(b) := getReserves(b) + 1
168 @act2:getStudent(getReserves(b)) := u
169 end
170
171 end

```

### 6.3 Examining the code

First, we have to declare the variables that we are going to use and we refined the OnlineLibrarySystem2.

```

1 machine OnlineLibrarySystem3
2 refines OnlineLibrarySystem2
3 sees OnlineLibrarySystemContext ResourceContext ReadingListContext
   BorrowingContext
4
5 /*We declare the variables that we are going to use */
6 variables
7   registeredUsers
8   administrators
9   lecturers
10  loggedInUsers
11  loggingPassword
12
13  resource
14  book
15  article
16  website
17  authors
18  urls
19  titles
20  getAuthor
21  getURL
22  getTitle
23  publisher
24  publishedYear
25  isbn
26  getPublisher
27  getPublishedYear
28  getISBN
29  lists
30  getResources
31  getCreator
32
33  getTotalTokens
34  getCurrentTokens
35  getBooks
36
37  getReserves
38  queue
39  bookQueues
40  getStudent

```

After declaring the variables we handle the invariants and and initialise these invariants.

```

1 /*Handling the invariants */
2 invariants
3 @inv25:getTotalTokens ∈ book → ℕ
4 @inv26:getCurrentTokens ∈ book → ℕ
5 @inv27:getBooks ∈ registeredUsers ↔ book
6 @inv28:getReserves ∈ book → ℕ
7 @inv29:getStudent ∈ ℕ → registeredUsers
8 @inv30:queue ⊆ QUEUE
9 @inv31:bookQueues ∈ book → queue

```

```

10  events
11  event INITIALISATION extends INITIALISATION
12  begin
13  /*Setting every invariant initially to an empty set
14  */
15  @beg25:getTotalTokens := ∅
16  @beg26:getCurrentTokens := ∅
17  @beg27:getBooks := ∅
18  @beg28:getReserves := ∅
19  @beg29:getStudent := ∅
20  @beg30:queue := ∅
21  @beg31:bookQueues := ∅
22  end

```

We extend the events from previous machines.

```

1
2  event LogIn extends LogIn
3  end
4
5  event LogOut extends LogOut
6  end
7
8  event RegisterUser extends RegisterUser
9  end
10
11  event ChangePassword extends ChangePassword
12  end
13
14  event ChangeToAdministrator extends ChangeToAdministrator
15  end
16
17  event ChangeToLecturer extends ChangeToLecturer
18  end
19
20  event CreateBook extends CreateBook
21  end
22
23  event RemoveBook extends RemoveBook
24  end
25
26  event CreateWebsite extends CreateWebsite
27  end
28
29  event CreateArticle extends CreateArticle
30  end
31
32  event RemoveWebsite extends RemoveWebsite
33  end
34
35  event RemoveArticle extends RemoveArticle
36  end
37
38  event SearchByTitle extends SearchByTitle
39  end

```

```

40
41 event SearchByISBN extends SearchByISBN
42 end
43
44 event CreateReadingList extends CreateReadingList
45 end
46
47 event RemoveReadingList extends RemoveReadingList
48 end
49
50 event AddResourceToList extends AddResourceToList
51 end
52
53 event RemoveResourceFromList extends RemoveResourceFromList
54 end
55

```

The first event we add to this machine is the BorrowBook event which allows a user to borrow a book if there are still available license for that book.

```

1  /*The event to borrow a book only a registered and logged in user can borrow a book if
   there is still license
2  * for that book
3  */
4  event BorrowBook
5  any u b where
6    @grd1:b ∈ book //Checking if b is part of book
7    @grd2:getTotalTokens(b) > getCurrentTokens(b) // Checking if there are still tokens
   available to borrow a book
8    @grd3:u ∈ registeredUsers // Checking if u is a registered user
9    @grd4:u ∈ loggedInUsers //Checking if u is logged in
10 then
11  /*We increase the currentTokens for this book by one and assign the book */
12  @act1:getCurrentTokens(b) := getCurrentTokens(b) + 1
13  @act2:getBooks(u) := b
14 end
15

```

The ReserveBook method lets a user to reserve a book if there is no token available to borrow it.

```

1  /*The event to reserve a book only a logged in user can reserve a book */
2  event ReserveBook
3  any u b where
4    @grd1:b ∈ book //Checking if b is part of book
5    @grd2:getTotalTokens(b) ≤ getCurrentTokens(b) //Checking if there are no more
   tokens available
6    @grd3:u ∈ registeredUsers // Checking if u is a registered user
7    @grd4:u ∈ loggedInUsers //Checking if u is logged in
8  then
9    @act1:getReserves(b) := getReserves(b) + 1
10   @act2:getStudent(getReserves(b)) := u
11 end

```

The ReturnBook method allows a user to return a book that was borrowed by him.

```

1  /*The event to return a book that can be done by a user */
2  event ReturnBook
3  any u b where
4    @grd1:b ∈ book//Checking if b is part of book
5    @grd2:u ∈ registeredUsers//Checking if u is a registered user
6    @grd3:u ∈ loggedInUsers//Checking if u is a logged in user
7  then
8    /*Returning the book and decreasing the used tokens*/
9    @act1:getBooks := getBooks \ {u ↦ b}
10   @act2:getCurrentTokens(b) := getCurrentTokens(b) - 1
11  end
12

```

The RevokeBook method allows an administrator to take a book from a user, the administrator needs to be logged in to perform this action.

```

1  /*The event to revoke a book and it can be done by the administrator */
2  event RevokeBook
3  any a u b where
4    @grd1:b ∈ book//Checking if b is part of book
5    @grd2:a ∈ administrators//Checking if a is an administrator
6    @grd3:u ∈ registeredUsers//Checking if u is a registered user
7    @grd4:a ∈ loggedInUsers//Checking if a is logged in
8  then
9    /*We revoke the book */
10   @act1:getReserves(b) := getReserves(b) + 1
11   @act2:getStudent(getReserves(b)) := u
12  end

```