Modelling an Online Library System using Event-B

COMP1216: Software Modelling and Design
University of Southampton
May 2021

Group 7
Victoria Dimitrova (vd2n20)
Archie French (af9g19)
Jordan Harle (jsh1g20)
Yaqin Hasan (ykh1e19)

Summary:

As part of the COMP1216 module and as a continuation of the first group coursework where a design specification for an online library system was created using UML, the library system was then formally modelled using Event-B as the second group coursework. The primary requirements of the system are summarized as follows:

- Users can login with their password (multiple device logins are ignored) and have different permission levels which determine what they're able to do within the system
- Books, websites and articles exist within the system as resources which have properties such as title, author and url.
- Reading lists can be created by lecturers within the system, which contain multiple resources.
- Books can be borrowed using a token system which limits the instances of that book being borrowed to the number of licenses it has, following which, users can request to be added to a first come first serve queue for that book.

Code:

The code is primarily split up into 3 files, the context file *LibraryContext.bucx* and the machine files *Library.bumx* and a refinement file *Library1.bumx*.

The code within the context file *LibraryContext.bucx* is as follows:

```
context
  LibraryContext
sets
  USER
  PASSWORD
  RESOURCE
  RESOURCETYPE
  TITLE
  AUTHOR
  URL
  PUBLISHER
  PUBLISHEDYEAR
  ISBN
  READINGLIST
  TOKEN
constants
  LEVEL // access level: 1 — student, 2 — lecturer, 3 — admin, 4 — root admin
  ROOTUSER // The root user
  ROOTPASS // The password of the root user
  BOOK // BOOK resource type
  ARTICLE // ARTICLE resource type
  WEBSITE // WEBSITE resource type
  POSITION
axioms
  @resource_partition: partition(RESOURCETYPE, {BOOK}, {ARTICLE}, {WEBSITE}) //
       3 RESOURCE types
  @def—LEVEL: LEVEL = 14 // 1 — student, 2 — lecturer, 3 — admin, 4 — root admin
  @def—ROOTUSER: ROOTUSER ∈ USER // rootuser is a user
  @def—ROOTPASS: ROOTPASS ∈ PASSWORD // rootpass is a password
  @def—POSITION: POSITION = ℕ// position is a natural number
end
```

The code within the machine file *Library.bumx* is as follows:

```
machine Library
sees LibraryContext

variables
  users // subset of USER container set
  resources // subset of RESOURCE container set
  register // mapping of users to their passwords
  permissions // mapping of users to their level
  loggedIn // subset of users that are currently logged in
  loggedOut // subset of users that are currently logged out
  title // relation assigning a resource to its title
  author // relation assigning a resource to its author
  url // relation assigning a resource to its url
  publisher // relation assigning a resource to its publisher
  publishedYear // relation assigning a resource to its published year
  isbn // relation assigning a resource to its isbn
  type // relation assigning a resource to its type


invariants
  @typeof—users: users ⊆USER
  @typeof—resources: resources ⊆RESOURCE
  @typeof—register: register ∈ users →PASSWORD
  @typeof—permissions: permissions ∈ users →LEVEL
  @typeof—loggedIn: loggedIn ⊆USER
  @typeof—loggedOut: loggedOut ⊆USER
  @typeof—title: title ∈ resources →TITLE
  @typeof—author: author ∈ resources →AUTHOR
  @typeof—url: url ∈ resources →URL
  @typeof—publisher: publisher ∈ resources ⇸ PUBLISHER
  @typeof—publishedYear: publishedYear ∈ resources ⇸ PUBLISHEDYEAR
  @typeof—isbn: isbn ∈ resources ⤔ ISBN
  @typeof—type: type ∈ resources →RESOURCETYPE
  @inv1: loggedIn ∩loggedOut = ∅
```

```
events

  event INITIALISATION // initialise variables
  begin
    @act1: register := {(ROOTUSER ↦ ROOTPASS)} // root user is registered at
           initialization
    @act2: permissions := {(ROOTUSER ↦ 4)} // root user is assigned root user
           permissions
    @act3: users := {ROOTUSER} // root user is added to users
    @act4: loggedIn := ∅
    @act5: loggedOut := {ROOTUSER} // root user starts logged out
    @act6: title := ∅
    @act7: author := ∅
    @act8: url := ∅
    @act9: publisher := ∅
    @act10: publishedYear := ∅
    @act11: isbn := ∅
    @act15: resources := ∅
    @act16: type := ∅
  end

  event NewUser
  any
    u // user to be registered
    p // password of the user
    l // level of the user
  where
    @grd1: u ∈ USER // @u is a user
    @grd2: p ∈ PASSWORD // @p is a password
    @grd3: l ∈ LEVEL // @l is a valid level
    @grd4: u ∉ users // @u isn't already registered
  then
    @act1: register(u) := p // assign password @p to the user @u
    @act2: permissions(u) := l // assign level @l to the user @u
    @act3: users := users ∪{u} // user is added to users set
    @act4: loggedOut := loggedOut ∪{u} // user starts logged out
  end

  event ChangePassword // changes the password of an existing user
  any
    u // user to have password changed
    p // new password
  where
    @grd1: u ∈ USER // @u is a user
    @grd2: p ∈ PASSWORD // @p is a password
    @grd3: u ∈ users // @u is already registered
  then
    @act1: register(u) := p // assign password @p to the user @u
  end
```

```
event AddBook // adds a book to the system
any
  b // book to be added
  t // title of book
  i // isbn
  u // url
  p // publisher
  y // published year
  a // author
where
  @grd1: b ∉ resources // @b isn't already a added
  @grd2: t ∈ TITLE // @t is a title
  @grd4: u ∈ URL // @u is a URL
  @grd5: p ∈ PUBLISHER // @p is a publisher
  @grd6: y ∈ PUBLISHEDYEAR // @y is a valid year
  @grd7: a ∈ AUTHOR // @a is an author
  @grd8: i ∉ ran(isbn) // @i isn't already in the isbn mapping
then
  @act1: resources := resources ∪{b} // book is added to resources set
  @act2: title(b) := t // book is assigned a title
  @act3: isbn(b) := i // book is assigned an isbn
  @act4: url(b) := u // book is assigned a url
  @act5: publisher(b) := p // book is assigned a publisher
  @act6: publishedYear(b) := y // book is assigned a published year
  @act7: author(b) := a // book is assigned an author
  @act8: type(b) := BOOK // book is assigned BOOK type
end

event RemoveBook // removes a book from the system
any
  b // book to be removed
where
  @grd1: b ∈ resources // @b is in the resources list
  @grd2: type(b) = BOOK // @b is a book
then
  @act1: resources := resources \{b} // book is removed from resources set
  @act2: title := {b} ⩤title // mapping of book to its title is removed
  @act3: isbn := {b} ⩤isbn // mapping of book to its isbn is removed
  @act4: url := {b} ⩤url // mapping of book to its url is removed
  @act5: publisher := {b} ⩤publisher // mapping of book to its publisher is removed
  @act6: publishedYear := {b} ⩤publishedYear // mapping of book to its published year
        is removed
  @act7: author := {b} ⩤author // mapping of book to its author is removed
  @act8: type := {b} ⩤type // mapping of book to its type is removed
end
```

```
event AddArticle // adds an article to the system
any
  a // Article to be added
  t // title of article
  u // url
  p // publisher
  y // published year
  w // author
where
  @grd1: a ∉ resources // @a isn't already added
  @grd2: t ∈ TITLE // @t is a title
  @grd3: u ∈ URL // @u is a URL
  @grd4: p ∈ PUBLISHER // @p is a publisher
  @grd5: y ∈ PUBLISHEDYEAR // @y is a valid year
  @grd6: w ∈ AUTHOR // @a is an author
then
  @act1: resources := resources ∪{a} // article is added to resources set
  @act2: title(a) := t // article is assigned a title
  @act3: url(a) := u // article is assigned a url
  @act4: publisher(a) := p // article is assigned a publisher
  @act5: publishedYear(a) := y // article is assigned a published year
  @act6: author(a) := w // article is assigned an author
  @act7: type(a) := ARTICLE // article is assigned ARTICLE type
end
event RemoveArticle // removes an article from the system
any
  a // article to be removed
where
  @grd1: a ∈ resources // @a is in the resource list
  @grd2: type(a) = ARTICLE // @a is an article
then
  @act1: resources := resources \{a} // article is removed from resources set
  @act2: title := {a} ⩤title // mapping of article to its title is removed
  @act3: url := {a} ⩤url // mapping of article to its url is removed
  @act4: publisher := {a} ⩤publisher // mapping of article to its publisher is removed
  @act5: publishedYear := {a} ⩤publishedYear // mapping of article to its published
       year is removed
  @act6: author := {a} ⩤author // mapping of article to its author is removed
  @act7: type := {a} ⩤type // mapping of article to its type is removed
end
```

```
event AddWebsite // adds a website to the system
any
  w // website to be added
  t // title of article
  u // url
  a // author
where
  @grd1: w ∉ resources // @w isn't already added
  @grd2: t ∈ TITLE // @t is a title
  @grd3: u ∈ URL // @u is a URL
  @grd4: a ∈ AUTHOR // @a is an author
then
  @act1: resources := resources ∪{w} // website is added to resources set
  @act2: title(w) := t // website is assigned a title
  @act3: url(w) := u // website is assigned a url
  @act4: author(w) := a // website is assigned a author
  @act5: type(w) := WEBSITE // @b is a website // website is assigned WEBSITE
        type
end
```

```
event RemoveWebsite // removes a website from the system
any
  a // website to be removed
where
  @grd1: a ∈ resources // @a is in the resources list
  @grd2: type(a) = WEBSITE // @a is a website
then
  @act1: resources := resources \{a} // website is removed from resources set
  @act2: title := {a} ⩤title // mapping of website to its title is removed
  @act3: url := {a} ⩤url // mapping of website to its url is removed
  @act4: author := {a} ⩤author // mapping of website to its author is removed
  @act5: type := {a} ⩤type // mapping of website to its type is removed
end
```

```
event SetAdmin
any
  u // user to have their permission level changed
  a // admin changing the permission level
where
  @grd1: u ∈ users // @u is a registered
  @grd2: a ∈ loggedIn // @a is a loggedIn user
  @grd3: permissions(a) = 3 ∨permissions(a) = 4 // @a is admin
then
  @act1: permissions(u) := 3 // set @u to admin account
end
```

```
event SetAdmin
any
  u // user to have their permission level changed
  a // admin changing the permission level
where
  @grd1: u ∈ users // @u is a registered
  @grd2: a ∈ loggedIn // @a is a loggedIn user
  @grd3: permissions(a) = 3 ∨permissions(a) = 4 // @a is admin
then
  @act1: permissions(u) := 3 // set @u to admin account
end

event SetLecturer
any
  u // user to have their permission level changed
  a // admin changing the permission level
where
  @grd1: u ∈ users // @u is a registerd user
  @grd2: a ∈ loggedIn // @a is a loggedIn user
  @grd3: permissions(a) = 3 ∨permissions(a) = 4 // @a is admin
then
  @act1: permissions(u) := 2 // set @u to lecturer account
end

event LogIn
any
  u // user to log in
where
  @grd1: u ∈ loggedOut // @u is logged out
then
  @act1: loggedIn := loggedIn ∪{u} //@u is now logged in
end

event LogOut
any
  u // user to log out
where
  @grd1: u ∈ loggedIn // @u is logged in
then
  @act1: loggedIn := loggedIn \{u} // @u is now not logged in
  @act2: loggedOut := loggedOut ∪{u} // @u is now logged out
end
```

The code within the machine file *Library1.bumx* is as follows:

```
machine Library1
refines Library
sees LibraryContext

variables
  users // subset of USER container set
  resources // subset of RESOURCE container set
  register // mapping of users to their passwords
  permissions // mapping of users to their level
  loggedIn // subset of users that are currently logged in
  loggedOut // subset of users that are currently logged out
  title // relation assigning a resource to its title
  author // relation assigning a resource to its author
  url // relation assigning a resource to its url
  publisher // relation assigning a resource to its publisher
  publishedYear // relation assigning a resource to its published year
  isbn // relation assigning a resource to its isbn
  type // relation assigning a resource to its type
  readingListCreator
  readingLists // subset of READINGLIST container set
  readingListContents // mapping of reading list to its resources
  licenseCount // mapping of resources (books) to their license count
  tokens // subset of TOKEN container set
  userToToken // mapping of users to tokens
  tokenToBook // mapping of tokens to resources (books)
  reservations // relation of resources (books) to users
  position // mapping of tokens to their position

invariants
  @typeof—readingLists: readingLists ⊆READINGLIST
  @typeof—readingListCreator: readingListCreator ∈ readingLists →users
  @typeof—readingListContents: readingListContents ∈ READINGLIST ↔ RESOURCE
  @typeof—licenseCount: licenseCount ∈ resources ⇸ ℕ
  @typeof—tokens: tokens ⊆TOKEN
  @typeof—userToToken: userToToken ∈ users ↔ tokens
  @typeof—tokenToBook: tokenToBook ∈ tokens →resources
  @typeof—reservations: reservations ∈ resources ↔ users
  @typeof—position: position ∈ tokens ⇸ POSITION
```

```
events
  event INITIALISATION extends INITIALISATION
  then
    @act18: readingLists := ∅
    @act19: readingListCreator := ∅
    @act20: readingListContents := ∅
    @act21: licenseCount := ∅
    @act22: tokens := ∅
    @act23: userToToken := ∅
    @act24: tokenToBook := ∅
    @act25: reservations := ∅
    @act26: position := ∅
  end

  event NewUser extends NewUser
  any x // admin creating the user
  where
    @grd5: x ∈ loggedIn // @x is logged in
    @grd6: permissions(x) = 3 ∨permissions(x) = 4 // @x is an admin
  end

  event ChangePassword extends ChangePassword
  where
    @grd4: u ∈ loggedIn // user is logged in
  end

   event AddBook extends AddBook
   any
     x // user adding the book
     l // number of licenses
   where
     @grd9: l ∈ ℕ1 // number of licenses is a positive natural number
     @grd10: x ∈ loggedIn // user is logged in
     @grd11: permissions(x) = 3 ∨permissions(x) = 4 // @x is an admin
   then
     @act9: licenseCount(b) := l // book is assigned a license count
   end

   event RemoveBook extends RemoveBook
   any
     x // user removing the book
   where
     @grd9: x ∈ loggedIn // user is logged in
     @grd10: permissions(x) = 3 ∨permissions(x) = 4 // @x is admin
   then
     @act9: licenseCount := {b} ⩤licenseCount // mapping of book to its license count is
            removed
   end
```

```
event AddArticle extends AddArticle
any
   x // user adding the book
where
   @grd9: x ∈ loggedIn // user is logged in
   @grd10: permissions(x) = 2 // @x is a lecturer
end

event RemoveArticle extends RemoveArticle
any
   x // user adding the book
where
   @grd9: x ∈ loggedIn
   @grd10: permissions(x) = 2 // @x is a lecturer
end

event AddWebsite extends AddWebsite
any
   x // user adding the book
where
   @grd9: x ∈ loggedIn
   @grd10: permissions(x) = 2 // @x is a lecturer
end

event RemoveWebsite extends RemoveWebsite
any
   x // user adding the book
where
   @grd9: x ∈ loggedIn
   @grd10: permissions(x) = 2 // @x is a lecturer
end

event SetAdmin extends SetAdmin
end

event SetLecturer extends SetLecturer
end

event LogIn extends LogIn
end

event LogOut extends LogOut
end
```

```
event CreateReadingList
any
  l // list to add
  u // user adding the list
where
  @grd1: l ∈ READINGLIST // @l is a reading list
  @grd2: u ∈ loggedIn // @u is logged in
  @grd3: permissions(u) = 2 // @u is a lecturer
  @grd4: l ∉ readingLists // @l isn't already added
then
  @act1: readingLists := readingLists ∪{l} // adding the reading list to its set
  @act2: readingListCreator(l) := u // assigning the reading list its creator
end

event RemoveReadingList
any
  l // list to remove
  u // user removing the list
where
  @grd1: l ∈ readingLists // @l is a reading list
  @grd2: u ∈ loggedIn // @u is logged in
  @grd3: permissions(u) = 2 // @u is a lecturer
then
  @act1: readingLists := readingLists \{l} // reading list it removed from its set
  @act2: readingListCreator := {l} ⩤readingListCreator // mapping of reading list to its
         creator is removed
end

event AddToReadingList
any
  l // list to add to
  u // user adding to the list
  r // resource to add to the list
where
  @grd1: l ∈ readingLists // @l is a reading list
  @grd2: u ∈ loggedIn // @u is logged in
  @grd3: permissions(u) = 2 // @u is a lecturer
  @grd4: r ∈ resources // @r is an existing resource
  @grd5: l ↦ r ∉ readingListContents // the resource isn't already a part of the reading
         list
  @grd6: readingListCreator(l) = u // mapping the reading list to its creator
then
  @act1: readingListContents := readingListContents ∪{l ↦ r} // reading list to resource
         mapping is added
end
```

```
event RemoveFromReadingList
any
  l // list to add to
  u // user adding to the list
  r // resource to add to the list
where
  @grd1: l ∈ readingLists // @l is a reading list
  @grd2: u ∈ loggedIn // @u is logged in
  @grd3: permissions(u) = 2 // @u is a lecturer
  @grd4: r ∈ resources // @r is an existing resource
  @grd5: l ↦ r ∈ readingListContents // resource exists in reading list
then
  @act1: readingListContents := readingListContents ⩥ {r} // resource is removed from
         reading list
end

event BorrowBook
any
  u // user requesting to borrow
  b // book to borrow
  t // token
where
  @grd1: u ∈ loggedIn // user is logged in
  @grd2: type(b) = BOOK // book is book
  @grd3: t ∈ TOKEN // generated token
  @grd4: t ∉ tokens // token doesn't already exist
  @grd5: licenseCount(b) ≥ 1 // there is still at least 1 license left
  @grd6: t ∉ dom(tokenToBook) // token doesn't already exist in the token to book
         function
  @grd7: t ∉ ran(userToToken) // token doesn't already exist in the user to token
         function
then
  @act1: userToToken := userToToken ∪{u ↦ t} // create a token for user borrowing
         book
  @act2: tokenToBook(t) := b // mapping token to book
  @act3: licenseCount(b) := licenseCount(b)  1 // decrement the licenseCount by 1
  @act4: tokens := tokens ∪{t} // add token to token set
end
```

```
event ReturnBook
any
  u // user requesting to return
  b // book to return
  t // token
where
  @grd1: $u \in loggedIn$ // user is logged in
  @grd2: $b \in resources$ // book is in resources
  @grd3: $t \in tokens$ // if token exists in tokens list
  @grd4: $type(b) = BOOK$ // book is book
  @grd5: $t \in ran(userToToken)$ // if token exists
  @grd6: $tokenToBook(t) = b$ // token is assigned to book
then
  @act1: $userToToken := userToToken \rhd \{t\}$ // remove user to token mapping
  @act2: $tokenToBook := \{t\} \lhd tokenToBook$ // remove token to book mapping
  @act3: $licenseCount(b) := licenseCount(b) + 1$ // increment the licenseCount by 1
  @act4: $tokens := tokens \setminus \{t\}$ // remove token from token set
end

event ReserveBook
any
  u // user requesting to reserve
  b // book to reserve
  p // position in the queue
  t // reservation token
where
  @grd1: $u \in loggedIn$ // user is logged in
  @grd2: $b \in resources$ // book is in resources
  @grd3: $type(b) = BOOK$ // book is book
  @grd4: $licenseCount(b) = 0$ // there are no licenses left
  @grd5: $p \in POSITION$
  @grd6: $t \notin tokens$ // t is not already in the tokens list
  @grd7: $b \mapsto u \notin reservations$ // @u hasn't already reserved @b
then
  @act1: $reservations := reservations \cup \{b \mapsto u\}$
  @act2: $position(t) := p$ // set the queue position of @t to @p (calculated above)
  @act3: $tokens := tokens \cup \{t\}$ // add token to tokens set
  @act4: $userToToken := userToToken \cup \{u \mapsto t\}$// create a token for user borrowing
        book
  @act5: $tokenToBook(t) := b$ // map token to book
end
```

```
event GetResourceByTitle // search for resource by title
any
  t // title of resource
  r // resource to get (return type)
where
  @grd1: t ∈ TITLE
  @grd2: r = title ∼[{t}]
end

event GetBookByISBN // search for book by ISBN
any
  i // isbn of book
  b // book to get (return type)
where
  @grd1: i ∈ ISBN
  @grd2: b ∈ RESOURCE
  @grd3: type(b) = BOOK
  @grd4: isbn(b) = i
end
```