MeteoCal
Requirements Analysis and Specification Document

Simone Graziussi

16th November 2014

Contents

I Introduction **3**

1 Purpose **3**

2 Objectives **3**

3 Definitions **4**

4 Stakeholders **4**

II The System **4**

5 Actors **5**

6 Assumptions **5**

7 Domain properties **6**

8 Scenarios **6**
8.1 Event Creation 6
8.2 Calendar browsing and Notifications 7
8.3 Invitations 7
8.4 Weather Forecast 7

III Requirements **7**

9 Features **7**

10 Functional Requirements **8**
10.1 Functions for Guests 8
10.2 Functions for Registered Users that do not own a calendar . . 9
10.3 Functions for Calendar Owners 9

11 Non Functional Requirements **9**
11.1 User interaction 9
11.2 User interface 9
11.3 Error handling 10
11.4 Quality of Service 10
11.5 Security 11
11.6 Documentation 11
11.7 Environment 11

IV UML Models **11**

12 Use Cases **12**
12.1 Registration and Log-in 12
12.2 Calendar browsing 14
12.3 Event management 15
12.4 Invitation management 17

13 Class Diagram **19**

14 Sequence Diagrams **20**
14.1 Registration and Log-in 20
14.2 Event management 21
14.3 Invitation management 22

15 State Charts **23**
15.1 Events 23
15.2 Notifications 23

V Alloy **23**

16 Model **23**

17 Analyzer **28**

18 Worlds **28**
18.1 Single user 28
18.2 Multiple users with invitations 28
18.3 Empty calendars 29
18.4 Users without calendar 29
18.5 Events without notifications 30
18.6 Calendar contains event but none organized by the owner . . 31

VI Conclusion **31**

19 Used Tools **31**

20 Change-log **31**

Part I
Introduction

1 Purpose

MeteoCal is an online calendar that offers the possibility to check real-time weather forecasts and manage events.

The system allows to easily browse through days, months and years of the calendar, with the possibility of reading, adding, deleting and updating everyday appointments.

MeteoCal's event management is an intuitive way to keep track of all important appointments: the user will be able to provide time and place information and then easily browse the daily schedule.

The calendar also offers the possibility to invite friends to an event. For each appointment the user will be able to send invitations to one or more other registered users, who will be notified and will accept or decline the proposal.

The relationship between calendar and weather forecast inside *MeteoCal* will prove very useful particularly for outdoor activities: in case of bad weather the system will promptly notify the user of the problem, allowing him/her to reschedule the event to the next sunny day.

2 Objectives

MeteoCal's goal is to provide the users an easy to use and reliable platform to store and track everyday events with particular regard to weather conditions. For this reason, the system should be able to provide the following functions

- register into the system
- easily browse the calendar
- read the weather forecast for the following days
- manage events
- read the schedule of created events for every day
- invite other users to events
- receive notifications about upcoming events or invitations

3 Definitions

- **Guest**: a user that has yet to register or log-in into the system
- **Registered user**: a user that logged into the system with a user-name and a password
- **Calendar owner**: a registered user that created his/her own calendar in the system
- **Event** or **Appointment**: an activity taking place in a specific place (outdoor or indoor) and during a specific interval of time
- **Organizer**: the registered user that created an event in his/her own calendar
- **Notification**: a system-generated message sent to a specific user that informs him/her of something related to an event
- **Invitation**: a special notification that asks whether the receiver will be able to come to an event organized by another user
- **Invited user**: a registered user that has been invited to an event and accepted the proposal
- **Weather Forecast**: a set of data describing the weather conditions (e.g. "raining") in a specific interval of time

4 Stakeholders

The main stakeholders for the *MeteoCal* project are the teachers of Politecnico di Milano's Software Engineering 2 class. They commissioned and will evaluate the project as part of the final examination of the course.

Considering the very high availability of similar products in the market, the hypothetical user interested in the system would be someone particularly interested in the *MeteoCal* key features: being able to send invitations to other users and to be notified of bad weather conditions in case of outdoor activities.

Part II
The System

5 Actors

- **Guest**: a user which is not registered to the system or not yet logged in
- **Registered user**: a user authenticated into the system. He/she can either own a calendar or not.
- **Calendar owner**: a registered user that owns a calendar, able to create, delete and update events.
- **Invited user**: a registered user that has been invited to an event by another user

6 Assumptions

Stakeholders didn't specify some properties in the request, so the developers made these assumptions:

- **User data**:
  - user-name: user-names must be at least four characters long; there cannot be two users with the same user-name
  - password: passwords must contain at least eight characters; they can contain letters, numbers and special characters
- **One calendar per user**: each registered user owns at most one calendar; calendars have a single owner
- **Bad weather**: the users will be able to specify their own idea of "bad" weather inside the calendar
- **Past events**: users cannot create events that take place before the current date
- **Simultaneous events**: two events cannot take place at the same time (in a generic time instant, there must be at most one event in progress)

7 Domain properties

- **User permissions**: a "Guest" user can only register or log into the system; a "Registered user" can create a calendar or manage it if it already exists and accept/decline other users' invitations
- **Calendar management**: a registered user can create, delete and update events only inside his/her own calendar. It is not allowed to change in any way the calendars of other users. Other users' events can be viewed only if invited by the calendar owner.
- **Event information**: for each event there must be
  - a date, with exact starting and ending time (of course, starting time must be before the ending time)
  - a place
  - a place description, specifying if it's outdoor or indoor

  Other non-essential information may be specified:
  - invited users: a list of registered users invited to event
  - weather forecast
- **Weather forecast linked to events**: each event is linked to a system-generated weather forecast (if available). The association is made at the event creation and the system can notify the user in case of bad weather during an outdoor event. These notifications are sent the day before the event.
- **Invitations**: a registered user may choose to invite one or more other users to an event. A user invited to an event may only accept or decline the invitation. Users cannot invite themselves or unregistered users to their events.
- **Notifications**: a registered user can read system-generated notifications about the events. Users cannot create notifications nor read other users' notifications. Notifications are received when the user logs into the system.

8 Scenarios

8.1 Event Creation

On Monday, John decides to try *MeteoCal* for managing his work and personal events. He opens the platform, registers into the system and then logs in. The first thing he does is creating some personal events like going to

the dentists on Wednesday at 2 PM and playing football with his friends on Sunday morning. For the latter appointment, he invites other 10 friends to join him.

8.2 Calendar browsing and Notifications

On Wednesday morning John opens *MeteoCal* and logs in again. He is immediately notified that five of his friends accepted his invitation for the football match. The system then allows him to check the day schedule and he is reminded of this afternoon dentist appointment at 2 PM.

8.3 Invitations

On Thursday, when John logs in, the system shows a notification to inform him that his friend Jack invited him to his birthday party, this Friday at 8 PM, and he accepts. John also reads other 5 notifications: 2 friends accepted and 3 declined the invitation to the football match next Sunday.

8.4 Weather Forecast

On Saturday John, after logging in, is notified that the weather forecast for tomorrow is bad and that the outdoor football match should be rescheduled. He checks the weather forecast for the next days and notices that Tuesday should be sunny. He decides to set the event on Tuesday at 6 PM.

Part III
Requirements

9 Features

For each goal, *MeteoCal* should provide the following features:

- **register into the system**
  - the systems should allow users to register with a user-name
  - after successful registration, the systems should allow users to log-in with their user-names and passwords
- **easily browse the calendar**
  - the systems should allow to switch month by clicking on the navigation arrows
  - the systems should allow to read day details by clicking on the day cell inside the month table

- **read the weather forecast for the following days**
  - for the 2/3 days following the current one, in the details there should be a recap about weather forecasts
- **manage events**
  - the system should allow to create new events
  - the system should allow to view the details of events previously created
  - the system should allow to update or delete existing events
- **read the schedule of created events for every day**
  - in each day details the system should provide a list of events ordered by time period
- **invite other users to events**
  - in the event creation form the system should allow the user to select one or more other users for invitation
  - the invited user(s) should be promptly notified of the invitation
  - the invited user(s) should be able to accept or decline the invitations
- **receive notifications about upcoming events or invitations**
  - at each log-in, the system should display a list of unread notifications

10 Functional Requirements

The *MeteoCal* system should provide different functions for each type of user. In particular, unregistered users or users that do not own a calendar yet will be able to use only a subset of the system features.

10.1 Functions for Guests

Unregistered users will be able to perform only two actions:

- register into the system, providing user-name, password and other optional information
- log into the system, by inserting correct user-name and password

10.2 Functions for Registered Users that do not own a calendar

Registered users that do not own a calendar yet can only:

- create a calendar, selecting their idea of "bad" weather
- read and manage notifications about other users' invitations

10.3 Functions for Calendar Owners

Registered users that already created a calendar will be able to explore the full *MeteoCal* system:

- view the whole calendar structure
- read existing events on his/her own calendar
- create new events
- delete events
- update events
- read and manage notification about his/her own calendar events
- read and manage notifications about other users' invitations

11 Non Functional Requirements

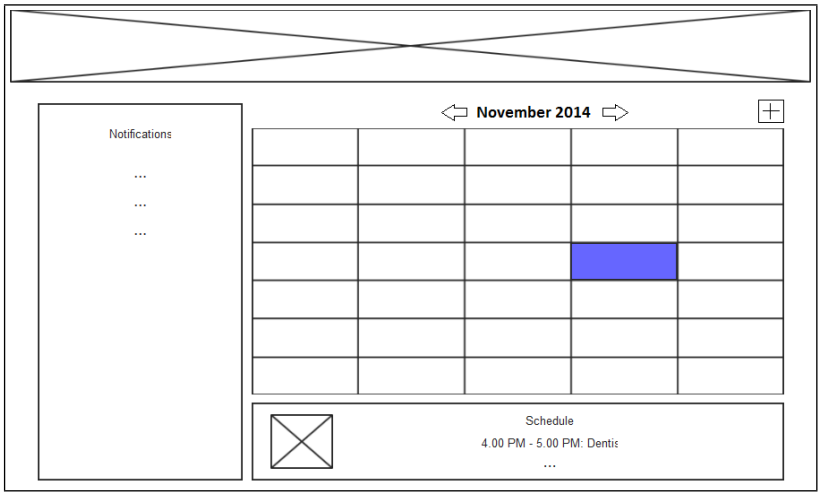
11.1 User interaction

- users only need a normal device (computer, smartphone, tablet) connected to the internet to access the system
- users should be able to easily learn how the system works
- all system features should be easily reachable

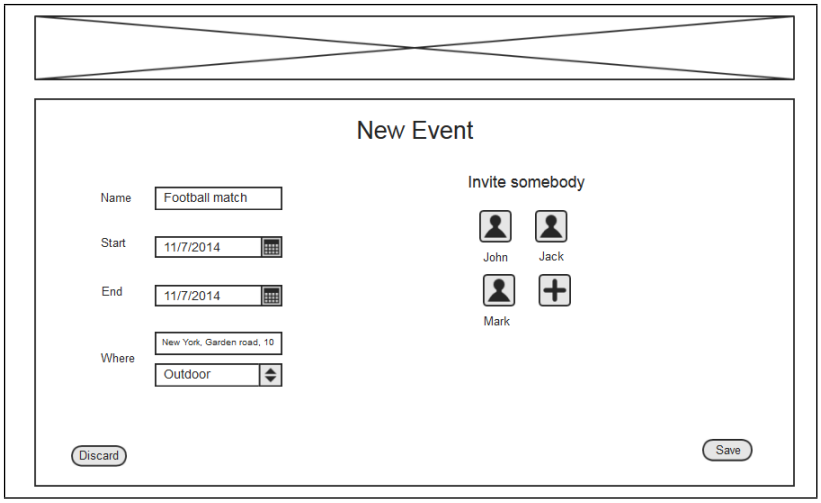
11.2 User interface

User interface should be minimal, providing easy access to all functions without disorienting the user. The general structure will be similar to the many digital calendars available on the market.

The main page will be a typical calendar representation, with a view of the days for each month. The user can browse the calendar month by month with the navigation arrows and check the details of each day clicking on its box. It will be possible to add an event by clicking on the plus sign on the top-right of the screen.



Event creation and update forms will allow the user to easily insert all required data in one single step:



11.3 Error handling

- the system should always notify the users about input errors and never create inconsistent data
- in case of extreme condition errors the system should at least notify the users about the possible causes before closing

11.4 Quality of Service

- the system should be available 24 hours a day
- the system should safely store all events and notifications without data loss

- the system should always be responsive (usually less then 3s for each action) and avoid freezes

11.5 Security

- user credentials should be safely stored in the system's database in order to avoid unauthorized access
- only the calendar owner or invited users can view an event
- only the organizer can update or delete an event

11.6 Documentation

- **User manual**: to explain how the system works to the new users
- **Requirements Analysis and Specification Document (RASD)**: for an introduction to *MeteoCal*'s functions and requirements
- **Design document (DD)**: to define the actual structure of the system and its functions
- **JavaDoc**: to document the Java code

11.7 Environment

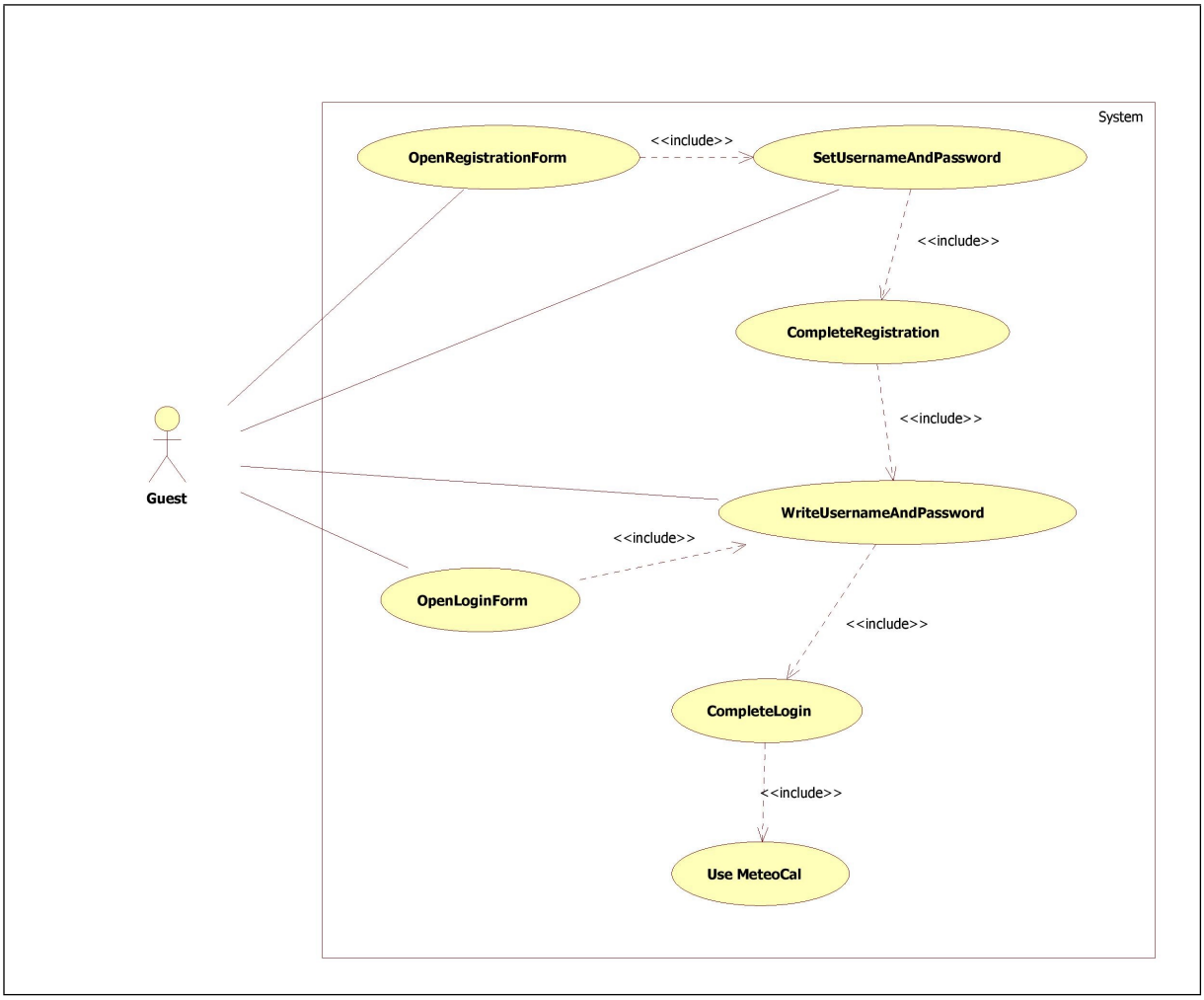
MeteoCal will be developed using the Java EE platform with database. The business logic will be implemented with EJBs, the user interface as a web application that interacts with the business logic.

Part IV
UML Models

12 Use Cases

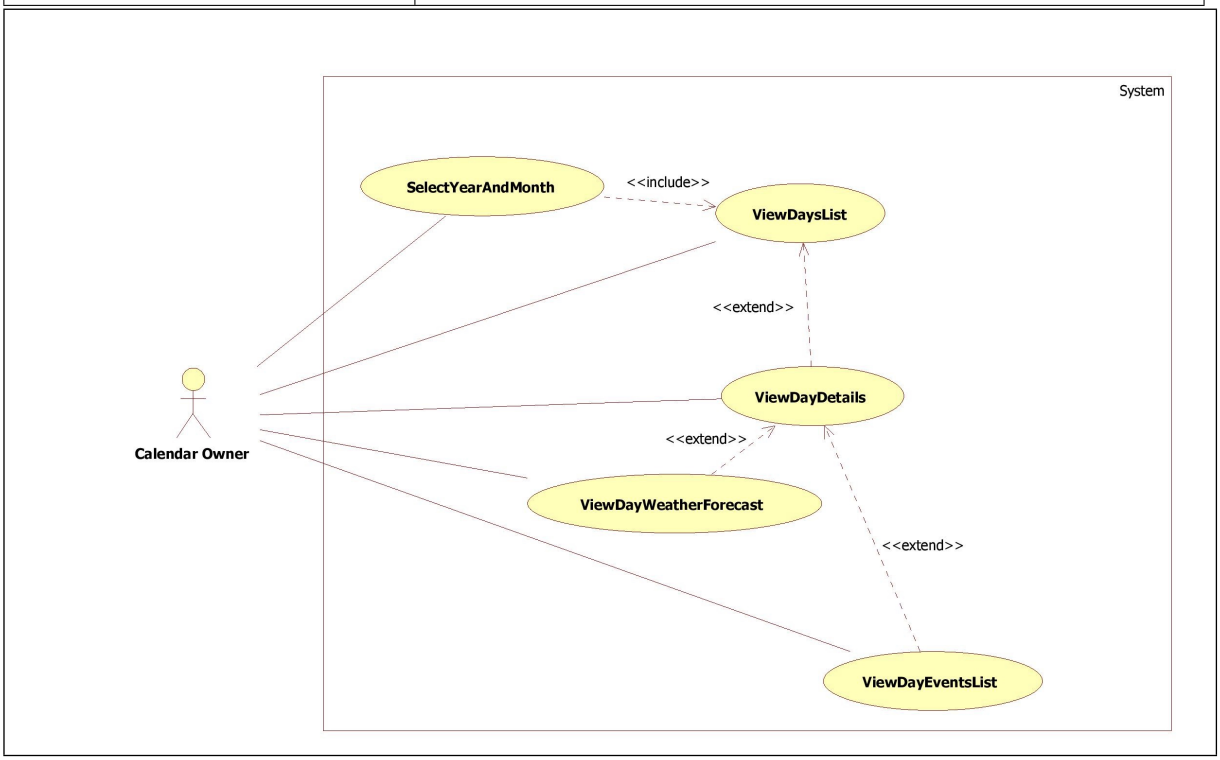
12.1 Registration and Log-in

Actors	Guest
Entry Conditions	The system has been opened by the user
Flow of events	• the system shows the log-in page, that also contains a "Register" button • if the user is not registered – clicks on the "Register" button – writes required data, like user-name and password – clicks on the "Complete registration" button – the system redirects to the log-in page • the user inserts user-name and password • the systems checks the input and then logs the user in • the user is now able to use *MeteoCal*
Exit conditions	To use *MeteoCal* the user must have successfully registered and logged into the system
Exceptions	• if during registration some input is not correct (e.g. user-name already present), the form notifies the user and waits for another submit • if during log-in user-name or password are not correct, the system informs the user and interrupts the log-in procedure


System
OpenRegistrationForm
<<include>>
SetUsernameAndPassword
<<include>>
CompleteRegistration
<<include>>
Guest
WriteUsernameAndPassword
<<include>>
OpenLoginForm
<<include>>
CompleteLogin
<<include>>
Use MeteoCal

12.2 Calendar browsing

Actors	Calendar owner
Entry Conditions	The system has been opened by the user
Flow of events	• log-in • the user browses the months through the navigation arrows until he/she finds the one he/she was looking for • the user clicks on a day of the month • the system shows the event details • the user reads the weather forecast (if available) • the user reads the day schedule (if at least one event has been created for that day)
Exit conditions	None
Exceptions	None

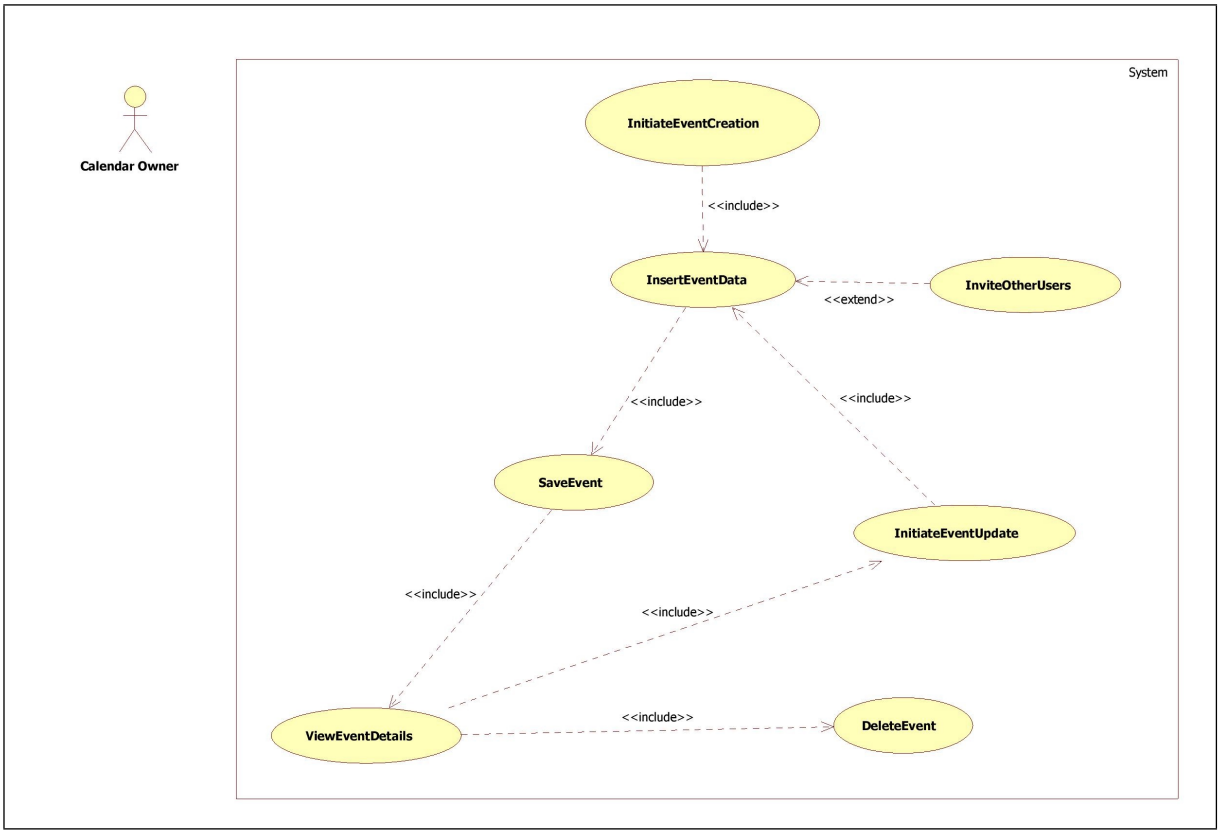


12.3 Event management

Actors	Calendar owner
Entry Conditions	The system has been opened by the user

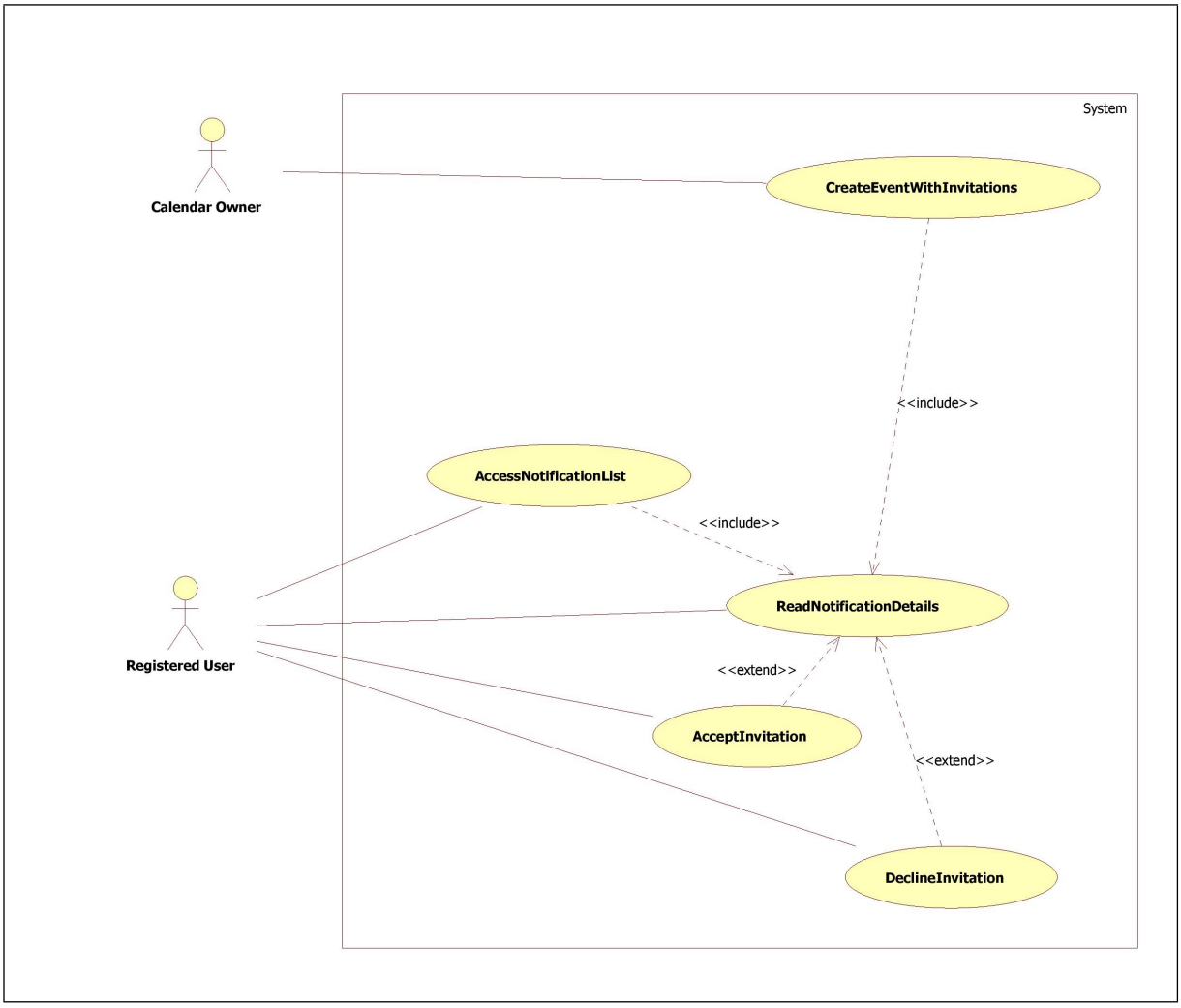
Flow of events	• log-in
	• new event creation
	  – click on "Add new event" button in the calendar home page
	  – fill event data, like place, time, etc.
	  – save event
	• event details
	  – click on a day with events
	  – click on an event listed in the day schedule
	  – read details
	• event update
	  – click "Update" button in the event details
	  – change event data
	  – save event
	• event delete
	  – click "Delete" button in the event details

Exit conditions	None
Exceptions	If event data is inconsistent (e.g. end time before start time) the system notifies the user and waits for another submit

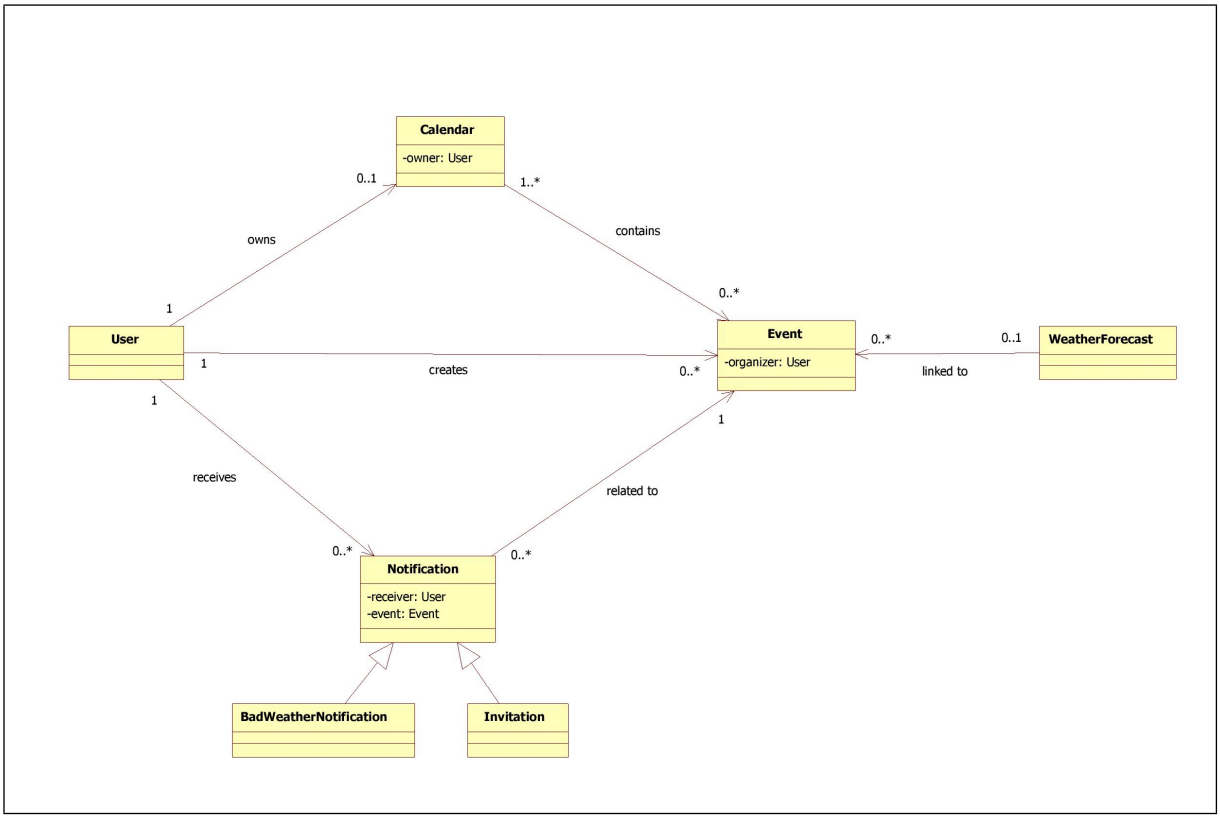

Calendar Owner
System
InitiateEventCreation
<<include>>
InsertEventData
InviteOtherUsers
<<extend>>
<<include>>
<<include>>
SaveEvent
InitiateEventUpdate
<<include>>
<<include>>
ViewEventDetails
<<include>>
DeleteEvent

12.4 Invitation management

Actors	Calendar owner (A), Registered user(s) (B)
Entry Conditions	The system has been opened by the users
Flow of events	• A logs in • A creates a new event inviting user(s) B • B logs in • B reads the notification list • B opens the invitation from A • B can accept or decline the invitation • if B accepts and he/she owns a calendar, the event is added to the calendar
Exit conditions	None
Exceptions	None

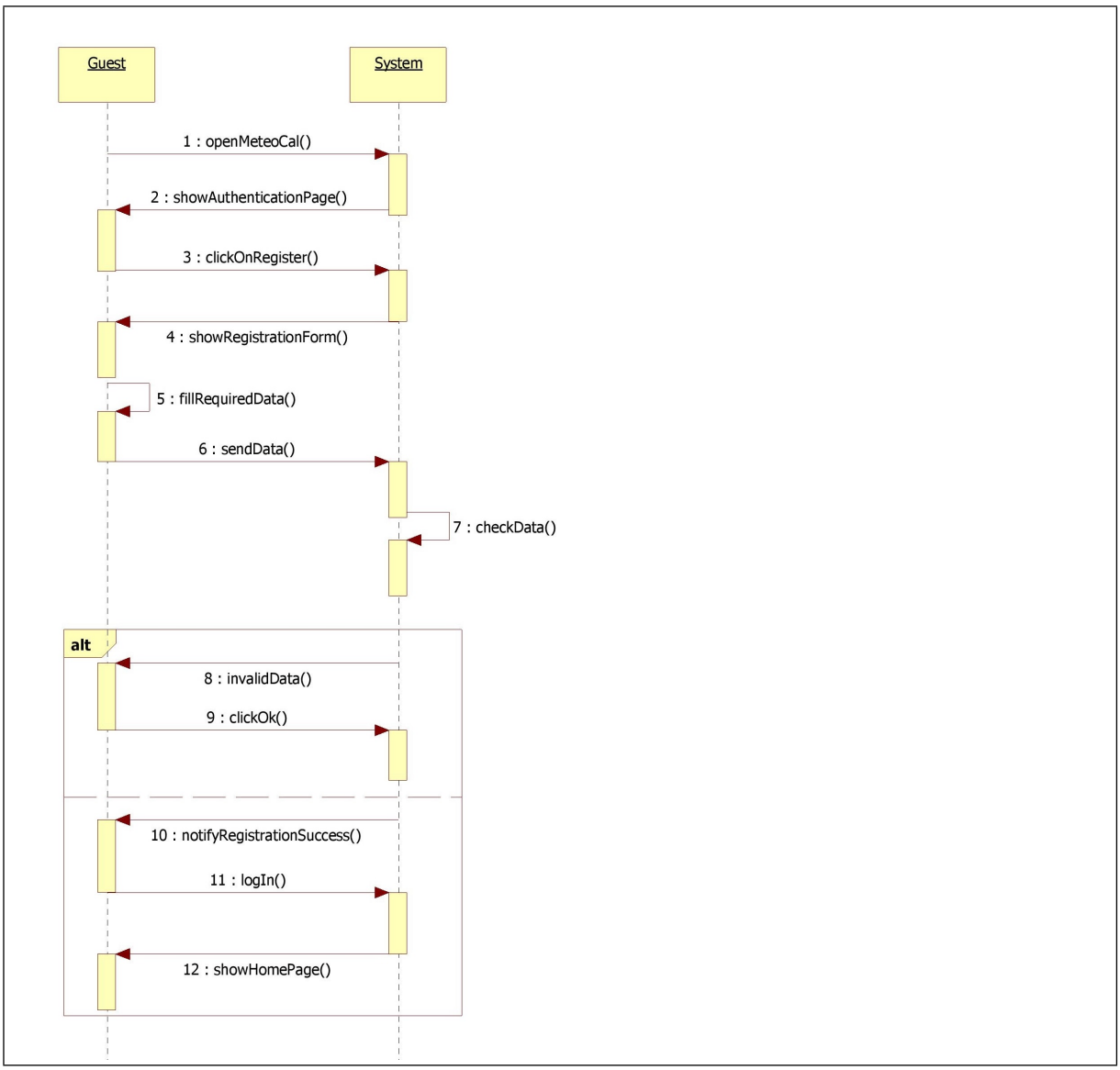

System
Calendar Owner
CreateEventWithInvitations
<<include>>
AccessNotificationList
<<include>>
Registered User
ReadNotificationDetails
<<extend>>
AcceptInvitation
<<extend>>
DeclineInvitation

13 Class Diagram

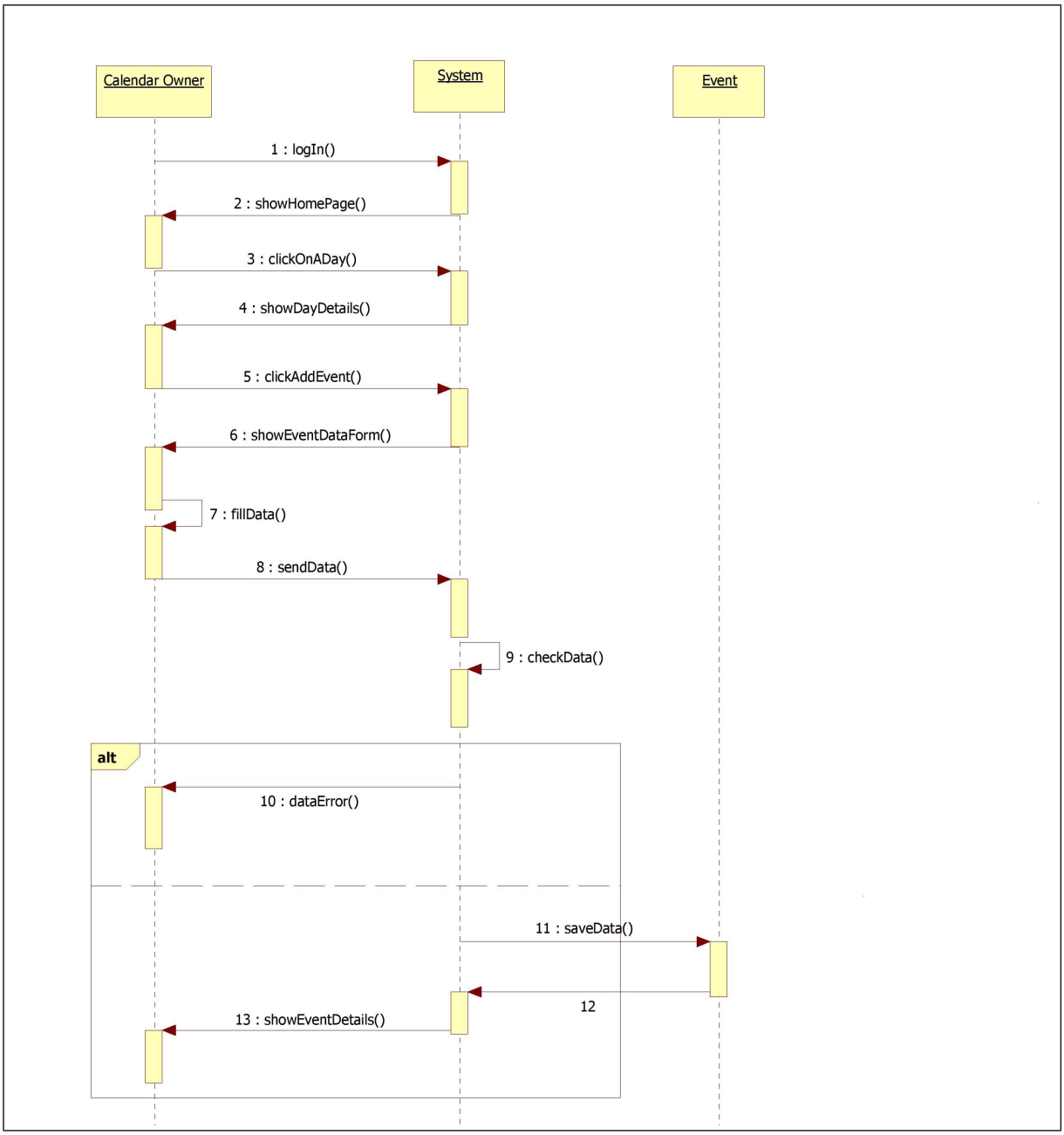


14 Sequence Diagrams

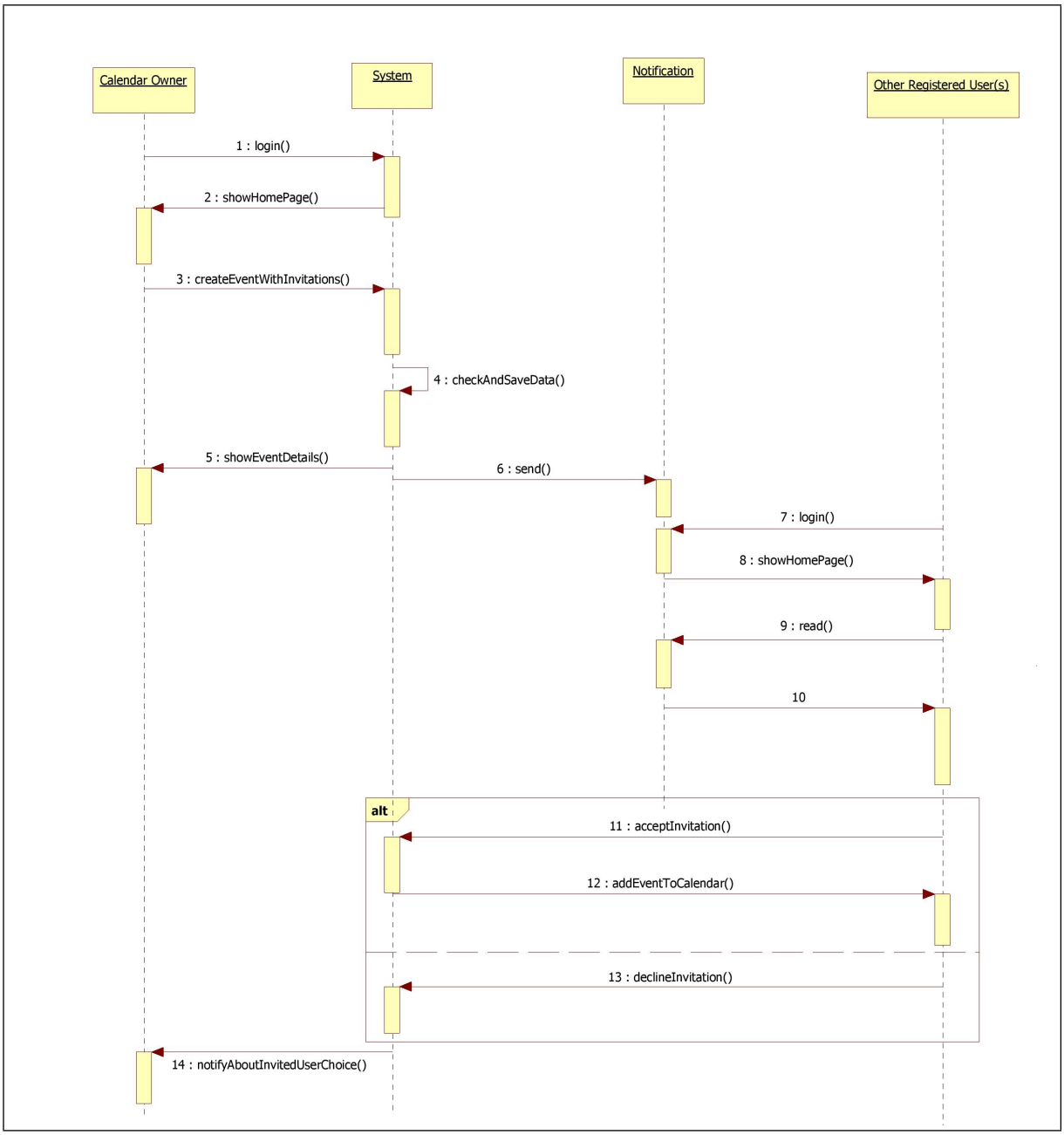
14.1 Registration and Log-in



14.2 Event management

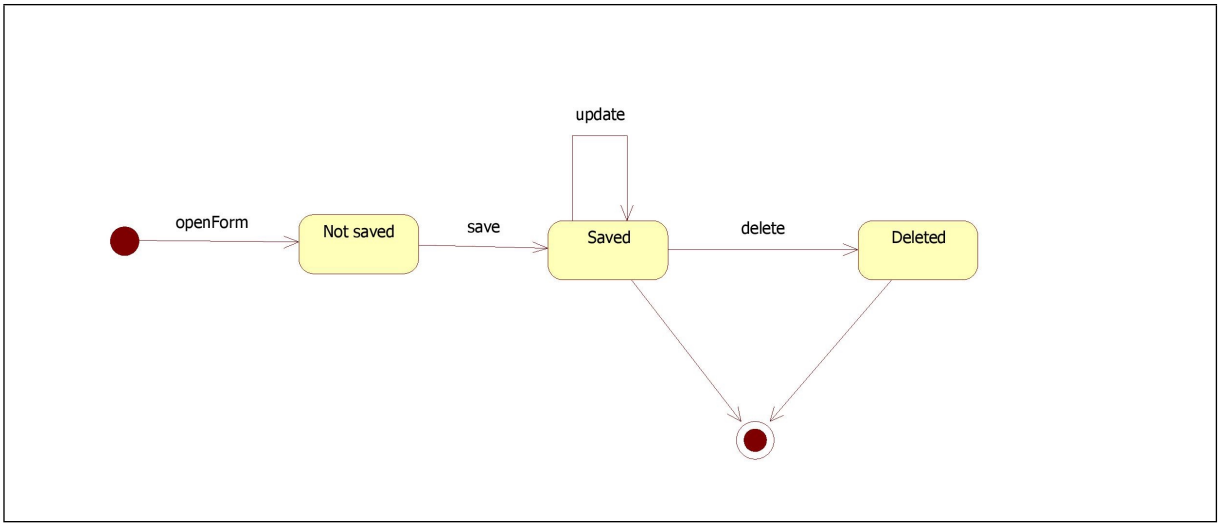


14.3 Invitation management

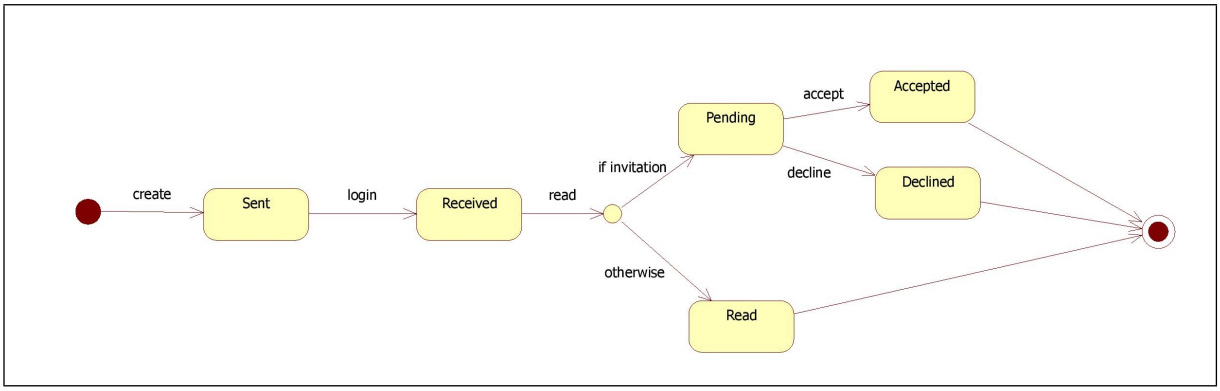


15 State Charts

15.1 Events



15.2 Notifications



Part V
Alloy

16 Model

```
/***************** SIGNATURES *****************/

abstract sig AbstractUser{}
sig Guest extends AbstractUser{}
```

```
sig RegisteredUser extends AbstractUser{}

sig Calendar
{
        owner: one RegisteredUser,
        contains: set Event
}

abstract sig WeatherForecast{}
sig GoodWeatherForecast extends WeatherForecast{}
sig BadWeatherForecast extends WeatherForecast{}

abstract sig Event
{
        organizer: one RegisteredUser,
        weather: lone WeatherForecast
}
sig OutdoorEvent extends Event{}
sig IndoorEvent extends Event{}

abstract sig Notification
{
        event: one Event,
        receiver: one RegisteredUser
}
sig Invitation extends Notification {}
sig BadWeatherAlert extends Notification{}


/******************** FACTS ********************/

// An event is inside a calendar only if the
   user is the organizer or if he/she received
   an invitation by the organizer
fact EventInCalendarProperties
{
        // Organizer must have event in calendar
        all e: Event | one c: Calendar |
           e.organizer=c.owner and e in
           c.contains

        // Non-organizers must have received an
           invitation
```

```
        all c: Calendar | all e: Event | ( e in
            c.contains and e.organizer!=c.owner=>
            one i: Invitation | i.event=e and
            i.receiver=c.owner )
}

// Users can have at most one calendar
fact OneCalendarPerUser
{
        all cal1: Calendar | no cal2: Calendar |
            cal1!=cal2 and cal1.owner=cal2.owner
}

// Users cannot invite themselves to events
fact NoSelfInvitations
{
        all e: Event | no i: Invitation |
            i.event=e and i.receiver=e.organizer
}

// Notifications are sent only if the linked
   event can be viewed by the receiver
fact NotificationsOnlyOnVisibleEvents
{
        all n: Notification |
            n.receiver=n.event.organizer or some
            c: Calendar | c.owner=n.receiver and
            n.event in c.contains
}

// There cannot be notifications about the same
   event sent to the same user
fact NoDuplicateNotifications
{
        all n1: Notification | no n2:
            Notification | n1!=n2 and
            n1.event=n2.event and
            n1.receiver=n2.receiver
}

// Weather forecast data must be linked to an
   event
fact WeatherForecastData
{
```

```
        all wf: WeatherForecast | some e: Event
            | e.weather=wf
}

// Bad weather alert notifications must be
    linked to an outdoor event with bad weather
    forecast
fact BadWeatherSettings
{
        all bwa: BadWeatherAlert | one bwf:
            BadWeatherForecast | one oe:
            OutdoorEvent | bwa.event=oe and
            bwa.event.weather=bwf
}


/***************** ASSERTIONS *****************/

// All events must be inside at least one
    calendar
assert noEventsWithoutCalendar
{
        all e: Event | some c: Calendar | e in
            c.contains
}


/***************** PREDICATES *****************/

// "Free" model generation
pred show() {}

// Calendars may be empty
pred emptyCalendars()
{
        some c: Calendar | no e: Event | e in
            c.contains
}

// Users may not have calendars
pred usersWithoutCalendar
{
        some u: RegisteredUser | no c: Calendar
            | c.owner=u
```

```
}

// Events may not have notifications
pred noEventNotifications
{
        some e: Event | no n: Notification |
           n.event=e
}

// User can have events in his/her calendar
   without ever creating one (invitation)
pred noOrganizedEvents
{
        some u: RegisteredUser | one c: Calendar
           | c.owner=u and all e: Event | e in
           c.contains and e.organizer!=u
}


/****************** COMMANDS ******************/

check noEventsWithoutCalendar
run show for 3 but exactly 1 RegisteredUser, 0
   Guest, exactly 5 Event, exactly 2
   BadWeatherAlert
run show for 3 but exactly 2 RegisteredUser,
   exactly 1 Guest, exactly 2 OutdoorEvent,
   exactly 2 IndoorEvent, 0 Notification,
   exactly 2 Invitation
run emptyCalendars for 2
run usersWithoutCalendar for 2
run noEventNotifications for 2
run noOrganizedEvents for 2
```

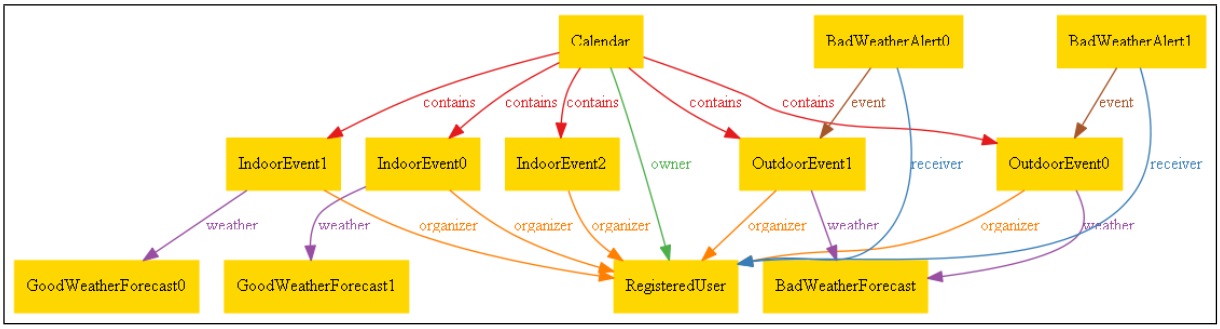
17 Analyzer

```
7 commands were executed. The results are:
  #1: No counterexample found. noEventsWithoutCalendar may be valid.
  #2: Instance found. show is consistent.
  #3: Instance found. show is consistent.
  #4: Instance found. emptyCalendars is consistent.
  #5: Instance found. usersWithoutCalendar is consistent.
  #6: Instance found. noEventNotifications is consistent.
  #7: Instance found. noOrganizedEvents is consistent.
```

18 Worlds

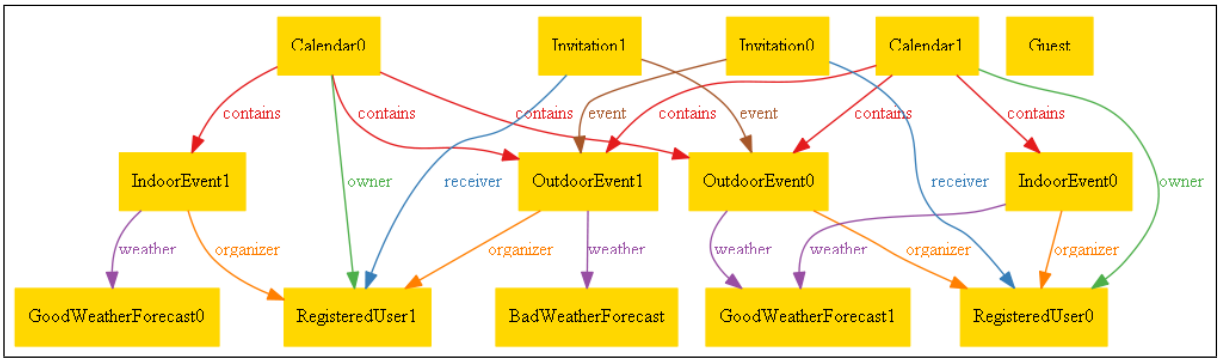
18.1 Single user

Basic "free" representation with one user and multiple events.



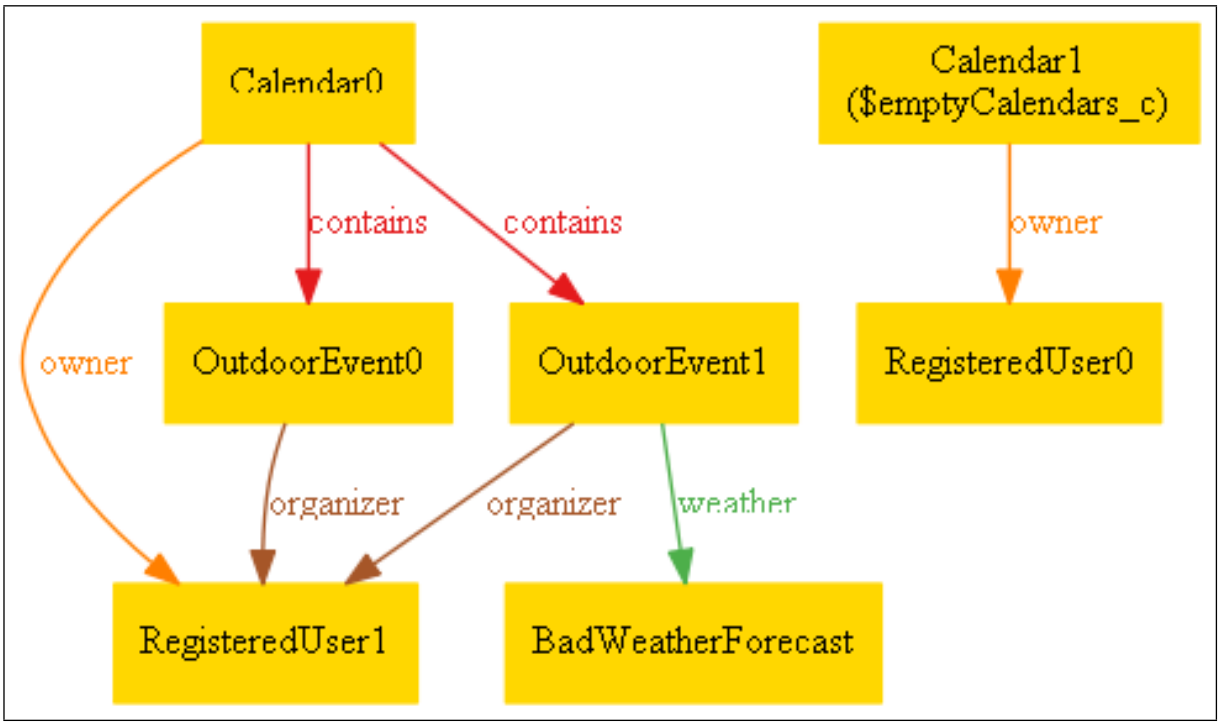
18.2 Multiple users with invitations

Basic "free" representation with more than one user and shared events due to invitations.



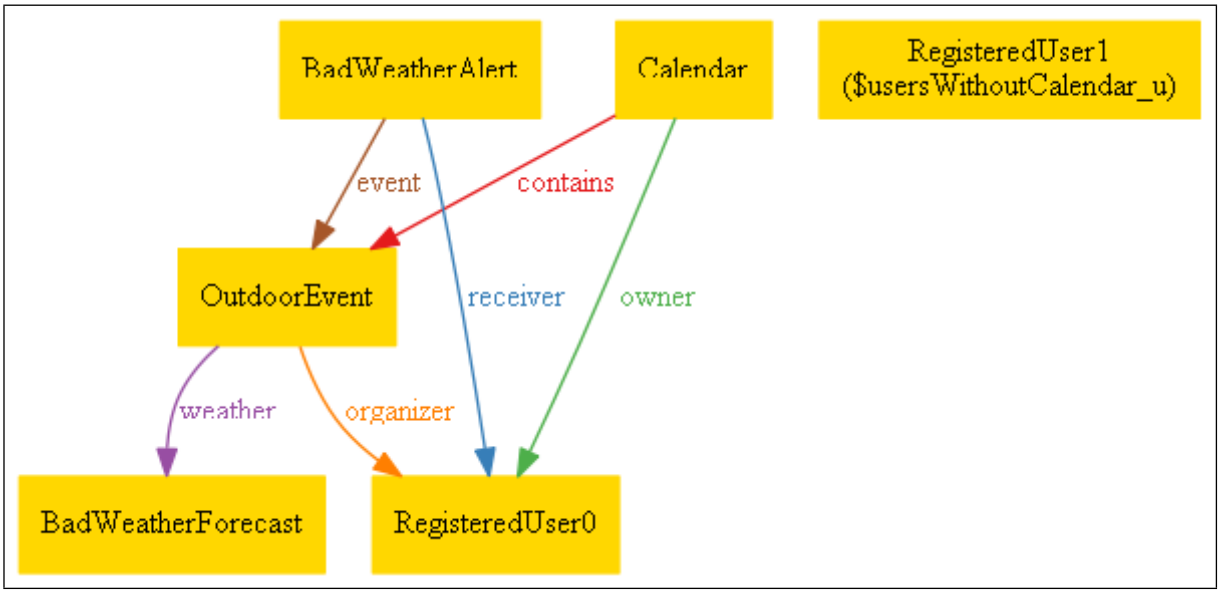
18.3 Empty calendars

Special case where a calendar is empty.



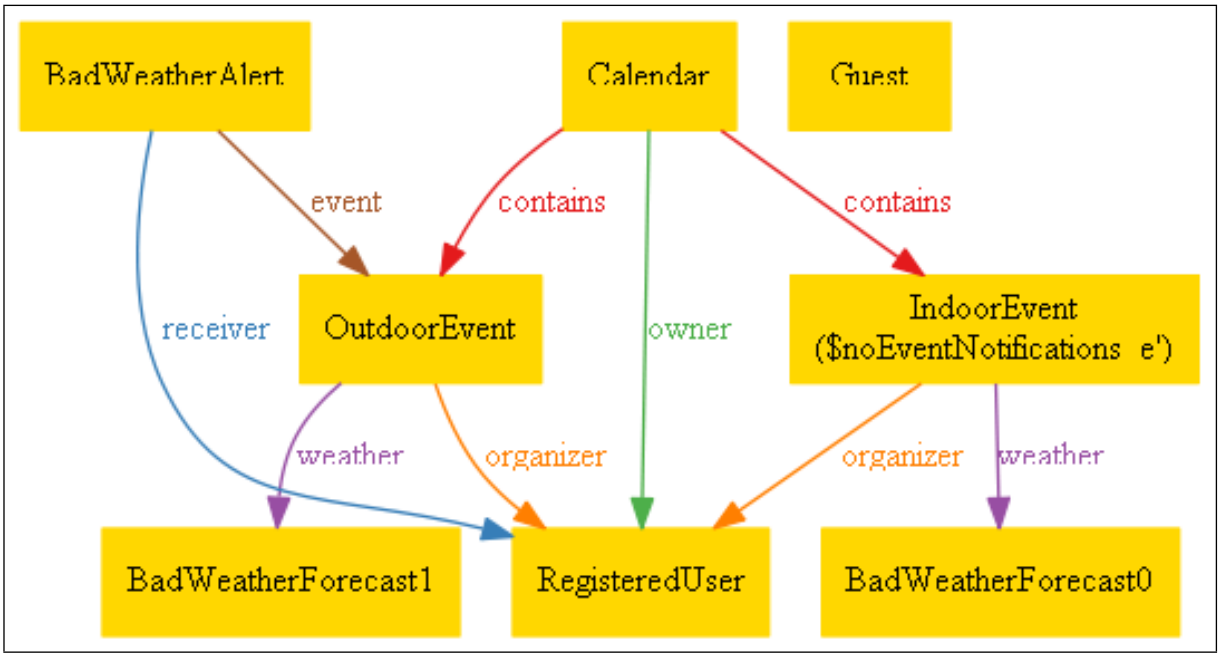
18.4 Users without calendar

Special case where a registered user hasn't created his/her calendar yet.



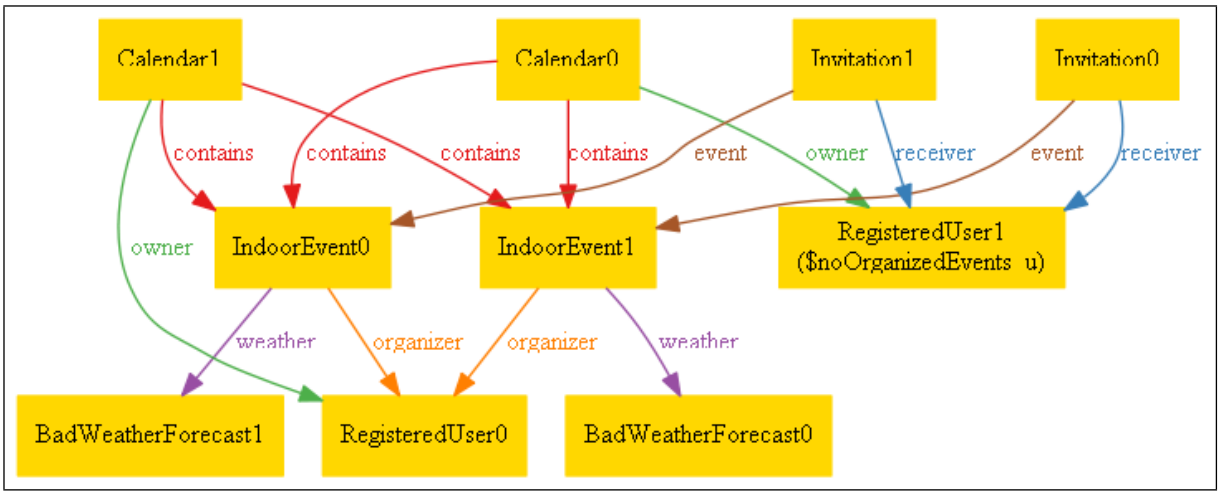
18.5 Events without notifications

Special case where an event has no notifications, because the organizer didn't invite anybody and there was no bad weather alert (yet).



18.6 Calendar contains event but none organized by the owner

Special case where a calendar contains some events but none of them organized by the owner (all invitations).



Part VI
Conclusion

19 Used Tools

To redact this document, the following tools have been used:

- **Texmaker**: to build this document using LaTeX
- **WhiteStarUML**: for all UML diagrams (Class Diagram, Use Cases, Sequence Diagrams, State Charts)
- **MIT's Alloy Tool**: for Alloy Analyzer and World Generation
- **Graphviz**: for better Alloy Worlds representation
- **GoMockingbird**: for UI mock-up

20 Change-log

Versions of this document:

- **16th November 2014**: first release