



POLITECNICO DI MILANO
Computer Science and Engineering
Project of Software Engineering 2

MeteoCal

Requirements Analysis and Specification Document

Authors: Alessandra Decaneto 836131
Alberto Marchesi 835919

Reference Professor: Mirandola Raffaella

SUMMARY

1. INTRODUCTION	3
1.1 DESCRIPTION OF THE GIVEN PROBLEM	3
1.2 GOALS.....	3
1.3 DOMAIN PROPERTIES	4
1.4 GLOSSARY	4
1.5 ASSUMPTIONS	5
1.6 PROPOSED SYSTEM.....	7
1.7 IDENTIFY STAKEHOLDERS	7
2. ACTORS IDENTITYFING	7
3. REQUIREMENTS	8
3.1 FUNCTIONAL REQUIREMENTS	9
3.2 NON-FUNCTIONAL REQUIREMENTS	10
3.2.1 <i>User Interface</i>	10
3.2.2 <i>Documentation</i>	13
4. SCENARIO IDENTITYFING.....	14
5. UML MODELS	16
5.1 USE CASE DIAGRAM	16
5.2 USE CASE DESCRIPTION	18
5.3 CLASS DIAGRAM	32
5.4 SEQUENCE DIAGRAM	33
5.4.1 <i>Log In</i>	33
5.4.2 <i>Create an Event</i>	34
5.4.3 <i>Receive a Three-Days Before Notification</i>	35
5.4.4 <i>Receive a One-Day Before Notification</i>	36
5.4.5 <i>Reply to an Invitation</i>	37
5.4.6 <i>Search for a User</i>	38
5.5 STATE CHART DIAGRAMS	39
5.5.1 <i>Event</i>	39
5.5.2 <i>Invitation</i>	40
5.5.3 <i>Event Weather Forecast</i>	41
6. ALLOY MODELLING	42
6.1 ALLOY CODE	42
6.2 ALLOY WORLDS	47
5.5.1 <i>General World</i>	47
5.5.1 <i>Invitation Properties</i>	48
5.5.1 <i>Notification Properties</i>	49
7. USED TOOLS	50

1.Introduction

1.1 Description of the given problem

We will project and implement METEOCAL, which is an online agenda that offers a new weather based calendar for helping people scheduling their personal events avoiding bad weather conditions in case of outdoor activities.

The system should be able to register new users with their information, such as name, surname, address and e-mail.

Once a user is registered to the application, he should be able to manage his calendar by adding new events or modifying and deleting existing ones. A user should be also able to invite other users to an event when he creates it. Users can search for the calendar of another user and visualize it according to the visibility assigned by the calendar's owner.

The peculiar feature of the system is that it is able to attach to events weather forecast automatically and notify the participants of an event one day before in case of bad weather conditions.

Three days before the event, in case of bad weather forecast, the system suggest to the creator of the event the closest sunny day in order to allow the user to change the date in which the event is scheduled.

1.2 Goals

We thought about the possible users and what METEOCAL should provide them, so we planned to give our system these features:

- Users should be able to:
 - Sign up into the system;
 - Log into the system;
 - Search for other people's personal page;
 - Visualize the invitation received and decide whether to participate or not;
 - Visualize other users' calendar, according to the visibility;
 - Create, delete, update an event;
 - Invite other users to an event which they are creating;
 - Be notified by the system of the weather forecast of outdoor events.
- The system should provide the weather forecast for each event that our users will schedule.

1.3 Domain Properties

We suppose that the following properties hold in the analysed domain.

- If an event is scheduled, it will actually take place.
- An event can be defined clearly as outdoor or indoor, there is no possibility of having events that are both outdoor and indoor.
- Events can only take place in the location for which it is possible to retrieve weather forecasts.
- Users accept the invitation to an event only if they are able to participate (they haven't already scheduled other events for that time).
- The weather forecast was to be reliable.
- It is possible to univocally define daily weather condition within a specific category.

1.4 Glossary

We want to make unequivocal some words that will be often used in our documentation of the project, so we are giving a precise definition of what will be the meaning of these words in the contest of this project.

- **USER:** for user we mean a person already registered in the system.

A user has a profile that includes all these information:

- Name;
- Surname;
- Email;
- Username;
- Password.

And optionally:

- Picture;
- Telephone Number;
- Address.

A user has the capability of doing everything that we mentioned in the goals, making exception for signing up.

- **GUEST:** a guest is a person who hasn't signed up yet.
Guests have less power in the system than users, they don't have the users' skills, and the only function they can use is to sign up.
- **EVENT:** for event we mean something that you have to do, an appointment, which is characterized by a name, a specific place, time and day, a property that says if it is public or private and finally another property that says if it is outdoor or indoor.
An event can be scheduled in the user's personal calendar.

- **CALENDAR:** for calendar we mean an agenda where the users can note their appointments, schedule events and see them all in a unique vision.
- **PARTECIPANT:** for participant we mean a user who has received an invitation and will participate to that event.
- **EVENT CREATOR:** for event creator we mean a user who creates an event, so he decides the specific characteristics of the event and he's able to invite other people to that event.
- **WEATHER FORECAST:** for weather forecast we mean the meteorological prediction given us by the API service of openweathermap.org. Our weather forecast can be of four different types: sunny, cloudy, rainy and snowy.
- **PLACE:** for place we mean the nation, the city and the address where the event will take place.
- **TYPE:** for type of an event we mean the characteristic of an event of being outdoor or indoor. The type is exclusive, so an event can't be both outdoor and indoor.
- **INVITATION LIST:** for invitation list we mean the list of all the events a user has been invited to, both the ones that the user hasn't answered to yet and the ones the user is participating to. If a user declines an invitation to an event he can't see it anymore in the invitation list.

1.5 Assumptions

There are few points that aren't really clear in the specification document, so we had to assume some facts.

We assume that:

- If a user decides to make his calendar public every user can see it and the events can be of two types:
 - **PUBLIC**, every user can see it,
 - **PRIVATE**, other users can only see that a time slot is occupied but can't see any detail of that.
- If a user decides to make his calendar private nobody can see it and the events can be of two types:
 - **Events with participants:**
If it is private the participants can't see the details of the event, if it's public they can.
 - **Events without participants:**
Whether it's public or private no one can see the details of the event.

- If an event is modified all the participants have to be notified by the system, and then they should be able to decide if they still want to participate or not.
- The system will provide weather forecast for each day, not making a distinction between hours.
- The search for other users is only made by usernames.
- If a user accepts to participate to an event, this event will be automatically scheduled in his calendar.
- The invited people are defined only at the act of the creation of the event, and it's no longer possible to modify the invitation list.
- In case of bad weather for an outdoor event the system proposes to the event creator the closest sunny day only one time, then if he accepts the proposed day the system reschedules it, otherwise he can modify the date of the event himself.
- The users' personal page it's visible only for users, not for guests.
- The system isn't allowed to reschedule an event in a day before the day that was selected by the event creator for that event.
- It's impossible for the user to change the type of an event, if he has created an outdoor event, even if he modifies it, it will always be an outdoor event.
- Once he has created an outdoor event, the User can't modify the weather forecast that he considers bad for that event.
- We assume that Change and Delete Notification are sent to all the invited people, the One Day Notification are sent only to the participant of the event.

1.6 Proposed System

We will implement an enterprise application based on the web using the platform JEE. It will be composed of a server, which runs the business logic, generates dynamic web pages and access to data sources. On the other side there will be several clients that interact with the server using a web browser and a graphical user interface.

Weather forecasts are taken from an online web application that provides free to use APIs to access a data base containing weather information from the whole world, both historical data and forecasts.

1.7 Identifying stakeholders

Our main stakeholder is the professor, who gave us the delivery of the project and expects us to be able to finish the project within this semester. The professor asks us to focus on the whole development process of a complex enterprise application, which involves the following phases: requirement analysis, design, implementation, testing and project reporting. So we will try to concentrate equally on each phase, not only on the implementation of the final product but also on the documentation. We will focus first on the main functionalities of the system, which are directly requested in the specification given to us by the professor, then we will try to extend the functionalities of the system in order to make the application as close as possible to an application that is ready to be launched in the market.

Starting from this consideration we can imagine that an hypothetical stakeholder for our system could be a company that would have asked us to realize a platform to manage employees appointment in a more efficient way.

2. Actors Identifying

The actors of our system are basically two:

- Guest: a guest as we already mentioned in the glossary section a guest is someone who hasn't signed up yet, and has only the capability of signing up.
- User: a user as we already mentioned in the glossary section is someone who has already signed up. The user can take advantage of every feature of our system, and depending on what his actions are he can be an EVENT CREATOR or a PARTECIPANT.

3. Requirements

We've determined the following requirements according to the Jackson and Zave analysis, so assuming that the domain properties, which are described above, holds for our purposes, we've written the requirements in order to satisfy goals.

- Registration of a person to the system
 - The system has to provide a sign up functionality.
- Searching for another user personal page
 - The system has to provide a function that allows users to search other users by name.
 - The system has to make possible for users to read personal pages belonging to other users.
- Visualization of other users' calendar
 - The system has to provide a function that makes possible to access the calendar of another user (from his personal page), if the calendar is public.
 - The visualization of the calendar must be done according to the visibility of events by applying the rules described in the assumptions.
- Visualization of the invitations and their acceptance
 - The system has to provide a function that allows a user to visualize the list of the events, which he's invited to.
 - The user should be able to accept or reject invitations directly from the list.
- Create, update, delete events and invite users to events
 - The system has to provide a function to allow a user to create a new event.
 - The system has to provide a function to allow a user to update an event created by him (except for the list of the participants).
 - The system has to provide a function to allow a user to delete an event created by him.
 - When an event is created, the system has to allow the user who creates the it to invite any number of users (even zero).
- Providing weather forecast
 - The system has to attach the weather forecast to each event.
- Weather forecast notification
 - The system has to notify a user, who has created an outdoor event, three days before that event, if the weather forecast is bad for that day, and it should propose to the user the closest sunny day.
 - The system has to notify a user, who is invited to an outdoor event, one day before the event, if the weather forecast for that day is bad.

3.1 Functional Requirements

Now that we have defined the main feature of METEOCAL, we can find some functional requirement concerning each defined actor:

- Guest: he can
 - Sign up.
- User: he can:
 - Log in;
 - Modify his profile information;
 - Search for users;
 - See other users' personal page;
 - Create an event;
 - Invite other people to his events;
 - Receive an invitation;
 - Modify an event;
 - Delete an event;
 - Receive a three days before notification from the system;
 - Receive a one day before notification from the system;
 - See his calendar;
 - See other users' calendar.

3.2 Non-functional Requirements

3.2.1 User Interface

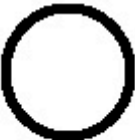

The interface of our application is thought to be used via web.
We want to provide three sketches of the pages we consider the most important.

The first is the HOME page, where the user can see his personal information and can modify them.

From the home page, as from any other page, the user can log out and search for other users.

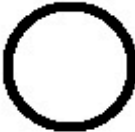
We can see below a first sketch of it.

The sketch shows a web interface for a user's home page. At the top left is a circular profile picture placeholder. To its right is the text 'USERNAME'. Further right is a search bar with a magnifying glass icon and the text 'SEARCH'. Below the profile picture is a blue link labeled 'LogOut'. Below these elements is a horizontal navigation bar with three buttons: 'HOME' (highlighted with a thick border), 'Calendar', and 'EventList'. Below the navigation bar are four form fields, each with a label and an input line: 'Name', 'Surname', 'Date of Birth', and 'Email'.


	USERNAME	 SEARCH <input type="text"/>
LogOut		
HOME	Calendar	EventList
Name	<input type="text"/>	
Surname	<input type="text"/>	
Date of Birth	<input type="text"/>	
Email	<input type="text"/>	

The second is the CALENDAR page, where the user can see the schedule of his week.

We can see that there are two kinds of events, the one in yellow, which are the public one, where we can see the detail of the event, and the one in pink, which are the private one, where we can only see that the time slots are occupied but we have no possibility to know y they're occupied



USERNAME

 SEARCH

[LogOut](#)

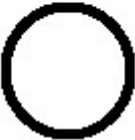
[Home](#)

[CALENDAR](#)

[EventList](#)


Mond	Tuesd	Wed	Thursd	Frid	Sat	Sund
					11.00-13.45 Ivy's Birthday	
14.30-16 Meeting John						

The third one is the INVITATION LIST page where we can see the list of the events we have been invited to.
From this page we also have the possibility to accept or to reject an invitation. Events are part of the list, both that we haven't answer the invitation yet or we have decided to accept.



USERNAME

[LogOut](#)

 SEARCH

[Home](#)

[Calendar](#)

[EVENT LIST](#)

Elizabeth's Birthday

Saturday

21.00-24.00

Piazza Duomo, Milan

WILL YOU PARTECIPATE?

YES

NO

List of the invited persons

3.2.2. Documentation

We will release the following documents in order to organize our work in each phase of the development process and keep in touch with the stakeholders.

- RASD, Requirement Analysis and Specification Document, which defines our goals and assumptions and contains an overall description of the system (using scenarios and use-cases) and the models describing requirements and specifications.
- DD, Design Document, which contains a functional description of the system using models such as UML diagrams.
- Installation manual, which explains how to deploy the web site.
- User manual, which explains users how to use the main functionalities of the web site.
- Testing report, which contains the results of the testing activity performed on a system developed by another group.
- Project reporting, which is the result of some analysis done on the project activity.

4. Scenarios Identifying

Here are some possible scenarios of METEOCAL:

- Jack needs an agenda where he can schedule his appointments, and because he really hates the rain he would like to find an application that automatically says what's the weather like for his appointments, so he digits on the web "calendar maker with weather forecast" and among the voices he finds METEOCAL.
He clicks on it and he discovers this new web platform.
In the home page of the platform there is a good explanation of what it is, he likes it and so he decides to sign up.
He has to complete a form, in which he has to declare his personal information, name, surname, email, username, password and optionally a picture, the address and a telephone number.
The system accepts his registration and Jack can visualize his personal page.
- Jack wants to organize a surprise party for his wife's birthday, Anne, so he decides to create an event on METEOCAL. He has to access his calendar and to click the button for creating a new event. The system asks him to specify the name, the place, the time and the type of the event, so he types "Anne' birthday surprise party" as name, Sunday 21 of June at 4.00 pm as time and his home address as place, he decides to choose the type outdoor for the event since it will take place in the garden behind the house, he also makes the event private in order to prevent Anne of being aware about the event. After that, he is asked by the system to specify in which cases of the weather conditions he wants to be notified, because he has created an outdoor event; he selects rainy from a list of possible weather conditions (cloudy, rainy, snow, ...) and clicks on the button create. Then the system asks him if he wants to invite someone to his event, Jack decides to ask all his and his wife's friends to participate to the party, so he has to search for them by usernames using a search field and when their profiles' icons are visualized by the system he clicks on them to send the invitations.
- Rose, Jack's wife's friend, is registered to METEOCAL, and in the evening she's always checking her account, so when she logs in she's notified of the invitation at Anne's Birthday Party and since she's free on 21/06/15 she decides to accept the invitation by clicking the "YES" button.

- Jack, talking to Anne, finds out that she has to leave for two days on 20 and 21 of June for work, so he has to rearrange the surprise birthday party to the next weekend. He logs in METEOCAL, then click in the event on his calendar and update it changing the date from Sunday 20 of June to Sunday 27 of June. The system notifies all the participant about the change.
- On 18/06/15 Jack logs in METEOCAL, he's immediately notified that the weather forecast for 21/06/15 says that it will rain, and the system proposes to him the closest sunny day which is 23/06/15 and asks him if he wants to reschedule it in that date, he accepts clicking the "YES" button.
- Rose on 22 of June logs in the METEOCAL web sites and the system visualizes to her a notification of bad weather forecast (in particular rain) for Anne's birthday party, which she was invited to. Then Rose click on the Ok button and she is redirected to her profile page.
- Jack wants to go to the gym with his friend Alex, so he logs into METEOCAL to see Alex's calendar to schedule an appointment in a day where they both are free.
He wants to search for his friend's personal page, but he reminds that for searching another user you must know his username, so he texts Alex asking him what his username is.
Alex answers him that his username is AlexWhite9, Jack inserts AlexWhite9 in the form and then he searches for him by clicking the "SEARCH" button.
The system shows him the results and he selects Alex, so the system loads Alex's personal page and Jack by clicking on "CALENDAR" can see his friend's agenda.

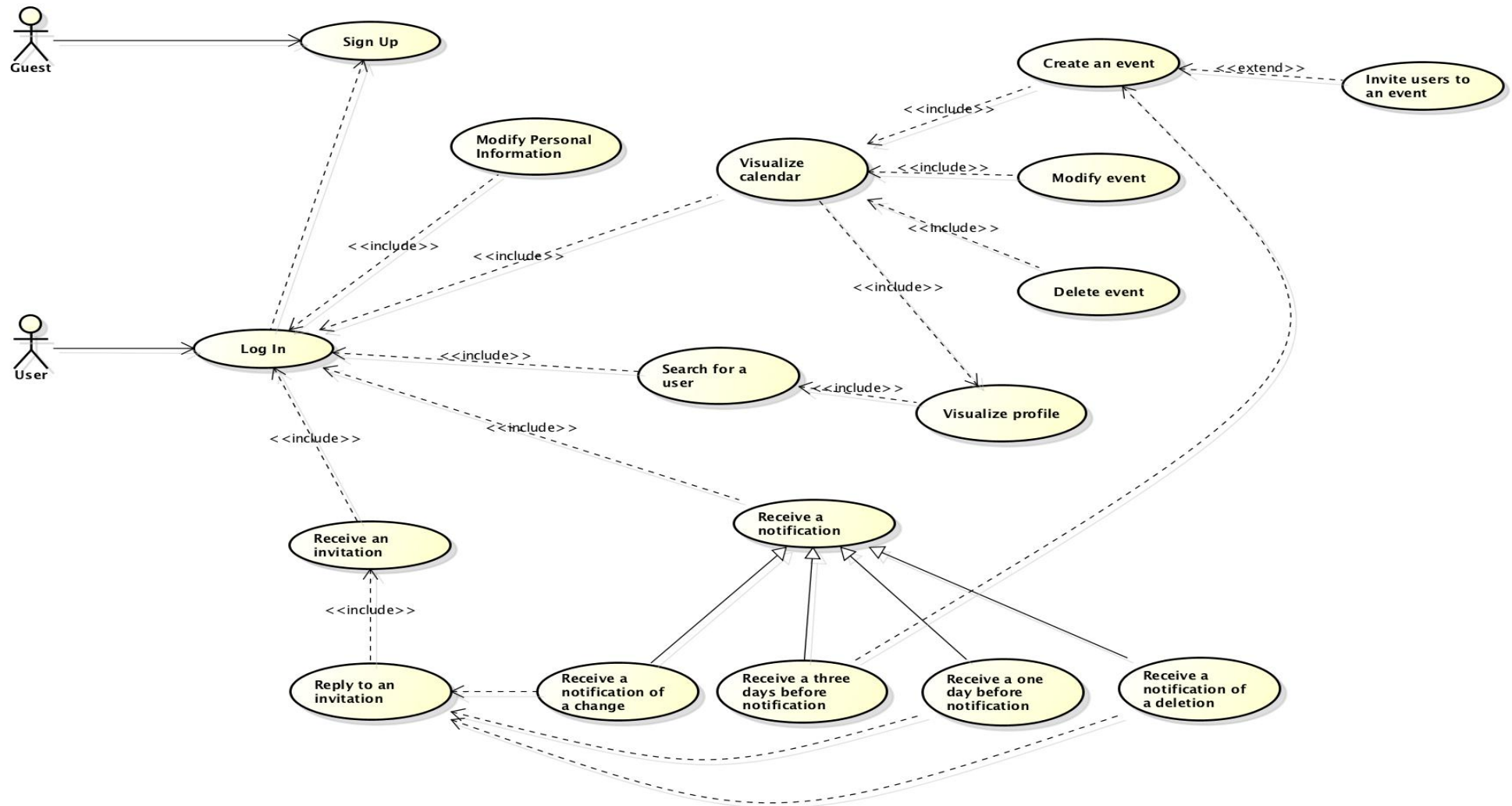
5. UML Models

5.1 Use Case Diagram

We can derive some use cases from the scenarios identified in the previous paragraph:

- Sign up;
- Log in;
- Search for a user;
- Visualize another user's personal page;
- Create a new event;
- Invite users to an event;
- Receive an invitation to an event;
- Reply to an invitation;
- Modify and event;
- Receive a notification of a change;
- Receive a notification of a deletion;
- Receive a three days before notification;
- Receive a day before notification;
- Modify personal information;
- Visualize a calendar;
- Visualize profile.

Here it is a Representation of the Use Case Diagram



5.2 Use Case Description

We describe in a detailed way the use cases that we derived from the scenarios. We try to define them all.

It is important to understand that all the references to “pages”, “buttons” or “input forms” are only hypothesis to make the situation as clear as possible and to help the reader to draw a visual picture in his mind of what we plan to do, but the real structures will be well defined in the Design Document.

We refine the use case “Sign up”:

Name	Sign up
Actors	Guests
Entry Conditions	The guest isn't registered to the website
Flow of events	<ul style="list-style-type: none">• The guest enters the website;• The guest clicks on the “SIGN UP” button;• The guest fills in the form where he has to write:<ul style="list-style-type: none">-Name-Surname-Email-Username-Password-Date of birth And optionally: <ul style="list-style-type: none">-Telephone number-Address-Picture <ul style="list-style-type: none">• The guest clicks the “DONE” button;• The system shows him his personal page.
Exit conditions	Registration successfully done.
Exceptions	An exception can be caused if the username the guest inserts already exists or if some field that are not optional aren't filled.

We refine the use case “Log in”:

Name	Log in
Actors	User
Entry Conditions	User has successfully signed up to the system
Flow of events	<ul style="list-style-type: none">• The user enters the web site.• The user fills in the text fields in the home page with username and password.• The user clicks on the “LOG IN” button.
Exit conditions	The system shows the user his personal page.
Exceptions	The password and/or username inserted by the user are wrong. The System shows an error message to the user.

We refine the use case “Search for a user”:

Name	Search for a user
Actors	User
Entry Conditions	The user must be logged in
Flow of events	<ul style="list-style-type: none">• The user inserts the username of the person he’s looking for in the search field next to the magnifier;• The guest clicks on the “MAGNIFIER”;• The system shows him the results.
Exit conditions	The user selects the one who’s interested in by clicking on him and starts the “VISUALIZE PERSONAL PAGE” use case.
Exceptions	The only exception that can happen is “NO RESULT FOUND”.

We refine the use case “Visualize another user’s personal page”:

Name	Visualize another user’s personal page
Actors	User
Entry Conditions	User has just finished the “SEARCH FOR A USER” use case
Flow of events	<ul style="list-style-type: none">• The user select the profile he is interested in;• The system shows him the personal page of the user he selected;
Exit conditions	The user visualizes the personal page of the user he was looking for.
Exceptions	No exceptions

We refine the use case “Create a new event”:

Name	Create a new event
Actors	User
Entry Conditions	User clicks on the button “ADD EVENT” in his personal calendar.
Flow of events	<ul style="list-style-type: none"> • The system shows him a form that the user has to fill in with the following information: <ul style="list-style-type: none"> – The name of the event – The place in which the event will take place – The time at which the event will take place – The type of the event (indoor or outdoor) – The visibility of the event (public or private) • If the user selects the type outdoor the system will show an additional box, which contains a set of weather conditions that the user may consider as bad for the event. The user has to select as least one weather forecast that he considers as bad. • The user selects the weather conditions that he considers bad for this specific event (at least one). • The user clicks the button “CREATE EVENT”.
Exit conditions	The system adds the event to user’s calendar and attaches to it the current weather forecast. The use case “INVITE PEOPLE TO AN EVENT” starts.
Exceptions	<ul style="list-style-type: none"> • The selected date is in the past or the selected time is not consistent, the system shows an error message. • The user aborts the operation by clicking the “CANCEL” button. The system shows the user’s personal page.

We refine the use case “Invite users to an event”:

Name	Invite users to an event
Actors	User
Entry Conditions	User has just finished the “CREATE A NEW EVENT” use case
Flow of events	<ul style="list-style-type: none">• System shows the user a page where the user can search by username other users and clicking on the “ADD” button he invites them to the event;• The invited user is visualized in the list of invitations, which is next to the search form.
Exit conditions	The user finishes inviting people to the event and clicks on the button “DONE”. The system visualizes user’s calendar.
Exceptions	The user searches the username of a person who is already invited to the event, the system doesn’t show any result.

We refine the use case “Receive an invitation”:

Name	Receive an invitation
Actors	User
Entry Conditions	User, who has received a new invitation for an event, logs in the web site.
Flow of events	<ul style="list-style-type: none">• The system shows the user a message that contains the list of the events, which he has been invited to.• The user clicks on the button “SHOW INVITATION LIST” and the use case “REPLY TO AN INVITATION” starts.
Exit conditions	The system adds new invitations to the invitation list.
Exceptions	The user clicks on the button “cancel” and the system shows him the personal page.

We want to refine the use case “Reply to an invitation”:

Name	Reply to an invitation
Actors	User
Entry Conditions	<ul style="list-style-type: none">• User clicks on “INVITATION LIST”.• User must have done the “RECEIVE AN INVITATION” use case.
Flow of events	<ul style="list-style-type: none">• The system shows the user the list of the events, which he is invited to.• If the user wants to accept an invitation, he will click on the button “ACCEPT” next to the event name.• If the user wants to reject an invitation, he will click on the button “DECLINE” next to the event name.• The system modifies the list by disabling the buttons associated to the event. Furthermore, if the user has rejected the invitation, the system removes it from the invitation list.
Exit conditions	If the user has accepted the event, the system adds it to his calendar.
Exceptions	No exceptions.

We refine the use case “Modify an event”:

Name	Modify an event
Actors	User
Entry Conditions	User must have done the “CREATE AN EVENT” use case. User has visualized his calendar.
Flow of events	<ul style="list-style-type: none">• The user select the event he wants to modify;• The user clicks on the “MODIFY” button;• The system shows him a page where he can fill in again the fields he has completed when he created the event;• The users modify the field that he wants to change;• The users clicks on the “SAVE” button.
Exit conditions	The user can see its modified event.
Exceptions	<ul style="list-style-type: none">- If the users violate date and time constraints the system will show them an error.- The user aborts the operation by clicking the “CANCEL” button. The system shows the user’s personal page.

We want to refine the use case “Delete an event”:

Name	Delete an event
Actors	User
Entry Conditions	User clicks on the button “DELETE EVENT” in his personal calendar.
Flow of events	<ul style="list-style-type: none"> • The system shows the user the list of his events. • The user clicks on the event he wants to eliminate. • The system shows him a message that asks for a confirmation of the operation. • The user clicks on the button “YES”.
Exit conditions	The system removes the event from his calendar, and then it notifies all the participants about the cancellation of the event.
Exceptions	If the user clicks on the button “NO” in the message, the system will reload the list of his events without doing any modification.

We want to refine the use case “Receive a notification of a change”:

Name	Receive a notification of a change
Actors	User
Entry Conditions	Another user, the event creator, has performed the “MODIFY AN EVENT” use case. User logs in the system.
Flow of events	<ul style="list-style-type: none"> • The system shows the user a box with the detail of the event that has been modified; • The system asks to the user “DO YOU STILL WANT TO PARTECIPATE?”; • The user clicks the “YES” or “NO” button.
Exit conditions	The system reschedules the event in the user’s calendar or it deletes the event.
Exceptions	No exceptions

We want to refine the use case “Receive a notification of event cancellation”:

Name	Receive a notification of event cancellation
Actors	User
Entry Conditions	<ul style="list-style-type: none">• The user logs in the web site.• An event, which the user was invited to, has been cancelled.
Flow of events	<ul style="list-style-type: none">• The system shows the user a message that notifies him the cancellation.• The user clicks on the button “OK”.• If the user has already decided to participate to the event, then the system eliminates it from his calendar; otherwise it simply removes it from his invitation list.
Exit conditions	The system shows the user his personal page.
Exceptions	No exceptions.

We want to refine the use case “Receive a three days before notification”:

Name	Receive a three days before notification
Actors	User
Entry Conditions	User must have done the “CREATE AN EVENT ” use case. User has to be logged in.
Flow of events	<ul style="list-style-type: none">• The system shows a box where there is written that the weather forecast for the day of the event is bad;• The system proposes to the event creator the closest sunny day;• The user clicks on the “YES” button if he accepts to reschedule the event in the day the system proposes or on the “NO” button not to accept the system proposal and can decide later to modify it by performing the “MODIFY AN EVENT” use case.
Exit conditions	The user can see his event rescheduled or it can see it not changed.
Exceptions	No exceptions

We want to refine the use case “Receive a day before notification”:

Name	Receive a day before notification
Actors	User
Entry Conditions	<ul style="list-style-type: none">• The user logs in the web site.• An outdoor event, which the user will participate to, has a weather forecast that is defined as bad by the event creator and it will take place within one day.
Flow of events	<ul style="list-style-type: none">• The system shows the user a notification that signals him that there is a bad weather forecast for the event.• The user clicks on the button “OK”.
Exit conditions	The system shows him the personal page.
Exceptions	No exceptions.

We want to refine the use case “Modify personal information”:

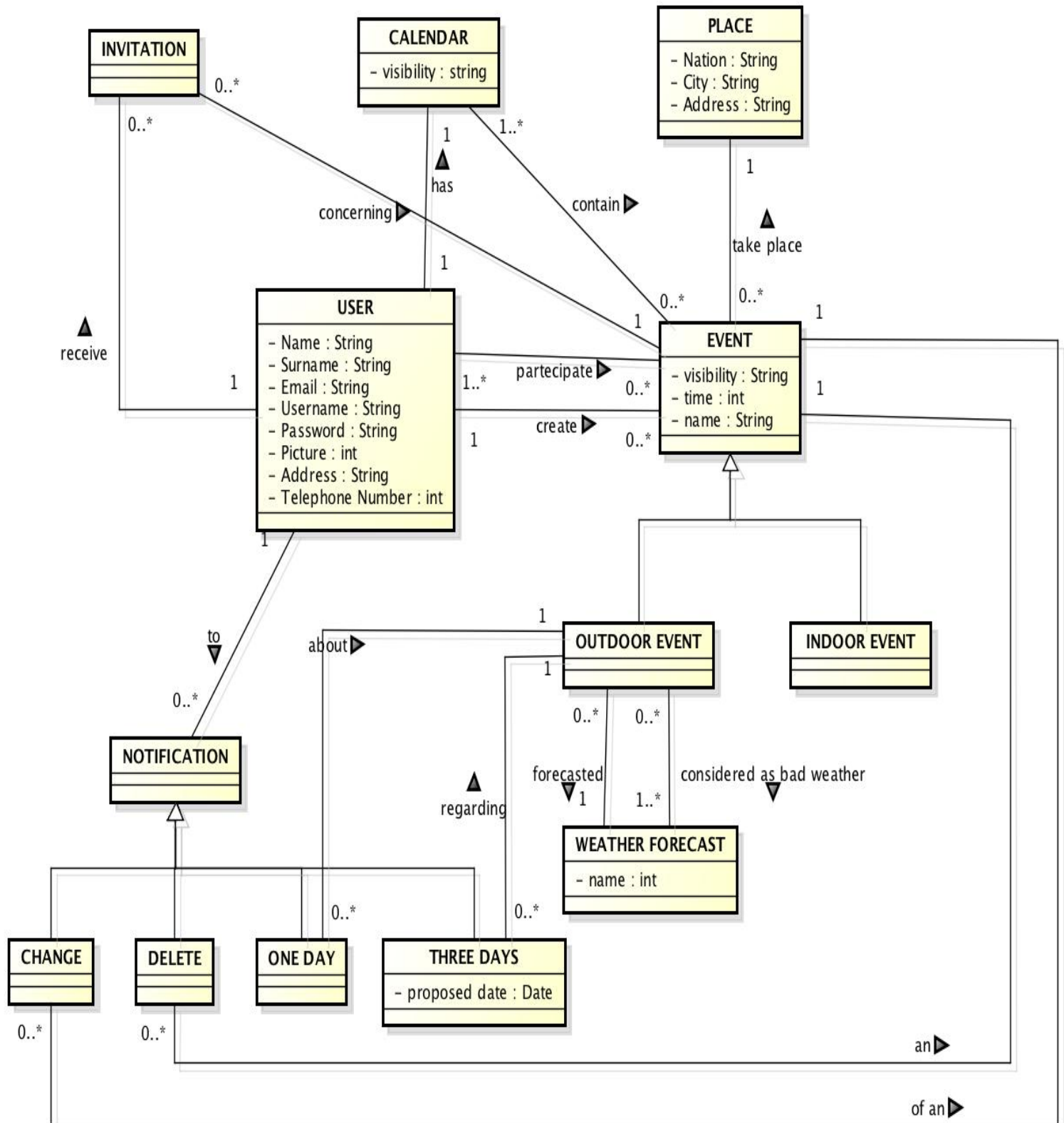
Name	Modify personal information
Actors	User
Entry Conditions	User must be logged in.
Flow of events	<ul style="list-style-type: none">• The user clicks on the “MODIFY PERSONAL INFORMATION” button;• The system shows him a page with the form the user has completed when he signed up;• The user can modify each field<ul style="list-style-type: none">-Name-Surname-Email-Date of birth And optionally: <ul style="list-style-type: none">-Telephone number-Address-Picture. The user can also change the visibility of his calendar, that is set to public by default. <ul style="list-style-type: none">• The user clicks on the “SAVE THE CHANGES” button.
Exit conditions	The system shows him his modified personal page.
Exceptions	An exception can be caused if some fields that are not optional aren’t filled.

We want to refine the use case “Visualize a Calendar”:

Name	Visualize a Calendar
Actors	User
Entry Conditions	The user clicks on the button “CALENDAR” either in his personal page or in the personal page of another user.
Flow of events	<ul style="list-style-type: none">• If the user clicks the button from his personal page:<ul style="list-style-type: none">– The system will show him the calendar containing all the events he has created and all the events, created by other users, which he will participate to.– The user can click on an event in order to obtain more information about it.• If the user clicks the button in an another user’s personal page:<ul style="list-style-type: none">– The system will show him the calendar containing all the events, which the other user will participate to. In this case all the information about public events will be visible, while the information about private ones will be hidden (substituted by a box containing the label “BUSY”).
Exit conditions	The calendar is visualized.
Exceptions	In the second case, if the other user has defined his calendar as private, the system will not show it, instead a message warning is visualized.

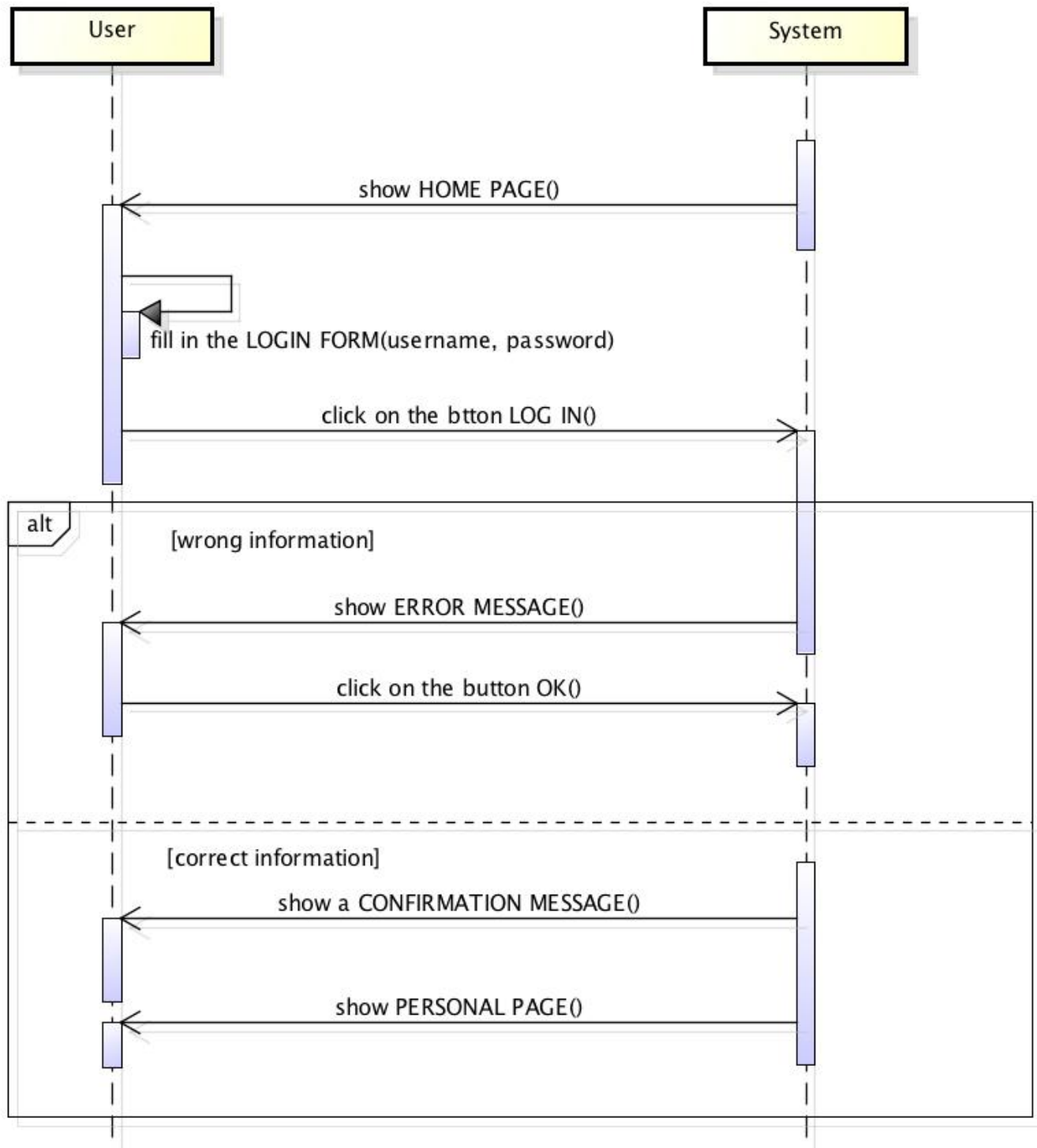
5.3 Class Diagram

Now that we have refined every use case we can draw a class diagram of our system:

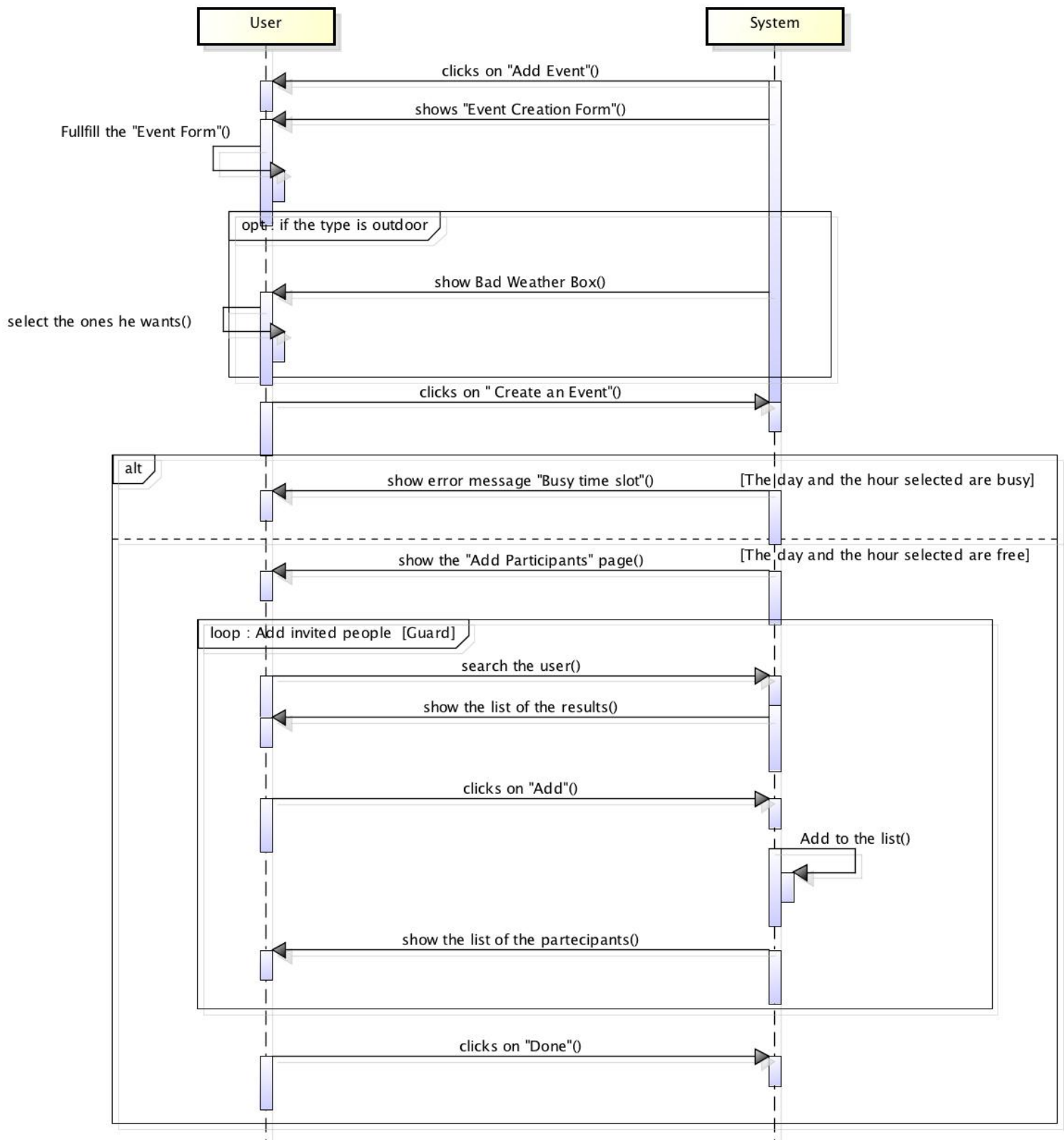


5.4 Sequence Diagrams

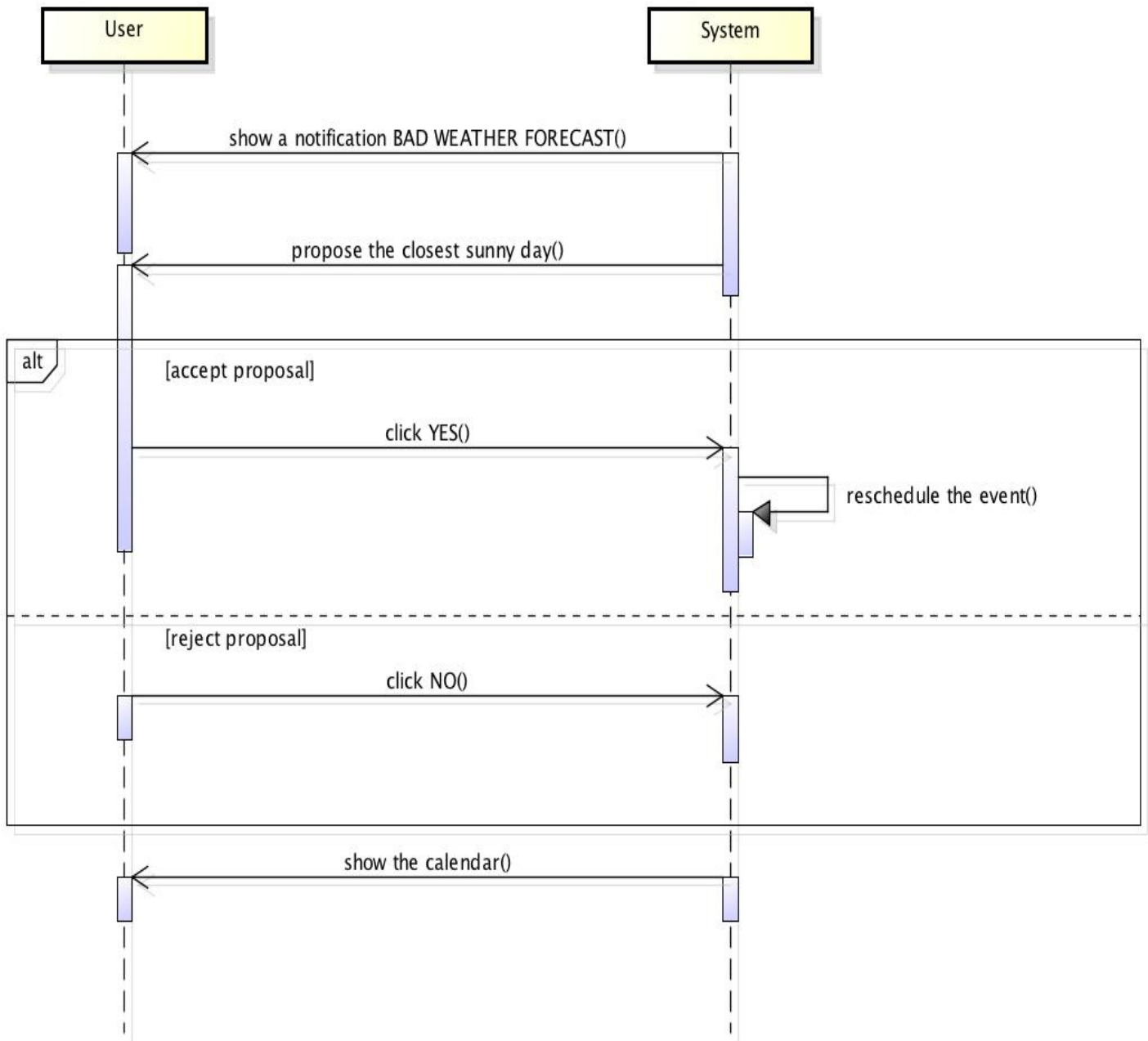
5.4.1 Log In



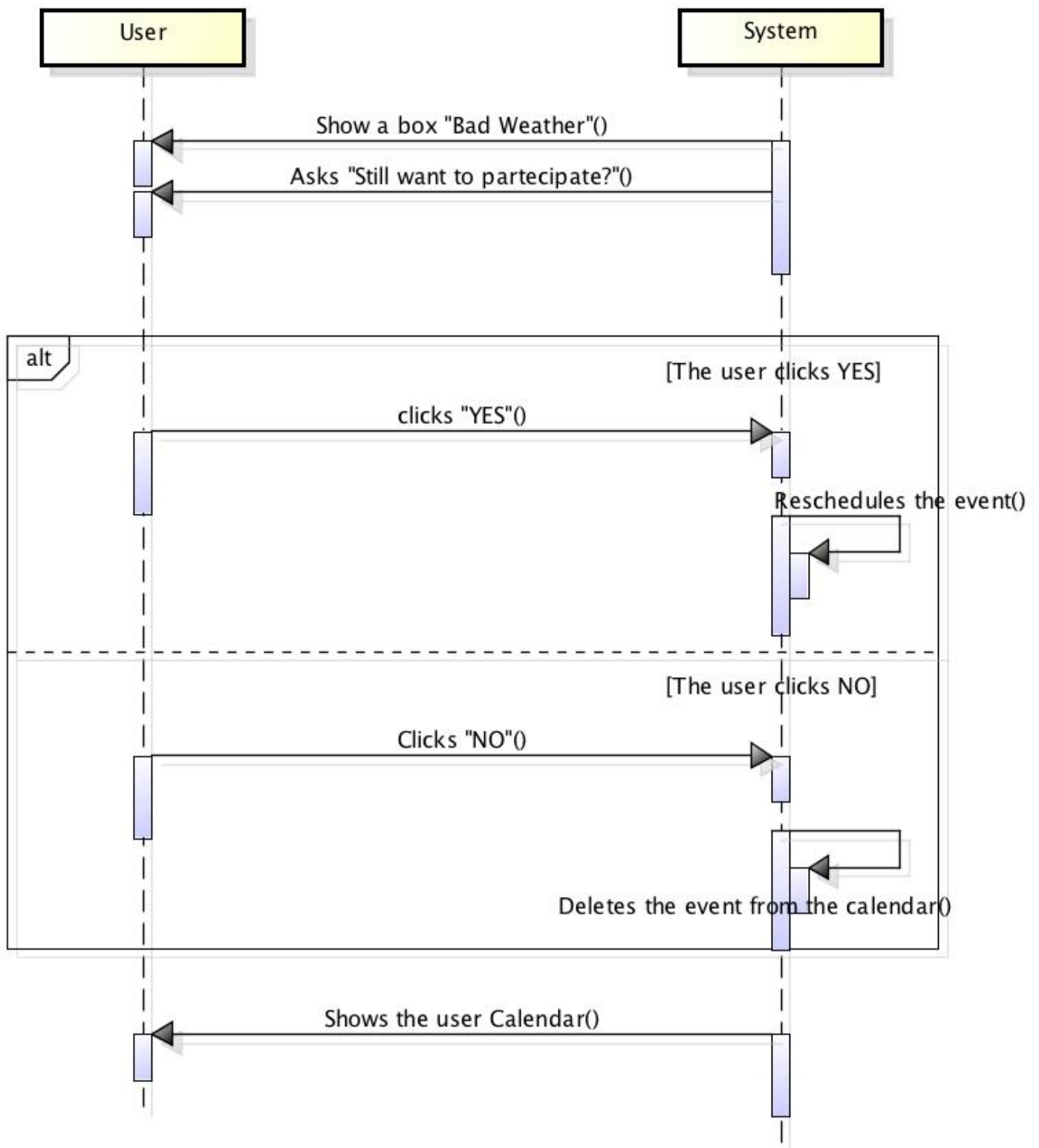
5.4.2 Create an Event



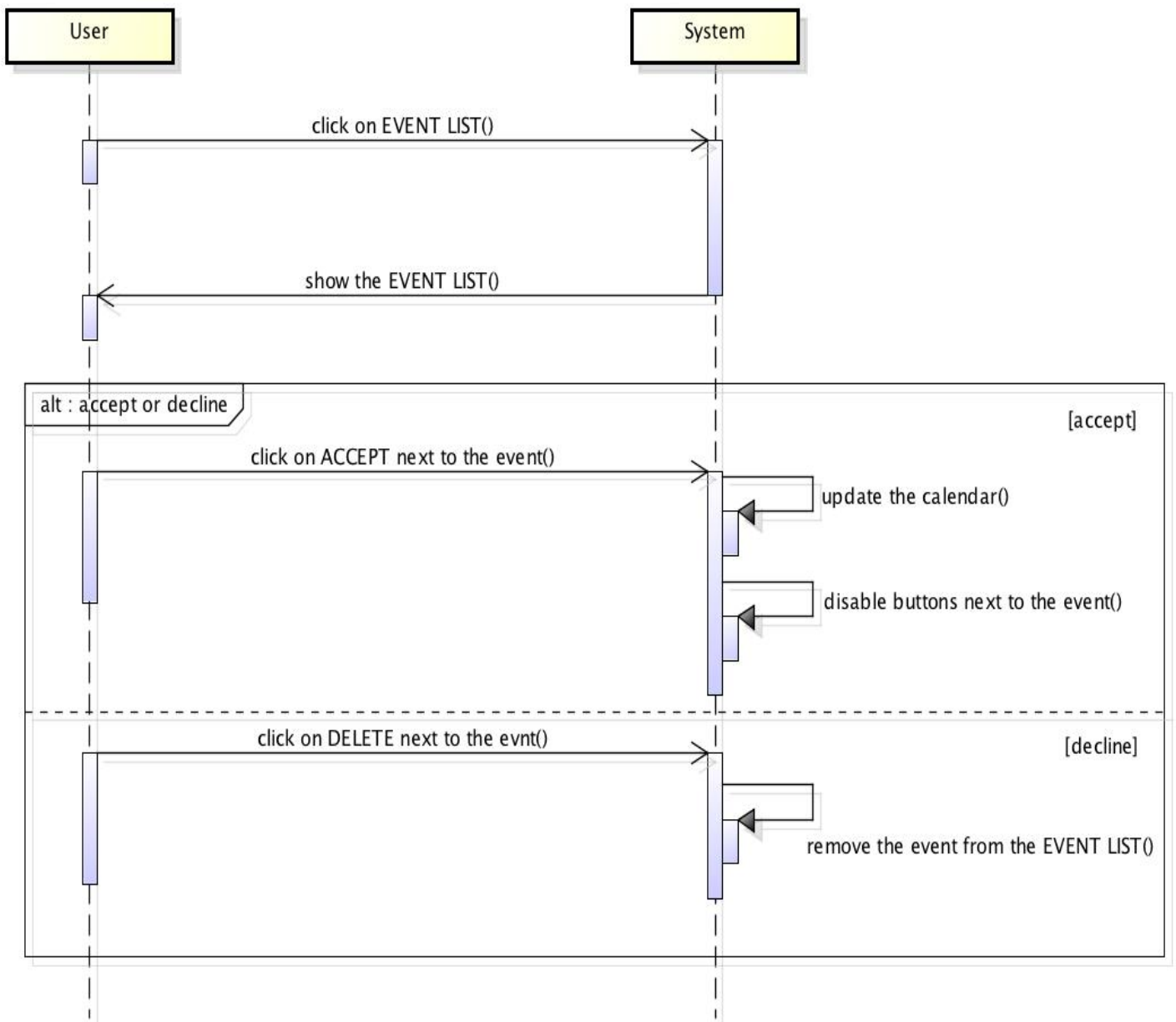
5.4.3 Receive a Three-Days Before Notification



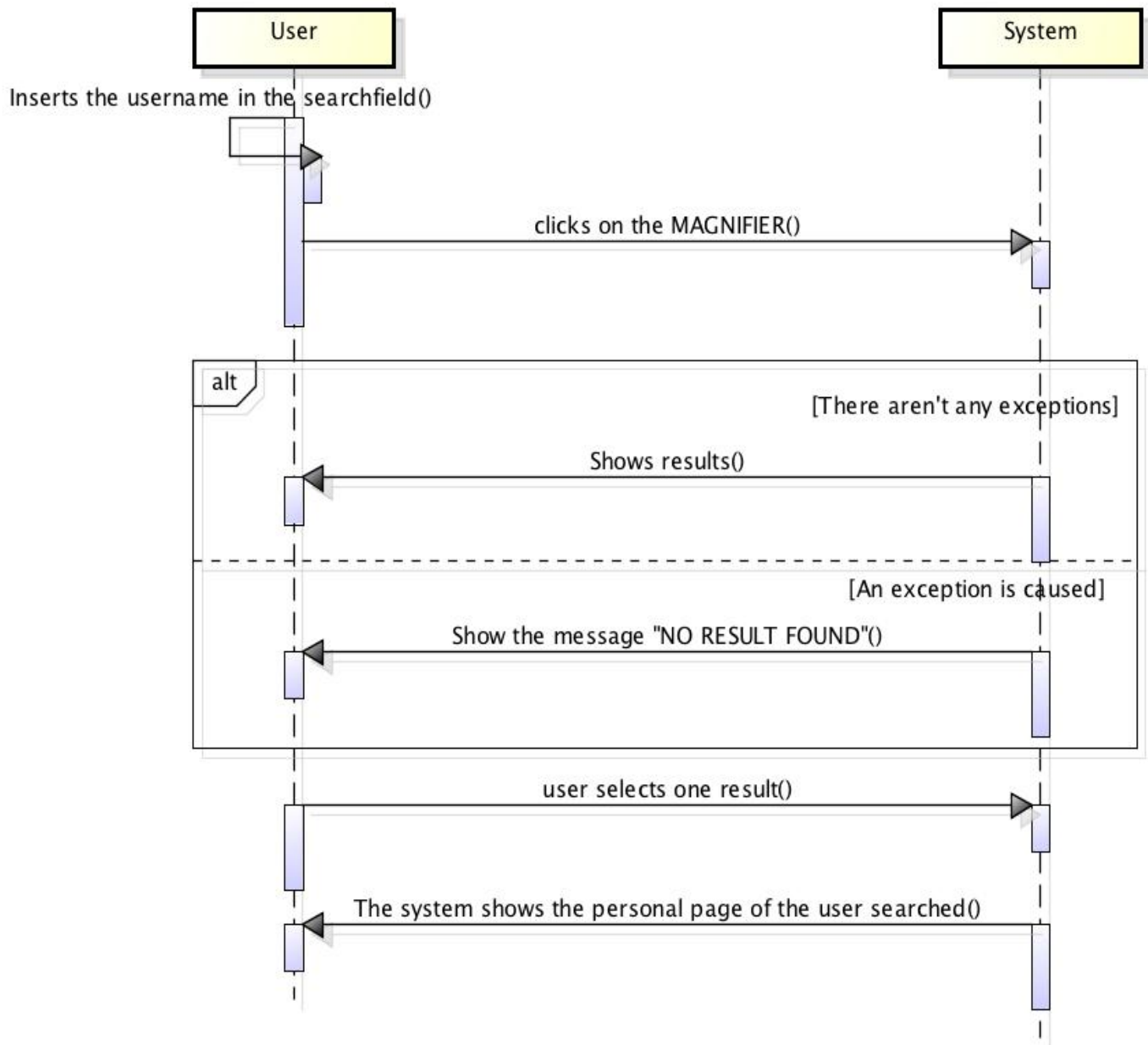
5.4.4 Receive a One-Day Before Notification



5.4.5 Reply to an Invitation



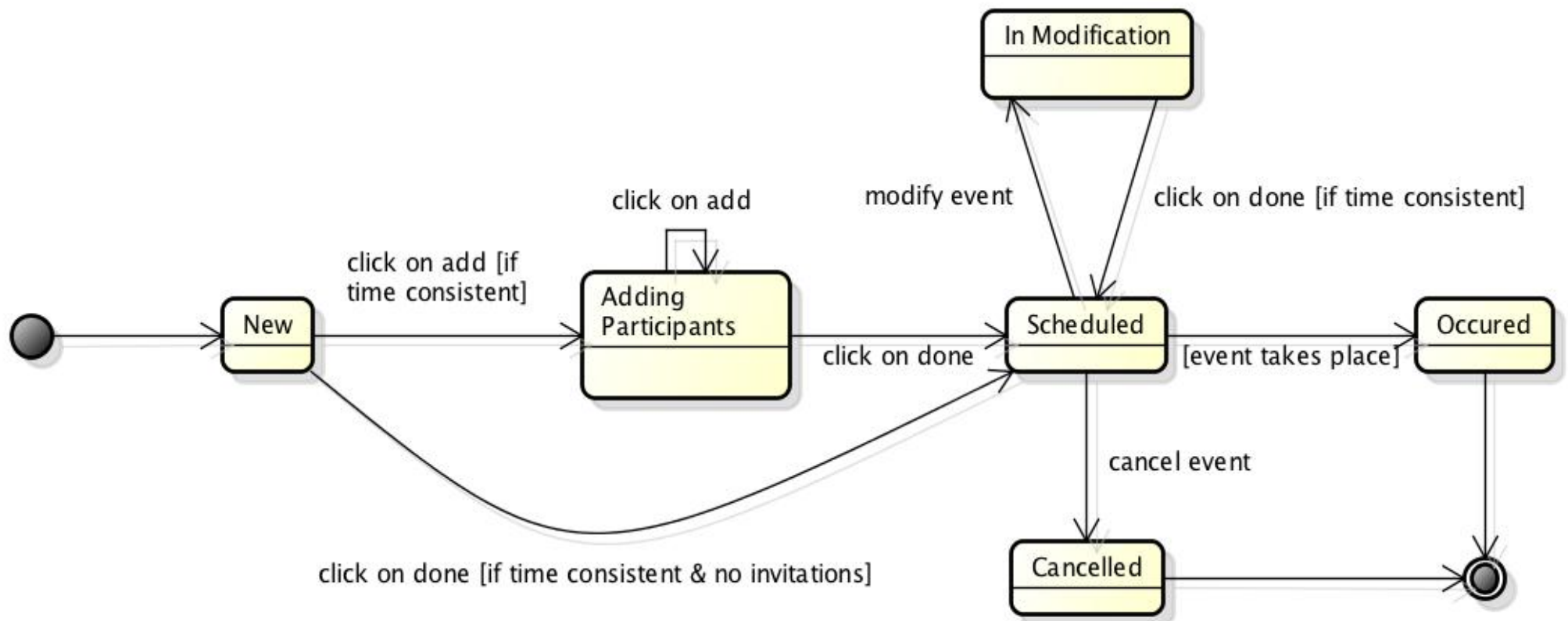
5.4.6 Search for a User



powered by Astah

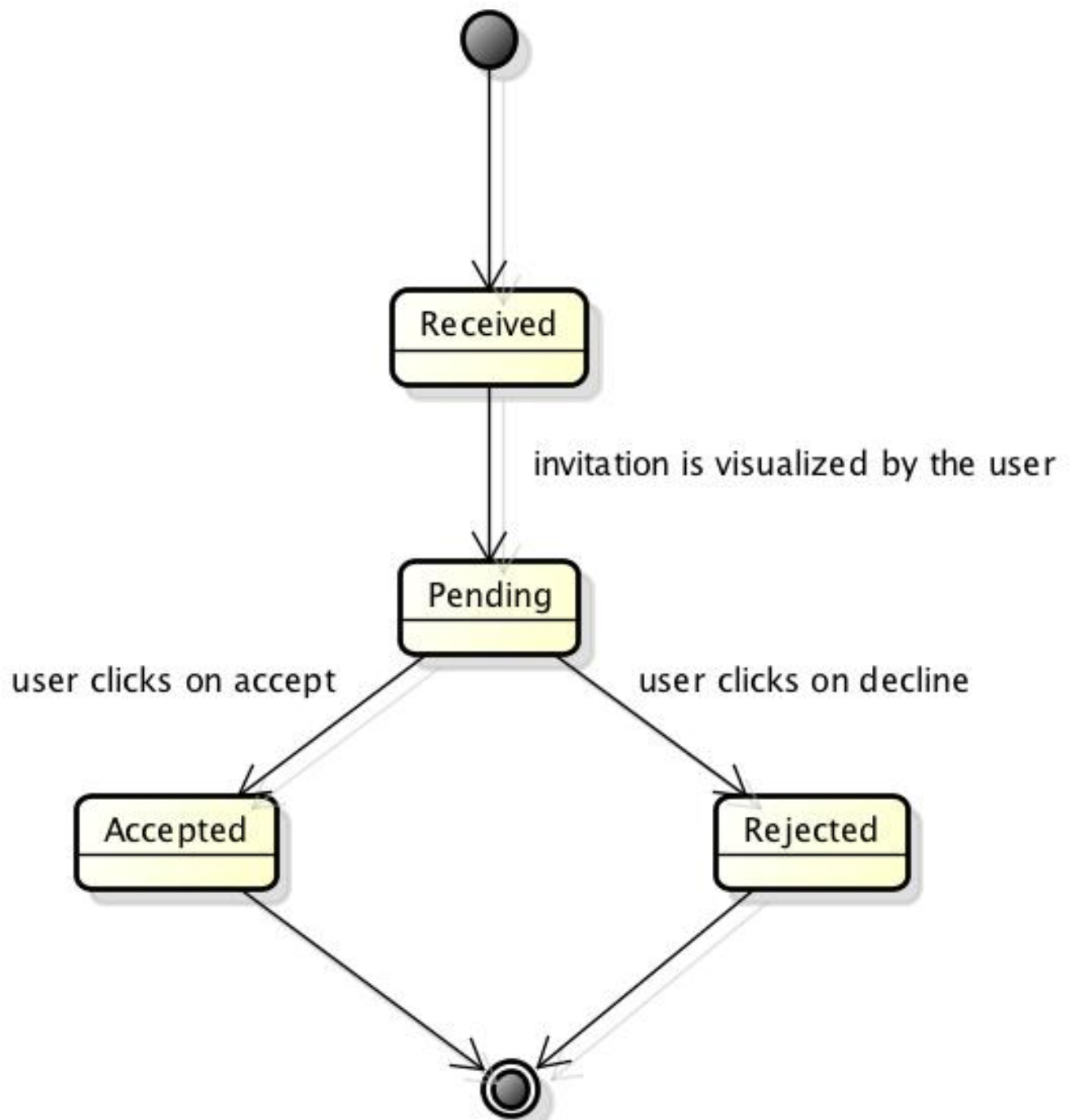
5.5 State chart Diagrams

5.5.1 Event



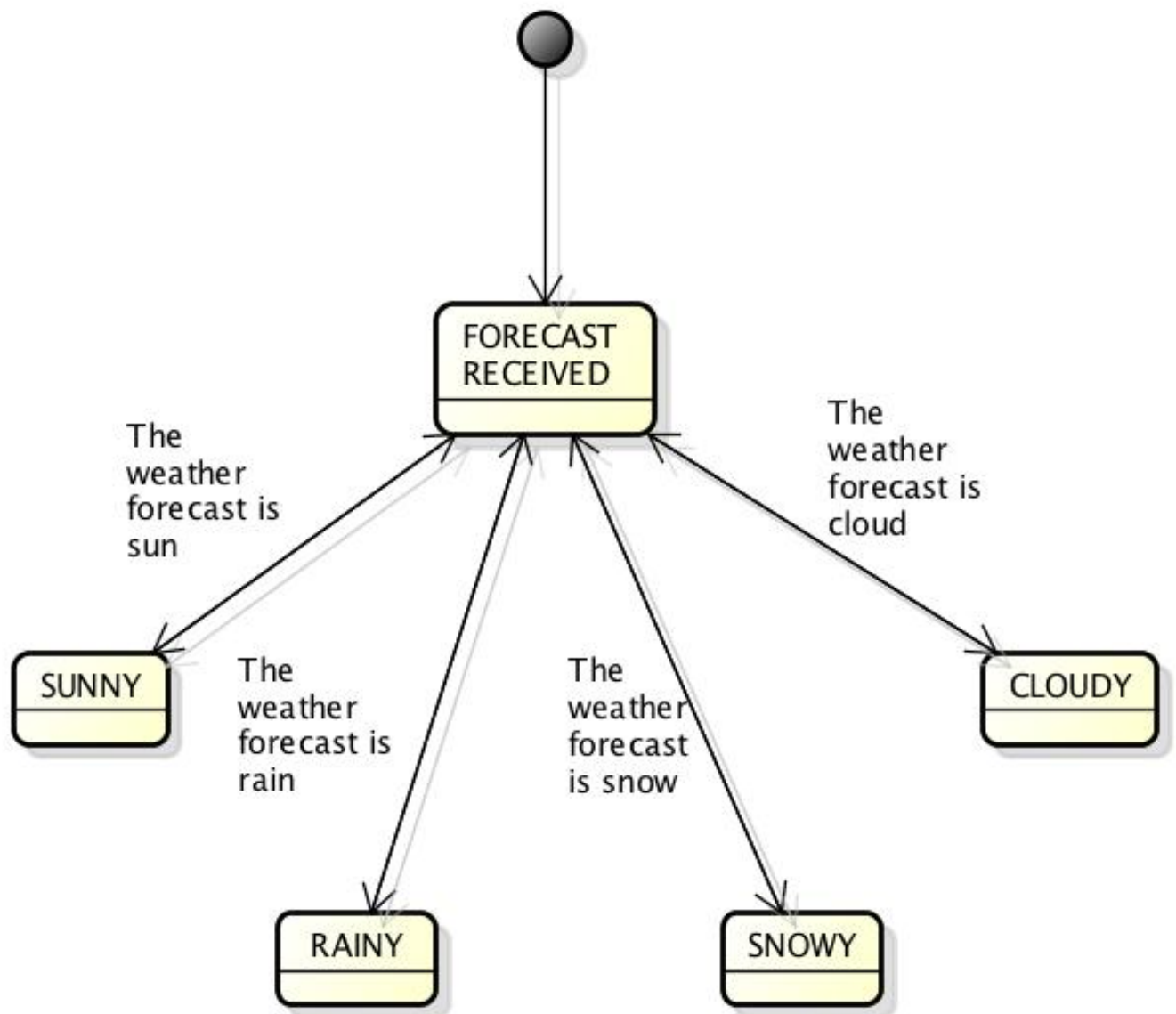
powered by Astah

5.5.2 Invitation



powered by Astah

5.5.3 Event Weather Forecast



powered by Astah 

6.Alloy Modelling

We've used Alloy Analyzer to specify the properties of our application, starting from the representation of the Class Diagram. In this paragraph we've reported the alloy code used for the analysis and some examples of worlds generated with the analyzer.

6.1 Alloy Code

//SIGNATURES

```
sig User {
    participate: set Event,
    calendar: one Calendar
}

abstract sig Event {
    place: one Place,
    creator: one User
}

sig OutDoorEvent extends Event {
    weather: one WeatherForecast,
    badweather: set WeatherForecast
} {
    #badweather > 0
}

sig InDoorEvent extends Event { }

sig Place { }

sig WeatherForecast {}

sig Invitation {
    receiver: one User,
    event: one Event
}

sig Calendar {
    contain: set Event
}

abstract sig Notification {
    sent: one User
}
```

```

sig OneDayNotification extends Notification {
    event: one OutDoorEvent
}

sig ThreeDaysNotification extends Notification {
    event: one OutDoorEvent
}

sig DeleteNotification extends Notification {
    event: one Event
}

sig ChangeNotification extends Notification {
    event: one Event
}

//FACTS

fact OneUserPerCalendar {
    //Every calendar belongs to exactly one user
    all c: Calendar | (one u: User | u.calendar = c)
}

fact EventInCreatorCalendar {
    //Every event is contained in owner's calendar
    all e: Event | e in e.creator.calendar.contain
    //If a user creates an event, he is also a participant
    all e: Event | e in e.creator.participate
}

fact EventInParticipantsCalendars {
    //All the events, which the user will participate to, are contained in his
    //calendar.
    all u: User | u.participate = u.calendar.contain
}

fact NoInvitationToHimself {
    //A user can't send an invitation to himself
    no i: Invitation | i.receiver = i.event.creator
}

fact ParticipateOnlyIfInvited {
    //A user can participate to an event only if he has previously received
    //an invitation for that event.
    no u: User | some e: Event |
        (e in u.participate and e.creator != u and
        (no i: Invitation | i.receiver = u and i.event = e))
}

```

```

fact OneInvitationPerUser {
    //There are no duplicate invitations
    no disj i1, i2: Invitation | i1.receiver = i2.receiver and
        i1.event = i2.event
}

fact ThreeDaysNoificationProperties{
    //Three days notifications are sent only to the creator of the event
    all n: ThreeDaysNotification | n.event.creator = n.sent
    //A three days notification is sent only if the weather forecast for the
    event is considered as bad by the creator
    all n: ThreeDaysNotification |
        n.event.weather in n.event.badweather
    //If the weather forecast for the event is bad, then there exists
    exactly one three day notification for that event
    all e: OutDoorEvent | (e.weather in e.badweather) implies
        (one n: ThreeDaysNotification | n.sent = n.event.creator)
}

fact OneDayNotificationProperties {
    //One day notifications are sent only to the participants of the event
    all n: OneDayNotification | n.event in n.sent.participate
    //If the weather forecast for the event is bad, then exactly one day
    notification is sent to each of the participants
    all e: OutDoorEvent | (e.weather in e.badweather) implies
        (all u: User | (e in u.participate) implies
            (one n: OneDayNotification | n.sent = u and n.event = e))
    //One day notifications are sent only if the weather forecast for the
    event is considered as bad by the event creator
    all n: OneDayNotification | n.event.weather in n.event.badweather
}

fact ChangeNotificationProperties {
    //Change notifications are sent only to the participants of the event
    all n: ChangeNotification | n.event in n.sent.participate
    //In case of modification, exactly one change notification is sent to all
    the participants of the event
    //Note: We're assuming that when the event is modified more than
    one time, only the notifications for the last modification are kept by the
    system
    all e: Event | (some n: ChangeNotification | n.event = e) implies
        (all u: User | (e in u.participate) implies
            (one n: ChangeNotification | n.event = e and n.sent = u))
}

```

```

fact DeleteNotificationProperties {
    //Delete notifications are sent only to the participants of the event
    all n: DeleteNotification | n.event in n.sent.participate
    //In case of cancellation, exactly one delete notification is sent to all
    the participants of the event
    all e: Event | (some n: DeleteNotification | n.event = e) implies
        (all u: User | (e in u.participate) implies
            (one n: DeleteNotification | n.event = e and n.sent = u))
}

//ASSERTIONS

//Check that there can't be calendar without an owner
assert NoCalendarWithoutUser {
    no c: Calendar | (no u: User | u.calendar = c)
}

check NoCalendarWithoutUser

//Check that one-day notifications must be sent to all the participants of the event
assert OneDayNotificationToAllParticipants {
    no disj u1, u2: User | (one n1: OneDayNotification |
        n1.sent = u1 and n1.event in u1.participate and n1.event in
        u2.participate and (no n2: OneDayNotification | n2.sent = u2
            and n2.event = n1.event))
}

check OneDayNotificationToAllParticipants

//Check that a user can't be invited twice to the same event
assert NoDoubleInvitations {
    no u: User, e: Event | (some disj i1, i2: Invitation |
        i1.receiver = u and i2.receiver = u and i1.event = e and
        i2.event = e)
}

check NoDoubleInvitations

//Check that if the weather forecast for an outdoor event is not bad, then there is
no One day notification or three day notification for that event
assert NoBadWeatherNoNotifications {
    all e: OutDoorEvent | (e.weather not in e.badweather) implies
        ((no n: OneDayNotification | n.event = e) and
            (no n: ThreeDaysNotifcation | n.event = e))
}

```

check NoBadWeatherNoNotifications

//PREDICATES

//Show a world in which there is an outdoor event with at least two participants and bad weather forecast

```
pred showParticipation {  
    some e: OutdoorEvent, u1: User, u2: User |  
        u1 != u2 and e in u1.participate and e in u2.participate and  
        e.weather in e.badweather  
}  
run showParticipation
```

//Show a world in which there is an event that has been modified

```
pred showChangeNotification {  
    some e: Event | some disj u1, u2: User |  
        e in u1.participate and e in u2.participate and  
        (some n: ChangeNotification | n.event = e)  
}  
run showChangeNotification
```

//Show a world that enlightens invitations properties

```
pred showInvitations {  
    #Event = 1  
    #Invitation >= 2  
    #WeatherForecast = 0  
}
```

```
run showInvitations for 3 but 0 Notification
```

```
pred show {  
    some u: User | some e: Event |  
        e in u.participate and e.creator != u  
}
```

```
run show
```

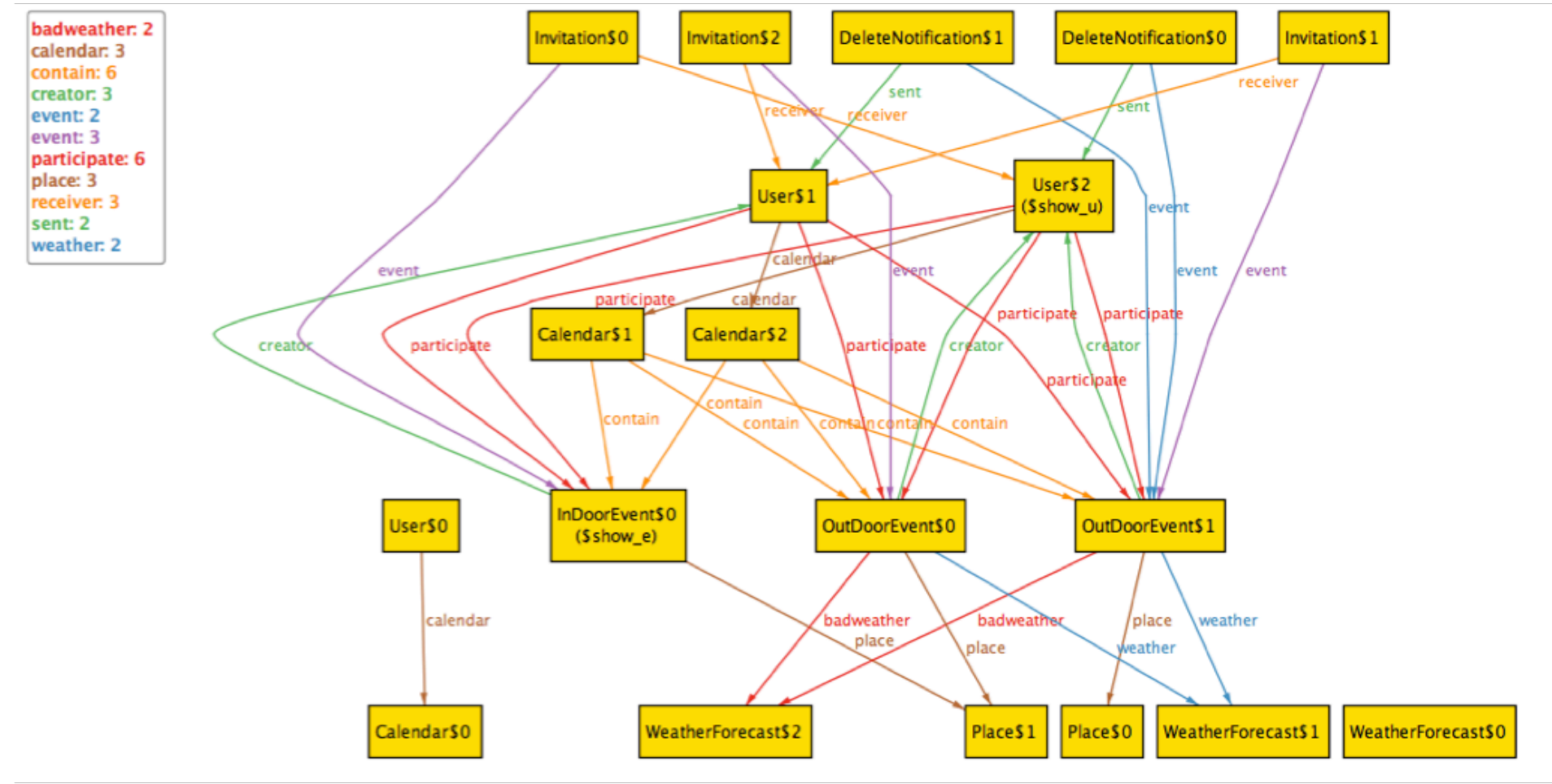
Here the result of the analysis.

```
#1: No counterexample found. NoCalendarWithoutUser may be valid.  
#2: No counterexample found. OneDayNotificationToAllParticipants may be valid.  
#3: No counterexample found. NoDoubleInvitations may be valid.  
#4: No counterexample found. NoBadWeatherNoNotifications may be valid.  
#5: Instance found. showParticipation is consistent.  
#6: Instance found. showChangeNotification is consistent.  
#7: Instance found. showInvitations is consistent.  
#8: Instance found. show is consistent.
```

6.2 Alloy Worlds

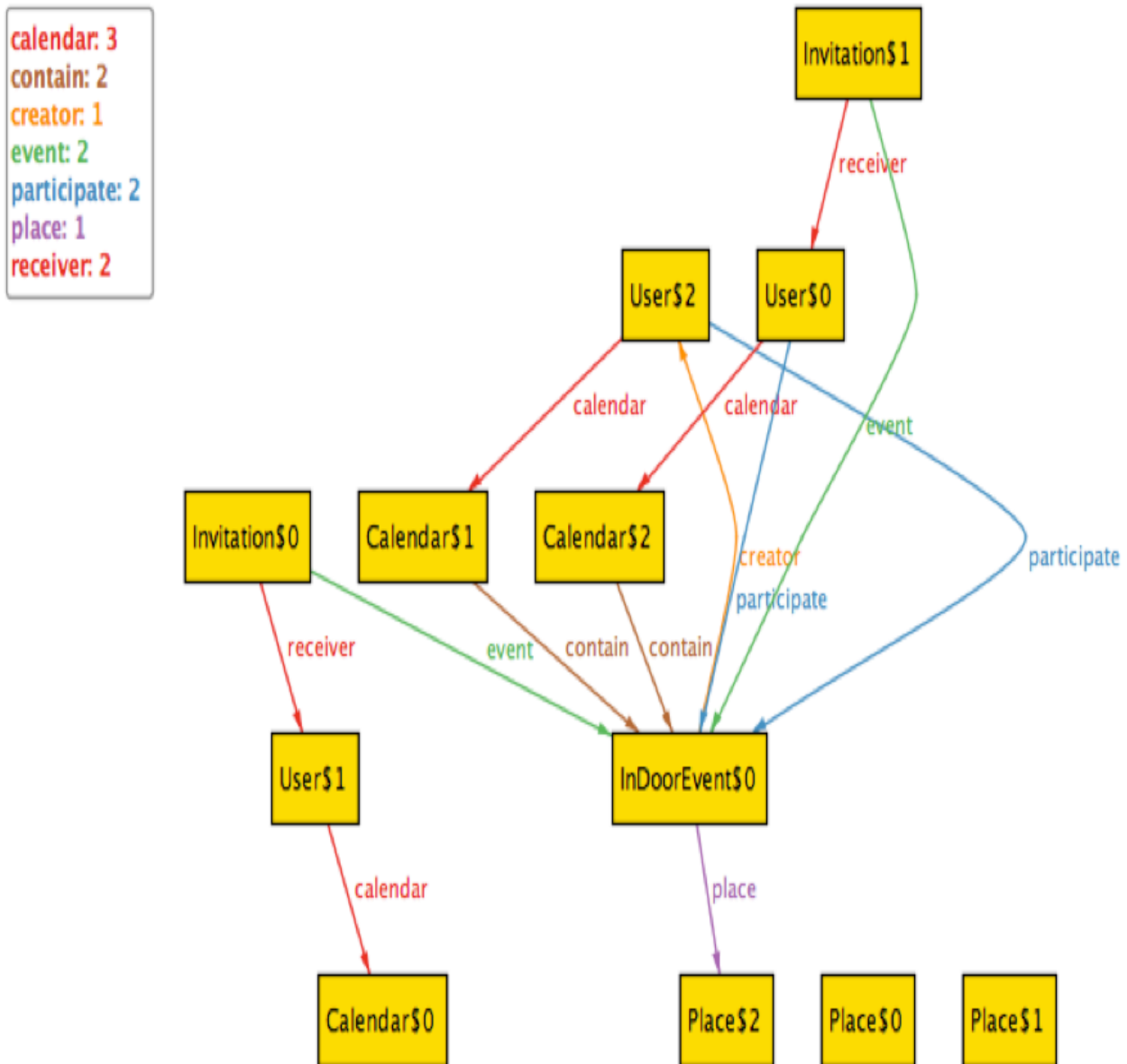
6.2.1 General World

The following world is a general world generated with the analyzer. It has been generated using the predicate show.



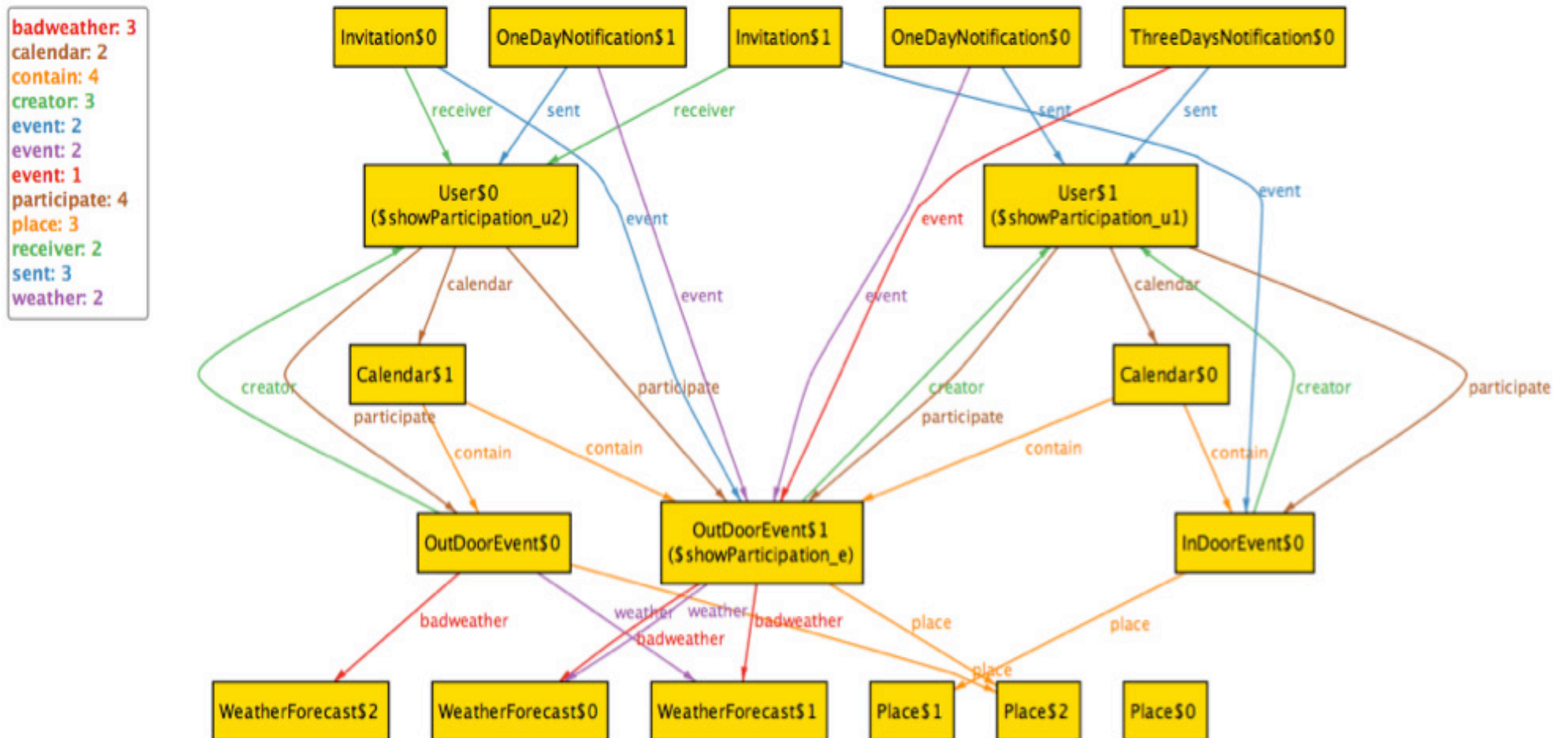
6.2.2 Invitations Properties

The following world shows how the model satisfies the main constraints on invitations, which are: no invitations to himself and participate only if invited. It has been generated using the predicate showInvitations.

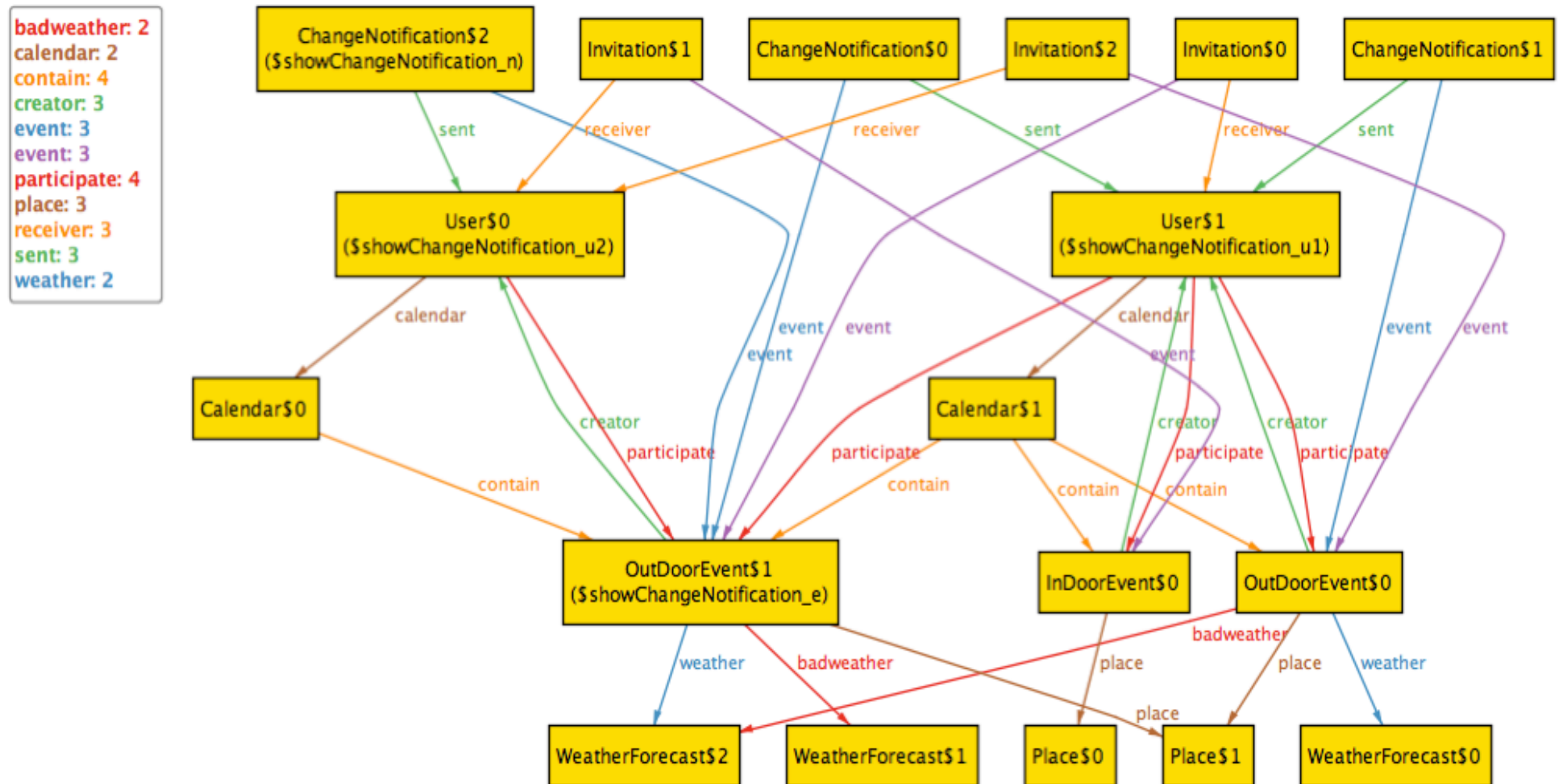


6.2.3 Notifications properties

The following worlds show how the model satisfies the main constraints on notifications. The first one shows that if there is a bad weather forecast for an outdoor event, then all the participants receives a One day before notification for that event. It has been generated using the predicate showParticipation.



The second one shows that if there is a modification to an event, it is signalled to all the participants. It has been generated using the predicate showChangeNotification.



7. Used tools

The tools we used to create the RASD document are:

- Microsoft Office Word 2011: to redact and to format this document;
- Astah: to create the State Charts, the Class Diagram, the Sequence Diagrams and the Use Case Diagram;
- Alloy Analyzer 4.2: to prove the consistency of our model.

For redacting and writing this document we spent 30 hours per person