Travlendar+ project Daverio Fiorillo

POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Daverio Fiorillo
Version:	1.0
Date:	29-October-2017
Download page:	https://github.com/FiorixF1/DaverioFiorillo.git
Copyright:	Copyright © 2017, Daverio Fiorillo – All rights reserved

Many endeavors require scheduling meetings at various locations all across a city or a region (e.g., around Lombardy), whether for work or personal reasons (e.g., meeting the CEO of a partner company, going to the gym, taking children to practice, etc.). The goal of this project is to create a calendar-based application that: (i) automatically computes and accounts for travel time between appointments to make sure that the user is not late for appointments; and (ii) supports the user in his/her travels, for example by identifying the best mobility option (e.g., use the train from A to B and then the metro to C), buying public transportation tickets, or by locating the nearest bike of a bike sharing system. Users can create meetings, and when meetings are created at locations that are unreachable in the allotted time, a warning is created. As mentioned, the application should also suggest travel means depending on the appointment (e.g., perhaps you bike to the office in the morning, but the bus is a better choice between a pair of afternoon meetings, and a car – either personal, or of a car-sharing system – is best to take children to practice) and the day (e.g., the app should suggest that you leave your home via car in the morning because meetings during a strike day will not be doable via public transportation; it could also take into account the weather forecast, and avoid biking during rainy days). Travlendar+ should allow users to define various kinds of user preferences. It should support a multitude of travel means, including walking, biking (own or shared), public transportation (including taxis), driving (own or shared), etc. A particular user may globally activate or deactivate each travel means (e.g., a user who cannot drive would deactivate driving). A user should also be able to provide reasonable constraints on different travel means (e.g., walking distances should be less than a given distance, or public transportation should not be used after a given time of day). Users should also be able, if they wish to, to select combinations of transportation means that minimize carbon footprint. Additional features could also be envisioned, for instance allowing a user to specify a flexible "lunch". For instance, a user could be able to specify that lunch must be possible every day between 11:30- 2:30, and it must be at least half an hour long, but the specific timing is flexible. The app would then be sure to reserve at least 30 minutes for lunch each day. Similarly, other types of breaks might be scheduled in a customizable way

Contents

Table of Contents **4**

List of Figures **5**

List of Tables **5**

1 Introduction **6**
1.1 Purpose 6
1.2 Scope 6
1.2.1 System description 6
1.2.2 World Phenomena 6
1.2.3 Goals 6
1.3 Definitions, Acronyms, Abbreviations 7
1.3.1 Definitions 7
1.3.2 Acronyms 7
1.3.3 Abbreviations 7
1.4 Revision history 7
1.5 References 7
1.6 Document Structure 7

2 Overall Description **8**
2.1 Product perspective 8
2.1.1 User interfaces 8
2.1.2 System interfaces 8
2.1.3 Software interfaces 9
2.2 Product functions 10
2.2.1 Preference profile creation 10
2.2.2 Appointment insertion 10
2.2.3 Appointment modification 10
2.2.4 Change travel route 10
2.2.5 Route choice 10
2.2.6 User Registration 10
2.2.7 User login 10
2.3 User characteristics 10
2.4 Assumptions 11

3 Specific Requirements **13**

4 Formal Analysis Using Alloy **15**
4.1 Signatures 15
4.2 Facts 18
4.3 Predicates 20
4.4 Results 21
4.5 World 21

5 Effort Spent **23**

References **24**

List of Figures

1 UML class diagram perspective 12
2 Alloy signatures 1/2 15
3 Alloy signatures 2/2 16
4 Alloy facts 1/2 18
5 Alloy facts 2/2 19
6 Alloy predicates 20
7 Alloy consistency results 21
8 Alloy generated world 22

List of Tables

1 Introduction

1.1 Purpose

The following document is the Requirement Analysis and Specification Document (RASD) for the formalization and description of all the needed features, constraints and recommendations that will constitute the proposed system.

The paper will focus on the elicitation and analysis of all functional and nonfunctional requirements, also verified by automated logic verification software (Alloy Analyzer [MIT]) and supported by UML schemes and scenarios. The provided document will also include a concise analysis of the environment and it tries to clarify all the interaction among system parts and external world.

The target audiences of this document are all the stakeholders involved in the development of the system and it can be used as contractual base for the project.

1.2 Scope

1.2.1 System description

The system that would be developed is a calendar-based application that provides support for planning of appointments, automating the travel arrangement process according to the user preferences. The system will be able to plan travel routes choosing transport options among public and private means, including trains, trams, taxis, bicycles, bike and car sharing, owned automobile and others. Therefore, the system will provide a set of possible route options, trying to minimize travel times, route length, carbon footprint or number of changes.

Also, it will have to recollect travel informations on public transportations and travelling conditions from different external sources. Beside, the application will give the possibility to users to register themselves, allowing them to access a series of extra features, including the possibility of setting personal preferences and backup personal agenda and routes.

User preferences include the possibility to set travel pauses, flexible launch (30 minute of travel pause between 11.30-14.30), enabling and disabling certains types of transportation means in a specific period of the day or anytime and in addition it will give the possibility to user to create profiles of preferences and link them to single appointments) Finally, system should be capable to interacting with any type of external public transportation service, although it will focus on make available them in the city of Milan.

1.2.2 World Phenomena

Many public transport service companies offers the possibility to obtains informations on timetables, routes, stations and status of services in aggregate way providing public Application Public Interfaces (APIs). Thus it is guaranteed the theoretical possibility to make queries in any moment and get the latest valid informations on transports.

Others doesn't offer the same possibilities; for instance Trenitalia hasn't yet a public API, but there are different opensource projects that allow to fill the gap.

Moreover, a lot of web mapping services makes available public APIs too. In this case it is possible to get GPS location from an address or vice-versa, or update a portion of map and even process a route given a starting point and a destination. All the required supported companies provide aforementioned services.

1.2.3 Goals

To be retrive from google docs

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- API(s) : Application Program Interface(s)
- RASD : Requirement Analysis Specification Document
- Appointment: location the user has to reach at a fixed date and time
- Route: journey between two appointments, it is composed by a series of paths
- Path: part of a route traversed with a specific mean of transport

1.3.2 Acronyms

- UI: User Interface

1.3.3 Abbreviations

- MA (Mobile Application): part of the system
- PT: Public Transportation

1.4 Revision history

1.5 References

- Specification Document assigned: `"Mandatory_Project_Assignments.pdf"`
- Software Requirements Specification Guidelines : IEEE 830-1998
- Alloy Website: http://alloy.mit.edu/alloy/

1.6 Document Structure

The document is composed of six different parts.

1. The introduction has the objective to support the reader into the reading of the document providing a brief description of the problem and the actual technological landscape description. It also include a list of definition, abbreviations and acronyms commonly employed through the rest of the document.
2. The second part offers an overall description of the system, then focused on the boundaries that divide the system from world, especially inspecting all interactions on shared phenomena. In addition, it is provided a short description of users. FInally, it is listed and described a set of core functionalities that will be supported and assumptions.
3. The third part provides a more complex and detailed description of functionalities, also in terms of requirement. This part is addressed specifically to development team
4. This section provides an Alloy model for verification of goals through a formal analysis of requirements and domain assumption. This is beneficial in assuring the consistency of the model. It also provide a representation of the world and system.
5. The fifth part provides the effort spent in term of time involved into the project by the authors. In the last part, there are references to external sources.

2 Overall Description

2.1 Product perspective

2.1.1 User interfaces

- A new user should be able to make an account, without training, in less than 10 minutes
- The number of steps requested to user in "appointment creation" should be less than 5
- User should be allowed to access system functions from mobile devices such as smart-phone and tablets
- User should be allowed to access system functionalities from PC
- UI for mobile devices should adapt to different screen sizes (from 5 to 11 inch)
- There shouldn't be need of training for new user to learn all application functionalities offered by UI

2.1.2 System interfaces

System have to communicate with external services, from public transportation means to map services. In order to guarantee future support to new services, also outside Milan area, it ought to be useful a modular, flexible but also extendible approach on the building of external interface. In the following part of the paragraph it will be shown a list of logic operation that should be accomplished by external services APIs in order to be interfaced and employed by the system.

- Train services and trams

  Looking for station list

  System request: station list

  Service response: list of station with locations

  Looking for train/tram

  System request: departure and arrival stations, departure time

  list of possible rides, cost of travel, times of leaving and arrival (at least 1 list's element)

- Taxi

  Looking for taxi

  System request: departure position and time

  Service response: disponibility and (hourly or journey cost)

- Bike sharing

  Looking for service parking spot

  System request: spaces

  Service response: list of spaces with available bicycles

  Looking for fees

  System request: fees

  Service response: hourly fee

- Map services

  System should be able to retrieve the nearest address given GPS coordinates and vice-versa by these services.

  Map services should be able to generate a route between two given address on system request.

  System should be allowed to download a portion of map.

- Weather forecast service

  Request forecast

  System request: forecast for a specific location and time

  Service response: weather conditions and temperature for requested locality and time

2.1.3 Software interfaces

In order to ship a complete functional software, at least for the city of Milan, a list of required supported external APIs is provided in this paragraph.

- Trenitalia

  Name: “Informazioni-Treni-Italiani”

  Description: unofficial API for Trenitalia, further details on source

  Version: testing

  Source: https://github.com/Razorphyn/Informazioni-Treni-Italiani

- ATM

  Name: “localizzazione delle fermate delle linee urbane di superficie”

  Description: list of metro stops and hours in data-sheet version, regularly updated

  Version: 29/12/2015

  Source: http://dati.comune.milano.it/

- Bike Sharing

  Name: “Mobilità: localizzazione delle rastrelliere per il bike sharing”

  Description: list of bikesharing areas in datasheet version, regularly updated

  Version: 05/12/2016

  Source: http://dati.comune.milano.it/

- Taxi

  Name: “appTaxi | Developers”

  Description: restricted API to book or calculate prices for a ride

  Version: rolling

  Source: www.apptaxi.it/developers/

- Weather

  Name: “Open Weather Map API”

  Description: open API to get weather conditions for a location

  Version: rolling

  Source: https://openweathermap.org/api

2.2 Product functions

It has been selected a set of functionalities, on one hand in order to provide to user a service that focus on easy of use, automation and simplification of user processes and interactions with the system, on the other hand allowing him to freely compose complex schemes of constraints in the choice of travel routes.

2.2.1 Preference profile creation

A logged user should be able to create some types of constraints (flexible launch, disabilitation of transport means, set pauses from transport and maximum travelling distance by types of transport mean). He should also be allowed to create many different preferences profiles, that are logical containers for sets of user preferences. These profiles can be applied to single appointments personalizing the research of a routes. Preferences with unspecified profiles belongs to "default" profile and they will ever be considered as constraint for route generation.

2.2.2 Appointment insertion

A user can register an appointment providing a description, a date, a time of begin and end. Beside, a registered user can optionally add a list of preferences profiles. The appointment shouldn't conflict with others of the same user, so if it happens, it will be rejected. Accepted appointment have to be processed, and route options prompted to user.

2.2.3 Appointment modification

A user can modify any detail of a previous inserted appointment. If the user is logged into the system, all modifications have to be synchronized with user account. Finally, system should provides new routes for reaching the modified appointment, deleting previous saved routed linked with the old one.

2.2.4 Change travel route

A user can choose to change a saved route. In this case, system will delete user current selected saved route and it will provides new routes to user.

2.2.5 Route choice

When requested, the system have to generate routes to reach appointment according to registered user preferences profiles and the previous appointment. System will provide a route that will minimize distance, another one minimizing duration and others minimizing carbon footprint or number of changes.

2.2.6 User Registration

User can choose to register an account into the system through MA indicating a valid email address, a secure password, name, surname and fiscal code.

2.2.7 User login

A registered user can log in using phone number and sms verification process or by email and password.

2.3 User characteristics

It's possible to divide application users into two groups:

- Unregistered

  He can access some app features, including appointment registration and route choices

- Registered

  He can access all the feature set that system can provide, so all the functionality provided to unregistered user, plus the possibility to set profiles of preferences, synchronization and backup of user data and preferences. This user can also receive weather forecast update.

The user of this system is not required to have IT skills.

2.4 Assumptions

D1 The timetables of the public transport means are retrieved by external API and are reliable.

D2 The route between two points is calculated by external map and routing services, they provide correct information.

D3 The GPS gives the correct position with an accuracy of few meters.

D4 The registered users are recognized by their email address, which is unique.

D5 If the system needs to send an SMS to a user, he will receive it.

D6 A route between two locations is composed by some paths, each of them is travelled by a specific mean of transport.

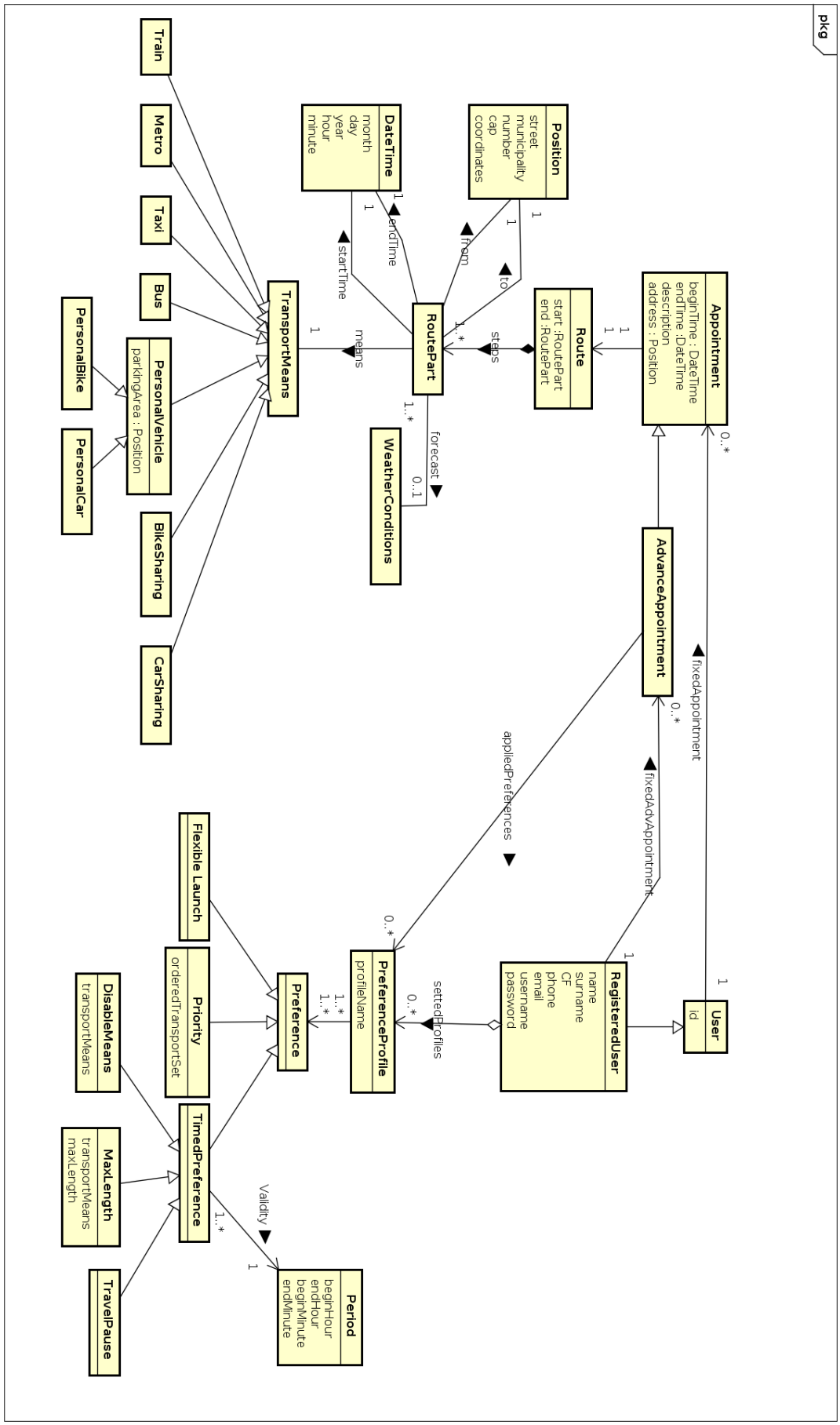

Figure 1: UML class diagram perspective

3 Specific Requirements

Organize this section according to the rules defined in the project description.

4 Formal Analysis Using Alloy

4.1 Signatures

```
//Signatures

sig User {
   id : Int,
   fixedAppointments : set Appointment
}

sig RegisteredUser extends User{
   username : one Username,
   password : one Password,
   email : one Email,
   name: one Name,
   surname : one Surname,
   cf : one CF,
   fixedAdvAppointments : set AdvancedAppointment,
   preferenceProfiles : set PreferenceProfile
}

sig Appointment{
   beginTime : one DateTime,
   endTime : one DateTime,
   description : one string,
   address : Position,
   route :  set Route
}

sig AdvancedAppointment extends Appointment{
   appliedPreferences : set PreferenceProfile,
}

sig Route {
   steps : some RoutePart,
   start : one RoutePart,
   end : one RoutePart
}

sig RoutePart {
   from : Position,
   to : Position,
   startTime : DateTime,
   endTime : DateTime,
   forecast : lone WeatherCondition,
   travelMeans : TravelMeans
}

sig Position {
   street : string,
   municipality : string,
   number : Int,
   cap : CAP,
   coordinates : Coord
}

sig PreferenceProfile{
   preferences : set Preference,
   name : string
}

one sig DefaultProfile extends PreferenceProfile {}
```

Figure 2: Alloy signatures 1/2

```
//Other Sig
sig Username{}
sig Password{}
sig Email{}
sig Name{}
sig Surname{}
sig CF {}
sig CAP {}
sig string{}
sig Coord{}
sig WeatherCondition{}
sig Preference {}
sig TravelMeans {}


sig DateTime{
  month : Int,
  day : Int,
  year : Int,
  hour : Int,
  minute : Int
}
```

Figure 3: Alloy signatures 2/2

4.2 Facts

```
//All normal appointment are linked to normal users, AdvancedAppointments to RegisteredUsers
fact AppointmentAndAdvancedAppointment{
   all a: Appointment, ad: AdvancedAppointment | ((a in (Appointment - AdvancedAppointment)) implies a in U
      && ad in RegisteredUser.fixedAdvAppointments
}

//Username, Email, CF for RegisteredUsers are keys
fact uniqueIdUsernameEmailCF{
   all u1, u2 : RegisteredUser | u1 != u2 implies
      (u1.id != u2.id && u1.username != u2.username && u1.cf != u2.cf && u1.email != u2.email)
}

//There aren't Usernames, Passwords, Email, Name, Surname, CF not linked to anyone
fact noOrphanUsernamePasswordEmailNameSurnameCF{
   all us : Username, c: CF, e : Email, p:Password, n:Name, s:Surname |
      us in RegisteredUser.username && c in RegisteredUser.cf && e in RegisteredUser.email
      && p in RegisteredUser.password && n in RegisteredUser.name && s in RegisteredUser.surname
}

//Constrains on Calendar Data
fact DateTimeCalendarConstrains{
   all dt : DateTime | dt.month <= Int[12] && dt.day <= 31 && dt.hour < 24 && dt.minute <60
      && dt.month > 0 && dt.day > 0 && dt.hour >= 0 && dt.minute >= 0
      && dt.year = 0 //Added to reduce scope and complexity
}

//Coords and WeatherCondition always linked to Position and RoutePart
fact noOrphanWeatherConditionCoord{
   all w: WeatherCondition, c : Coord | c in Position.coordinates && w in RoutePart.forecast
}

//Preferences are always linked to a PreferenceProfile as well as that is always linked to a RegisteredUser
fact PreferenceProfileConstraints{
   all p : Preference | p in PreferenceProfile.preferences
   all pp : PreferenceProfile | pp in RegisteredUser.preferenceProfiles
}

//WeatherCondition are guaranteed to RegisteredUser
fact WeatherConditionForRegisteredUser{
   all adst: AdvancedAppointment.route.(steps+start+end) | adst.forecast != none
}

//Coord are keys for Position
fact uniqueCoords{
   all p1, p2 : Position | p1 != p2 implies p1.coordinates != p2.coordinates
}

//RoutePart are not of null length or equal segments
fact lengthNotNull{
   all rp : RoutePart | rp.from != rp.to
   no disj rp, rp1 : RoutePart | rp.from = rp1.from and rp.to = rp1.to
}

//intemediate steps aren't start or end ones
fact {
   all r : Route | (no s : r.steps | r.start = s or r.end = s)
}
```

Figure 4: Alloy facts 1/2

```
//Route Parts are linked to form a Route
fact {
  all r : Route | (
    (lone s : r.steps | r.start.to = s.from)
    && (lone s : r.steps | r.end.from = s.to)
    && (no s : r.steps | r.start.from = s.to)
    && (no s : r.steps | r.end.to = s.from)
    && (all s : r.steps | s.to = r.end.from or one s1: r.steps | s.to = s1.from)
    && (all s : r.steps | s.from = r.start.to or one s1 : r.steps | s.from = s1.to)
    && (#r.steps = 0 implies (r.start = r.end or r.start.to= r.end.from) else r.start != r.end)
  )
}
```

Figure 5: Alloy facts 2/2

4.3 Predicates

```
//A new preference is added to DefaultProfile
pred createPreference(p : Preference, dp : DefaultProfile, dp' : DefaultProfile) {
  //prec
  !(p in dp.preferences)
  //postc
  all op : dp.preferences | op in dp'.preferences && p in dp'.preferences && #op = (#dp'.preferences -1)
}

//Add a preference to a profile (This implies the removal from DefaultProfile if present)
pred addPreference(p : Preference, dp : DefaultProfile, dp' : DefaultProfile, pp : PreferenceProfile, pp' : Prefer
  //prec
  !(p in pp.preferences)
  //postc
  all op : dp.preferences | op != p implies op in dp'.preferences else !(op in dp'.preferences)
  all np : pp.preferences | np in pp'.preferences && p in np
}

//Add preferenceProfile to AdvancedAppointment
pred addPreferenceProfileToAppointment(pp : PreferenceProfile, aa : AdvancedAppointment, aa' : AdvancedAp
  //prec
  !(pp in aa.appliedPreferences)
  //postc
  all pr : aa.appliedPreferences | pr in aa'.appliedPreferences &&  pp in aa'.appliedPreferences
}

//Add an Appointment to User (AdvancedAppointment <> RegisteredUser)
pred addAppointment(a : Appointment, u : User, u' : User) {
  //prec
  a in AdvancedAppointment iff u in RegisteredUser
  !( a in u.fixedAdvAppointments+u.fixedAdvAppointments)
  //postc
  a in AdvancedAppointment implies (all ad : u.fixedAdvAppointments | ad in u.fixedAdvAppointments && a in
    else (all ab : u.fixedAppointments | ab in u.fixedAppointments && a in u.fixedAppointments)
}

//Remove an Appointment (AdvancedAppointment <> RegisteredUser)
pred removeAppointment(a : Appointment, u : User, u' : User) {
  //prec
  a in AdvancedAppointment iff u in RegisteredUser
  a in u.fixedAdvAppointments+u.fixedAdvAppointments
  //postc
  a in AdvancedAppointment implies
    (all ad : u.fixedAdvAppointments | ad != a implies ad in u'.fixedAdvAppointments else !(ad in u'.fixedAdvA
      else (all ba : u.fixedAppointments | ba != a implies ba in u'.fixedAppointments else !(ba in u'.fixedApp
}
```

Figure 6: Alloy predicates

4.4 Results

```
Executing "Run addRoutePart for 5 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  186838 vars. 9829 primary vars. 504389 clauses. 292ms.
  Instance found. Predicate is consistent. 1924ms.

Executing "Run createPreference for 5 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  186605 vars. 9821 primary vars. 503581 clauses. 378ms.
  Instance found. Predicate is consistent. 2392ms.

Executing "Run addPreference for 5 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  186724 vars. 9831 primary vars. 503888 clauses. 386ms.
  Instance found. Predicate is consistent. 914ms.

Executing "Run addPreferenceProfileToAppointment for 5 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  186691 vars. 9829 primary vars. 503813 clauses. 340ms.
  Instance found. Predicate is consistent. 1181ms.

Executing "Run addAppointment for 5 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  186724 vars. 9829 primary vars. 504008 clauses. 373ms.
  Instance found. Predicate is consistent. 922ms.

Executing "Run removeAppointment for 5 but 8 int"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  186808 vars. 9829 primary vars. 504202 clauses. 852ms.
  Instance found. Predicate is consistent. 1089ms.

6 commands were executed. The results are:
  #1: Instance found. addRoutePart is consistent.
  #2: Instance found. createPreference is consistent.
  #3: Instance found. addPreference is consistent.
  #4: Instance found. addPreferenceProfileToAppointment is consistent.
  #5: Instance found. addAppointment is consistent.
  #6: Instance found. removeAppointment is consistent.
```

Figure 7: Alloy consistency results

4.5 World

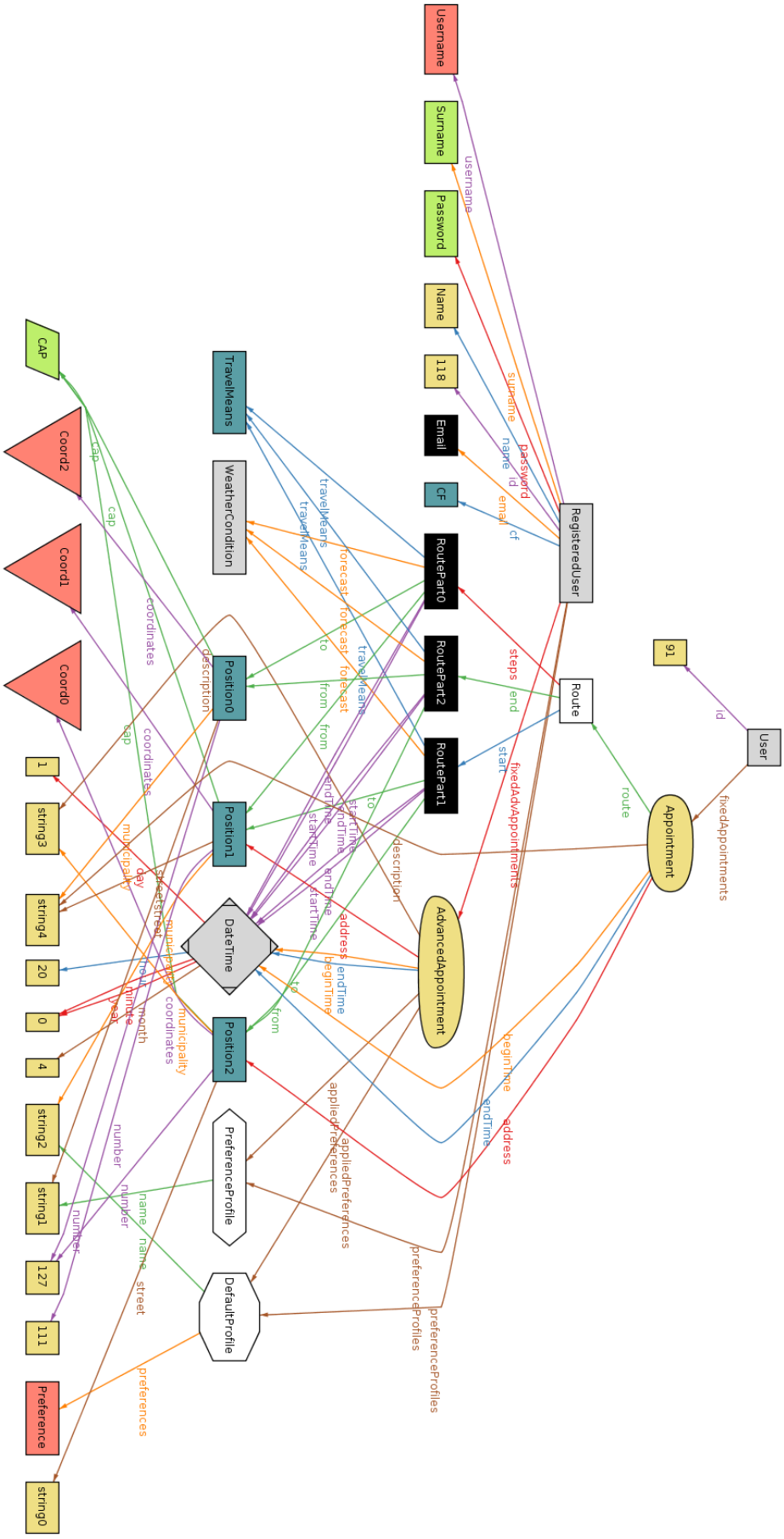

Figure 8: Alloy generated world

5 Effort Spent

Provide here information about how much effort each group member spent in working at this document. We would appreciate details here.

References

[1] S. Bernardi, J. Merseguer, and D. C. Petriu. A dependability profile within MARTE. *Software and Systems Modeling*, 10(3):313–336, 2011.

