Abdul Hafeez CS14B031 Harshith Reddy CS14B011
P Sai Mohith CS14B047 Sachin Sridhar CS14B057
Y Deekshit Reddy CS12B030

February 4, 2017

Real Office Z Schema

Here we model an Event Scheduler system for CSE Department Office in Z. Scheduling a number of events in multiple venues on a daily basis is painstaking task for anyone.The purpose of this system is make event scheduling , event availability automated for the CSE Department Office.
Here is an excerpt from the informal description:

> Suppose the office staff would like to schedule an event they give the required parameters as input. The system then checks for availability of different resources for the given parameters and allocates resources. All information about an event is also stored for future reference in a database.

We begin the Z model by introducing four basic sets that hold everything of interest in this universe, namely events, venues, times, strings(errors,reports and output notifications from a query):

$$[\mathbf{EVENTS}, \mathbf{VENUE}, \mathbf{TIME}, \mathbf{CAPACITY}, \mathbf{STRING}]$$

The set of state variables are as follows:

> **eventname**: set of all event names which are recorded in the database
> **eventvenue**: a function which when applied to certain meetings gives back the venues associated with the respective events

evnentime: a function which when applied to certain events gives back the time associated with the respective event
venueproj: a function which when applied to certain venues gives back whether a projector is present or not present
venuecap: a function which when applied to certain venues gives back the capacity of a particular
name: name is an input variable which takes the event name from user.
venue: venue is an input variable which takes the venue required for an event from user.
time: time is an input variable which takes the time required for an event from user.
capacity: input variable which is specified by the user
projector: input variable where user specifies projector requirement.
refreshment_req: input variable where user specifies refreshment requirements.
stationary: input variable where user specifies stationary requirements.
error: output variable which contains the error messages for various input parameters(eventname,venue,time,projector requirement).
report: output variable returned when an event is cancelled.
info: output variable which contains the info of a particular event when queried for.

The following schema is a state space schema defining all the static variables and functions associated with them.

REALOFFICE

$$\begin{aligned}
&\mathbf{eventname} : \mathbf{EVENTS} \\
&\mathbf{eventvenue} : \mathbf{EVENTS} \pfun \mathbf{VENUE} \\
&\mathbf{eventtime} : \mathbf{EVENTS} \pfun \mathbf{TIME} \\
&\mathbf{venueproj} : \mathbf{VENUE} \pfun \{\mathbf{present}, \mathbf{notpresent}\} \\
&\mathbf{venuecap} : \mathbf{VENUE} \pfun \mathbf{CAPACITY} \\
\hline
&\mathbf{eventname} = \mathrm{dom}\, \mathbf{eventvenue} \\
&\mathbf{eventname} = \mathrm{dom}\, \mathbf{eventtime} \\
&\mathbf{eventvenue} = \mathrm{dom}\, \mathbf{venueproj} \\
&\mathbf{eventvenue} = \mathrm{dom}\, \mathbf{venuecap}
\end{aligned}$$

We then define various function schema for atomic operations like adding an event, cancelling an event, and notifications. The modify event operation can be achieved by calling CANCELEVENT and then ADDEVENT with the new parameters.

The Add Event schema takes inputs name, venue, time, projector requirement, refreshment requirement, stationary requirements .It does checks on the validity of these input arguments. If the arguments are valid, it updates the list of events.

ADDEVENT

Δ**REALOFFICE**
name? : **EVENTS**
venue? : **VENUE**
time? : **TIME**
projector? : {**required**, **notrequired**}
capacity : **CAPACITY**
refreshment_req? : {**present**, **notpresent**}
stationary_req? : {**present**, **notpresent**}
error? : **STRING**

(**name**? $\notin$ **eventname** $\wedge$
venue? $\in$ dom **eventvenue** $\wedge$
$\phi = \{\mathbf{x} : \mathbf{EVENTS} \mid \mathbf{eventvenue}(\mathbf{x}) = \mathbf{venue}? \wedge \mathbf{eventtime}(\mathbf{x}) = \mathbf{time}?\} \wedge$
capacity <= **venuecap**(**venue**) $\wedge$
eventname$'$ = **eventname** $\bigcup \{\mathbf{name}? \pfun \mathbf{venue}?\} \bigcup \{\mathbf{name}? \pfun \mathbf{time}?\}$)
$\vee$
(**name**? = **eventname** $\wedge$
error? = **EventAlreadyExists**)
$\vee$
(**venue**? $\notin$ dom **eventvenue** $\wedge$
error? = **VenueDoesNotExist**)
$\vee$
($\{\mathbf{x} : \mathbf{EVENTS} \mid \mathbf{eventvenue}(\mathbf{x}) = \mathbf{venue}? \wedge$
eventtime(**x**) = **time**?$\} \neq \phi \wedge$
error? = **EventClashesWithAnotherEvent**)
$\vee$
(**projector**? = **required** $\wedge$
venueproj(**venue**?) = **notpresent** $\wedge$
error? = **ProjectorNotPresent**)
$\vee$
(**capacity** <= **venuecap**(**venue**) $\wedge$
error? = **VenueCapacityInsufficient**)

The Cancel Event schema checks if the input event name exists and removes it if it does.

CANCELEVENT

Δ**REALOFFICE**
name? : **EVENTS**
report! : **STRING**

(**name**? $\notin$ **eventname** $\wedge$
report! = **EventDoesNotExist**)
$\vee$
(**name**? $\in$ **eventname** $\wedge$
eventname$'$ = **eventname** $\setminus$ {**event**} $\wedge$
report! = **EventCancelled**)

The Find Event schema does a query for a given input event name and returns the info associated with it ,gives a report if event does not exist.

FINDEVENT

Ξ**REALOFFICE**
name? : **EVENTS**
info! : **STRING**

(**name**? $\notin$ **eventname** $\wedge$
info! = "**Name** : " + **name**+
"**Venue** : " + **eventvenue**(**name**)+
"**Time** : " + **eventtime**(**name**)+
"**ProjectorAtVenue** : " + **venueproj**(**name**))
$\vee$
(**name**? $\in$ **eventname** $\wedge$
report! = **EventDoesNotExist**)

The Notify schema notifies the user if there an event happening at time *duration* from now. The value of the variable *duration* can be statically set.

NOTIFY

Ξ**REALOFFICE**
time? : **TIME**
events! : **P**(**EVENTS**)

---

events! = {**e** : **EVENTS** | **eventtime**(**e**) = **time**? + **duration**}