A formal Domain Model for Restaurant Reservations

Oliver Reinhard

March 17, 2020

Abstract

This essay explores a common everyday situation as a short case study by ways of a formal domain model.

The situation is a party of people entering a restaurant and asking for a free table.

The analysis reveals two main entities, two roles and two domain events and establishes their properties and relationships in a precise and consistent manner.

1 Situation

A party of n people enter a restaurant at which they would like to eat but don't have a table reservation.

1.1 Business Domain

The business domain this scenario is *restaurant management*.

1.2 Party

One of the two central entities in this scenario is the party of people, short party. In this context, party is a virtual concept referring to a number of people who have a common purpose like travelling or a some social gathering. A party exists only as long as there are members sharing a purpose. There is no physical evidence of the party as such.

1.3 Table

The other central entity in this scenario is the table. Tables are physical objects that have a life in the physical world. Since this domain model is not about manufacturing a table, it only captures the existence and the properties of a table but it is not concerne with, for example, creating a table. The tables of a restaurant simply exist, however, the domain model is concerned with their presence, their size, and maybe their locations, e.g. at the window, on the terrace, by the bar, etc.

1.4 Party enters Restaurant

A Part enters a restaurant and steps up to the signpost reading "Wait to be seated".

1.5 Information Model

```
1 initial state ARRIVED
2         state WAITING
3         state SEATED
4         final state LEFT
```

Listing 1: Some Code

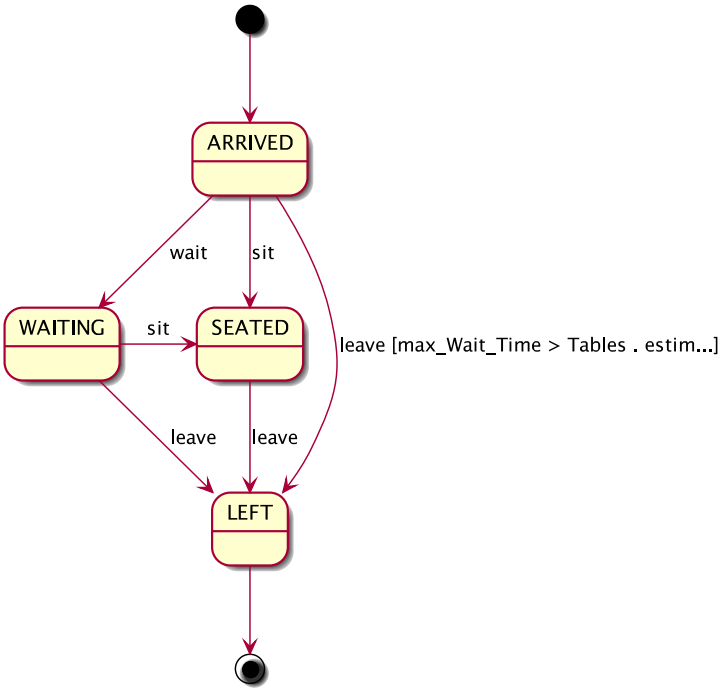

Figure 1: Party Life Cycle

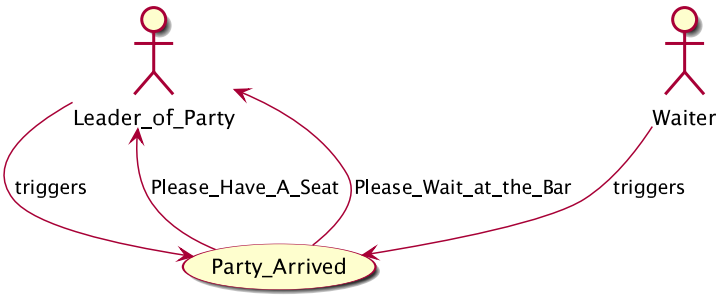

Figure 2: Use Case: Party arrived

```
case Party_At_Table
when Tables.has_Free_Table(p.size):

  Party_Seated:
    p.state = Party.SEATED

  Party_Assigned_To_Table:
    Table.all.exists(t | t.seats ≥ p.size AND t.party = p AND t.state = Table.USED)

  Message_Delivered:
    Please_Have_A_Seat.delivered
```

Listing 2: Some included Code

Here's an example on non-titled code:

```
PAID
```

No title here!

```
case Party_At_Table
when Tables.has_Free_Table(p.size):

  Party_Seated:
```

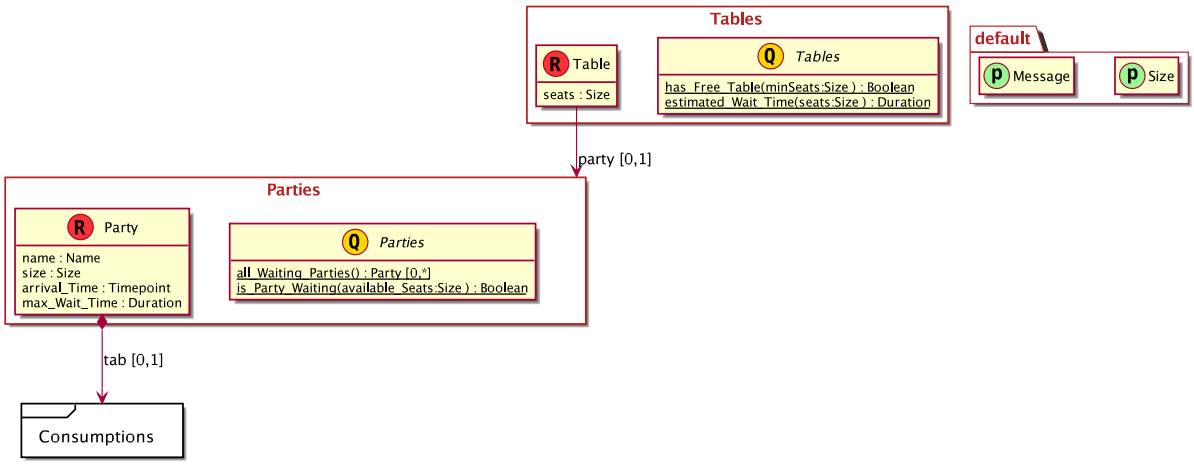

Figure 3: Reservations Information Model

```
  p.state = Party.SEATED

Party_Assigned_To_Table:
  Table.all.exists(t | t.seats ≥ p.size AND t.party = p AND t.state = Table.USED)

Message_Delivered:
  Please_Have_A_Seat.delivered
```

Listing 3: More included Code