26/08/2020

TEA-ROOM COVID-19: REQUIREMENT ANALYSIS

1. SETTING

The **teaRoom** is described as a rectangular room composed of:

A. **N teaTable**: tables placed inside the tearoom, where the admitted **Client** can consume his tea.

B. **serviceArea**, composed of:

   i. **serviceDesk**: where the entity <u>barman</u> prepares the tea after receiving a request by the <u>waiter</u>;

   ii. **home**: where the <u>waiter</u> can rest if it has no tasks to do.

C. **hall**: where, when he arrives, a **Client** has to wait here before entering the teaRoom (client behaviour explained in point 2). It is equipped with:

   i. **a presenceDetector,** which can detect the presence of a person or other entity;

   ii. **a smartbell,** which measures the temperature of the <u>Client</u> that wants to enter the tearoom and, if the Client's temperature is less than 37.5°, ***sends a request message*** *to the <u>waiter</u>*.

      *1. **clientIdentifier:** value to univocally represent a client's request of entering the tearoom. It is assigned by the smartbell to the client and it is given to the waiter with the aforementioned request.*

   iii. an **entranceDoor**, through which the Client will be admitted inside the tearoom;

   iv. an **exitDoor**, through which the Client can leave the tearoom.

A **tearoom is considered safe** if there are no people inside with a temperature greater or equal than 37.5° and if there are clean tea-tables posed at a proper distance.

2. CLIENT

A. **notifiy interest in entering the tearoom**: once in the **hall**, the Client ***has to send a notification*** to the **smartbell**, whose behavior has been described in point [1.A.iii.2].

B. ***maxWaitTime***: time value that will be given by the **waiter** if there are <u>no free and clean tables</u> available and he's not been sent away because of the temperature, after which either he has entered, or he has to leave.

C. ***maxStayTime***: maximum time that the Client can spend at a teaTable. After it expires, the client has to leave, no matter if he's finished the tea or not.

3. WAITER

The **waiter tasks**, listed in the requirements, are a set of actions the waiter should be able to perform, one at the time, *optimizing as much as possible the execution so that the waiting time of the requests coming from each client is minimized*.

A. **Convoy the Client**: once a Client is free to enter the tearoom or ready to leave it, the waiter has to accompany them to and from the table, from and to the entranceDoor and exitDoor, respectively.

B. **Clean the (tea)table**: once the client leaves the table, the waiter has to clean it before another client can occupy it.

The waiter is also a ***Robot***. This means that there needs to be a system to interact with the robot hardware to make it move around the room.

4. BARMAN

A. ***receive* order from waiter**: the orders are *transmitted through a WIFI* device by the waiter to the barman.

B. ***notify* waiter of drink ready**: the barman has to send a notification to the waiter once the drink order by a Client is ready. The drink should be prepared in a time that is significantly smaller than the client's maxStayTime.

5. ENTITIES AS ACTORS

What we can infer from the requirements is that:

A. we have identified different entities that will come into play in the system (barman, waiter, client, smartbell);

B. all of these entities have a behavior that can be represented as a **Finite State Machine** while also have to keep a **readable state** for the manager;

C. they will need to interact with each other through different kinds of messages.

These three points have highlighted the need for a model representation through the concept of *Actor as a Finite State Machine*. To do so, we introduce the **QAktor meta-modeling language**, referenced here, so as to close the abstraction gap as much as possible from the beginning and also develop working prototypes fast and easily.

6. DEFINING THE STATES

A. During this first analysis, we have identified the main states the various actors can be in:

  i. **Waiter** can be answering the client's request to enter, deploying the client, cleaning the table, moving or waiting for a new task to execute;
  ii. **Barman** can be preparing the beverage or waiting for a new order;
  iii. **Client** can be Outside, waitingInTheHall, seated, finished, paid and left;
  iv. **SmartBell** can be evaluating a client or free.

B. Even though teaTable is not an actor, we need to keep its state memorized for the system to work properly and for the current state of the teaRoom to be complete. The states the teaTable can be in are:

  i. ***tableClean***: without food residue and sanitized and available;
  ii. ***tableDirty***: not clean but available;
  iii. ***tableBusy***: occupied by a client.

7. MESSAGES

As we have already explained in [5.C], the actors need messages to interact. As visible in the next paragraph, we use different kinds of messages depending on the exchange required. Here is a summary of the messages used:

A. When the client arrives, she sends a request ringBell, specifying her temperature in the `TEMP` variable, to the Smartbell and waits for the reply tempStatus, which contains a variable `STATUS`, that is 0 if the temperature is too high to enter the tearoom or 1 if it's ok. The variable `ClientID` contains the *clientidentifier* to identify the client.

```
Request ringBell : ringBell(TEMP)

Reply tempStatus : tempStatus(STATUS, CLIENTID)
```

B. If the Client's temperature is fine, the Smartbell emits an event to notify that there is a client waiting to be admitted.

```
Event clientID : clientID (CLIENTID)
```

C. Once the Waiter handles the previous event, it sends a Dispatch message to the client to let them know how long they have to wait through the variable `MAXWAITTIME`. If 0, the waiter is moving to take them to the table right away.

```
Dispatch admission : admission(MAXWAITTIME, CLIENTID)
```

D. When the Waiter arrives at the entrance, he sends an Event so that the Client knows he's there. This is when the deployment to the table begins.

```
Event waiterAtEntrance : waiterAtEntrance(OK)
```

E. Once the Client is ready to order the tea, she emits an event and waits for the waiter to come at the table.

```
Event readyToOrder: readyToOrder(TABLE, CLIENTID)
```

F. As soon as the Waiter reaches the table, it requests the order to the Client, which Replies with what kind of tea she wants

```
Request getOrder : getOrder(TABLE, CLIENTID)
Reply  order : order(TEA)
```

G. When the Client replies with the desired tea, the Waiter sends a Dispatch message to the Barman.

```
Dispatch sendOrder : sendOrder(TEA, TABLE)
```

H. When the Barman has done preparing the tea, he emits an Event to notify it

```
Event orderReady : orderReady(TEA, TABLE)
```

I. When the Tea is ready the waiter notifies the Client and brings it to him.

```
Dispatch deliver : deliver (TEA)
```

J. When the Client is ready to pay, he emits an Event to notify it

```
Event readyToPay : readyToPay(TABLE, CLIENTID)
```

K. As soon as he receives the event notifying that the Client's ready to pay, he goes to the table, sends a Request for the money owed in the variable `MONEY` to the Client and it answers with a Reply to complete it.

```
Request pay : pay(MONEY, CLIENTID)
Reply  paid : paid(MONEY)
```

L. When the Client has done the payment, the Waiter deploys her to the ExitDoor. To notify the Client that they have arrived a Dispatch message is sent.

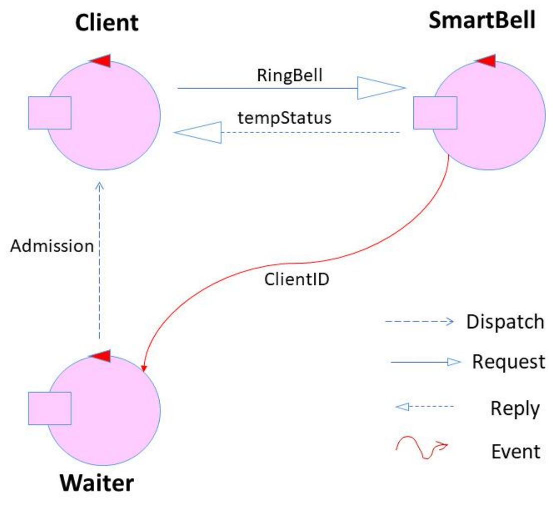
```
Dispatch exit : exit(OK)
```

M. Once the Client has paid, the table becomes dirty and an "auto-Event" is raised for the Waiter, so that he can clean it. The number of the table dirty is in the payload variable `N`.

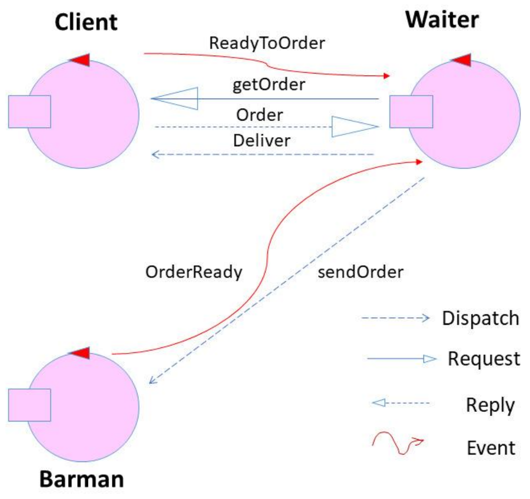
```
Event tableDirty : tableDirty(N)
```

8. INTERACTION DIAGRAMS

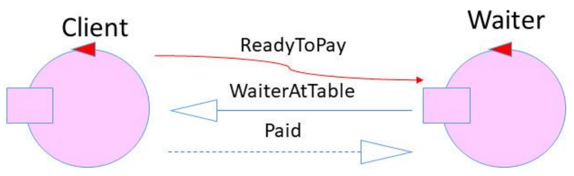
A. Client arrival



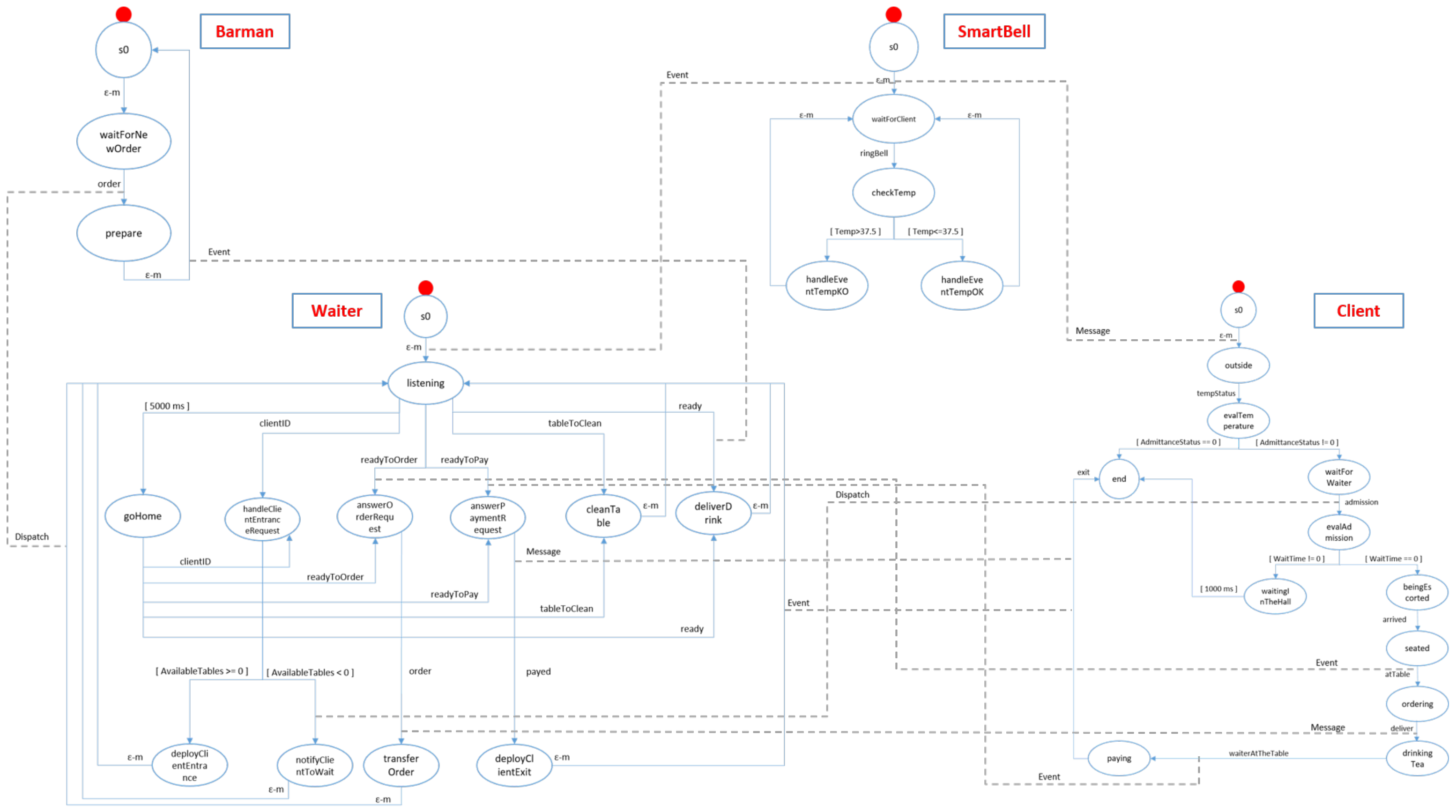
B. Client orders



C. Client pays



9. STATE DIAGRAMS



11. TEST PLAN

A. We plan to test various activities:

   i. The **smartBell** must let in the **hall** only **clients** that have a temperature below 37.5° Celsius.

   ii. When a client enters the tearoom, he may only sit at a table that is in the state **tableClean**. If there is no such table available, then the **waiter** must inform the **client** about the **maximum waiting time**.

   iii. The client must leave the **hall** when the **maximum waiting time** is over.

   iv. The client must receive the drink he ordered.

   v. The client must **pay and leave when the maximum stay time** is over.