REQUIREMENT ANALYSIS

We're still basing our following Problem Analysis on the SPRINT1 Requirement Analysis.

One thing that has been added is the need for a User Interface for the Client.

The UI should present the Client with the possibility to:

- ring the smartbell;
- ask the waiter to order, what to order and to pay;
- be notified if the Maximum Stay Time has been exceeded and he should leave.

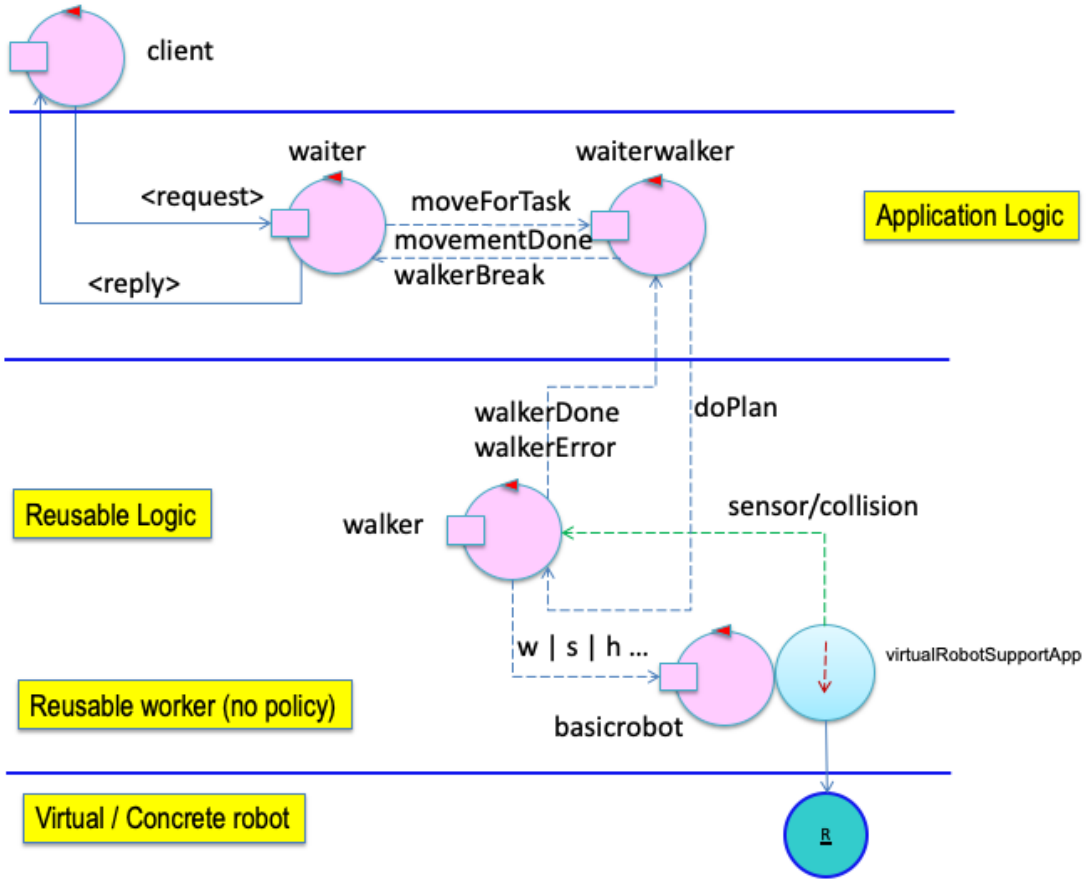
PROBLEM ANALYSIS

REFACTORING WAITERWALKER

As it was noticed in the Project Implementation during SPRINT1, the need to divide the Waiterwalker actor from a more simple Walker has risen to ensure **reusability** of components developed and also the **Single Responsibility Principle**, which will help greatly with the support for the system if the need for new requirements arises.

The division is meant to bring to light the separation between the actor that handles the interaction with the basicrobot and the actor that is meant to find the best path to the destination that the waiter has to reach, depending on the task the Waiter is executing.

The actors should then behave like the following diagram:



CLIENT-WAITER INTERACTION

Another issue that has come to light is the necessity to separate and make completely independent the Client from the Waiter.

As a matter of fact, as of SPRINT1 we had a Client deeply bound to the waiter's system through a direct exchange of messages from the Waiter to the Client, meaning that the Waiter would have had to know directly the Client. Since we want to build clients that are not necessarily QActors, but may be any kind of *alien*, we need to change the logic of the interaction. In particular, what makes for a better exchange of

messages is a series of Request-Reply, where the Client requests a service and the Waiter, once he executes it, replies with the proper payload.

The Client can also send Dispatch messages, while the Waiter can, at best, emit events.

The following image represents the new interaction diagram:

Client User Interface

Lastly, we discuss the Client User Interface. Keeping in mind that we also have to develop an Interface for the Manager, we want to build a reusable Interface for the Model (the Tearoom System) and the Views. This Interface will need to enable the View to:

- **Specify if the View is for interaction or monitoring**. For example, the Client would need to interact with the system while the Manager only needs to Monitor it;
- **Handle the communication with the Model**;
- **Update the View accordingly**.

An Architecture that can easily handle all this and that can last through any change in Requirements should provide a way to configure the Controller depending on needs, leaving it as independent as possible from both the Model and the View.

Mockups

The following images are intended as Mockups of a Web Application for the Client.

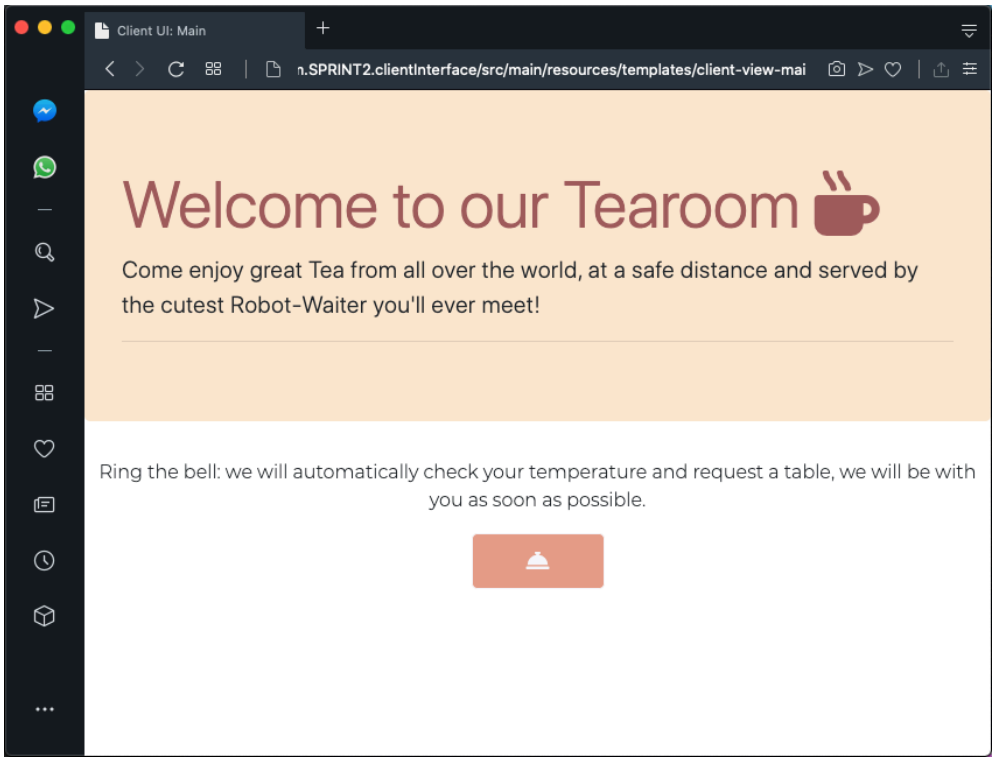


FIGURE 1 MAIN GREETINGS PAGE

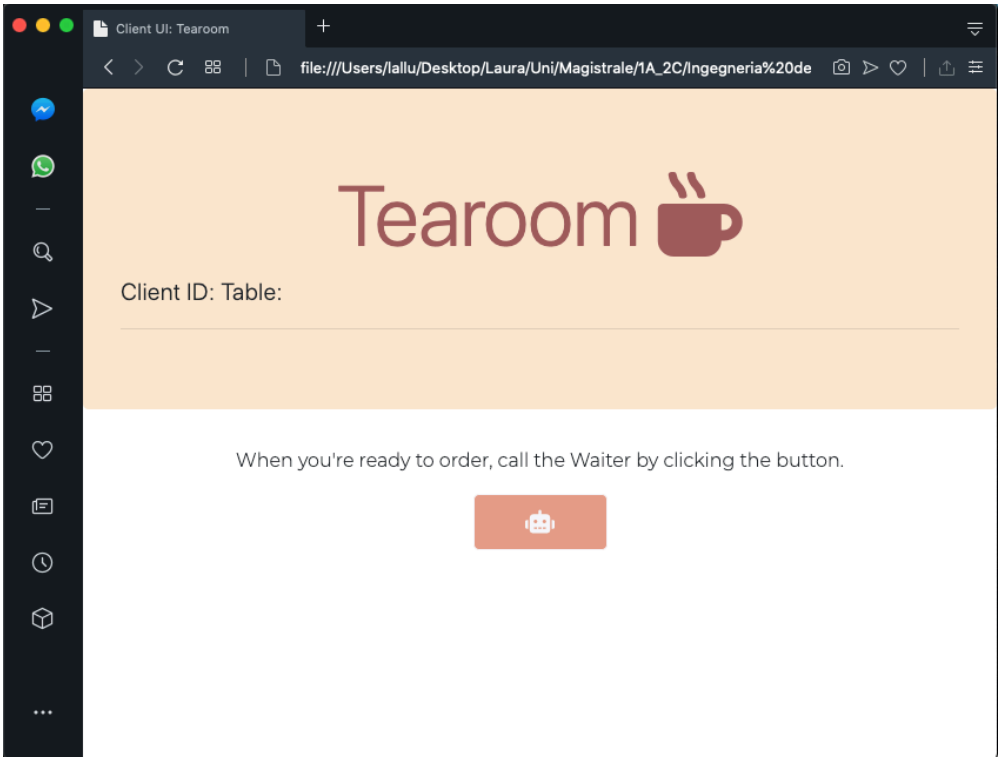


FIGURE 2 WAITER INTERACTION PAGE

TESTS

The **GUI** will need to be tested for:

- Pressing the button sends the correct message, through the Controller, to the Model;
- the correct View should be presented when the reply arrives from the Model.

The **waiterwalker** should be tested for:

- correct handling of moveForTask messages.

The barman and smartbell actors should be tested for:

- ability to answer unexpected messages.

The waiter should be tested for:

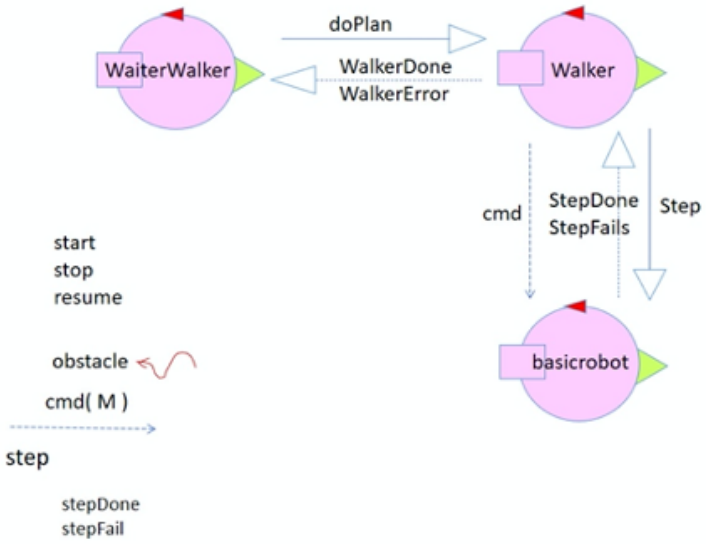
- its behaviour when he receives Requests that are different from the ones in the main schedule.

PROJECT

WAITERWALKER - WALKER

To implement the separation, we have left only the Knowledge Base to the Waiterwalker, while the Walker handles the planning of the route and the communication with the Basicrobot.

The interaction between the two Actors is handled through an exchange of Request-Reply messages.



CLIENT - WAITER INTERACTION

The Client, that was previously part of the Tearoom Context and System, is now independently defined in the `simclient.qak`, which grants the possibility to simulate the Client's behavior (only the "correct" one) without using a Graphic User Interface.

The communication between the Client and the Waiter has been radically changed. Now, the Client always sends either a Request, to ask for a task to be executed, or a Dispatch, to send ulterior information, if required, while the Waiter send a Reply when the Task is completed.

This separation has enabled a clearer implementation of tests and will greatly help with the Client User Interface, which will be developed in the next SPRINT.

TESTS

The tests have been implemented in the two classes:

- `SPRINTS/tearoom.SPRINT2/test/TestRobotPosition.kt`
- `SPRINTS/tearoom.SPRINT2/test/TestTearoomSys.kt`