Assignment 2.2 documentation

Péter Zaváczki
30433

November 7, 2018

0.1 Requirements

Design, implement and test a client-server distributed system that uses RPC to compute taxes and selling prices for cars.

0.2 Functional requirements

- Users introduce the information of their cars using a simple form (Web or Desktop):
  - int year – fabrication year
  - int engineSize – engine size
  - double price- purchasing price
- The application uses RPC to send the car information to the distributed object from the server that computes the following information depending on the client request:
  - Tax for a car, where the sum depends on the engine size. The equation for calculating the tax, and the sum's variation table can be seen on figure 1, respectively figure 2.

$$tax = \left(\frac{engineSize}{200}\right) * sum$$

Figure 1: The equation for calculating the tax

Engine Size	Sum (in RON)
<1600	8
1601-2000	18
2001-2600	72
2601-3000	144
>3001	290

Figure 2: The sum's variation table

  - Selling price for a car, for which you can find the equation on figure **??**.

$$price_{selling} = \begin{cases} price_{purchasing} - \frac{price_{purchasing}}{7} * (2018 - year) & if\ 2018 - year < 7 \\ 0 & otherwise \end{cases}$$

Figure 3: The equation for calculating the selling price

- The result of the invoked operation, tax, respectively selling price, is displayed on the client GUI.

0.3 Implementation technologies

- Java RMI
- Swing

0.4 Conceptual architecture

This application has a Client-Server architecture, which uses RPC. The Client being represented by a window containing the user's GUI, and the server, which can serve multiple clients simultaneously. On the server we bind the implementations of the methods invoked remotely by the client to the RMI register. The RMI register holds the implementations, which the client looks up and uses. On the client we can find a GUI implemented in Swing, which uses the remote implementations for the ITaxComputationService and ISellingPriceComputationService interfaces.

This can be seen on figure 4.

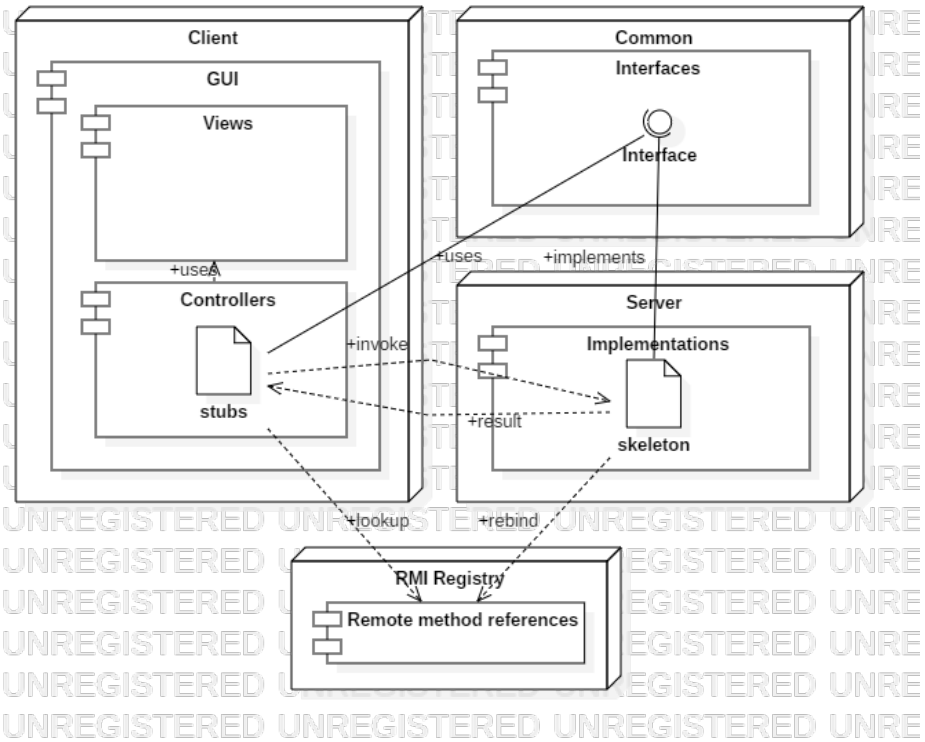

Figure 4: The conceptual architecture diagram of the project

0.5 Deployment

The application is deployed in three parts: The Client has the application locally, consisting of the GUI, with which he can interact. The Server has the implementations of

the ITaxComputationService and ISellingPriceComputationService interfaces, which are bound to the RMI register. The RMI register has the references to the implemented remote methods.

This deployment can be seen on figure 5.

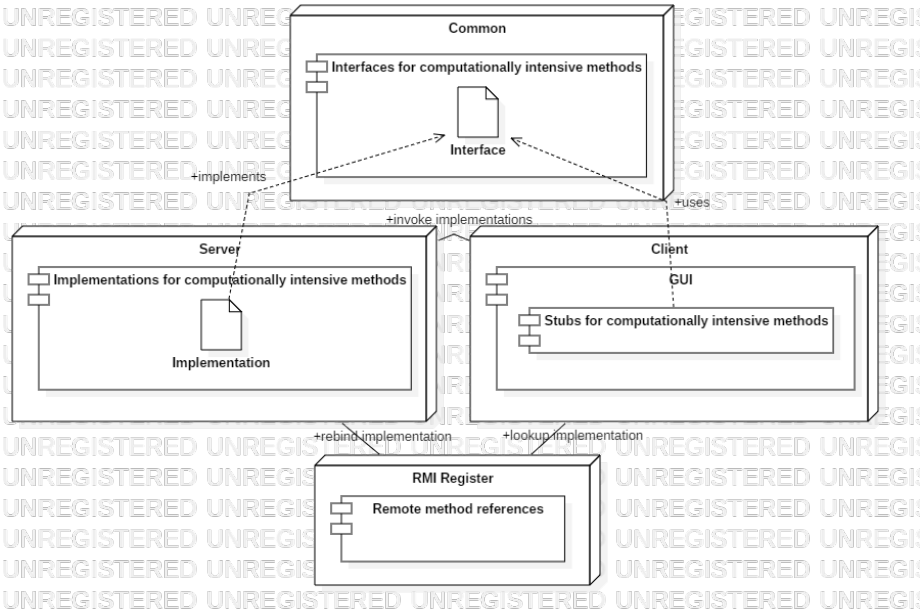

Figure 5: The deployment diagram of the project

0.6 Build considerations

- The client's GUI is made in Swing for easier development. As an alternative, a web based GUI could have been made, similarly to the previous assignment.
- Java RMI was used instead of .NET Remoting, due to already having knowledge in developing using Java, as opposed to .NET.