How to Model Multi-Party Service in SoaML?

Written Date : September 4, 2013

Multi-party service contracts refer to services that involve the participation of more than two participants, and with interaction between one another. An example would be an escrow purchase service where an order is made by purchaser, mediated through escrow agent and delivered by product provider. Escrow purchase service is a multi-party service not just because it involves the participation of three different parties, but also because each participant has to interact with all the other participants throughout the service.

What is this Tutorial about?

This tutorial is written to explain what is meant by multi-party service in terms of SoaML and how to specify such service with various SoaML diagrams in Visual Paradigm.

The example that will be used in this tutorial is about paying tax through online bank account. You will draw different SoaML diagrams to specify the tax payment service.

Preparation

In order to complete this tutorial, make sure you have Visual Paradigm downloaded and installed. Click here to download Visual Paradigm if you do not have it installed.

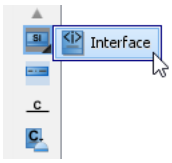
You are expected to have basic understanding of SOA, SoaML and how to develop SOA with SoaML using Visual Paradigm. If you are unclear about any of these topics, please read the tutorial - How to Draw SoaML Diagrams? first.

Part I - Define Interfaces in Service Interface Diagram

In a multi-service contract, all participants provide their own interface and use the interfaces of each party they call. Let's draw a service interface diagram for the three interfaces - Tax Payer, Tax Receiver, Bank.

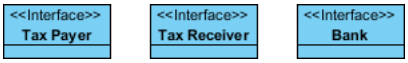
1. In a new project, create a service interface diagram by selecting **Diagram > New** from the toolbar. In the **New Diagram** window, enter *Service Interface Diagram* in the search field, click **Next**. Then, fill in the **Diagram Name** and **Description** (if any), click **OK** to confirm diagram creation.

2. We are going to draw the three interfaces. Expand the Service Interface tool in the diagram toolbar and select Interface.



3. Let's create an interface for tax payer. Click on the diagram to create an interface and name it Tax Payer.

4. Create two more interfaces - *Tax Receiver* and *Bank*.



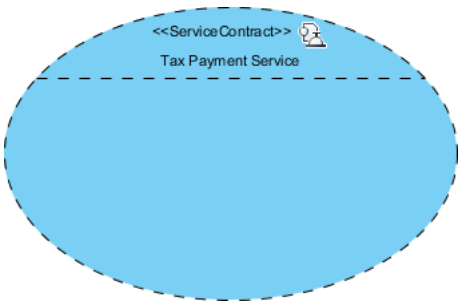
That's all for now. In each interface, there should be operations (or signals) to be invoked by others, but we are not going to specify them now. When we define the choreography of the service in sequence diagram, those operations will be automatically generated. And this will be done in coming sections.

Part II - Drawing Service Contract Diagram

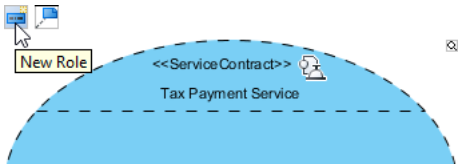
Multi-party service contracts are those involve two or more participants. Let's draw a service contract diagram for the tax payment (multi-party) service.

1. To create a service contract diagram, select **Diagram > New** from the toolbar. In the **New Diagram** window, enter **Service Contract Diagram** in the search field, click **Next**. Then, fill in the **Diagram Name** and **Description** (if any), click **OK** to confirm diagram creation.

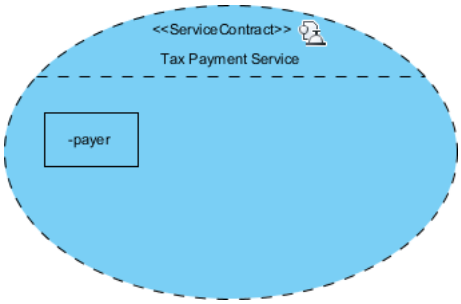
2. Select **Service Contract** from the diagram toolbar and click on the diagram to create a service contract. Name it *Tax Payment Service*.



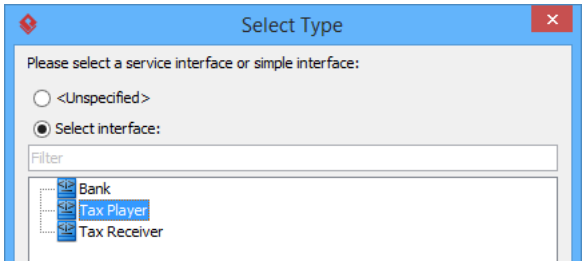
3. Visualize the roles of participants in the tax payment service. Click on the **New Role** resource to create a role in *Tax Payment Service* contract.



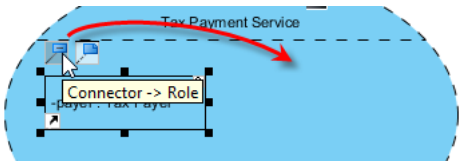
4. Name the role *payer*.



5. Let's type the role. Right click on the role and select **Select Type...** from the popup menu.
6. In the **Select Type...** window, select *Tax Payer* and click **OK**.

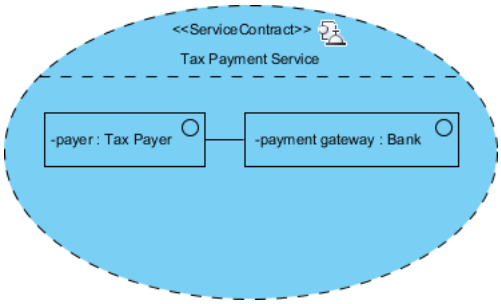


7. Visualize the role of bank. Make use of the **Connector -> Role** resource to create a new role from the *payer* role.

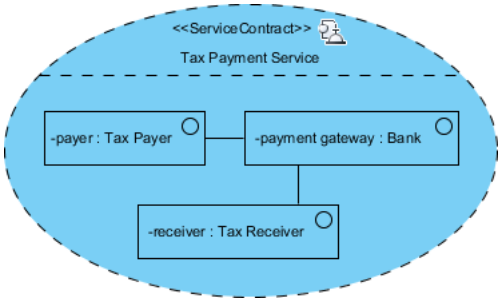


8. Name the role payment gateway
9. Let's type the role. Right click on the role and select **Select Type...** from the popup men

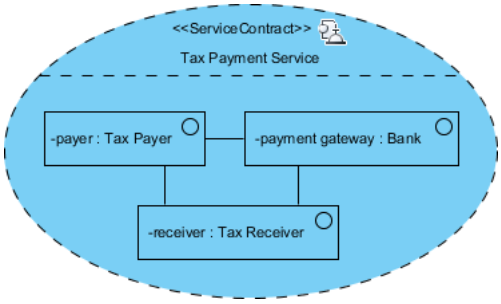
10. In the **Select Type...** window, select *Bank* and click **OK**.



11. From the role *payment gateway*, create the role *receiver*. Select *Tax Receiver* to be its type.



12. Tax receiver may interact with the tax payer to let him know about the payment status. So connect the *payer* and *receiver* role. Finally, your service contract diagram should look like this:

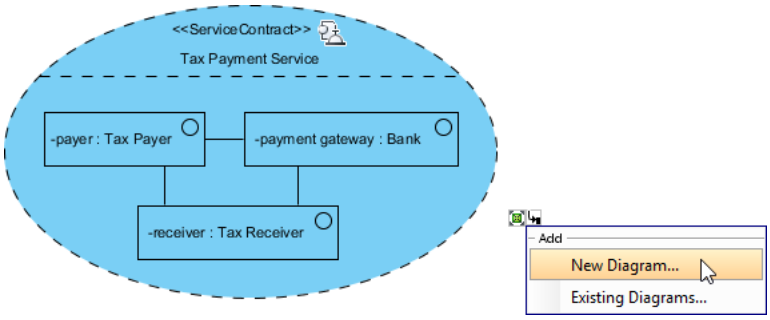


Part III - Specifying Multi-Party Choreography with UML Sequence Diagram

In a multi-service contract, the communication between parties and the choreography can be shown in an interaction diagram. Interaction diagram like UML sequence diagram shows who calls who and when the calls are made. Let's draw a sequence diagram to specify the choreography of the tax payment service.

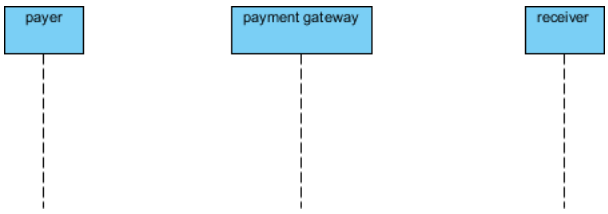
1. Click on the service contract *Tax Payment Service*.

2. Click on the tiny resource icon at the bottom right of the shape and select **New Diagram...** from the popup menu.

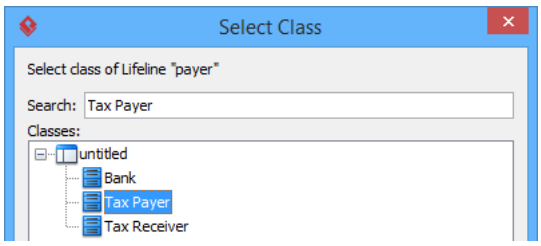


3. The **New Diagram** window is opened. In the **New Diagram** window, enter *sequence diagram* in the search field, click **Next**. Then, fill in the **Diagram Name** and **Description** (if any), click **OK** to confirm diagram creation.

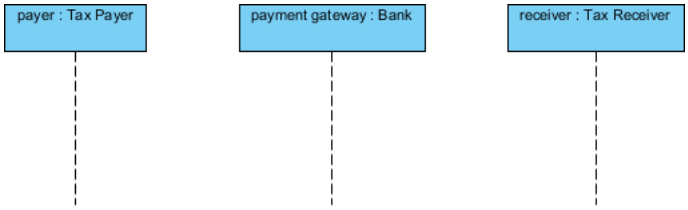
4. Create three lifelines in diagram. Name them *payer, payment gateway* and *receiver*.



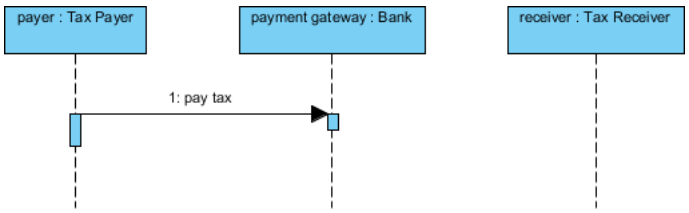
5. Set the classifier for the lifelines. Right click on the *payer* lifeline and select **Select Class > Select Class...** from the popup menu. In the **Select Class** window, select **Tax Payer** and click **OK**.



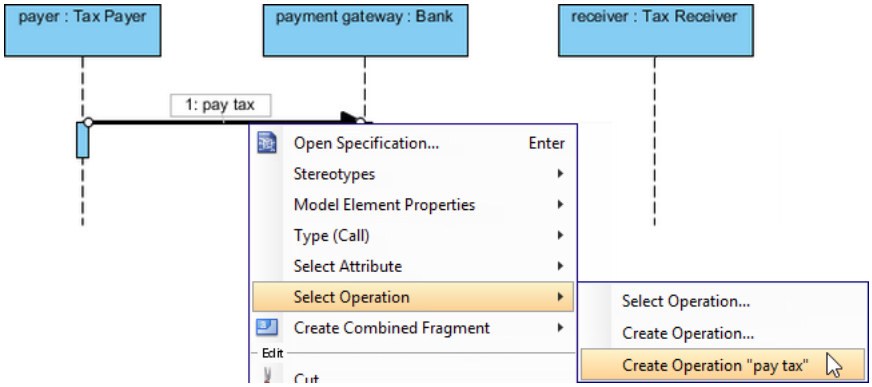
6. Set the *Bank* and *Tax Receiver* to be the classifier of *payment gateway* and *receiver* lifeline.



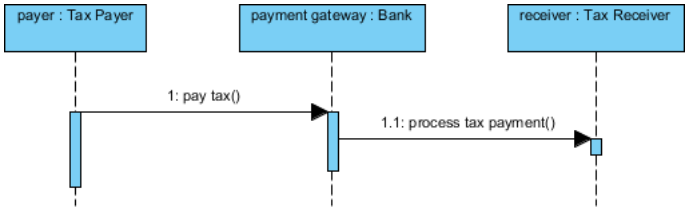
7. Time to model the interaction between the lifelines. The interaction starts with a payment request made by the payer on a bank account. So, create a message pay tax between *payer* and *payment gateway*.



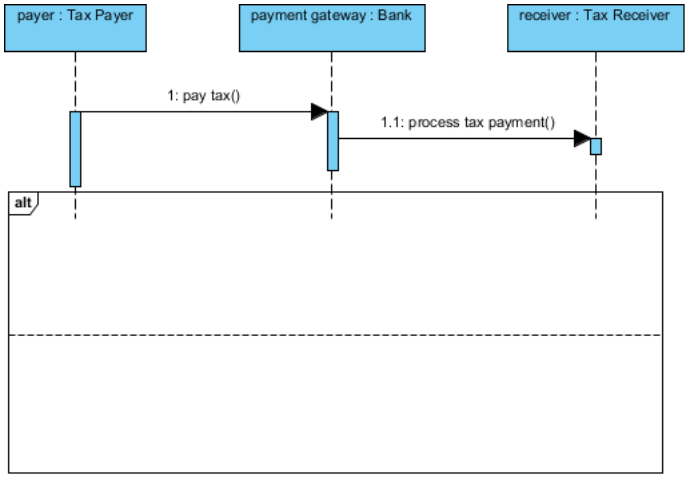
8. In order for 'pay tax' to be an operation in the *Bank* interface, we have to create an operation from the sequence message. Right click on the message and select **Select Operation > Create Operation "pay tax"** from the popup menu.



9. Having received the payment request, the bank will request the tax receiver to process the payment. Create a message process *tax payment* from *payment gateway* to *receiver*. Again, create an operation from the message.

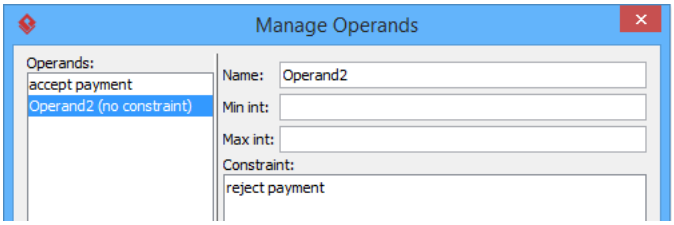


10. If the payment is made correctly, the tax receiver will send a confirmation message to both the bank and the tax payer. Otherwise, the tax receiver will send a reject message to the bank and the bank will forward the message to the tax payer. To represent a conditional flow, draw an alternative combined fragment that covers the lifelines.

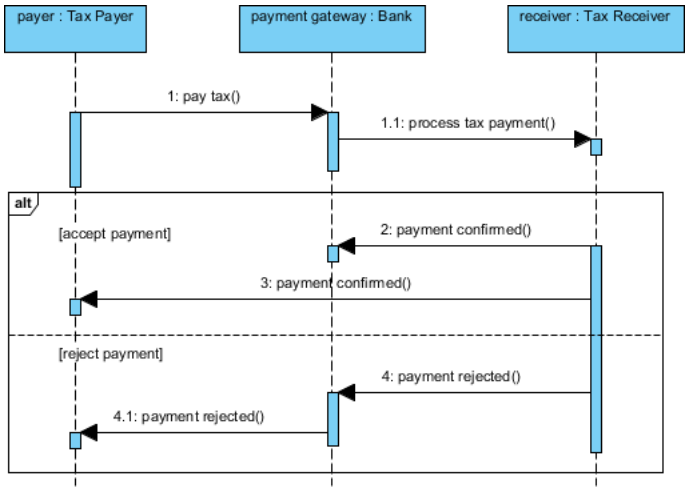


11. Right click on the **alt** tag at the top left of the combined fragment. Select **Operand > Manage Operands...** from the popup menu.

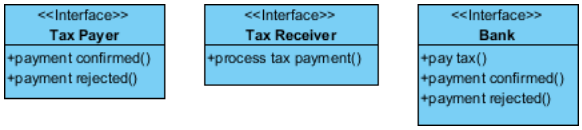
12. Enter the constraints of both operands in the **Manage Constraints** window. For the first operand, enter *accept payment* to be the constraint. For the second operand, enter *reject payment* to be the constraint. Click **OK** to confirm the change.



13. Create the messages between the lifelines. Remember to create operations for all the sequence messages you created. When finish, your sequence diagram should look like this:



14. When you draw the UML sequence diagram, you have created operations for the three lifelines. If you check the service interface diagram now, you can see the interfaces have operations listed.

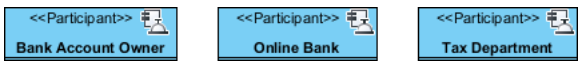


Part IV - Drawing Service Participant Diagram

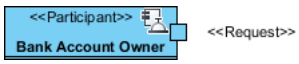
In a multi-party service, each participant provides their own interface and uses the interfaces of the other parties. This information can be represented with a service participant diagram. Let's draw a service participant diagram.

1. To create a service participant diagram, select **Diagram > New** from the toolbar. In the **New Diagram** window, enter *Service Participant Diagram* in the search field, click **Next**. Then, fill in the **Diagram Name** and **Description** (if any), click **OK** to confirm.

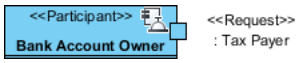
2. There are three participants in the tax payment service, the bank account owner, online bank and the tax department. Draw them in the diagram.



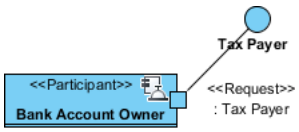
3. The bank account owner is a consumer of the tax payment service. Create a **<<Request>>** port in the *Back Account Owner* participant. You can create a **<<Request>>** port via the resource centric interface of a participant.



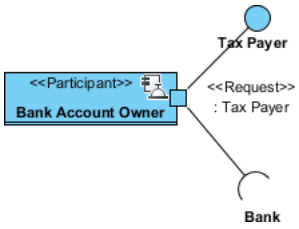
4. Let's type the port. Right click on the port and select **Select Type...** from the popup menu.
5. In the **Select Type** window, select *Tax Payer* and click **OK**.



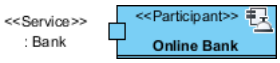
6. The bank account owner provides its interface, which is the *Tax Payer* interface, and according to the interaction modeled in the UML sequence diagram, we know that it uses the *Bank* interface. Draw the provided interface from the **<<Request>>** port first. Name the interface *Tax Payer*.



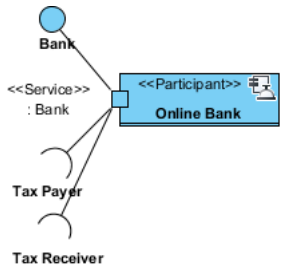
7. Draw the required interface from **<<Request>>** port. Name it *Bank*.



8. The participant *Online Bank* is a provider of the tax payment service. Create a **<<Service>>** port in it. Then, select *Bank* to be the port type.

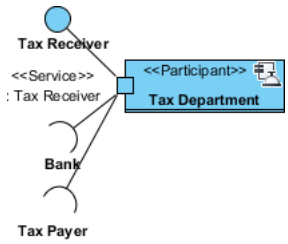


9. *Online Bank* provides the *Bank* interface and uses both the *Tax Payer* and *Tax Receiver* interface. Draw the provided and required interfaces. For this special case, you have to draw two required interfaces for the *Tax Payer* and *Tax Receiver* interfaces.

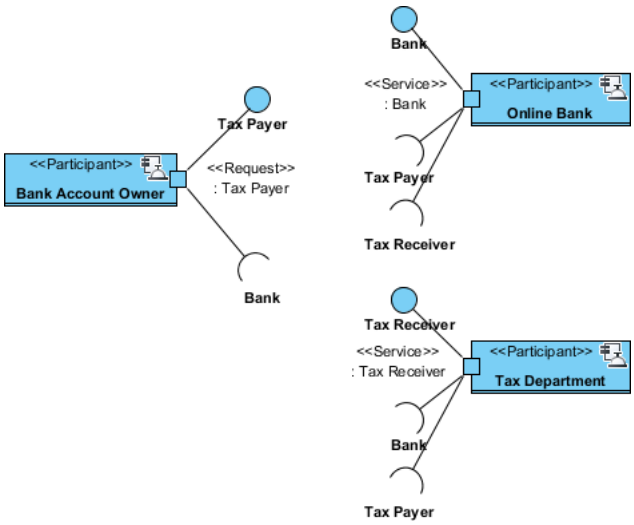


10. The participant *Tax Department* is also a provider of the tax payment service. Create a **<<Service>>** port in it. Then, select *Tax Receiver* to be the port type.

11. *Tax Department* provides the *Tax Receiver* interface and uses both the *Bank* and *Tax Payer* interface. Draw the provided and required interfaces.



When finish, your diagram should look like this:



Resources

1. Complete SoaML Sample Project - Tax-Payment_SoaML.vpp

Related Links

- Visual Paradigm Feature - SoaML Modeling
- Tutorial - How to Draw SoaML Diagrams?

Attributions

- Object Management Group - Service oriented architecture Modeling Language (SoaML) Specification

Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm tutorials
(https://www.visual-paradigm.com/tutorials/)