**[Homework 18](https://github.com/hendraanggrian/IIT-ITM511/blob/assets/assignments/hw18.pdf): Service-oriented software engineering**

**Problem 1**

*Why is it important to define exceptions in service engineering?*

It is unwise to expect the system in service-oriented engineering to work flawlessly at all times. Hardware periodically fails and software regularly encounters bugs. On top of this, users may feed unpredictable input to the service. Defining exceptions helps in identifying known errors and unchartered conditions during deployment. The exceptions can then be handled using applicable measures to restore the system and minimize service disruption.

**Problem 2**

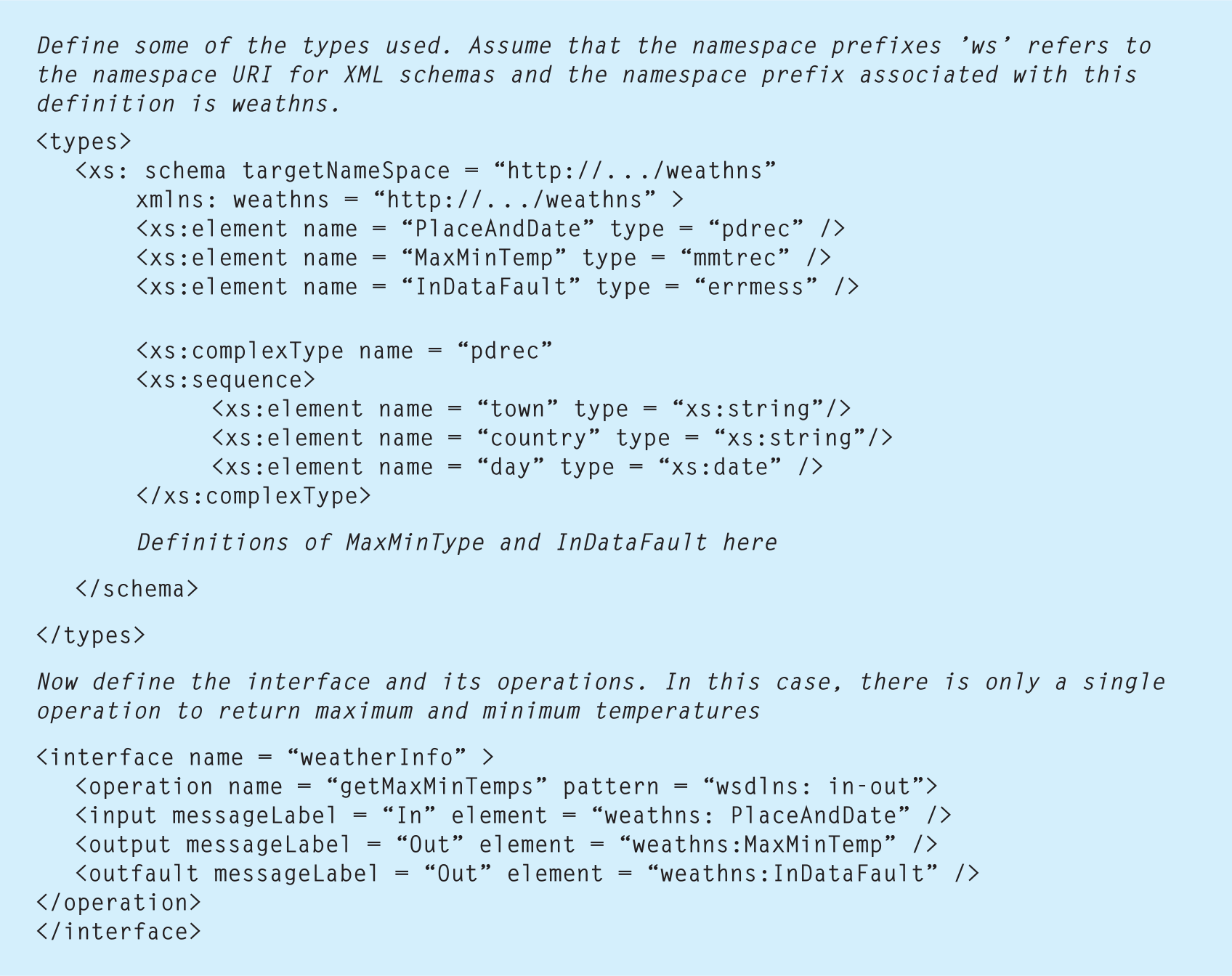
*Standards are fundamental to service-oriented architectures, and it was believed that standards conformance was essential for successful adoption of a service-based approach. However, RESTful services, which are increasingly widely used, are not standards-based. Discuss why you think this change has occurred and whether or not you think that the lack of standards will inhibit the development and take up of RESTful services.*

Service protocol like Simple Object Access Protocol (SOAP) is designed with system engineering standards with a collaborative architectural approach (REST vs. SOAP, 2019). On the other hand, Representational State Transfer (REST) does not follow the shared guidelines to be highly adaptable in a web environment where it excels. SOAP has built-in transactional models whereas the customization of REST services is managed according to the API blueprint leveraging HTTPS methods like GET, POST, PUT and DELETE.

Despite lacking a standardized communication protocol, REST services can follow community best practices to enforce interoperability. Other features including caching and security can be integrated into REST, making it a popular option in modern services. Wide adaptability also means an abundance of third-party REST full-fledged frameworks and lightweight libraries.

**Problem 3**

*Extend* ***Figure 18.5*** *to include WSDL definitions for MaxMinType and InDataFault. The temperatures should be represented as integers, with an additional field indicating whether the temperature is in degrees Fahrenheit or degrees Celsius. InDataFault should be a simple type consisting of an error code.*

**

***Figure 18.5*** *Part of a WSDL description for a web service*

<types>

<xS:schema

targetNameSpace="http://.../weathns"

xmIns:weathns="http://.../weathns">

<xs:element name="PlaceAndDate" type="pdrec"/>

<xs:element name="MaxMinTemp" type="mmtrec"/>

<xs:element name="InDataFault" type="errmess"/>

<xs:complexType name="pdrec">

<xs:sequence>

<xs:element name="town" type="xs:string"/>

<xs:element name="country" type="xs:string"/>

<xs:element name="day" type="xs:date"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="mmtrec">

<xs:sequence>

<xs:element name="maxTemp" type="tempType"/>

<xs:element name="minTemp" type="tempType"/>

</xs:sequence>

</xs:complexType>

<xs:simpleType name="tempType">

<xs:restriction base="xs:int">

<xs:enumeration value="0"/>

<xs:enumeration value="1"/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="errmess">

<xs:restriction base="xs:int">

<xs:minInclusive value="100"/>

<xs:maxInclusive value="999"/>

</xs:restriction>

</xs:simpleType>

</xS:schema>

</types>

[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/web_service_wsdl.xml)

Below is the interface corresponding to **MaxMinTemp** and **InDataFault** defined in the Web Services Description Language (WSDL) configuration above.

<interface name="weatherInfo">

<operation name="getMaxMinTemps" pattern="wsd1ns:in-out">

<input messageLabel="In" element="weathns:PlaceAndDate"/>

<output messageLabel="Out" element="weathns:MaxMinTemp"/>

<outfault

messageLabel="Out"

element="weathns:InDataFault"/>

</operation>

</interface>

**Problem 4**

*Suggest how the SimpleInterestCalculator service could be implemented as a RESTful service.*

The **SimpleInterestCalculator** is defined in an API blueprint format. Users can calculate interest rates given a pre-defined amount or by supplying a yearly range. The server then responds with a relevant response code determining whether or not the request is successful.

FORMAT: 1A

# Simple Interest Calculator API

Welcome to the `https://simple-interest-calculator.com/api/v1`, a free and

straightforward way to assess return on investment.

## Calculate using a fixed value [POST /value]

+ Request (application/json)

+ Attributes

- principal: 100 (number, required) - The base amount.

- rate: 5 (number, required) - The interest rate.

- time: 2 (number, required) - The time period in years.

+ Response 200 (application/json)

+ Attributes

- principal: 100 (number) - The principal amount.

- rate: 5 (number) - The interest rate.

- time: 2 (number) - The time period in years.

- interest: 10 (number) - The calculated simple interest.

+ Response 400 (application/json)

+ Attributes

- message: "Bad Request" (string) – When input is negative.

## Calculate using yearly range [POST /range]

+ Request (application/json)

+ Attributes

- principal: 100 (number, required) - The base amount.

- rate: 5 (number, required) - The interest rate.

- start: 2015 (year, required) – The beginning year.

- end: 2015 (year, required) – The final year.

+ Response 200 (application/json)

+ Attributes

- principal: 100 (number) - The principal amount

- rate: 5 (number) - The interest rate

- start: 2015 (year, required) – The beginning year.

- end: 2015 (year, required) – The final year.

- interest: 10 (number) - The calculated simple interest

+ Response 400 (application/json)

+ Attributes

- message: "Bad Request" (string) – When input is negative or starting year

exceeds final year.

[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/simple_interest_calculator.apib)

**Problem 5**

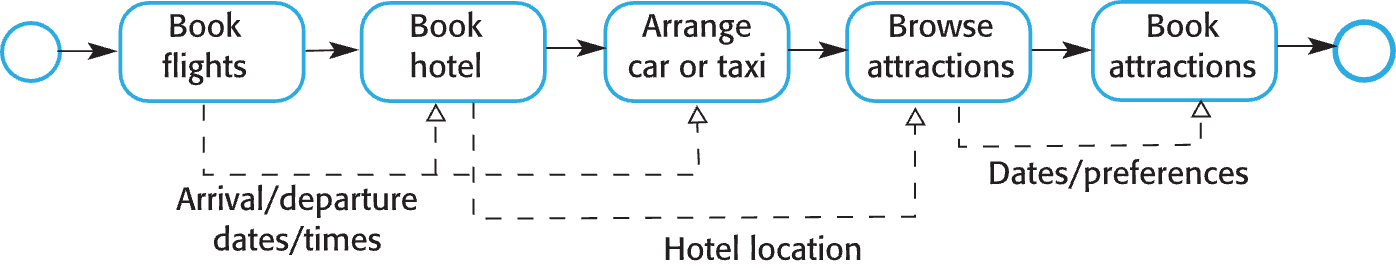
*What is a workflow? List out the key stages in the process of system construction by composition.*

The workflow of service-oriented software is the composite service stages from start to finish. The stages are connected using appropriate notation to depict a condition or an object. The key stages of constructing the system are:

1. **Workflow outline:** Propose an initial abstract for an idealistic service satisfying user requirements (Sommerville, 2016).
2. **Services discovery:** Search existing third-party services for software reuse, prominent service providers such as Oracle and Microsoft are preferred.
3. **Services selection:** Filter and integrate the service candidates into the main service.
4. **Workflow refinement:** Deliver service details not initially outlined, edge cases and other enhancements.
5. **Workflow creation:** Produce the workflow applying modeling standards like Business Process Modeling and Notation (BPMN).
6. **Final testing:** Simulate the combined services illustrated in the final workflow draft.

**Problem 6**

*Design possible input and output messages for the services shown in* ***Figure 18.13****. You may specify these in the UML or in XML.*

**

***Figure 18.13*** *Vacation package workflow*

Start with a sample procedure to book a round-trip flight in JSON format. In case of a single trip, the **book\_flight\_response** would only have one flight entry.

**book\_flight\_request.json**

{

"departure\_date": "2024-07-14",

"return\_date": "2024-07-17",

"origin": "Chicago",

"destination": "New York",

"passengers": [

{

"name": "John Doe",

"passport\_no": "E00007730"

}

],

"payment": {

"amount": 80,

"method": "VISA",

"account": "0987654321"

}

}

**book\_flight\_response.json**

{

"booking\_id": "xa12gw3va0",

"flights": [

{

"no": "BA212",

"when": "2024-07-14",

"from": "ORD",

"to": "JFK"

},

{

"no": "AB121",

"when": "2024-07-21",

"from": "JFK",

"to": "ORD"

}

],

"passengers": [

{

"name": "John Doe",

"passport\_no": "E00007730",

"seat": "12C"

}

]

}

[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/book_flight.json)

Continuing into the hotel booking process, the **book\_hotel\_response** returns the list of hotel services and their availability.

**book\_hotel\_request.json**

{

"check\_in\_date": "2024-07-14",

"check\_out\_date": "2024-07-17",

"room\_count": 1,

"hotel": {

"name": "Hard Rock Hotel",

"address": "159 West 48th St.",

"city": "New York",

"state": "NY"

},

"payment": {

"amount": 360,

"method": "paypal",

"account ": "johndoe"

}

}

**book\_hotel\_response.json**

{

"booking\_id": "k12vzd80v2",

"check\_in\_date": "2024-07-14",

"check\_out\_date": "2024-07-17",

"room\_count": 1,

"hotel": {

"name": "Hard Rock Hotel",

"address": "159 West 48th St.",

"city": "New York",

"state": "NY",

"services": {

"wifi": true,

"breakfast": true,

"room\_service": true,

"nightclub": false

}

}

}

[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/book_hotel.json)

When reserving transportation, users provide the coordinate generated by client devices and the **book\_taxi\_response** replies with an associated address.

**book\_taxi\_request.json**

{

"time": "2024-07-14T13:00:00",

"passenger\_count": 2,

"pickup\_location": {

"lat": 9.96233,

"lng": 49.80404

},

"dropoff\_location": {

"lat": 6.11499,

"lng": 50.67891

},

"payment": {

"amount": 32,

"method": "cash"

}

}

**book\_taxi\_response.json**

{

"booking\_id": "9dvb1324bn",

"time": "2024-07-14T13:00:00",

"passenger\_count": 2,

"pickup\_location": {

"address": "959 N Western Ave",

"city": "Chicago",

"state": "IL",

"lat": 9.96233,

"lng": 49.80404

},

"dropoff\_location": {

"address": "O'Hare International Airport",

"city": "Chicago",

"state": "IL",

"lat": 6.11499,

"lng": 50.67891

}

}

[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/book_taxi.json)

Lastly, users browse for available attractions given the date and budget. To reserve a spot at an attraction, users enter a special code they get from **browse\_attraction\_response**.

**browse\_attraction\_request.json**

{

"date": "2024-07-15",

"city": "New York",

"state": "NY",

"budget": 30

}

**book\_attraction\_request.json**

{

"date": "2024-07-15",

"attraction\_id": "nyc-museum2",

"guest\_count": 1,

"payment": {

"amount": 0

}

}

**browse\_attraction\_response.json**

{

"date": "2024-07-15",

"city": "New York",

"state": "NY",

"budget": 30,

"attractions": [

{

"id": "nyc-museum2",

"name": "Museum of Modern Art",

"fee": 0

},

{

"id": "nyc-statue1",

"name": "Statue of Liberty",

"fee": 25

}

]

}

**book\_attraction\_response.json**

{

"date": "2024-07-15",

"guest\_count": 1,

"attraction": {

"id": "nyc-museum2",

"name": "Museum of Modern Art",

"open\_time": "7:30:00",

"close\_time": "21:00:00"

}

}

[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/book_attraction.json)

**Problem 7**

*Giving reasons for your answer, suggest two important types of application where you would not recommend the use of service-oriented architecture.*

1. **Small applications:** Service-oriented architecture (SOA) standards call for additional development overhead not worth implementing in simple applications with a limited scope. For instance, a stock calculator application on our devices does not need an external service to perform a simple calculation.
2. **Legacy applications:** Mature applications are bound to obsolete technology hindering the transition to SOA. For example, a monolithic application with a custom communication protocol would need a major rewrite to follow SOA standards.

**Problem 8**

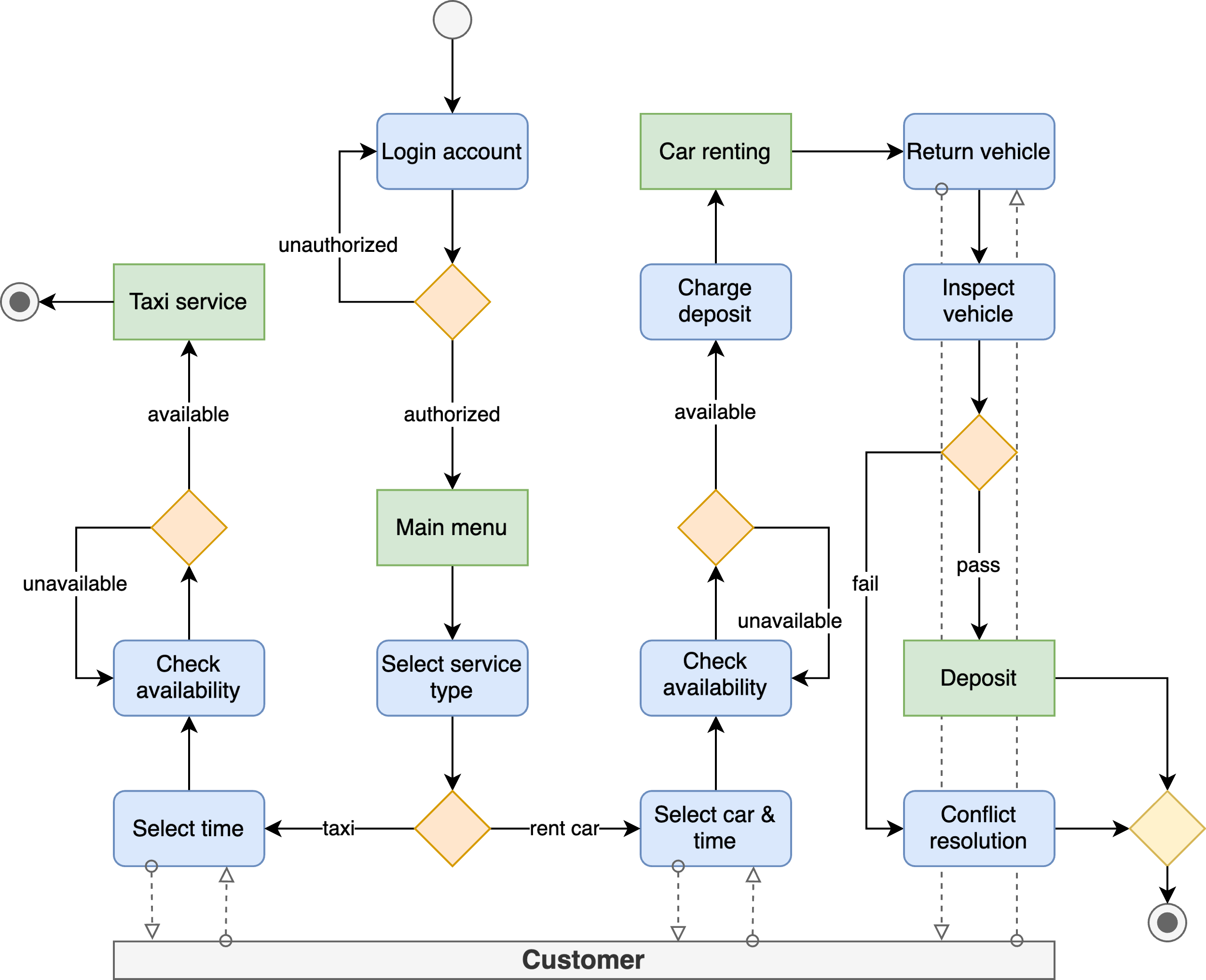
*Explain what is meant by a “compensation action” and, using an example, show why these actions may have to be included in workflows.*

When consuming a service, users might make choices they later regret. Software bugs could also lead to errors which the administrators need to fix.  
The SOA workflow definition provides the ability to revert such actions with compensation actions.

For instance, consumers go through several steps to purchase a product in an online marketplace. They start by placing an item into the shopping cart and finalizing the order by verifying the payment account. However, unlike traditional shopping, the item amount is not deducted from the warehouse when users place it in the cart, opening up the possibility of multiple purchases of the same item. The marketplace can correct this overselling issue using compensation actions and maintain a high-quality service.

**Problem 9**

*For the example of the vacation package reservation service, design a workflow that will book ground transportation for a group of passengers arriving at an airport. They should be given the option of booking either a taxi or a hire car. You may assume that the taxi and rental car companies offer web services to make a reservation.*



[View source](https://github.com/hendraanggrian/IIT-ITM511/blob/main/assignments/hw18/figures.drawio)

**Problem 10**

*Using an example, explain in detail why the thorough testing of services that include compensation actions is difficult.*

In services engineering, a compensation action undoes previous activities when the primary service fails or misbehaves. Because these actions often rely on interacting with other services, testing them requires mimicking external services to reproduce errors in a test environment. In addition, the test simulations must be revised whenever the underlying services are modified.

For example, a customer of a food delivery service places an order from a restaurant that currently does not have the necessary ingredients. In response to this situation, the automated compensation mechanism within the system can execute various actions. These compensation actions may include canceling the order, redirecting the order to other restaurants, or substituting the item with a replacement of equal value.

# Bibliography

*REST vs. SOAP*. (2019, April 8). Retrieved from Red Hat: https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest/

Sommerville, I. (2016). Software Engineering. In *Software engineering diversity* (10 ed., p. 543). Pearson Education.