CSCI927 Service-Oriented Software Engineering (Project Report)

An application for online car-hailing services

Group Members (Group 17): Wenrui Chen, Lu Xiong, Bing Xia, Yunbo Zhang

This project is an application for online car-hailing services. Our project describes the whole process of online car-hailing services, which are related to qualification view, customers, allotment of the order and drivers.
We have four group members.
Wenrui Chen is responsible for the process of qualification view.
Lu Xiong is responsible for the process of customer hailing and the writing of the report.
Bing Xia designs the process of the allotment of the order to cars.
Yunbo Zhang designs the process of drivers.
Our process is designed by BPMN, semantic effect annotation, CMMN, XML, DMN, TDM, Process Mining, Petri nets from Process Mining, Process Compliance and RuleML.
We make some modifications to BPMN, which is adding some conditions about ExclusiveGateway.
The part of the XML of BPMN picture above. Because the whole XML code is too long, so we put the part of it in our appendix. And we modify the XML by the new BPMN picture.
We write the semantic effect annotation of our BPMN picture to analyse the semantics of process, which determines the effects achieved. And because it is too long, so we just write part of the semantic effect annotation.
CMMN is designed to describe the important competencies in our project, and it can easy to model the knowledge-intensive tasks, especially for activities started in an ad-hoc way.
DMN is added because it can provide analysis with a tool for separating business decision logic from business processes, which helps greatly reduce the complexity and readability of business process models. The semantics of decision tables in DMN is more expressive than TDM. It can return multiple values and can specify how multiple values are aggregated.
Process mining shows process management by analyzing event execution logs and data mining. It can improve the process efficiency and understanding of our processes.
Formal language based on deontic logic can show normative positions stemming from compliance requirements.
And we also use RuleML to express business rules of our business process compliance. It allows the development, execution, and exchange of our rules.

The details of our project are in the following appendix.

Appendix

Project Title: An application for online car-hailing services

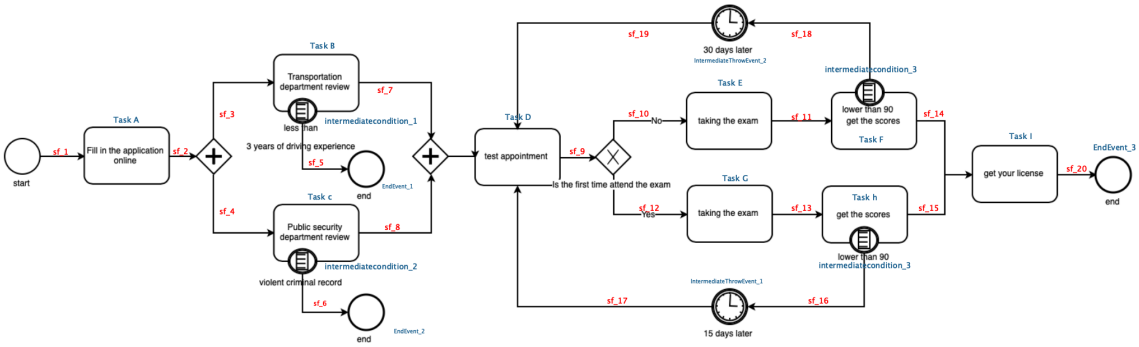
Group Members (Group 17): Wenrui Chen, Lu Xiong, Bing Xia, Yunbo Zhang

This project is an application for online car-hailing services. Our project describes the whole process of online car-hailing services, which are related to qualification view, customers, allotment of the order and drivers.
We have four group members.
Wenrui Chen is responsible for the process of qualification view.
Lu Xiong is responsible for the process of customer hailing and the writing of the report.
Bing Xia designs the process of the allotment of the order to cars.
Yunbo Zhang designs the process of drivers.
Our process is designed by BPMN, semantic effect annotation, CMMN, XML, DMN, TDM, Process Mining, Petri nets from Process Mining, Process Compliance and RuleML.
We make some modifications to BPMN, which is adding some conditions about ExclusiveGateway.
The part of the XML of BPMN picture above. Because the whole XML code is too long, so we put the part of it in our appendix. And we modify the XML by the new BPMN picture.
We write the semantic effect annotation of our BPMN picture to analyse the semantics of process, which determines the effects achieved. And because it is too long, so we just write part of the semantic effect annotation.
CMMN is designed to describe the important competencies in our project, and it can easy to model the knowledge-intensive tasks, especially for activities started in an ad-hoc way.
DMN is added because it can provide analysis with a tool for separating business decision logic from business processes, which helps greatly reduce the complexity and readability of business process models. The semantics of decision tables in DMN is more expressive than TDM. It can return multiple values and can specify how multiple values are aggregated.
Process mining shows process management by analyzing event execution logs and data mining. It can improve the process efficiency and understanding of our processes.
Formal language based on deontic logic can show normative positions stemming from compliance requirements.
And we also use RuleML to express business rules of our business process compliance. It allows the development, execution, and exchange of our rules.

1 In the process of qualification review (Wenrui Chen)

1.1 BPMN

This BPMN picture depicts the whole process of qualification. The qualification examination is carried out in the early stage and the test is carried out in the later stage.



1.2 XML for BPMN

*Notice: We cannot print the right “ ” in our codes, so the all ” ” are “ ”.

```
<bpmn:process id="Process_1" isEcecutable="true">
    <bpmn:task id="Task_1" name="Task_A">
        <bpmn:incoming>SequenceFlow_1</bpmn:incoming>
        <bpmn:outgoing>SequenceFlow_2</bpmn:outgoing>
    </bpmn:task>
    <bpmn:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1"
        targetRef="Task_1" />
    ......
    ......
    <bpmn:intermediateCatchevent id="IntermediateThrowEvent_1" name="15
        _days_later">
        <bpmn:incoming>SequenceFlow_16</bpmn:incoming>
        <bpmn:outgoing>SequenceFlow_17</bpmn:outgoing>
        <bpmn:timerEventDefinition />
    </bpmn:intermediateCatchEvent>
    <bpmn:boundaryEvent id="intermediatecondition_4">
        <bpmn:outcoming>Sequenceflow_18</bpmn:outcoming>
    </boundaryEvent>
    ......
    ......
    <bpmn:endEvent id="EndEvent_3">
        <bpmn:incoming>SequenceFlow_20</bpmn:incoming>
    </bpmn:endEvent>
</bpmn:process>
```

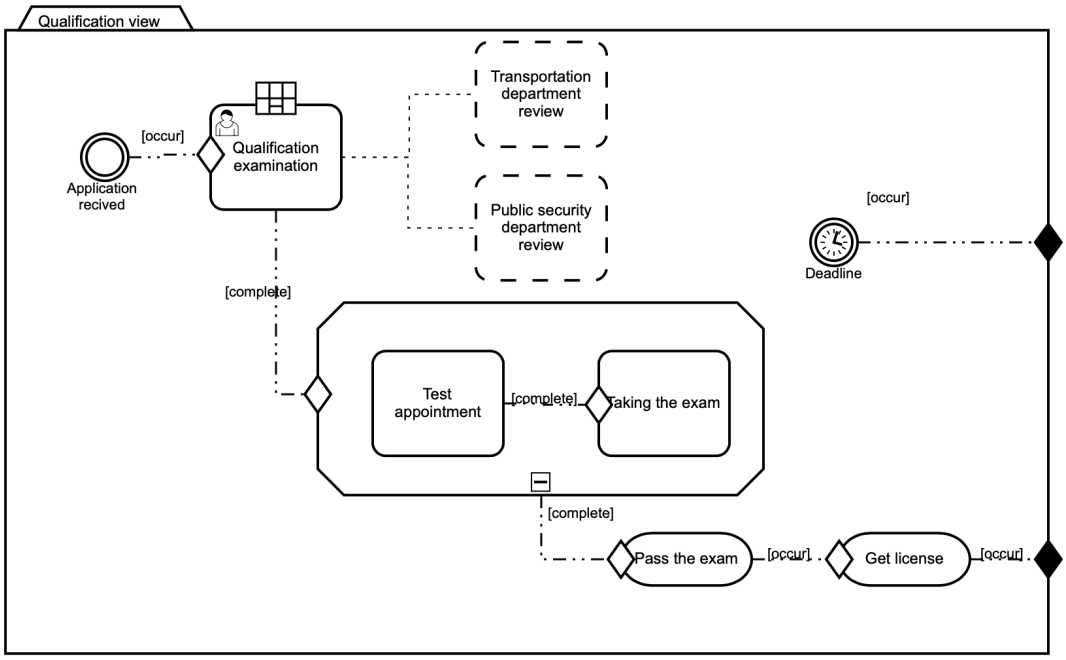
1.3 Semantic Effect Annotation

Cumulative Effect Scenarios:
t9: Scenario(1): ⟨{t1, {⟨t2⟩, ⟨t3⟩}, t4, t5, t7, t9⟩, {t1, {⟨t2⟩, ⟨t3⟩}, t4, t6, t8}⟩

t9: Scenario(2): ⟨⟨t1, {⟨t2⟩, ⟨t3⟩}, t4, t6, t8, t9⟩, {t1, {⟨t2⟩, ⟨t3⟩}, t4, t5, t7}⟩
Cumulative Effects of Tasks/Activities:
Objects of interests:
Application a, Review r1,r2, Appointment p, Times t, Exam e1,e2, Outcome o1,o2, Grade g1,g2, Result s, License l.
States:
Application rejected or approved rejected(a), approved(a), Review pass - pass(r1) and pass(r2), Result pass or not pass success(s), fail(s), License get(l)
Relationships between objects:
Review investigated with an outcome investigated(r1, o),investigated(r2,o)
Appointment with times times(p, t)
Exam with an grade grade(e, g)
The cumulative effects :
(approved(a)∨(investigated(r1,o1)∧(investigated(r2,o2)))∨times(p, t)∨((exam(e1,g1)∨ success(s))∧exam(e2,g2)∨success(s))∨get(l))

1.4 CMMN

This CMMN describes the same process as BPMN. The qualification examination is conducted in the early stage. After passing the examination, the driver can take the test and obtain the score.



1.5 XML for CMMN

The XML of CMMN

```
<?xml version="1.0" encoding="UTF-8"?>
    <cmmn:case id="Case_1">
        <cmmn:casePlanModel id="CasePlanModel_1" name="Qualification_
            view">
            <cmmn:planItem id="PlanItem_1" definitionRef="
                EventListener_1" />
            <cmmn:planItem id="PlanItem_2" definitionRef="Stage_1">
                <cmmn:entryCriterion id="EntryCriterion_1" sentryRef="
                    Sentry_1" />
            </cmmn:planItem>
            <cmmn:planItem id="PlanItem_3" definitionRef="
                TimerEventListener_1" />
            <cmmn:planItem id="PlanItem_4" definitionRef="Milestone_1">
                <cmmn:entryCriterion id="EntryCriterion_2" sentryRef="
                    Sentry_2" />
            </cmmn:planItem>
        ......
        ......
    <cmmn:task id="Task_3" name="Transportation_department_review" />
            <cmmn:task id="Task_4" name="
                Public_security_department_review" />
            <cmmn:exitCriterion id="ExitCriterion_1" sentryRef="
                Sentry_5" />
            <cmmn:exitCriterion id="ExitCriterion_2" sentryRef="
                Sentry_6" />
        </cmmn:casePlanModel>
    </cmmn:case>
</cmmn:definitions>
```

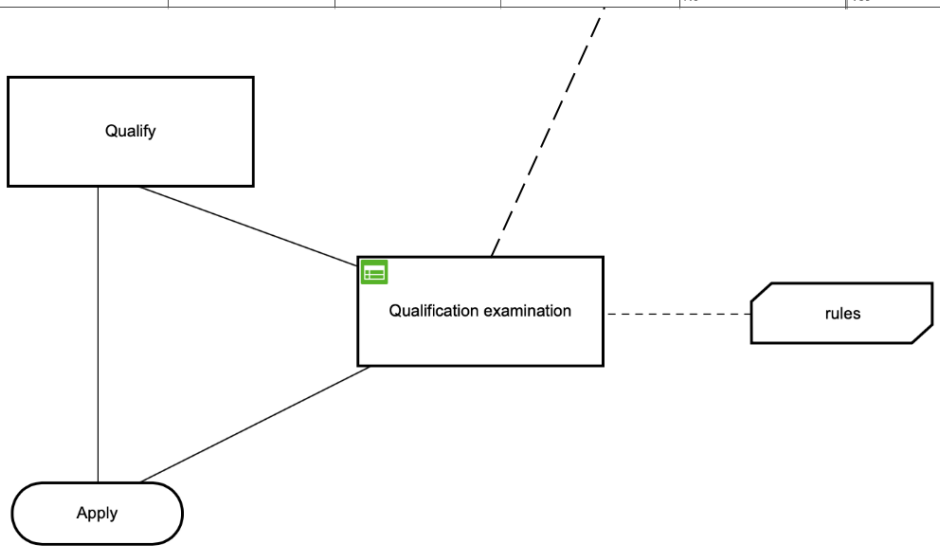
1.6 DMN

This DMN describes the decision-making process in the qualification examination process. In the picture, the examination body needs to consider multiple variables and reach a final conclusion. If one of them is not satisfied, it cannot pass the examination, otherwise, it can be passed.
The Qualification Decision Model and Notation

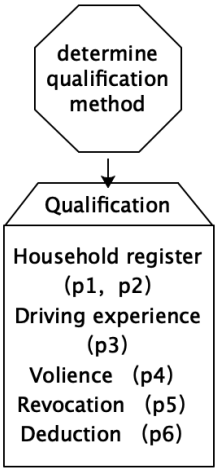
Qualification examination

`decision`

A	Input +					Output +
	Household register	Driving experience	Violence record	Revocation of license in 3 years	Deduction of driver's license	Qualifications
	`string`	`integer`	`integer`	`string`	`string`	`string`
1	six mouths longer register	-	-	-	-	Yes
2	one year longer temporary register	-	-	-	-	Yes
3	-	≥3	-	-	-	Yes
4	-	-	≥12	-	-	No
5	-	-	[0..11]	-	-	Yes
6	-	-	-	No	-	Yes
7	-	-	-	-	No	Yes



The Decision Model



Conditions										Conclusions	
household register		driving experience		volience record		revocation		deduction		qualification	
is	temporary ≥12 months									is	Yes
is	formal ≥6 months									is	Yes
		is	≥3 years							is	Yes
				is	NO					is	Yes
						is	No			is	Yes
								is	≥12 point	is	No

1.7 Process Mining

The process mining describes the process of obtaining a driver's qualification after passing the qualification certification.
Hxecution Log:
case 1:task D
case 2:task D

case 1:task E
case 2:task G
case 1:task F
case 2:task H
case 1:task I
case 2:task I

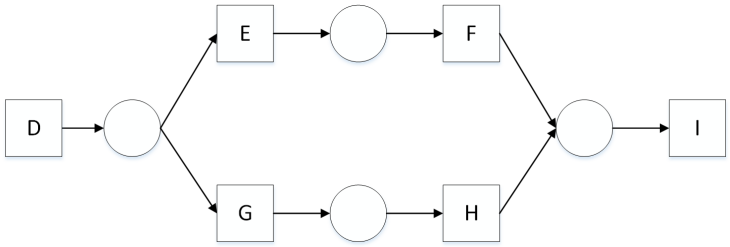
W={DEFI, DGHI} T={D, E, F, G, H, I}
T_I={D} T_O={I}
Girect Succession: D>E E>F F>I D>G G>H H>I
Causality: D→E E→F F→I D→G G→H H→I
Concurrency:
Choice: D#F D#H D#I E#G E#H E#I F#G F#H G#I

	D	E	F	G	H	I
D	#	→	#	→	#	#
E	←	#	→	#	#	#
F	#	←	#	#	#	→
G	←	#	#	#	→	#
H	#	#	#	←	#	→
I	#	#	←	#	←	#

Table 1. Footprint Table

X_L={({D},{E}),({E},{F}),({F},{I}),({D},{G}),({G},{H}),({H},{I}),({D},{E,G}),({F,H},{I})}
Y_L={({E},{F}),({G},{H}),({D},{E,G}),({F,H},{I})}

1.8 Petri Net From Process Mining



1.9 Business Process Compliance and RuleML

This is the rules for the verification section and describes the verification process.
Censors: ApplicationReceived⊢OL VerifyQualification
Censors: VerifyQualification⊢OL InformResult
Merge:
A: ApplicationReceived⊢OL VerifyQualification⊗ OL InformResult

```
<Imp label='A'>
    <body>
        <Atom>
            ApplicationReceived
```

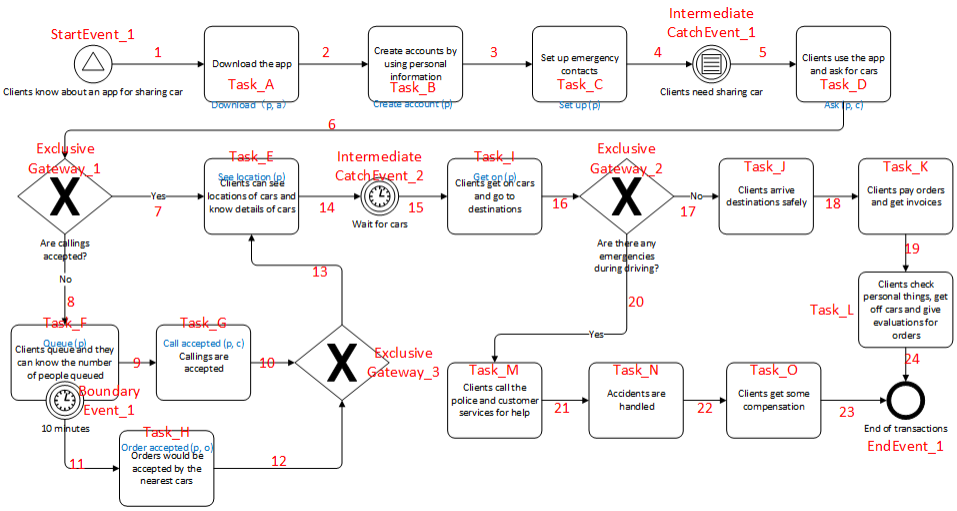
```
5            </Atom>
6        </body>
7        <head>
8            <Behaviour>
9                <Obligation>VerifyQualification</Obligation>
10               <Obligation>InformResult</Obligation>
11           </Behaviour>
12       </head>
13   </Imp>
```

This rule describes the driver's behavior after passing a review. Driver: ThroughReview⊢OL Test
Driver: Test⊢OL Grade
Driver: Grade⊢OL ObtainLicense
Merge:
D: ThroughReview⊢OL Test⊗OL Grade⊗OL ObtainLicense

```
1    <Imp label='D'>
2        <body>
3            <Atom>
4                ThroughReview
5            </Atom>
6        </body>
7        <head>
8            <Behaviour>
9                <Obligation>Test</Obligation>
10               <Obligation>Grade</Obligation>
11               <Obligation>ObtainLicense</Obligation>
12           </Behaviour>
13       </head>
14   </Imp>
```

2 In the process of customers (Lu Xiong)

2.1 BPMN



2.2 XML for BPMN

```
<bpmn:process id="Process" isExecutable="true">
        <bpmn:task id="Task_A" name="Task_A">
                <bpmn:incoming>SequenceFlow_1</bpmn:incoming>
                <bpmn:outgoing>SequenceFlow_2</bpmn:outgoing>
        </bpmn:task>
        <bpmn:task id="Task_B" name="Task_B">
                <bpmn:incoming>SequenceFlow_2</bpmn:incoming>
                <bpmn:outgoing>SequenceFlow_3</bpmn:outgoing>
        </bpmn:task>
        ...
        <bpmn:sequenceFlow id="SequenceFlow_1" sourceRef="StartEvent_1"
             targetRef="Task_A" />
        <bpmn:sequenceFlow id="SequenceFlow_2" sourceRef="Task_A"
            targetRef="Task_B" />
        <bpmn:sequenceFlow id="SequenceFlow_7" sourceRef="
            ExclusiveGateway_1" targetRef="Task_E">
                <bpmn:conditionExpression
                 xsi:type="bpmn:tFormalExpression">Yes</
                    bpmn:conditionExpression>
        </bpmn:sequenceFlow>
        ...
        <bpmn:startEvent id="StartEvent_1">
                <bpmn:outgoing>SequenceFlow_1<bpmn:outgoing>
                <bpmn:signalEventDefinition>
        <bpmn:startEvent />
        <bpmn:boundaryEvent id="BoundaryEvent_1" attachedToRef="Task_F"
            >
                <bpmn:outgoing>SequenceFlow_11</bpmn:outgoing>
                <bpmn:timerEventDefinition />
        </bpmn:boundaryEvent>
        <bpmn:endEvent id="EndEvent_1">
                <bpmn:incoming>SequenceFlow_23</bpmn:incoming>
                <bpmn:incoming>SequenceFlow_24</bpmn:incoming>
        </bpmn:endEvent>
        <bpmn:exclusiveGateway id="ExclusiveGateway_1">
                <bpmn:incoming> SequenceFlow_6</bpmn:incoming>
                <bpmn:outgoing> SequenceFlow_7</bpmn:outgoing>
                <bpmn:outgoing> SequenceFlow_8</bpmn:outgoing>
        </bpmn:exclusiveGateway>
</bpmn:process>
```

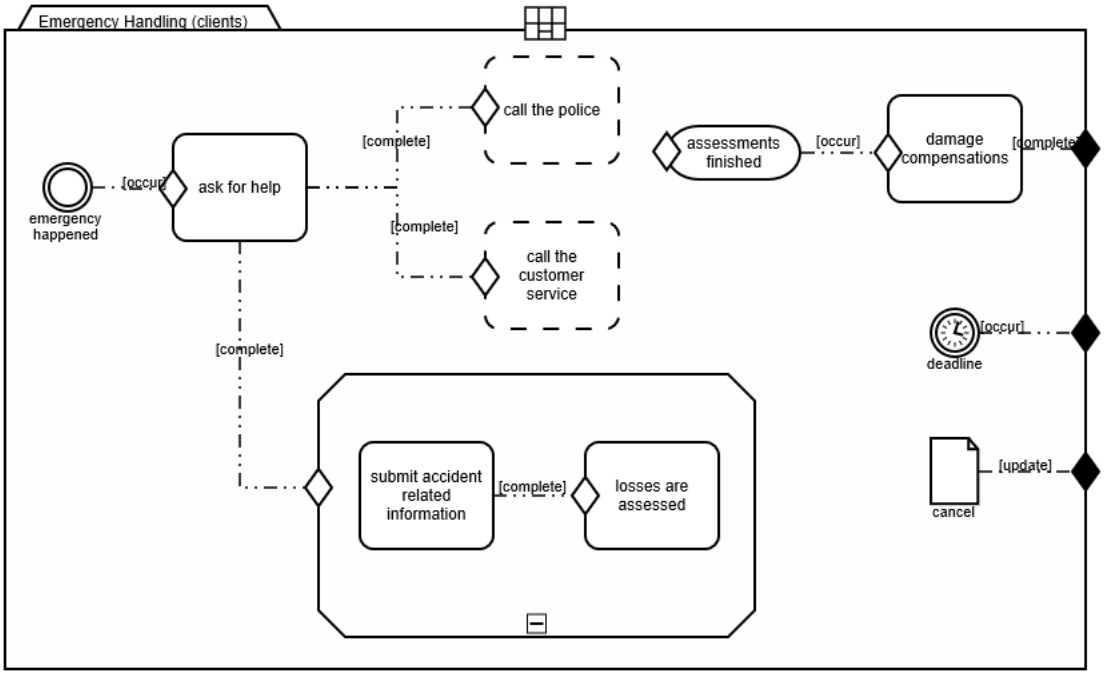
2.3 Semantic Effect Annotation

The states are marked in the BPMN picture.
Cumulative Effect Scenarios:
TE: Scenario(1): ⟨ ⟨TA,TB,TC,TD,{TE}⟩,{ TA,TB,TC,TD,TF,TG } ⟩
TE: Scenario(2): ⟨ ⟨TA,TB,TC,TD,{TF,TG},TE⟩,{ TA,TB,TC,TD,} ⟩
TI: Scenario(1): ⟨ ⟨TA,TB,TC,TD,{TE},TI⟩,{ TA,TB,TC,TD,TF,TG } ⟩
TI: Scenario(2): ⟨ ⟨TA,TB,TC,TD,{TF,TG},TI⟩,{ TA,TB,TC,TD,TE } ⟩

TJ: Scenario(1): ⟨ ⟨TA,TB,TC,TD,{TE},TI,TJ⟩,{ TA,TB,TC,TD,TF,TG,TM} ⟩
TJ: Scenario(2): ⟨ ⟨TA,TB,TC,TD,{TF,TG},TI,TJ⟩,{ TA,TB,TC,TD,TE,TM } ⟩
TM: Scenario(1): ⟨ ⟨TA,TB,TC,TD,{TE},TI,TM⟩,{ TA,TB,TC,TD,TF,TG,TJ} ⟩
TM: Scenario(2): ⟨ ⟨TA,TB,TC,TD,{TF,TG},TI,TM⟩,{ TA,TB,TC,TD,TE,TJ } ⟩
Cumulative Effects of Tasks/Activities:
Ask for cars: download(p,a) ⋀ creat account(p) ⋀ setup(p) ⋀ ask(p,c)
Callings are accepted: download(p,a) ⋀ creat account(p) ⋀ setup(p) ⋀ ask(p,c) ⋀ queue(p) ⋀ Call accepted(p,c)
See location: download(p,a) ⋀ creat account(p) ⋀ setup(p) ⋀ ask(p,c) ⋀ ((queue(p) ⋀ Call accepted(p,c) ⋀ see location(p)) ⋁ see location(p))

2.4 CMMN

The picture describes the emergency handling of clients.



2.5 DMN

This DMN describes how the passenger chooses which car to ride during the car-hailing process.

asking car

`decision_ask-car`

U	Input +			Output +
	people number	comfort	carpooling	Vehicle type
	integer	string	string	string
1	≥5	-	-	MPV
2	-	engoyment	-	Luxury Cars
3	-	-	Yes	Saloon car
4	-	comfortable	-	SUV

2.6 Process Mining

The process mining is about the situations of clients after driving. Whether the client is safety or not.
Execution Log:
case 1:task I
case 2:task I
case 1:task J
case 2:task M
case 1:task K
case 2:task N
case 1:task L
case 2:task O

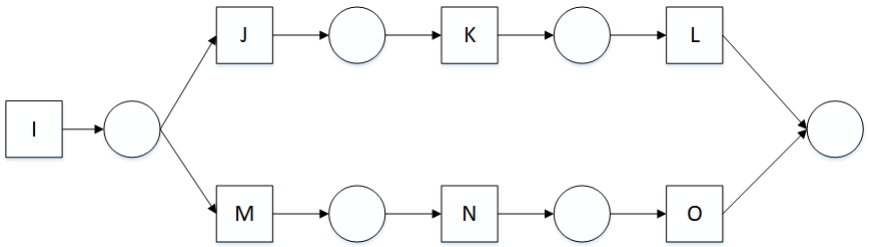
Direct Succession: I>J I>M J>K M>N K>L N>O
W={IJKL, IMNO} T={I, J, K, L, M, N, O}
T_I={I} T_O={L, O}
Causality: I→J I→M J→K M→N K→L N→O
Concurrency:
Choice: I#K I#L J#L I#N I#O M#O J#M J#N J#O K#M K#N K#O L#M L#N L#O

	I	J	K	L	M	N	O
I	#	→	#	#	→	#	#
J	←	#	→	#	#	#	#
K	#	←	#	→	#	#	#
L	#	#	←	#	#	#	#
M	←	#	#	#	#	→	#
N	#	#	#	#	←	#	→
O	#	#	#	#	#	←	#

Table 2. Footprint Table

X_L={({I},{J}),({I},{M}),({J},{K}),({M},{N}),({K},{L}),({N},{O}),({I},{J,M})}
Y_L={({J},{K}),({M},{N}),({K},{L}),({N},{O}),({I},{J,M})}

2.7 Petri Net From Process Mining



2.8 Business Process Compliance

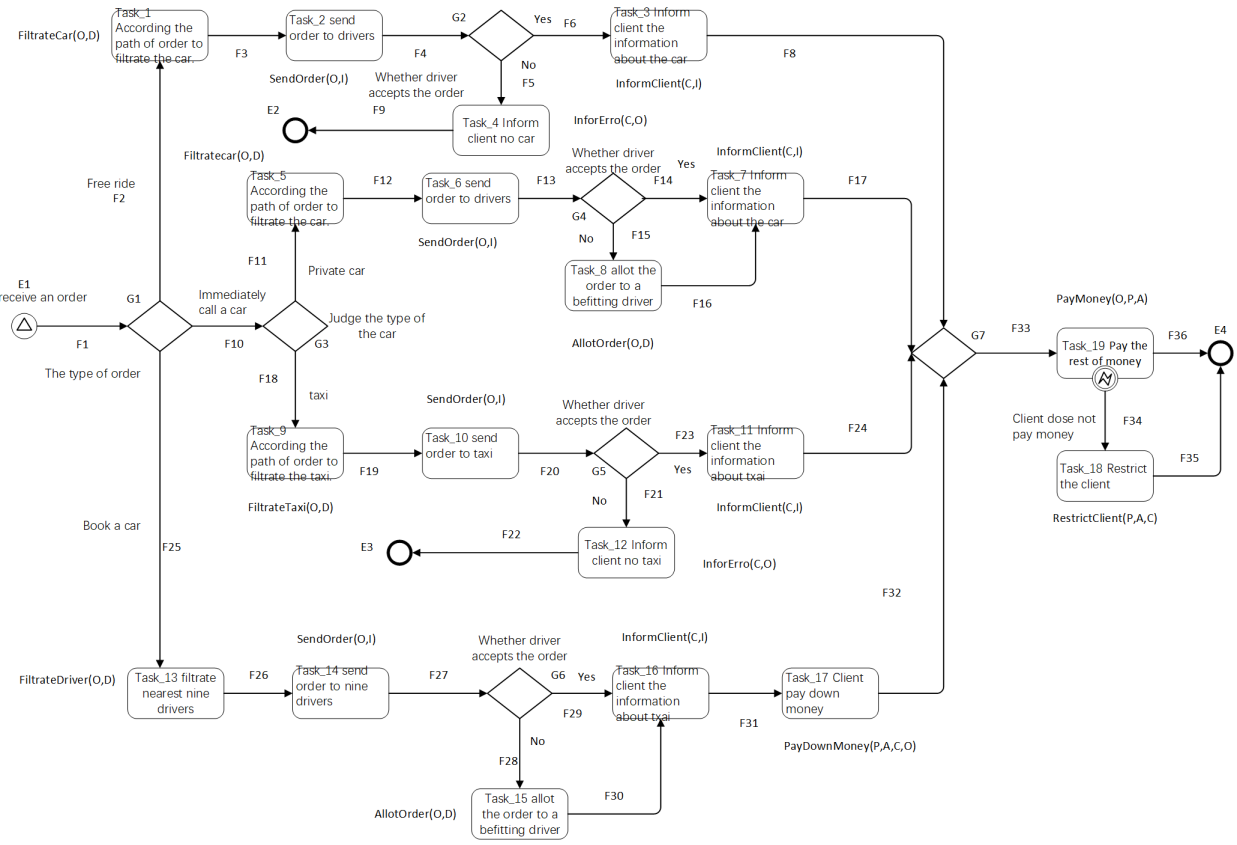
This part describes the logic of what clients will do when they are calling for cars.
Client, CallCars $\vdash$ OL OrderReceived
$\neg$ OrderReceived $\vdash$ OL Queue
Merge:
r: Client, CallCars $\vdash$ OL OrderReceived $\otimes$ OL Queue

2.9 RuleML

```
<Imp label='r'>
        <body>
                <And>
                        <Atom>
                        Client
                        <\Atom>
                        <Atom>
                        CallCars
                        <\Atom>
                <\And>
        </body>
        <head>
                <Behaviour>
                        <Obligation>OrderReceived</Obligation>
                        <Obligation>Queue</Obligation>
                </Behaviour>
        </head>
</Imp>
```

3 In the process of the allotment of the order (Bing Xia)

3.1 BPMN



3.2 XML for BPMN

```
<bpmn:process id="Process_order" isExecutable="true">
        <bpmn:task id="Task_1" name="Task_1">
                <bpmn:incoming>F2</bpmn:incoming>
                <bpmn:outgoing>F3</bpmn:outgoing>
        </bpmn:task>
        <bpmn:task id="Task_2" name="Task_2">
                <bpmn:incoming>F3</bpmn:incoming>
                <bpmn:outgoing>F4</bpmn:outgoing>
        </bpmn:task>
        ...
        <bpmn:sequenceFlow id="F1" sourceRef="E1" targetRef="G1" />
        <bpmn:sequenceFlow id="F2" sourceRef="G1" targetRef="Task_1" />
        <bpmn:sequenceFlow id="F3" sourceRef="Task_1" targetRef="Task_2
            " />
        ...
        <bpmn:startEvent id="E1">
                <bpmn:outgoing></bpmn:outgoing>
                <bpmn:siganlEventDefinition/>
        </bpmn:startEvent>
```

```
19          <bpmn:boundaryEvent id="boundaryEvent_1">
20                  <bpmn:outgoing>F34</bpmn:outgoing>
21                  <bpmn:errorEventDefinition/>
22          </bpmn:boundaryEvent>
23          <bpmn:exclusiveGateway id="G1">
24                  <bpmn:incoming>F1</bpmn:incoming>
25                  <bpmn:outgoing>F2</bpmn:outgoing>
26                  <bpmn:outgoing>F10</bpmn:outgoing>
27                  <bpmn:outgoing>F25</bpmn:outgoing>
28          </bpmn:exclusiveGateway>
29          ...
30          <bpmn:endEvent id="E2">
31                  <bpmn:outgoing>F9</bpmn:outgoing>
32                  <bpmn:endEventDefinition/>
33          </bpmn:endEvent>
34  </bpmn:process>
```

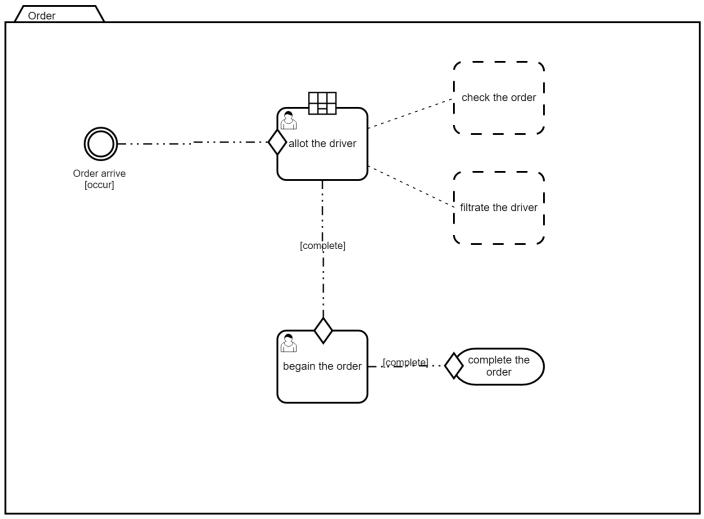
3.3 Semantic Effect Annotation

Cumulative Effect Scenarios:
-Task_1:Scenarios:$<< Task\_1 >>$
-Task_2:Scenarios:$<< Task\_1, Task\_2 >>$
-Task_3:Scenarios:$<< Task\_1, Task\_2, Task\_3 >>$
-Task_4:Scenarios:$<< Task\_1, Task\_2, Task\_4 >>$
-Task_5:Scenarios:$<< Task\_5 >>$
-Task_5:Scenarios:$<< Task\_5, Task\_6 >>$
-Task_8:Scenarios:$<< Task\_5, Task\_6, Task\_8 >>$
-Task_7:Scenarios:$<< Task\_5, Task\_6.Task\_7 >, \{< Task\_5, Task\_6, Task\_8 >\} >$
$<< Task\_5, Task\_6, Task\_8, Task\_7 >>$
-Task_9:Scenarios:$<< Task\_9 >>$
-Task_10:Scenarios:$<< Task\_9, Task\_10 >>$
-Task_11:Scenarios:$<< Task\_9, Task\_10, Task\_11 >>$
-Task_12:Scenarios:$<< Task\_9, Task\_10, Task\_12 >>$
-Task_13:Scenarios:$<< Task\_13 >>$
-Task_14:Scenarios:$<< Task\_13, Task\_14 >>$
-Task_15:Scenarios:$<< Task\_13, Task\_14, Task\_15 >>$
-Task_16:Scenarios:$<< Task\_13, Task\_14, Task\_16 >, \{< Task\_13, Task\_14, Task\_15 >\} >$
$<< Task\_13, Task\_14, Task\_15, Task\_16 >>$
Cumulative Effects of Tasks/Activities:
Obiect:
Order-O
DataBase-D
Client-C
InformationOfDrive-I
Payment-P
Account-A
States:
Task_1 According the path of order to filtrate the car.- FiltrateCar(O,D)
Task_2 send order to drivers- SendOrder(O,I)
Task_3 Inform client the information about the car- InformClient(C,I)
Task_4 Inform client no car- InforErro(C,O)

Task_5 According the path of order to filtrate the car.- Filtratecar(O,D)
Task_6 send order to drivers- SendOrder(O,I)
Task_7 Inform client the information about the car- InformClient(C,I)
Task_8 allot the order to a befitting driver- AllotOrder(O,D)
Task_9 According the path of order to filtrate the taxi.- FiltrateTaxi(O,D)
Task_10 send order to taxi- SendOrder(O,I)
Task_11 Inform client the information about txai- InformClient(C,I)
Task_12 Inform client no taxi- InforErro(C,O)
Task_13 filtrate nearest nine drivers- FiltrateDriver(O,D)
Task_14 send order to nine drivers- SendOrder(O,I)
Task_15 allot the order to a befitting driver- AllotOrder(O,D)
Task_16 Inform client the information about txai- InformClient(C,I)
Task_17 Client pay down money- PayDownMoney(P,A,C,O)
Task_19 Pay the rest of money- PayMoney(O,P,A)
Task_18 Restrict the client- RestrictClient(P,A,C)
Effect:
Task_1= $FiltrateCar(O, D)$
Task_2= $FiltrateCar(O, D) \wedge SendOrder(O, I)$
Task_3= $FiltrateCar(O, D) \wedge SendOrder(O, I) \wedge InformClient(C, I)$
Task_4= $FiltrateCar(O, D) \wedge SendOrder(O, I) \wedge InforErro(C, O)$
Task_5= $Filtratecar(O, D)$
Task_6= $Filtratecar(O, D) \wedge SendOrder(O, I)$
Task_7= $Filtratecar(O, D) \wedge SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I)$
Task_8= $Filtratecar(O, D) \wedge SendOrder(O, I) \wedge AllotOrder(O, D)$
Task_9= $FiltrateTaxi(O, D)$
Task_10= $FiltrateTaxi(O, D) \wedge SendOrder(O, I)$
Task_11= $FiltrateTaxi(O, D) \wedge SendOrder(O, I) \wedge InformClient(C, I)$
Task_12= $FiltrateTaxi(O, D) \wedge SendOrder(O, I) \wedge InforErro(C, O)$
Task_13= $FiltrateDriver(O, D)$
Task_14= $FiltrateDriver(O, D) \wedge SendOrder(O, I)$
Task_15= $FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge AllotOrder(O, D)$
Task_16= $FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I)$
Task_17=$FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I) \wedge$
$PayDownMoney(P, A, C, O)$
Task_18=$((FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge InformClient(C, I)) \vee ((Filtratecar(O, D) \wedge$
$SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I)) \vee (FiltrateTaxi(O, D) \wedge SendOrder(O, I) \wedge$
$InformClient(C, I))) \vee (FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I) \wedge$
$PayDownMoney(P, A, C, O))) \wedge PayMoney(O, P, A) \wedge RestrictClient(P, A, C) \wedge RestrictClient(P, A, C)$
Task_19=$((FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge InformClient(C, I)) \vee ((Filtratecar(O, D) \wedge$
$SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I)) \vee (FiltrateTaxi(O, D) \wedge SendOrder(O, I) \wedge$
$InformClient(C, I))) \vee (FiltrateDriver(O, D) \wedge SendOrder(O, I) \wedge (AllotOrder(O, D)) \wedge InformClient(C, I) \wedge$
$PayDownMoney(P, A, C, O))) \wedge PayMoney(O, P, A)$

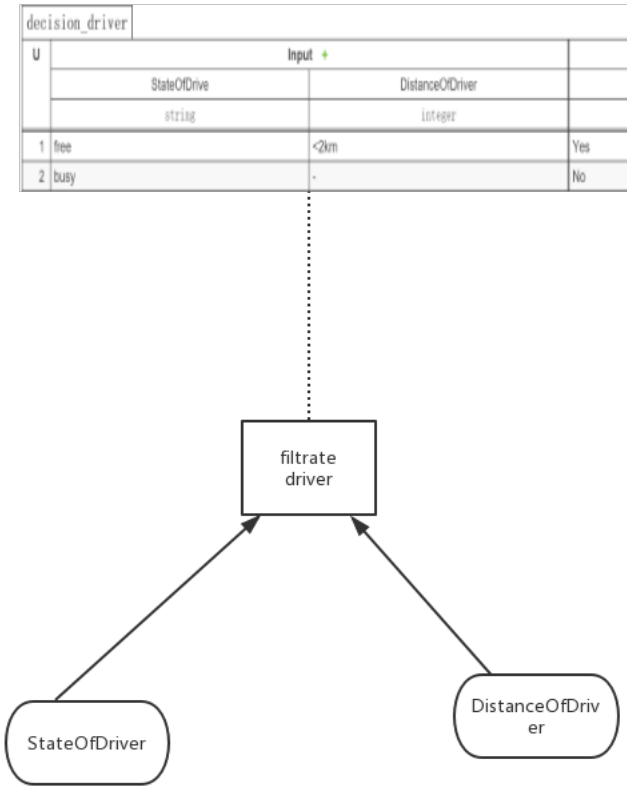
3.4 CMMN

This CMMN describes the process of how to complete an order when the client calls an immediate car.



3.5 DMN

This DMN describes how to filtrate the applicable driver to complete the order.



3.6 Process Mining

This part is a process mining about the order that the client books a car.
Execution Log:
case 1:task A
case 2:task A
case 1:task B
case 2:task B
case 1:task D
case 2:task C
case 1:task E
case 2:task D
case 2:task E

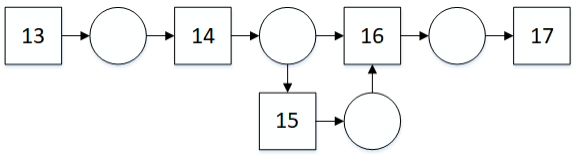
W={ABDE, ABCDE}[<13,14,16,17>,<13,14,15,16,17>] T={13, 14, 15, 16, 17}
T_I={13} T_O={17}
Direct Succession: 13>14 14>16 16>17 14>15 15>16
Causality: 13→14 14→16 16→17 14→15 15→16
Concurrency:
Choice: 13#15 13#16 13#17 14#17 15#17

	13	14	15	16	17
13	#	→	#	#	#
14	←	#	→	→	#
15	#	←	#	→	#
16	#	←	←	#	→
17	#	#	#	←	#

Table 3. Footprint Table

X_L={({13},{14}),({14},{16}),({16},{17}),({14},{15}),({15},{16})}
Y_L={({13},{14}),({14},{16}),({16},{17}),({14},{15}),({15},{16})}

3.7 Petri Net From Process Mining and RuleML



3.8 Business Process Compliance

When an order arrives, if no drivers receive the order, the system will allot a driver to receive the order.
When an order is complete, the client must pay money, otherwise, the account of the client will be restricted.
$Order \vdash OBLDriverReceive \otimes OBLAllotDriver$
$Client, OrderComplete \vdash OBLPayment \otimes OBLRestrictClient$

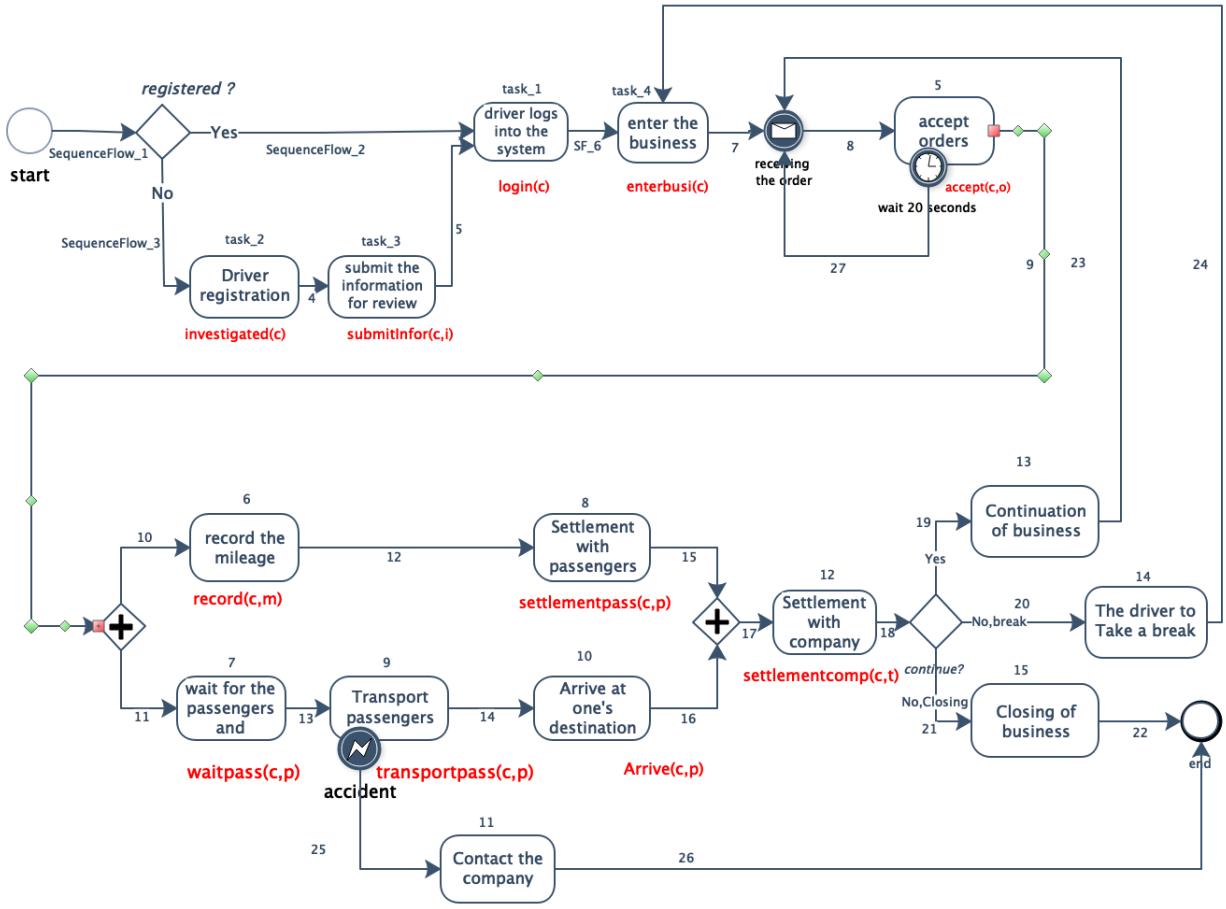
```
<Imp label='r1'>
        <body>Order</body>
        <head>
                <Behaviour>
                        <Obligation>DriverReceive</Obligation>
                        <Obligation>AllotDriver</Obligation>
                </Behaviour>
        </head>
</Imp>
<Imp label='r2'>
        <body>
                <And>
                        <Atom>
                        Client
                        <\Atom>
                        <Atom>
                        OrderComplete
                        <\Atom>
                <\And>
        </body>
        <head>
                <Behaviour>
                        <Obligation>Payment</Obligation>
                        <Obligation>RestrictClient</Obligation>
                </Behaviour>
        </head>
</Imp>
```

4 In the process of drivers (Yunbo Zhang)

4.1 BPMN

This BPMN describes the driver's process. The first part is qualification certification and the second part is order confirmation. The third part is that the driver picks up the passenger after confirming the order and records the mileage. After finishing this order, the driver can choose whether to continue the business or not. In case of any accident in the delivery process, the driver

needs to contact the company to deal with it.



4.2 XML for BPMN

```
1  <bpmn:process id="Process_1"> isExecutable="true">
2          <bpmn:startEvent id="startEvent_1">
3                  <bpmn:outgoing>SequenceFlow_1</bpmn:outgoing>
4                  <bpmn:conditionalStartDefinition />
5          </bpmn:startEvent>
6          <bpmn:SequenceFlow id="SequenceFlow_1" sourceRef="StartEvet_1"
               targetRef="exclusiveGateway_1">
7          <bpmn:exclusiveGateway id="exclusiveGateway_1" name="
               regisitered?">
8                  <bpmn:incoming>SequenceFlow_1</bpmn:incoming>
9                  <bpmn:outgoing>SequenceFlow_2</bpmn:outgoing>
10                 <bpmn:outgoing>SequenceFlow_3</bpmn:outgoing>
11         </bpmn:exclusiveGateway>
12         <bpmn:SequenceFlow id="SequenceFlow_2" sourceRef="
               exclusiveGateway_1" targetRef="Task_1">
13         <bpmn:SequenceFlow id="SequenceFlow_3" sourceRef="
               exclusiveGateway_1" targetRef="Task_2">
14          ...
```

```
15          <bpmn:task id="Task_14" name="The_driver_to_take_break">
16                  <bpmn:incoming>SequenceFlow_20</bpmn:incoming>
17                  <bpmn:outgoing>SequenceFlow_24</bpmn:outgoing>
18          </bpmn:task>
19          <bpmn:task id="Task_15" name="Closing_of_business">
20                  <bpmn:incoming>SequenceFlow_21</bpmn:incoming>
21                  <bpmn:outgoing>SequenceFlow_22</bpmn:outgoing>
22          </bpmn:task>
23          ...
24          <bpmn:SequenceFlow id="SequenceFlow_23" sourceRef="Task_13"
                targetRef="IntermediateCatchEvent_1" />
25          <bpmn:SequenceFlow id="SequenceFlow_24" sourceRef="Task_14"
                targetRef="Task_4" />
26          <bpmn:SequenceFlow id="SequenceFlow_22" sourceRef="Task_15"
                targetRef="endEvent_1" />
27  </bpmn:process>
```


4.3 Semantic Effect Annotation

The states are marked in the BPMN picture.
Cumulative Effect Scenarios:
t13: Scenario(1): ⟨ ⟨ t1,t4,t5,{ ⟨t6,t8⟩,⟨t7,t9,t10⟩ },t12,t13 ⟩, { ⟨t2, t3,⟩ } ⟩

t13: Secnario(2): ⟨ ⟨ t2,t3,t1,{⟨t6,t8⟩,⟨t7,t9,t10⟩ }⟩,t12,t13⟩ ⟩

t14: Secnario(1): ⟨ ⟨ t1,t4,t5,{⟨t6,t8⟩,⟨t7,t9,t10⟩ }⟩,t12,t14⟩, { ⟨t2, t3,⟩ } ⟩

t14: Secnario(2): ⟨ ⟨ t2,t3,t1,{⟨t6,t8⟩,⟨t7,t9,t10⟩ }⟩,t12,t14⟩ ⟩

t15: Secnario(1): ⟨ ⟨ t1,t4,t5,{⟨t6,t8⟩,⟨t7,t9,t10⟩ } ⟩,t12,t15⟩, { ⟨t2, t3,⟩ } ⟩

t15: Secnario(2): ⟨ ⟨ t2,t3,t1,{⟨t6,t8⟩,⟨t7,t9,t10⟩ } ⟩,t12,t15⟩ ⟩ Cumulative Effects of Tasks/Activities:
enter business: (login(c) ⋁ investigated(c) ⋀ submitInfor(c,i)) ⋀enterbusi(c)

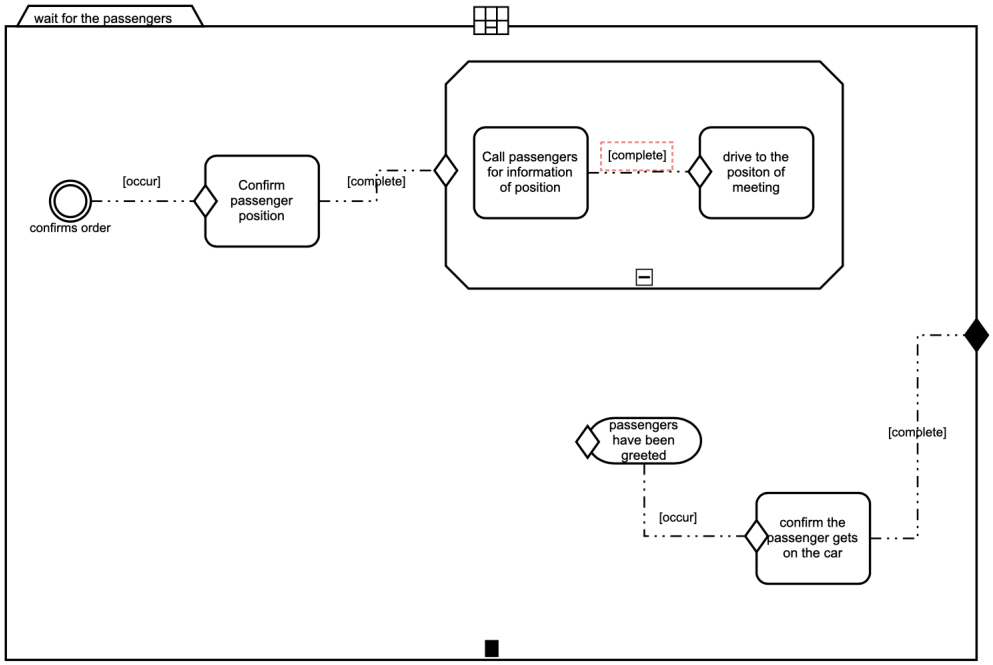
accept order: (login(c) ⋁ investigated(c) ⋀ submitInfor(c,i)) ⋀enterbusi(c) ⋀ accept(c,o)

settlement with company: (login(c) ⋁ investigated(c) ⋀ submitInfor(c,i)) ⋀enterbusi(c) ⋀ accept(c,o) ⋀ ((record(c,m)⋀ settlementpass(c,p)) ⋁(waitpass(c,p)) ⋀transportpass(c,p)⋀Arrive(c,p)) ⋀settlementcomp(c,t)

4.4 CMMN

This picture is about the subprocess of drivers waiting for the passengers. First, the driver needs to confirm passengers' position, there is a stage to describe the process. It is necessary to connect with passengers by phone to get information about the position, then the driver drives to the position. It is a milestone of meeting the passenger, the next task is to confirm the passengers have got in

the car.



4.5 DMN

This DMN describes the situation of the driver accepting the order. If the order distance is greater than 2 km, the driver would refuse. If the order shows bad passenger credit, the driver would refuse. If the order shows more than 5 passengers, the driver would refuse. If the order is greater than 50km, the driver would refuse. If the order is greater than 50km, but the driver received a surcharge, the driver would accept. If the order shows good passenger credit, the driver would accept.

Order

`decision_order`

View DRD

U	Input +					Output +
	distance	credit	people number	destination	surcharge	order
	integer	string	integer	integer	integer	string
1	≥2	-	-	-	-	Refuse
2	-	bad	-	-	-	Refuse
3	-	-	≥5	-	-	Refuse
4	-	-	-	≥50	-	Refuse
5	-	-	-	≥50	≥5	Accept
6	≥2	good	-	-	-	Accept

4.6 Process Mining

Execution Log:

case 1:task 5	case 2:task 5	case 3:task 5	case 4:task 5	case 5:task 5
case 1:task 6	case 2:task 7	case 3:task 7	case 4:task 7	case 5:task 7
case 1:task 7	case 2:task 6	case 3:task 9	case 4:task 9	case 5:task 9
case 1:task 9	case 2:task 9	case 3:task 6	case 4:task 6	case 5:task 10
case 1:task 8	case 2:task 8	case 3:task 8	case 4:task 10	case 5:task 6
case 1:task 10	case 2:task 10	case 3:task 10	case 4:task 8	case 5:task 8
case 1:task 12	case 2:task 12	case 3:task 12	case 4:task 12	case 5:task 12

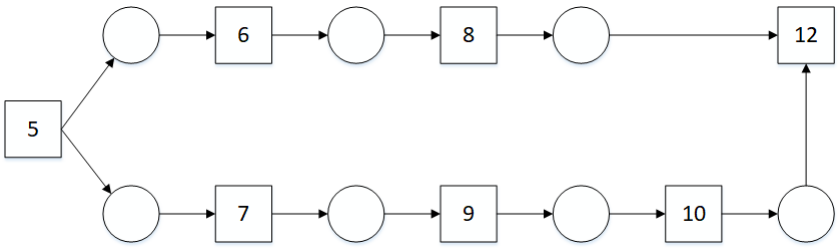
	5	6	7	8	9	10	12
5	#	→	→	#	#	#	#
6	←	#	‖	→	‖	‖	#
7	←	‖	#	→	#	#	#
8	#	←	#	#	#	‖	→
9	#	‖	←	#	#	→	#
10	#	‖	#	‖	←	#	→
12	#	#	#	←	#	←	#

Table 4. Footprint Table

W=[<5,6,7,9,8,10,12>,<5,7,6,9,8,10,12>,<5,7,9,6,8,10,12>,<5,7,9,6,10,8,12>,<5,7,9,10,6,8,12>]
T={5,6,7,8,9,10,12}
T_I={5} T_O={12}
Direct Succession: 5>6 5>7 6>7 6>8 6>9 6>10 7>6 7>9 8>10 8>12 9>8 9>6 9>10 10>6 10>8 10>12
Causality: 5→6 5→7 6→8 7→9 8→12 9→10 10→12 9→8
Concurrency: 6‖7 7‖6 6‖9 9‖6 6‖10 10‖6 8‖10 10‖8
Choice: 5#8 5#9 5#10 5#12 6#12 7#8 7#10 7#12 9#12
X_L={({5},{6}),({5},{7}),({6},{8}),({7},{9}),({8},{12}),({9},{10}),({10},{12}),({9},{8})}
Y_L={({5},{6}),({5},{7}),({6},{8}),({7},{9}),({8},{12}),({9},{10}),({10},{12}),({9},{8})}

4.7 Petri Net From Process Mining

This Petri Net describes the process of picking up the passenger and record the mileage until the settlement.



4.8 Business Process Compliance and RuleML

After receiving an order, it is obligatory to comfirm for the driver. If the driver does not confirm the order, the driver has to wait for the next order.
If the driver confirms an order, it is obligatory for the driver to pick up the passenger.
r:DriverReceiveOrder ⊢ OBL ConfirmOrder⊗ OBL WaitNextOrder
v:DriverConfirmOrder ⊢ OBL PickUpPassengers

```
<Imp label='r'>
        <body>
                DriverReceiveOrder
        </body>
        <head>
                <Behaviour>
                        <Obligation>ConfirmOrder</Obligation>
                        <Obligation>WaitNextOrder</Obligation>
                </Behaviour>
        </head>
</Imp>
<Imp label 'v'>
        <body>
                DriverConfirmOrder
        </body>
        <head>
                <Behaviour>
                        <Obligation>PickUpPassengers</Obligation>
                </Behaviour>
        </head>
</Imp>
```