Information Management I - Group Project Report
Group 18 - Car Insurance Broker

1. Introduction
As part of this project, we were tasked with creating some XML documents describing some classes from our class diagram. We also had to create interesting XQuery queries based of our XML documents based of the use case diagrams which we made in the previous project.

2. Did we make any design changes to our UML diagrams to suit the XML format?
No, we found that the design of our UML diagrams fit well into our XML design and as a result, no changes were made to the diagrams when writing our DTD and XML files.

3. Who did what
After having presented our UML diagrams, we met up the next week and divided the work between the members of the group that were constantly present in our weekly meetings. We then met up the next week with the DTDs and XML files done, that is when we divided the work for the different queries amongst ourselves.

Samuel Petit : I personally worked on the Vehicle DTD and XML file. I also wrote both the queries that use built in xquery functions.

Everlyn Poh : I worked on the Company XML and DTD files and wrote 3 of the interlinked XQueries.

Conor Ryan: I worked on the Claim and QuoteSystem XML and DTD files and wrote the user-defined xquery.

Kamil Przepiórowski : I worked on the Driver XML and DTD files, and wrote the “penalty points info” xquery.

Sean Roche : I worked on the Policy XML and DTD files and wrote the xquery which uses the LET function. I also assessed and documented the strengths of XML and Xquery design.

Con Óg Ó Laoghaire: I wrote Customer Support.xml and analysed and wrote the weaknesses of our XML approach

4. XML and DTD files

a. Claims

- The DTD file:

```
<!ELEMENT claim (privateInformation)>
<!ATTLIST claim
  xmlns CDATA #FIXED ''
  type CDATA #REQUIRED>

<!ELEMENT privateInformation (name,synopsis,driverName,driverID,
                              thirdPartyInsurer)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT synopsis (#PCDATA)>

<!ELEMENT driverName (title,firstName,surname)>

<!ELEMENT driverID (#PCDATA)>

<!ELEMENT thirdPartyInsurer (#PCDATA)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT firstName (#PCDATA)>

<!ELEMENT surname (#PCDATA)>
```

- The XML file:

```
<?xmlversion = "1.0"?>
<!DOCTYPE claim SYSTEM "claim.dtd">
<claim type ="3">
  <privateInformation>
    <name> Crash </name>
    <synopsis>
      "Third party driver drove into the
      back of claimant's vehicle on the 3rd November"</synopsis>
      <driverName>
        <title>Mr.</title>
        <firstName>John</firstName>
        <surname>Doe</surname>
      </driverName>
      <driverID>1234567</driverID>
      <thirdPartyInsurer>AXA</thirdPartyInsurer>
  </privateInformation>
</claim>
```

b. Company

- The DTD file :

```
<!DOCTYPE company [
<!ELEMENT company (branch*)>
<!ATTLIST company type CDATA #IMPLIED>
<!ATTLIST company name CDATA #IMPLIED>
<!ATTLIST company country CDATA #IMPLIED>
<!ELEMENT branch (local_branch*)>
<!ATTLIST branch county CDATA #IMPLIED>
<!ELEMENT local_branch (branch_details,department+)>
<!ATTLIST local_branch area CDATA #IMPLIED>
<!ELEMENT branch_details (address,phone,email,manager)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
<!ELEMENT department (employee*)>
<!ATTLIST department name CDATA #IMPLIED>
<!ELEMENT employee (firstname,lastname,title)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT title (#PCDATA)>

]>
```

- The XML file

```
<?xml version="1.0"?>
<company type="Insurance" name="AXA" country="Ireland">
   <branch county="Dublin">
      <local_branch area="City Centre">
        <branch_details>
           <address>Wolfe Tone House Wolfe Tone Street Dublin 1</address>
           <phone>01 872 6444</phone>
           <email>axa.dublincity@axa.ie</email>
           <manager>Mark Shields</manager>
        </branch_details>
        <department name="Car Insurance">
           <employee>
              <firstname>John</firstname>
              <lastname>Smith</lastname>
              <title>Customer Support Manager</title>
           </employee>
           <employee>
              <firstname>Connail</firstname>
              <lastname>O'Donnell</lastname>
              <title>Receptionist</title>
           </employee>
        </department>
      </local_branch>
      <local_branch area="Blanchardstown">
         <branch_details>
            <address>Unit 4, Blanchardstown Plaza,Main Street</address>
            <phone>01 8179733</phone>
            <email>axa.blanchardstown@axa.ie</email>
            <manager>Tony Higgins</manager>
         </branch_details>
         <department name="Car Insurance">
            <employee>
              <firstname>Nicole</firstname>
              <lastname>Smith</lastname>
              <title>Department Manager</title>
            </employee>
         </department>
      </local_branch>
   </branch>
</company>
```

c. Customer Support

- The DTD file:

```
<!ELEMENT CustomerSupport (employee,query)>
<!ELEMENT employee (employee.personalInfo,employee.contactDetails,
                    employee.businessRecord)>
<!ELEMENT query (customer)>
<!ATTLIST query criteria CDATA #IMPLIED>
<!ELEMENT employee.personalInfo (employee.firstName,employee.lastName,
                                 employee.DOB)>
<!ELEMENT employee.contactDetails (employee.phoneNumber,employee.email)>
<!ELEMENT employee.businessRecord (employee.speciality,
                                   employee.dateJoined,
                                   employee.remarks)>
<!ELEMENT customer (customer.personaInfo,customer.history)>
<!ELEMENT employee.firstName (#PCDATA)>
<!ELEMENT employee.lastName (#PCDATA)>
<!ELEMENT employee.DOB (#PCDATA)>
<!ELEMENT employee.phoneNumber (#PCDATA)>
<!ELEMENT employee.email (#PCDATA)>
<!ELEMENT employee.speciality (#PCDATA)>
<!ELEMENT employee.dateJoined (#PCDATA)>
<!ELEMENT employee.remarks (#PCDATA)>
<!ELEMENT customer.personaInfo (customer.firstName,customer.lastName,
                                customer.DOB)>
<!ELEMENT customer.history (customer.previousQuerys,customer.plan)>
<!ELEMENT customer.firstName (#PCDATA)>
<!ELEMENT customer.lastName (#PCDATA)>
<!ELEMENT customer.DOB (#PCDATA)>
<!ELEMENT customer.previousQuerys (customer.previousQuerys.date,
                                   customer.previousQuerys.notes,
                                   customer.previousQuerys.resolution)>
<!ELEMENT customer.plan (customer.plan.type,customer.plan.expiry)>
<!ELEMENT customer.previousQuerys.date (#PCDATA)>
<!ELEMENT customer.previousQuerys.notes (#PCDATA)>
<!ELEMENT customer.previousQuerys.resolution (#PCDATA)>
<!ELEMENT customer.plan.type (#PCDATA)>
<!ELEMENT customer.plan.expiry EMPTY>
```

- The XML file:

```
<!DOCTYPE CustomerSupport SYSTEM "CustomerSupport.dtd">
<CustomerSupport insuranceCompany="AXA">
  <employee>
→  <employee.personalInfo>
→→<employee.firstName>John</employee.firstName>
→→<employee.lastName>Smith</employee.lastName>
→→<employee.DOB>05/06/1973</employee.DOB>
→  </employee.personalInfo>
→  <employee.contactDetails>
→→  <employee.phoneNumber>123456789</employee.phoneNumber>
→→  <employee.email>johnsmith@gmail.com</employee.email>
→  </employee.contactDetails>
→  <employee.businessRecord>
→→  <employee.speciality> Handling angry customers </employee.
speciality>
→→  <employee.dateJoined> 10/11/2018 </employee.dateJoined>
→→  <employee.remarks> vegetarian </employee.remarks>
→  </employee.businessRecord>
  </employee>
  <query criteria="policy enquiry">
→<customer>
→    <customer.personaInfo>
    →→<customer.firstName>Nicola</customer.firstName>
    →→<customer.lastName>Barry</customer.lastName>
    →→<customer.DOB> 05/06/1973 </customer.DOB>
→    </customer.personaInfo>
→  <customer.history noClaimsBonus="no">
→→<customer.previousQuerys>
→→→<customer.previousQuerys.date> asd</customer.previousQuerys.date>
→→→<customer.previousQuerys.notes> asda</customer.previousQuerys.
notes>
→→→<customer.previousQuerys.resolution>asd </customer.previousQuerys.
resolution>
→→</customer.previousQuerys>
→→<customer.plan>
→→→<customer.plan.type>asd </customer.plan.type>
→→→<customer.plan.expiry> </customer.plan.expiry>
→→</customer.plan>
→ </customer.history>
→</customer>
  </query>
</CustomerSupport>
```

d. Driver

- The DTD file :

```
<!ELEMENT Driver (person*)>
<!-- there can be none, one, or multiple drivers in the class -->

<!ELEMENT person (personalInfo,penaltyPoints,contactDetails)>
<!ATTLIST person age CDATA #IMPLIED>
<!ATTLIST Driver idNumber CDATA #IMPLIED>

<!ELEMENT personalInfo (firstName,lastName,DOB)>
<!-- a driver has to provide one and only one value for each -->
<!ELEMENT penaltyPoints (#PCDATA)>

<!ELEMENT contactDetails (phoneNumber+,email+)>
<!-- a driver has to provide one phone number, but may provide more, same for the email -->

<!--  every driver has a first name, last name, date of birth (DOB), a phone number & email -->
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT DOB (#PCDATA)>
<!ELEMENT phoneNumber (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

- The XML file :

```
<?xml version="1.0"?>
<!DOCTYPE Driver SYSTEM "driver.dtd">
<Driver idNumber='1234567'>
  <person age='45'>
→  <personalInfo>
→→<firstName>John</firstName>
→→<lastName>Doe</lastName>
→→<DOB>05/06/1973</DOB>
→  </personalInfo>
→  <penaltyPoints>69</penaltyPoints>
→  <contactDetails>
→→  <phoneNumber>123456789</phoneNumber>
→→  <email>johndoe@gmail.com</email>
→  </contactDetails>
  </person>
</Driver>
```

e. Policy

- The DTD file:

```
<!-- Defines the Policy element-->
<!DOCTYPE Policy [
<!ELEMENT policy (customer*)>
<!ATTLIST policy type CDATA #IMPLIED>

<!-- Customer details -->
<!ELEMENT customer(personalInfo,contactDetails,policyInfo)>
<!ATTLIST customer idNumber CDATA #IMPLIED>

<!-- The insured driver's details which appear on the policy document -->
<!ELEMENT personalInfo(firstName,lastName,DOB,penaltyPoints)>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT DOB (#PCDATA)>
<!ELEMENT penaltyPoint (#PCDATA)>

<!ELEMENT contactDetails(phoneNumber,email)>
<!ELEMENT phoneNumber(#PCDATA)>
<!ELEMENT email(#PCDATA)>

<!-- Details of the policy cover-->
<!ELEMENT policyInfo (cover,startDate,endDate)>
<!ATTLIST policyInfo id CDATA #IMPLIED>
<!ELEMENT cover (#PCDATA)>
<!ELEMENT startDate (#PCDATA)>
<!ELEMENT endDate (#PCDATA)>

]>
```

- The XML file:

```
<?xml version="1.0"?>
<!DOCTYPE Policy SYSTEM "policyDTD.dtd">

  <policy type="car insurance">
    <customer idNumber="1234567">
      <personalInfo>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <DOB>1973-05-06</DOB>
        <penaltyPoints>69</penaltyPoints>
      </personalInfo>
      <contactDetails>
        <phoneNumber>123456789</phoneNumber>
        <email>johndoe@gmail.com</email>
      </contactDetails>
      <policyInfo id="5989">
        <cover>third party</cover>
        <startDate>2018-11-15</startDate>
        <endDate>2019-11-19</endDate>
      </policyInfo>
    </customer>
  </policy>
```

f. Vehicle

- The DTD file :

```
<!-- Defines the Vehicle element-->
<!DOCTYPE Vehicle [
<!ELEMENT Vehicle (Owner+, Dealer, brand, colour+, model, year, registration)>
<!-- a vehicle has one single brand, name (model), year and registration and is bought from one dealer,
it can however have multiple owners/drivers & multiple colours -->
<!ATTLIST Vehicle id CDATA #IMPLIED>
<!-- we give each vehicle an id in our database -->

<!ELEMENT Owner (Driver)>
<!-- The owner element is imported from the driver.dtd, so the same attributes/elements are used -->

<!ELEMENT Dealer (Name, Address, phone+)>
<!ATTLIST Dealer id CDATA #IMPLIED>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!-- elements /attributes for  the dealer: ever dealer has an id (in our DB) & every single one of
its elements is unique (name, address). It can however have multiple phone numbers -->

<!ENTITY % driver SYSTEM "driver.dtd">
%driver;

<!ELEMENT brand (#PCDATA)>
<!ELEMENT colour (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT registration (#PCDATA)>
<!-- defining the rest of the elements used for a vehicle : brand, model, year, registration, colour -->
]>
```

- The Vehicle XML file :

```
<?xml version="1.0"?>
<!DOCTYPE Vehicle SYSTEM "vehicleDTD.dtd">
<!-- Define new vehicle-->
<Vehicle id="50">
    <!-- first owner -->
    <Owner>
        <Driver idNumber="233">
            <person age='38'>
                <personalInfo>
                    <firstName> Eve </firstName>
                    <lastName> McDonald </lastName>
                    <DOB> 10/10/1980 </DOB>
                </personalInfo>
                <penaltyPoints> 75 </penaltyPoints>
                <contactDetails>
                    <phoneNumber> 0872661773 </phoneNumber>
                    <email> eve-mcdonald@gmail.com </email>
                </contactDetails>
        </person>
        </Driver>
    </Owner>
    <!-- second owner -->
    <Owner>
        <Driver idNumber="40">
            <person age='33'>
                <personalInfo>
                    <firstName> Mike </firstName>
                    <lastName> McDonald </lastName>
                    <DOB> 05/05/1985 </DOB>
                </personalInfo>
                <penaltyPoints> 20 </penaltyPoints>
                <contactDetails>
                    <phoneNumber> 07221188227 </phoneNumber>
                    <email> mike-mcdonald@gmail.com </email>
                </contactDetails>
            </person>
        </Driver>
    </Owner>
    <!-- unique information -->
    <Dealer id="124">
    <Name> Fred's Auto Shop</Name>
    <Address> 15 Dame Street, Dublin</Address>
    <phone> 0812049281</phone>
    </Dealer>

    <brand> Peugeot </brand>
    <model> 206 </model>
    <year> 2010 </year>
    <registration> 99-D-123456 </registration>
</Vehicle>
```

g. Quote System

- The DTD file:

```
<!DOCTYPE quoteSystem [

<!ELEMENT quoteSystem (website,input*)>

<!ELEMENT website (#PCDATA)>

<!ELEMENT input (vehicle,registration,driver*)>

<!ELEMENT vehicle (#PCDATA)>

<!ELEMENT registration (#PCDATA)>

<!ELEMENT driver (name*,dateOfBirth,firstReceivedLicense)>

<!ELEMENT name (title, firstName, surname)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT firstName (#PCDATA)>

<!ELEMENT surname (#PCDATA)>

<!ELEMENT dateOfBirth (#PCDATA)>

<!ELEMENT firstReceivedLicense (#PCDATA)>

]>
```

- The XML file:

```
<?xmlversion = "1.0"?>
<!DOCTYPE quoteSystem SYSTEM "quoteSystem.dtd">
<quoteSystem>
  <website>"https://www.axa.ie"</website>
  <input>
    <vehicle>"Opel Corsa"</vehicle>
    <registration>"03-D-3434"</registration>
    <driver>
      <name>
        <title>Mr.</title>
        <firstName>John</firstName>
        <surname>Doe</surname>
      </name>
      <dateOfBirth>01/01/1996</dateOfBirth>
      <firstReceivedLicense>04/12/2017</firstReceivedLicense>
    </driver>
  </input>
</quoteSystem>
```

5. XQueries

a. Query 1 (Interlinked)

```
for $w in doc("company.xml")/company/branch/local_branch/department/employee ,
$x in doc("Customer Support.xml")/CustomerSupport/employee

where $w/firstname = $x/employee.personalInfo/employee.firstName

return <employeeContactDetails>
  {$w/firstname}
  {$w/lastname}
  {$x/employee.contactDetails/employee.phoneNumber}
  {$x/employee.contactDetails/employee.email}
</employeeContactDetails>
```

- Identification of UML Case that it supports : Company class and Customer Support Class.
- Purpose of Query : To retrieve contact details of an employee from another interlinked XML file by searching their name.
- Example of output when query is executed :

```
<employeeContactDetails>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
  <employee.phoneNumber>123456789</employee.phoneNumber>
  <employee.email>johnsmith@gmail.com</employee.email>
</employeeContactDetails>
```

b. Query 2 (Interlinked)

```
for $w in doc("claim.xml")/claim ,
$x in doc("driver.xml")/Driver

where $w/privateInformation/driverName/firstName = $x/person/personalInfo/firstName

return
<claimInformation>
  {$x/person/personalInfo/firstName}
  {$x/person/personalInfo/lastName}
  {$w/privateInformation/name}
  {$w/privateInformation/synopsis}
  {$w/privateInformation/driverID}
</claimInformation>
```

- Identification of UML Case that it supports : Claim class and Driver Class.
- Purpose of Query : To retrieve claim information of a driver from another interlinked XML file by searching the driver's name.
- Example of output when query is executed :

```
<claimInformation>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
  <name>Crash</name>
  <synopsis>"Third party driver drove into the
      back of claimant's vehicle on the 3rd November"</synopsis>
  <driverID>1234567</driverID>
</claimInformation>
```

c. Query 3 (Interlinked)

```
for $w in doc("driver.xml")/Driver ,
$x in doc("policy.xml")/policy

where $w/@idNumber = $x/customer/@idNumber

return
<PolicyInformation>
  {$w/@idNumber}
  {$x/customer/policyInfo/cover}
  {$x/customer/policyInfo/startDate}
  {$x/customer/policyInfo/endDate}
</PolicyInformation>
```

- Identification of UML Case that it supports : Policy class and Driver Class.
- Purpose of Query : To retrieve policy information of a driver from another interlinked XML file by searching the driver's customer ID number.
- Example of output when query is executed :

```
<PolicyInformation idNumber="1234567">
  <cover>third party</cover>
  <startDate>15/11/18</startDate>
  <endDate>14/11/19</endDate>
</PolicyInformation>
```

d. FOR query

```
for $s in doc("vehicle.xml")/Vehicle/Owner/Driver/person
where $s/penaltyPoints = 75 (: 75 used as random example :)

return <details>
  {$s/personalInfo}
</details>

(: returns driver information with x penalty points :)
```

- Identification of UML Case that it supports : Registering / Getting a quote
- Purpose of Query : To retrieve the information of a Driver with x amount of penalty points.
- Example of output when query is executed : If there exists a driver in the database, with x penalty points, the output will be:

```
<details>
  <personalInfo>
    <firstName>Eve</firstName>
    <lastName>McDonald</lastName>
    <DOB>10/10/1980</DOB>
  </personalInfo>
</details>
```

- Otherwise, the query will not return anything as the driver doesn't exist.

e. LET query

```
let $s := "2018-11-15"
let $e := "2019-11-19"

for $p in doc("policy.xml")/policy/customer/policyInfo/startDate
for $q in doc ("policy.xml")/policy/customer/policyInfo/endDate
where $s = $p and $e = $q
return
$p and $q
```

- This query may be useful when a claim is being made to check its validity. It could be implemented to ensure that the policy is active and the vehicle is covered for the the date of the claim or incident.
- Therefore, this query supports our UML Use case #1 which was “Making a Claim”.
- This query has two potential outputs; true or false.

f. Built in function query #1

```
(:  Get Vehicle Data :)
for $c in doc("vehicle.xml")
(: here "3456" would be user input
This method may be useful for customer support to identify a customer :)
where ends-with($c/Vehicle/registration, "3456")
return <RegistrationData>
  (:return the actual registration :)
  {$c/Vehicle/registration}
</RegistrationData>
```

- This query may be useful for customer support when dealing with a customer. The employee may ask for the customer to provide certain information to verify that he is indeed who he claims to be before processing the customers query.
- As a result of the above explanation, this query illustrates part of our UML Use case #1 which was “Making a Claim”. But it also is relevant for our 3rd UML Use case : “Customer Service”.
- Here is a sample output when this specific query is executed:

```
<RegistrationData>
  <registration>99-D-123456</registration>
</RegistrationData>
```

g. Built in function query #2

```
(: use data from Policy :)
for $c in doc("policy.xml"),
(:define $x to be used for WHERE condition & return data from $c:)
$x in $c/policy/customer/policyInfo/endDate

(:check from policies expiring in the next 3 months:)
where xs:date($x) < current-date() + xs:yearMonthDuration("P30M")

(:return personal data as well as contact details to get in touch with newer policy:)
return <data>
  {$c/policy/customer/personalInfo/firstName}
  {$c/policy/customer/personalInfo/lastName}
  {$c/policy/customer/contactDetails/phoneNumber}
  {$c/policy/customer/contactDetails/email}
</data>
```

- This query would be used for renewing customer policies before they expire. It returns data and contact details from every customer with policies expiring in the next 3 months from when the query is executed. A computer system could make use of this query, run it once a day or so and get in touch with those customers the query returns.
- As a result of the above explanation, this query is useful for our 2nd UML Use case : Insurance Renewal System.
- Here is a sample output of this query :

```
<data>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
  <phoneNumber>123456789</phoneNumber>
  <email>johndoe@gmail.com</email>
</data>
```

h. User defined function query

```
declare function local:findRegisteredDrivers()
{
for $s in doc("vehicle.xml")/Vehicle
where $s/@id = "50"

return <details>
  {$s/Owner/Driver/person}
</details>
};
<Driver>{local:findRegisteredDrivers()}</Driver>
(:Returns the details and contact info of drivers registered to any particular
vehicle based on the in-system vehicle id:)
```

- This query could be used for finding the drivers on any vehicle that is involved in a claim, or if a driver's policy may be changing.
- Example of output:

```
<Driver>
  <details>
    <person age="38">
      <personalInfo>
        <firstName>Eve</firstName>
        <lastName>McDonald</lastName>
        <DOB>10/10/1980</DOB>
      </personalInfo>
      <penaltyPoints>75</penaltyPoints>
      <contactDetails>
        <phoneNumber>0872661773</phoneNumber>
        <email>eve-mcdonald@gmail.com</email>
      </contactDetails>
    </person>
    <person age="33">
      <personalInfo>
        <firstName>Mike</firstName>
        <lastName>McDonald</lastName>
        <DOB>05/05/1985</DOB>
      </personalInfo>
      <penaltyPoints>20</penaltyPoints>
      <contactDetails>
        <phoneNumber>07221188227</phoneNumber>
        <email>mike-mcdonald@gmail.com</email>
      </contactDetails>
    </person>
  </details>
</Driver>
```

6. Strengths and Weaknesses of the XML design and the XQueries design.

Strengths

- Relatively straightforward to write, it is much like HTML however it has more advanced linking abilities and does not use acronyms which increases it's understandability.
- It is easily extendable.
- It is essentially a human language as opposed to a computer language and therefore can be easily understood by anyone.
- Allows for quick prototyping.
- XQueries allow for quick lookup of data in a database in a relatively simple manner

Weaknesses

- Information portrayed in a very raw manner ie. impossible to write or test without knowledge of XML and XPath
- Layout style dependant on who is writing it
- Can be overwhelming to someone who isn't familiar with it