Deployment of a Contract Compliant Checker for Executable Contracts : (User's Guide ver 1.2, 10 Oct 2018)

Carlos Molina–Jimenez[1] and Ioannis Sfyrakis[2]

[1] Computer Laboratory, University of Cambridge,
carlos.molina@cl.cam.ac.uk
[2] School of Computing Science, Newcastle University, UK,
Ioannis.Sfyrakis@newcastle.ac.uk

Abstract. This document is a walk through description of the deployment of version 1.2 of the Contract Compliant Checker (CCC). With respect to the 15 Feb 2018 release, this document has less spelling mistakes, no fundamental changes are included.
The CCC is a software tool whose implementation started at Newcastle University UK around 2003 and is under further development.
The CCC can be used for monitoring and enforcing **executable contracts** (now called **smart contracts**) that business partners use to regulate their business interactions. Examples of such interactions are contractual agreements signed between buyers and sellers of goods and contractual agreements signed between providers of computing services and their consumers. The CCC is a centralised smart contract enforcer in the sense that it is deployed on a Trusted Thrid Party.
Version 1.1 of the CCC was released in 2012 and was implemented as a web server in Java with Red Hat Drools and Red Hat jboss Application Server 7 (AS 7). The current version (version 1.2) uses jboss EPA (Enterprise Application Platform) instead of AS 7.
In both versions, the CCC is loaded with a set of ECA rules that represent the contractual clauses of the contract under monitoring and deployed as a web server within a trusted third party or within one of the business partners.
Being a web server, i) the CCC listens to events (RESTful messages) produced by the application under monitoring, ii) processes them using its ECA rules and iii) produces a response (a RESTful message) indicating whether a given event was found to be either contract compliant or non–contract compliant.
This document describes the installation procedure of version 1.2. It is aimed at potential users interested in locally deploying the CCC after downloading it from a public repository such as GitHub where it appears as the *carlos–molina/tecomate* project and trying it by means of running the provided examples. The old *carlos–molina/conch* repository stores version 1.1 of the CCC. Also, the reader might like to have a look at the *carlos–molina/contraval* repository which hosts a contract validator directly related to the CCC.
The installation discussed were conducted on a MacBook Air computer running macOS High Sierra version 10.13.2, it involes only free licence

software available from the Internet and takes about 60 min to execute. We expect that users of Windows and Linux should be able to deploy and run the CCC after minor adjustments.

1 Introduction

In our research [1], we define an **executable contract** as a conventional contract that can be converted into executable code, executed and enforced programmatically at run–time. Observe that currently, the term **smart contract** suggested in [2] is now widely accepted to refer to what we called executable contracts in our papers.

To detect potential violation of an executable contract, we need tools to analyse the event produced from the execution of the contract. The CCC that we discuss in this document was implemented to adress this issue.

The CCC is a software tool whose implementation started at Newcastle University UK. Version 1.1 of the CCC was released in 2012 and was implemented as a web server in Java with Red Hat Drools and Red Hat jboss Application Server 7 (AS 7). The current version (version 1.2) uses jboss EPA (Enterprise Application Platform) instead of AS 7.

It can be deployed as a contract monitor or alternatively, as a contract enforcer, By *monitor* we mean that the CCC acts as a passive observer of the interaction whereas by *enforcer* we mean that the CCC actively interferes with the interaction to prevent business partner to execute contractually illegal actions.

In both deployments, the CCC is provided with the set of Event Condition Action rules (ECA rules) that represent the contractual clauses of the contract of interest and deployed as a web service. It can be physically deployed withing a trusted third party or within one of the business partners. Its job is to listen to and process events and determine if the business partners are observing their contract clauses. We will use two examples to explain the operation of the CCC.

This user's guide is organised as follows. Section 2 explains that the CCC can be deployed to perform monitoring (passive observer) or as an enforcer that interfers with the business process. Section 3 presents an abstract view of its architecture. The material included in these sections is also included in Sections 1 and 2 of the User's Guide of version 1.1 [3]. Section 4 and subsequent sections discuss the particularities of the deployment of the current version.

2 Monitoring and enforcement of contracts

The CCC can be deployed under different configurations. For example, it can be deployed as a monitor that passively listen to the business events exchanged by the contractual parties, analyses them and sends notfications about detected non–contract compliance events to the parties interested in. Another alternative is to deploy the CCC between the contractual parties to actively intercep the business events, analyse them and take some actions on events found to be

non–contract compliant. A potential action is to prevent non–contract compliant events from reaching their intended target. Subsections 2.2 and 2.3 discuss motivating examples of these two alternatives.

2.1 Centralised vs blockchain–based descentralised approaches

It is worth emphasising that the version 1.2 of the CCC follows a centralised approach. This implies that a single instance of the sotware of the executable contract is deployed within a Trusted Third Party (TTP) that is responsible for hosting and running it. The advantage of using a centralised approach is simplicity. The executable contract is essentially a single Finite State Machine (FSM) that has unique global view of the development of the contractual interactiuon and it current state. Thus the CCC is free from issues related to consensus. The disadvantage of this approach is that it suffers from all the issues (for example, single point of failure, bottle–necks, etc.) that afflict centralised systems. In contrast, in the descentralised approach alternative, several replicas (each running a local FSM) of the executable contract are deployed on different computers. To progress from a given state S_i to next state (let us say S_{i+1}) all the FSM involved need to reach consensus over the value of S_{i+1}. Consensus is not a trivial issue to solve and require the execution of sophisticated algorithms. To addree consensus and other issues, designers of executable contracts following the distributed approach can use the consensus middleware services offered by blockchain technology [4].

2.2 Monitoring Example

Let us assume that a buyer and store have agreed to trade under the following contract. This contract example is oversimplified and incomplete, yet it it good enough for explaining our ideas.

1. *The buyer can place a* ***buy request*** *with the store to buy an item.*
2. *The store is obliged to respond with either* ***buy confirmation*** *or* ***buy rejection*** *within 3 days of receiving the buy request.*
   (a) *No response from the store within 3 days will be treated as a buy rejection.*
3. *The buyer can either* ***pay*** *or* ***cancel*** *the buy request within 7 days of receiving a confirmation.*
   (a) *No response from the buyer within 7 days will be treated as a cancellation.*

Imagine that the two business partners decide to monitor their contractual interaction. A typical deployment of the CCC for addressing this question is shown in Fig. 1.

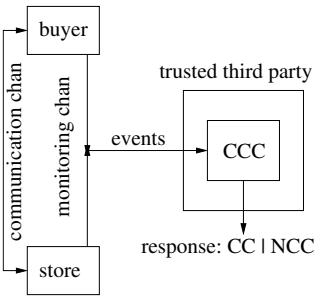

Fig. 1. The CCC deployed as a contract monitor.

In the figure, *buyer* and *store* represent the two parties involved in the contract. The *trusted third party* is a third party that operates the CCC which is assumed to be loaded with the ECA rules that represent the contractual clauses. As shown in the figure, the business partners use a communication channel (*communication chan*) for exchanging their business messages. In addition they use a monitoring channel (*monitoring chan*) for notifying events of interest to the CCC. Examples of *events* are events that notify of the execution of a contractual business operation such as the execution of a buy request operation by buyer or the execution of a confirmation operation by the store. Upon receiving an event(for example, *BuyRequest*), the CCC processes it to determine if the event is contract compliant (CC) or non–contract compliant (NCC). The results (*response: CC | NCC*) is sent to interested in parties such as the business partners.

2.3 Enforcement Example

Imagine service providers (providers for short) that offers services to clients under the stipulation of a contract. As a more specific example, let us think of a provider that sells pre–paid cards to clients that grant access to its service N (for example, five) times. Naturally, such a provider would need to deploy a mechanism to allow legal request reach its service and reject illegal ones (those that exceed the agreed number).

An potential solution to this problem is shown in Fig. 2, where the *client* and *provider* represent the business partners.

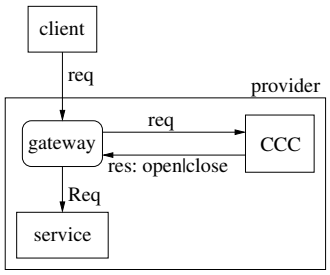

Fig. 2. The CCC deployed as an enforcer.

In this scenario, the the CCC is deployed as an enforcer—it opens or closes the *gateway* that grants access to the *service*.

1. The client sends a request (*req*) to the gateway.
2. The gateway intercepts *req* and forwards it to the CCC which is loaded with the ECA rules that represent the contract between the client and provider.
3. The CCC processes *req* and determines if the client has not exceeded yet his prepaid access (five requests in this example).
4. If *req* is declared legal by the CCC, it responds with *open*, otherwise it produces *close*.
5. The gateway forwards the request to the service only when the CCC responds with *open*.

3 Abstract Architecture of the CCC

We have implemented the CCC as a RESTful web service. Fig. 3 shows an abstract view of its architecture. In this section we will present and overview of the functionality of its components. Details about their implementations, deployments and configurations will be presented in subsequent sections.

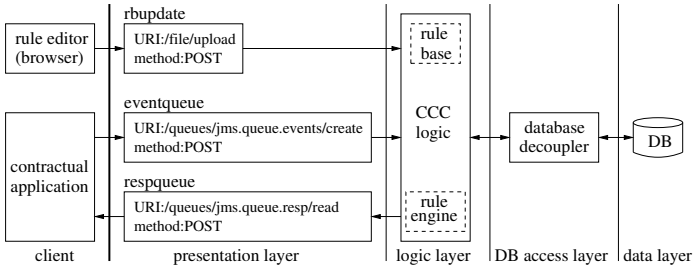

Fig. 3. Abstract architecture of the CCC.

As shown in the figure, conceptually, the CCC consists of four layers (**presentation, logic, DB access** and **data** layers) and is expected to interact with external entities that are represented by a **client** tier.

Client: The *client* represents the external entity to the CCC and consists of a *rule editor* (for example, a browser) and a *contractual application*. The *rule editor* is used by rule administrators for updating the rule base of the CCC. It offers editing facilities and means for sending the edited file to the CCC as a conventional HTTP POST request. The *contractual application* represents the contractual application under monitoring or enforcement. For instance, in Figs. 1, the *contractual application* corresponds to buyer and store. Similarly, regarding Fig. 2, the *contractual application* corresponds to the client.

Presentation layer: The CCC interacts with external entities through its presentation layer which we have implemented as three RESTful endpoints:

- A *rbupdate* (rule base update) point that accepts POST request sent by administrator to update the current rule base of the CCC.
- An *eventqueue* that accepts and stores events produced by the *contractual application* and sent as RESTful POST requests. Examples of events produced by the buyer–store contract example would be *BuyReq*, *BuyConf* and *BuyPay* that correspond, respectively, to the execution of buy request, buy confirmation and payment operations. To support portability of events, the *eventqueue* accepts events tagged with XML tags. For example, the *BuyConf* and *BuyPay* events are expected to be formatted as follows:

```
<event>
 <originator>store</originator>
 <responder>buyer</responder>
 <type>BuyConf</type>
 <status>success</status>
</event>

<event>
 <originator>buyer</originator>
 <responder>store</responder>
 <type>BuyPay</type>
 <status>success</status>
</event>
```

  The *originator* specifies the business partner that initiated the execution of the operation; likewise, *responder* specifies the business partner that responded to the operation; finally, *status* specifies the outcome of the operation (we will elaborate on this parameter later). Thus the *BuyConf* event notifies that the execution of a buy request operation was originated by the store, responded by the buyer and completed in success. Similarly, the *BuyPay* event notifies that the execution of a payment operation completed in success and was originated by the buyer, responded by the store.
- An *respqueue* (response queue) where the CCC stores the results (contract compliant or non contract compliant) of the evaluation of the events. To support portability of results, the CCC produces results tagged with XML tags like in the following two examples:

```
<result>
 <contractcompliance>true</contractcompliance>
</result>

<result>
 <contractcompliance>false</contractcompliance>
</result>
```

The first example is the response to an event that was declared contract compliant (*true*) by the CCC. In contrast, the second example is the response to an event declared non contract compliant (*false*) by the CCC.

Logic layer: The *Logic layer* is represented by the *CCC logic* which consist of a *rule base*, *rule engine* and ancillary Java classes (not shown in the figure). The *rule base* represents the ECA rules that encode the contractual clauses. The *rule engine* represents the rule engine (for example, Drools engine) that upon arriving of events, triggers the execution of the corresponding rules.

DB layer: The *DB layer* represents a data base that is used by the CCC for storing permanent records (for example, events notified to the CCC) about the development of the contractual interaction.

DB access layer: The *DB access layer* is represented by a *database decoupler*. Its job is to hide from the designer the details of the communication between the CCC and the particular database technology used.

The functionality of the CCC as a web service can be summarised as follows:

1. The CCC retrieve and event from the *eventqueue*, sent by the *contractual application*.
2. The *rule engine* of the CCC processes the event with the help of the rules in the *rulebase*.
3. The CCC produces a response (RESTful message) that indicates if the event is contract compliant or not, and enqueues in the *respqueue*.

4 Deployment of Components

The deployment of the CCC is platform independent. The functionality of the current version has been tested in a MacBook Air computer running macOS High Sierra version 10.13.2 with 8GB of memory and 1.7 GHz Intel Core i7 cpu. We will use this settings in our discussions.

The distribution software includes two independent working folders that can be downloaded from GitHub [5] (*carlos–molina/conch* project). To be independent from specific Integrated Development Environments (IDE), we follow manual deployment. We rely on command line commands issued from terminals running bash shells, as opposed to using eclipse [6] like in the deployment of version 1.1 [3].

The architecture to operate the CCC is shwon in Fig. 3 and consists of two interactive modules. In the figure, they are separated by the thickest vertical line: on the left side is the client while the actuall CCC is on the right side. In the git repository the code of these two modules is as follows:

- *CCCRest–ear*: contains the software related to the presentation, logic, DB access and data layers.
- *CCCRestClient*: contains the software related to the client.

To deploy the CCC, we recommend the following prodecure:

4.1 Clone the conch repository

We assume that you have Git running on your local computer and a git account. If you are new to Git, go to [5].

Onoce you are set up to run git, proceed to clone the conch repository `https://github.com/carlos-molina/conch`

A clone of the local version will produce the following files:

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch

bash-3.2$ ls -l
total 108288
drwxr-xr-x   6 carlosmolina  staff       192 22 Feb 00:44 CCCRest-ear/
drwxr-xr-x   5 carlosmolina  staff       160 23 Feb 20:11 CCCRest-ear-commons/
drwxr-xr-x   6 carlosmolina  staff       192 23 Feb 20:11 CCCRest-ear-ear/
drwxr-xr-x   6 carlosmolina  staff       192 23 Feb 20:11 CCCRest-ear-ejb/
drwxr-xr-x   5 carlosmolina  staff       160 23 Feb 20:11 CCCRest-ear-web/
drwxr-xr-x  15 carlosmolina  staff       480 23 Feb 22:15 CCCRestClient/
drwxr-xr-x   7 carlosmolina  staff       224 23 Feb 19:58 JBOSS-EAP-6.4 configuration/
-rw-r--r--   1 carlosmolina  staff      1051 22 Feb 14:37 README.md
-rw-r--r--   1 carlosmolina  staff        87 22 Feb 00:44 README.me
-rw-r--r--@  1 carlosmolina  staff    875138 22 Feb 00:44 UsersGuide.pdf
-rw-r--r--   1 carlosmolina  staff  54246846 24 Feb 01:56 drools.log
drwxr-xr-x   6 carlosmolina  staff       192 22 Feb 00:44 example contracts/
-rw-r--r--   1 carlosmolina  staff      3562 22 Feb 00:44 install.md
-rwxr-xr-x   1 carlosmolina  staff      1213 22 Feb 00:44 logging.properties*
-rw-r--r--   1 carlosmolina  staff     10649 22 Feb 00:44 pom.xml
-rw-r--r--   1 carlosmolina  staff       168 22 Feb 00:44 run.bat
-rwxr-xr-x   1 carlosmolina  staff       112 23 Feb 19:55 run.sh*
bash-3.2$
```

4.2 Install MySql

The CCC needs a data base for permanently storing records about the contractual interaction.

Free versions of MySQL data base servers can be downloaded from [7].

The current version uses a MySql 5.6 data base which we downloaded from
`https://dev.mysql.com/downloads/mysql/5.6.html#downloads`

Installation instructions for MacOS are detailed at
`https://dev.mysql.com/doc/refman/5.6/en/osx-installation-pkg.html`

A successful installation of the MySql server should result in the screen shown in Fig. 4 produced from System Preferences of your Mac.

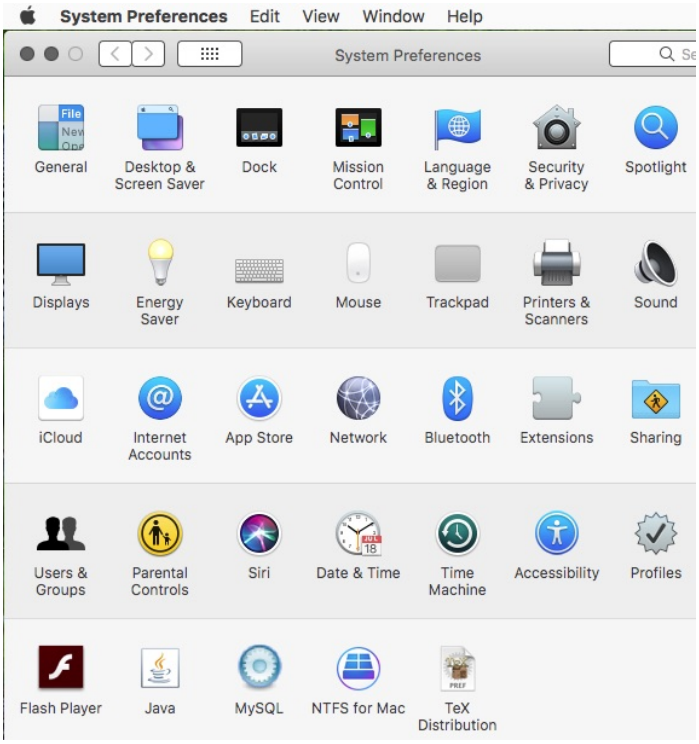

Fig. 4. Deployment of MySql server.

As shown in Fig. 5, the server can be started and stopped.

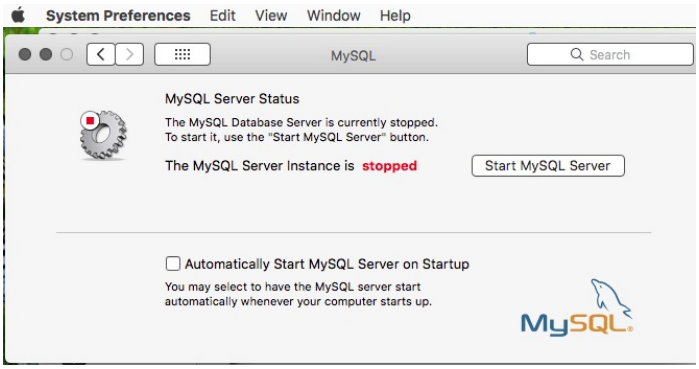

Fig. 5. Start and stop MySql server.

Once the MySql Server is deployed you need to create a database and initialise it with a database with corresponding tables and a user. To administer the MySql server we dowloaded MySQLWorkbench for Mac from `https://dev.mysql.com/downloads/workbench/`.

Fig. 6 shows a screen shot of MySqlWorkbench. The MySql server was previously started as suggested in Fig. 5.

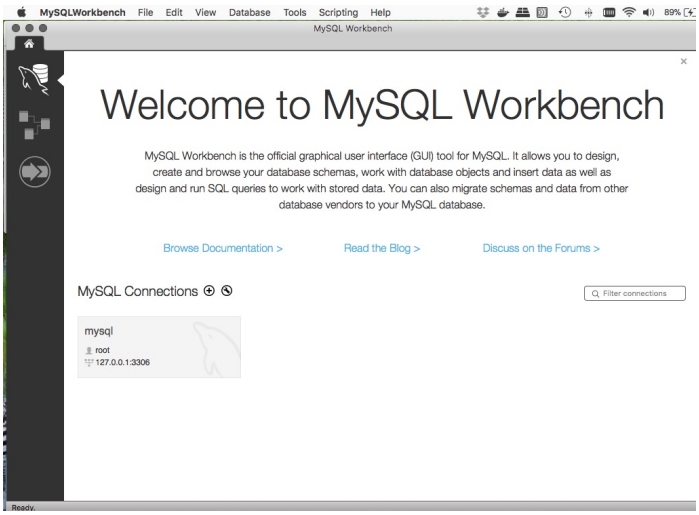

Fig. 6. MySqlWorkbecnch running.

Use the menu offered by MySqlWorkbench (see Fig. 6) to create:

1. A database called **rope_historical**. Initiate the database with a table called *eventhistory* for storing the history of the development of the contractual

interaction. Note that as of now, the history is not cancelled upon completion of a contractual interation as it makes sense to allow historical checks on activities of past transactions. The facility to cancel it on request would in any case be useful. Create the *eventhistory* table with the following fields:

type –VARCHAR(80): Same as field eventtypename of eventtypes: name of event type.

timestamp –DATETIME: The time stamp of the event.

originator –VARCHAR(50): Same as field rolename of rolename: name of originating role player.

responder –VARCHAR(50): Same as field rolename of rolename: name of responding role player.

status –VARCHAR(30): Same as field outcomedescription of statusoutcomes: outcome status of event.

2. A user (with a corresponding password) to access the data base. Since we were not concerned about security and for simplicity, we used the *root* user without a password.

4.3 Deploy jboss EAP 6.4

In the current version, the CCC is deployed as a REST web server within the jboss Enterprise Application Platform 6.4 (jboss EAP–6.4).

1. Download the $jboss - eap - 6.4.0 - installer.jar$ file from `https://developers.redhat.com/download-manager/file/jboss-eap-6.4.0.GA-installer.jar` You need an account with RedHat to be granted access to the jboss EAP 6.4. It can be created at no cost.
2. Run the installer and deploy the jboss EAP–6.4 is a local folder, for example:

```
bash-3.2$ pwd
/Users/carlosmolina/local/jboss/EAP-6.4.0
bash-3.2$
bash-3.2$ ls -l
total 808
-rw-r--r--   1 carlosmolina  staff     419 27 Mar  2015 JBossEULA.txt
-rw-r--r--   1 carlosmolina  staff   26530 27 Mar  2015 LICENSE.txt
drwxr-xr-x   3 carlosmolina  staff      96 23 Feb 19:47 Uninstaller/
drwxr-xr-x   3 carlosmolina  staff      96 23 Feb 19:47 appclient/
drwxr-xr-x  22 carlosmolina  staff     704 23 Feb 19:47 bin/
drwxr-xr-x   3 carlosmolina  staff      96 23 Feb 19:47 bundles/
drwxr-xr-x   5 carlosmolina  staff     160 23 Feb 19:47 docs/
drwxr-xr-x   5 carlosmolina  staff     160 23 Feb 19:47 domain/
drwxr-xr-x  30 carlosmolina  staff     960 23 Feb 19:47 icons/
drwxr-xr-x   4 carlosmolina  staff     128 23 Feb 19:47 installation/
-rw-r--r--   1 carlosmolina  staff  363815 27 Mar  2015 jboss-modules.jar
drwxr-xr-x   4 carlosmolina  staff     128 23 Feb 19:58 modules/
drwxr-xr-x  10 carlosmolina  staff     320 23 Feb 20:00 standalone/
```

```
-rw-r--r--   1 carlosmolina  staff      66 27 Mar  2015 version.txt
drwxr-xr-x  10 carlosmolina  staff     320 23 Feb 19:47 welcome-content/
bash-3.2$
```

At some point, the installation process requests to define an *admin user name* and a corresponding *password*.

4.4 Configure the MySql connectors

We assume that your *conch* folder is located on your local file system, in this running example, it is located at

`/Users/carlosmolina/local/git/conch`
and includes the following files:

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch

bash-3.2$ ls -l
total 108288
drwxr-xr-x   6 carlosmolina  staff       192 22 Feb 00:44 CCCRest-ear/
drwxr-xr-x   5 carlosmolina  staff       160 23 Feb 20:11 CCCRest-ear-commons/
drwxr-xr-x   6 carlosmolina  staff       192 23 Feb 20:11 CCCRest-ear-ear/
drwxr-xr-x   6 carlosmolina  staff       192 23 Feb 20:11 CCCRest-ear-ejb/
drwxr-xr-x   5 carlosmolina  staff       160 23 Feb 20:11 CCCRest-ear-web/
drwxr-xr-x  15 carlosmolina  staff       480 23 Feb 22:15 CCCRestClient/
drwxr-xr-x   7 carlosmolina  staff       224 23 Feb 19:58 JBOSS-EAP-6.4 configuration/
-rw-r--r--   1 carlosmolina  staff      1051 22 Feb 14:37 README.md
-rw-r--r--   1 carlosmolina  staff        87 22 Feb 00:44 README.me
-rw-r--r--@  1 carlosmolina  staff    875138 22 Feb 00:44 UsersGuide.pdf
-rw-r--r--   1 carlosmolina  staff  54246846 24 Feb 01:56 drools.log
drwxr-xr-x   6 carlosmolina  staff       192 22 Feb 00:44 example contracts/
-rw-r--r--   1 carlosmolina  staff      3562 22 Feb 00:44 install.md
-rwxr-xr-x   1 carlosmolina  staff      1213 22 Feb 00:44 logging.properties*
-rw-r--r--   1 carlosmolina  staff     10649 22 Feb 00:44 pom.xml
-rw-r--r--   1 carlosmolina  staff       168 22 Feb 00:44 run.bat
-rwxr-xr-x   1 carlosmolina  staff       112 23 Feb 19:55 run.sh*
bash-3.2$
```

Similarly, the jboss EAP–6.4 is located at `/Users/carlosmolina/local/jboss/EAP-6.4.0`, thus:

```
bash-3.2$ pwd
/Users/carlosmolina/local/jboss/EAP-6.4.0
bash-3.2$
```

```
bash-3.2$ ls -l
total 808
-rw-r--r--   1 carlosmolina  staff     419 27 Mar  2015 JBossEULA.txt
-rw-r--r--   1 carlosmolina  staff   26530 27 Mar  2015 LICENSE.txt
drwxr-xr-x   3 carlosmolina  staff      96 23 Feb 19:47 Uninstaller/
drwxr-xr-x   3 carlosmolina  staff      96 23 Feb 19:47 appclient/
drwxr-xr-x  22 carlosmolina  staff     704 23 Feb 19:47 bin/
drwxr-xr-x   3 carlosmolina  staff      96 23 Feb 19:47 bundles/
drwxr-xr-x   5 carlosmolina  staff     160 23 Feb 19:47 docs/
drwxr-xr-x   5 carlosmolina  staff     160 23 Feb 19:47 domain/
drwxr-xr-x  30 carlosmolina  staff     960 23 Feb 19:47 icons/
drwxr-xr-x   4 carlosmolina  staff     128 23 Feb 19:47 installation/
-rw-r--r--   1 carlosmolina  staff  363815 27 Mar  2015 jboss-modules.jar
drwxr-xr-x   4 carlosmolina  staff     128 23 Feb 19:58 modules/
drwxr-xr-x  10 carlosmolina  staff     320 23 Feb 20:00 standalone/
-rw-r--r--   1 carlosmolina  staff      66 27 Mar  2015 version.txt
drwxr-xr-x  10 carlosmolina  staff     320 23 Feb 19:47 welcome-content/
bash-3.2$
```

To configure jboss EAP–6.4 with the needed MySql driver copy conch/JBOSS-EAP-6.4 configuration/mysql folder to EAP-6.4.0/modules/com

4.5 Configure jboss EAP–6.4 with datasource and MySql drivers

Copy `conch/JBOSS-EAP-6.4 configuration/standalone-full.xml` to `EAP-6.4.0/standalone/configuration`

The most relevant lines of the `EAP-6.4.0/standalone/configuration/standalone-full.xml` are shown next:

```
bash-3.2$ pwd
/Users/carlosmolina/local/jboss/EAP-6.4.0/standalone/configuration

bash-3.2$ ls
application-roles.properties standalone-full-ha.xml
application-users.properties standalone-full.xml
logging.properties standalone-ha.xml
mgmt-groups.properties standalone-osgi.xml
mgmt-users.properties standalone.xml
standalone-full-backup.xml standalone_xml_history/

bash-3.2$ more standalone-full.xml
<?xml version='1.0' encoding='UTF-8'?>

<server xmlns="urn:jboss:domain:1.7">
```

```
... ... ...

<datasources>
   <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enable
      <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE</connect
      <driver>h2</driver>
      <security>
         <user-name>sa</user-name>
         <password>sa</password>
       </security>
   </datasource>

   <datasource jta="false" jndi-name="java:jboss/datasources/RopeDS" pool-name="RopeDS"
      <connection-url>jdbc:mysql://127.0.0.1:3306/rope_historical</connection-url>
      <driver-class>com.mysql.jdbc.Driver</driver-class>
      <driver>mysql</driver>
      <security>
        <user-name>root</user-name>
      </security>
      <validation>
           <validate-on-match>false</validate-on-match>
           <background-validation>false</background-validation>
      </validation>
      <statement>
         <share-prepared-statements>false</share-prepared-statements>
      </statement>
    </datasource>
    <drivers>
       <driver name="mysql" module="com.mysql"/>
       <driver name="h2" module="com.h2database.h2">
       <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
        </driver>
     </drivers>
</datasources>

...

<deployments>
   <deployment name="CCCRest-ear.ear" runtime-name="CCCRest-ear.ear">
       <content sha1="e4ee218d278dc5a41df60f896ba819272521a960"/>
   </deployment>
</deployments>
</server>
```

Some lines deserve further explanation:

1. In

```
<security>
   <user-name>sa</user-name>
   <password>sa</password>
 </security>
```

user-name and *password* correspond to ..... They can be set by ... and can altered by ... by means of ...

2. In

   ```
   <connection-url>jdbc:mysql://127.0.0.1:3306/rope_historical</connection-url>
   ```

   rope_historical is the name of the MySql database that we created in Section 4.2.
3. In

   ```
   <security>
       <user-name>root</user-name>
   </security>
   ```

   user-name correspons to the user that is authorised to manipulate the *rope_historical* database. Observe that we use the default *root* user with no password.

I need help from Ioannis here to explain the meaning of the user–name called sa and its sa password. Who, where and when created the sa user and the pass and how can I change it if I need to.

4.6 Deploy drools with the jboss EAP–6.4

Copy the `conch/JBOSS-EAP-6.4 configuration/drools` folder to `/EAP-6.4.0/standalone` Afer copying, my EAP-6.4.0 contain the following files:

```
bash-3.2$ pwd
/Users/carlosmolina/local/jboss/EAP-6.4.0/standalone
bash-3.2$
bash-3.2$ ls
configuration/ deployments/ lib/ tmp/
data/ drools/ log/
bash-3.2$
```

4.7 Configure jboss EAP–6.4 with drools

Edit the `EAP-6.4.0/standalone/drools/upload/change-set.xml` file to indicate the full path to the *Rule.drl* file

```
bash-3.2$ pwd
/Users/carlosmolina/local/jboss/EAP-6.4.0/standalone/drools/upload

bash-3.2$ ls
BuyerStoreContractEx.drl change-set.xml
Rule.drl
```

The *Rule.drl* file that contains the rules used by the CCC is the example of executable contract that we use to demonstrate the functionality of the CCC. The file *BuyerStoreContractEx.drl* contains another set of rules (for a different contract) that we do not use in these experiments. The following lines are in drool language and are an excerpt from the actual *Rule.drl* file.

```
bash-3.2$ more Rule.drl
import uk.ac.ncl.core.*;
...
import uk.ac.ncl.state.RopState.ObligationState;
import uk.ac.ncl.state.RopState.RightState;
import uk.ac.ncl.state.RopState.ProhibitionState;
import uk.ac.ncl.rop.Obligation;
import uk.ac.ncl.rop.Prohibition;
import uk.ac.ncl.rop.Right;
...
import java.util.Date;
import java.util.Calendar;

global RelevanceEngine engine;
global EventLogger logger;
global TimingMonitor timingMonitor;
...
rule "Registration"
    when
       $e:Event (operation.getName() == OperationName.register, status == EventStatus.su
       $user:User(role == "PI")
    then

      CCCLogger.logInfo("Registration rule is triggered!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
       Right uploadData = new Right(new Operation(OperationName.upload, "data", null));
       Right uploadTool = new Right(new Operation(OperationName.upload, "tool", null));
       $user.addRight(uploadData);
       $user.addRight(uploadTool);
       responder.setContractCompliant(true);
       CCCLogger.logInfo("Registration rule is done!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
end
...
```

The *change-set.xml* needs to include a line to inform jboss EAP-6.4 where the *Rule.drl* file is.

```
bash-3.2$ more change-set.xml
<?xml version="1.0" encoding="UTF-8"?>
...
 <add>
```

```
   <resource type="DRL"
      source="file:////Users/carlosmolina/local/jboss
       /EAP-6.4.0/standalone/drools/upload/Rule.drl"/>
 </add>
</change-set>
bash-3.2$
```

4.8 Export jboss EAP–6.4 home

Edit your ./bash_profile to export the path to the jboss EAP–6.4 folder, to `/Users/carlosmolina/local/jboss/EAP-6.4.0` in my installation.

```
bash-3.2$ pwd
/Users/carlosmolina

bash-3.2$ more .bash_profile
...
export JBOSS_HOME="/Users/carlosmolina/local/jboss/EAP-6.4.0"
...
```

4.9 Install maven 3.3.9

We use maven projects to build both the CCC and its Client counterpart (see Fig. 3).

maven 3.3.9 is currently available from `http://maven.apache.org/download.cgi`

1. After dowloading maven to a local folder, for example, to `/Users/carlosmolina/local/apache-maven-` edit your *.bash_profile* to include the the `/Users/carlosmolina/local/apache-maven-3.5.2/bin` in the search path.
2. Create the *.mavenrc* file under your home folder and add the following lines to indicate maven what jdk machine to use.

```
bash-3.2$ pwd
/Users/carlosmolina

bash-3.2$ ls -la
total 1352
drwxr-xr-x+ 110 carlosmolina  staff    3520 25 Feb 01:21 ./
drwxr-xr-x    5 root          admin     160 23 Jan 20:53 ../
...
-rw-------    1 carlosmolina  staff    5241 24 Feb 15:07 .bash_history
-rw-r--r--    1 carlosmolina  staff    2719 23 Feb 19:51 .bash_profile
-rw-r--r--    1 carlosmolina  staff    1821 23 Oct 22:31 .bash_profile-error
...
drwxr-xr-x    4 carlosmolina  staff     128 22 Feb 21:00 .m2/
-rw-r--r--    1 carlosmolina  staff     152 22 Feb 19:21 .mavenrc
```

```
...

bash-3.2$
bash-3.2$ more .mavenrc
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home
JAVA_OPTS="-Xms512m -Xmx1024m -Xss512k -XX:PermSize=64m -XX:MaxPermSize=128m"
bash-3.2$
```

Observe that we are using jdk1.7.0_79.jdk. We had some difficulties to run jboss EAP–6.4 with jdk1.8.0_20.jdk. Since it ran well with jdk1.7.0_79.jdk we did not investigate the issue.

4.10 Configure maven to build the CCC counter part

Copy `conch/JBOSS-EAP-6.4 configuration/settings.xml` to `~/.m2`

Your `./m2` now should look like this:

```
bash-3.2$ pwd
/Users/carlosmolina/.m2
bash-3.2$ ls
repository/ settings.xml
bash-3.2$
```

4.11 Run maven to build and deploy the CCC within jboss EAP–6.4

1. Verify that the MySql server is running (see Fig. 5)
2. Go to the *conch* folder.

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch

bash-3.2$ ls
CCCRest-ear/ UsersGuide.pdf
CCCRest-ear-commons/ drools.log
CCCRest-ear-ear/ example contracts/
CCCRest-ear-ejb/ install.md
CCCRest-ear-web/ logging.properties*
CCCRestClient/ pom.xml
JBOSS-EAP-6.4 configuration/ run.bat
README.md run.sh*
README.me
bash-3.2$
```

3. Execute the *run.sh* shell scrypt to initiate jboss EAP–6.4 to have it ready to host and instance of the CCC presented as a web server by maven. Observe that EAP–6.4 initiates a standalone–full server.

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch

bash-3.2$ more run.sh
#!/bin/sh
# run standalone Jboss EAP 6.4
exec $JBOSS_HOME/bin/standalone.sh -c standalone-full.xml
#-b 0.0.0.0
bash-3.2$


bash-3.2$ ./run.sh
=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: /Users/carlosmolina/local/jboss/EAP-6.4.0

  JAVA: /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java

  JAVA_OPTS:  -server -XX:+UseCompressedOops -verbose:gc -Xloggc:"/Users/carlosmolina

=========================================================================

02:02:12,113 INFO  [org.jboss.modules] (main) JBoss Modules version 1.3.6.Final-redha
...
02:10:19,965 INFO  [org.jboss.as.server] (Controller Boot Thread) JBAS015859: Deploye
02:10:19,973 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015961: Http managemen
02:10:19,973 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console
02:10:19,973 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss EAP 6.4.
```

The jboss EAP–6.4 is now ready to accept deployments of instancess of the CCC.

4. Run maven to build the CCC web server and present it to jboss EAP–6.4.

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch
bash-3.2$
bash-3.2$ mvn clean package jboss-as:deploy
bash-3.2$ mvn clean package jboss-as:deploy
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for uk.ac
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be uniq
...
Feb 25, 2018 2:18:17 AM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 3.2.12.GA
```

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] CCCRest-ear ....................................... SUCCESS [  0.811 s]
[INFO] CCCRest EAR: Commons Module ....................... SUCCESS [  3.626 s]
[INFO] CCCRest EAR: EJB Module ........................... SUCCESS [  1.506 s]
[INFO] CCCRest EAR: WAR Module ........................... SUCCESS [  3.001 s]
[INFO] CCCRest EAR: EAR Module ........................... SUCCESS [ 11.128 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 21.464 s
[INFO] Finished at: 2018-02-25T02:18:25Z
[INFO] Final Memory: 43M/509M
[INFO] ------------------------------------------------------------------------
```

The jboss EAP–6.4 has accepted the deployment of the CCC. The CCC is now ready to receive and process contractual events generated by the client.

4.12 Build the client counter part module with maven

Go to the folder that contains the files of the client modue and run maven to create the Client.

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch/CCCRestClient
bash-3.2$
bash-3.2$ ls
ExecSequencesSamples/ runClient.sh*
SeqIncorrectCanc3BF-xml/ src/
pom.xml target/
readme.txt trails-xml/
restClient.log transaction.log
bash-3.2$
```

```
bash-3.2$ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building CCCRestClient 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ CCCRestClient ---
[INFO] Deleting /Users/carlosmolina/local/git/conch/CCCRestClient/target
```

```
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ CCCRestClient ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. buil
[INFO] skip non existing resourceDirectory /Users/carlosmolina/local/git/conch/CCCRestCl
...
-------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.957 s
[INFO] Finished at: 2018-02-25T02:24:09Z
[INFO] Final Memory: 21M/223M
[INFO] ------------------------------------------------------------------------
```

4.13 Build client with maven and run it against the CCC

We have written the runClient.sh to build and execute the client. Upon starting, the client starts producing and sending contractual event to the CCC which is assumed to be running and waiting for these events (see Section 4.11).

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch/CCCRestClient
bash-3.2$ ls
ExecSequencesSamples/ runClient.sh*
SeqIncorrectCanc3BF-xml/ src/
pom.xml target/
readme.txt trails-xml/
restClient.log transaction.log
```

```
bash-3.2$ more runClient.sh
#!/bin/sh
# 1 build the CCC client that communicates with the CCC server
# 2 execute the jar file using a sequence of events that we have
#   stored in the ccTestSeq-xml fplder
# 2. execute jar with sequences

mvn clean package assembly:single  &&
java -jar target/CCCRestClient-1-jar-with-dependencies.jar /Users/carlosmolina/local/git

bash-3.2$
```

The *runClient.sh* assumes the exustance of a sequence of contractual event stored in the *ccTestSeq-xml* folder. We have generated this sequence of event a priori and independently from this installation procedure. The sequence is a test—case that we generated using the SPIN model checker in combination with its Promela language (see [8], [9], and [10], [11]).

The following lines show the format of a contractual event.

```
bash-3.2$ pwd
/Users/carlosmolina/local/git/conch/CCCRestClient/
  ExecSequencesSamples/ccTestSeq-xml

bash-3.2$ ls
correctchoreExecSeq1/ correctchoreExecSeq12/

bash-3.2$ pwd
/Users/carlosmolina/local/git/conch/CCCRestClient/
  ExecSequencesSamples/ccTestSeq-xml/correctchoreExecSeq1

bash-3.2$ ls
event1.xml event2.xml event3.xml event4.xml


bash-3.2$ cat event1.xml
<event>
  <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
  <originator>buyer</originator>
  <responder>store</responder>
  <type>BUYREQ</type>
  <status>success</status>
</event>


bash-3.2$ cat event2.xml
<event>
  <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
  <originator>store</originator>
  <responder>buyer</responder>
  <type>BUYCONF</type>
  <status>success</status>
</event>


bash-3.2$ cat event3.xml
<event>
    <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
  <originator>buyer</originator>
  <responder>store</responder>
  <type>BUYPAY</type>
  <status>success</status>
</event>
```

```
bash-3.2$ cat event4.xml
<event>
    <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
  <originator>reset</originator>
  <responder>reset</responder>
  <type>reset</type>
  <status>reset</status>
</event>
bash-3.2$
```

The meaning of the event fields is as follows:

- `<sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>`: is the identifier of the event.
- `<originator>buyer</originator>`: is the name of the contractual party that originated the event, the *buyer* in this example.
- `<responder>store</responder>`: is the name of the party targetted by the event, the *store* in this example.
- `<type>BUYREQ</type>`: is the event, a *buy request* in this eample.
- `<status>success</status>`: is the outcome of the execution of the operation corresponding to this event, a *success* in this example, other alternatives that we do not use in this example, are *business failure* and *technical failure*.

Upon reading these four xml–like files, the Client produces the sequence of events: *BUYREQ* followed by *BUYCONF*, followed by *BUYPAY* followed by *reset*, that stand for buy request, buy confirmation, buy payment and reset. the reset event indicates the end of the contractual interaction.

Run the *runClient.sh* script to create and execute the client.

```
bash-3.2$ ./runClient.sh
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building CCCRestClient 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ CCCRestClient ---
[INFO] Deleting /Users/carlosmolina/local/git/conch/CCCRestClient/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ CCCRestClient ---

...

[INFO] Building jar: /Users/carlosmolina/local/git/conch/CCCRestClient/target/CCCRestCli
[INFO] ------------------------------------------------------------------------
```

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.702 s
[INFO] Finished at: 2018-02-26T20:46:00Z
[INFO] Final Memory: 53M/735M
[INFO] ------------------------------------------------------------------------


folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event1.xml


BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='buyer', responde
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='buyer', responde


-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
    <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
</result>


-------- End Response from CCC service ----------



folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event2.xml

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='store', responde
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='store', responde
```

```
-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
    <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
</result>


-------- End Response from CCC service ----------



folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event3.xml

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='buyer', responde
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='buyer', responde


-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
    <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
</result>


-------- End Response from CCC service ----------
```

```
folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event4.xml

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='reset', responde
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='reset', responde


-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
</result>


-------- End Response from CCC service ----------



folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event1.xml

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq12', originator='buyer', respond
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq12', originator='buyer', respond


-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
```

```
    <sequenceId>buyerstoreincorrectContExecSeq12</sequenceId>
</result>


-------- End Response from CCC service ----------



folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event2.xml

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq12', originator='store', respond
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq12', originator='store', respond


-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
    <sequenceId>buyerstoreincorrectContExecSeq12</sequenceId>
</result>


-------- End Response from CCC service ----------



folder: /Users/carlosmolina/local/git/conch/CCCRestClient/ExecSequencesSamples/ccTestSeq

filename: event3.xml

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='reset', responde
-------- Begin Request to CCC service ----------

BusinessEvent{sequenceId='buyerstoreincorrectContExecSeq1', originator='reset', responde
```

```
-------- End Request to CCC service ----------



-------- Begin Response from CCC service ----------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
</result>


-------- End Response from CCC service ----------

bash-3.2$
```

Observe that the one event after another is sent by the cliend to the CCC and that the CCC responds with his decision about the event being contract compliant (true) or not. In this example, all the events are contract compliance:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<result>
    <contractCompliant>true</contractCompliant>
    <sequenceId>buyerstoreincorrectContExecSeq1</sequenceId>
</result>
```

Examination of the jboss EAP–6.4 outputs An examination of the outs produced on the terminal from where the jboss EAP–6.4 was initiated reveals that the CCC has received and examined (in accordance with the *Rule.drl* file) the contractual events sent by the client.

Ioannis: what drl file is activated in this example BuyerStoreContractEx.drl or Rule.drl?

```
20:46:02,035 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:02,036 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:02,041 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:02,042 INFO  [class uk.ac.ncl.erop.ContractComplianceChecker] (Thread-3 (HornetQ-c
20:46:02,042 INFO  [class uk.ac.ncl.erop.ContractComplianceChecker] (Thread-3 (HornetQ-c
20:46:02,044 INFO  [org.drools.compiler.kie.builder.impl.ClasspathKieProject] (Thread-3
20:46:02,044 INFO  [org.drools.compiler.kie.builder.impl.ClasspathKieProject] (Thread-3
20:46:02,052 INFO  [org.drools.compiler.kie.builder.impl.KieRepositoryImpl] (Thread-3 (H
20:46:03,182 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) TimeKe
20:46:03,182 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,183 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,184 FATAL [uk.ac.ncl.logging.CCCLogger] (Thread-3 (HornetQ-client-global-thread
20:46:03,185 ERROR [uk.ac.ncl.logging.CCCLogger] (Thread-3 (HornetQ-client-global-thread
20:46:03,185 WARN  [uk.ac.ncl.logging.CCCLogger] (Thread-3 (HornetQ-client-global-thread
20:46:03,185 INFO  [uk.ac.ncl.logging.CCCLogger] (Thread-3 (HornetQ-client-global-thread
```

```
20:46:03,185 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,185 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,208 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-3 (Ho
20:46:03,209 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,210 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-3 (HornetQ-c
20:46:03,213 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) * Init
20:46:03,214 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) * Init
20:46:03,214 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,280 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,318 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,326 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-3 (Ho
20:46:03,326 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,328 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-3 (HornetQ-c
[fact 0:2:1043117159:1043117159:2:DEFAULT:NON_TRAIT:Event{sequenceId='buyerstoreincorrec
20:46:03,330 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))   Dead
20:46:03,330 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) * Buy
20:46:03,331 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,357 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,357 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) <resul
20:46:03,357 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,357 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))     <s
20:46:03,358 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,387 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,449 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,450 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,451 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,451 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,451 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,452 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-4 (HornetQ-client-glob
20:46:03,456 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-4 (Ho
20:46:03,456 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-4 (HornetQ-client-glob
20:46:03,458 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-4 (HornetQ-c
[fact 0:3:1264964433:1264964433:3:DEFAULT:NON_TRAIT:Event{sequenceId='buyerstoreincorrec
20:46:03,461 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))   Dead
20:46:03,462 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))   Dead
20:46:03,462 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))   Dead
20:46:03,462 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) * Buy
20:46:03,463 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,468 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,469 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) <resul
20:46:03,469 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,469 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))     <s
20:46:03,469 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,475 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,517 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
```

```
20:46:03,517 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,519 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,519 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,519 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,521 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,523 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-3 (Ho
20:46:03,524 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-3 (HornetQ-client-glob
20:46:03,525 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-3 (HornetQ-c
[fact 0:4:1598278538:1598278538:4:DEFAULT:NON_TRAIT:Event{sequenceId='buyerstoreincorrec
20:46:03,526 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))   Dead
20:46:03,527 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))   Dead
20:46:03,527 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) * Paym
20:46:03,528 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-3 (HornetQ-client-global-thre
20:46:03,533 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,533 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) <resul
20:46:03,533 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,534 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931))     <s
20:46:03,534 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,538 INFO  [stdout] (Thread-3 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,574 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,574 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,576 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,576 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,576 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,577 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-4 (HornetQ-client-glob
20:46:03,577 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-4 (Ho
20:46:03,578 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-4 (HornetQ-client-glob
20:46:03,579 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-4 (HornetQ-c
[fact 0:5:1752996649:1752996649:5:DEFAULT:NON_TRAIT:Event{sequenceId='null', originator=
20:46:03,588 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) * rese
20:46:03,589 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) * Rese
20:46:03,589 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,594 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,594 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) <resul
20:46:03,595 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,595 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,601 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,636 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,636 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,637 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,638 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,638 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,640 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-1 (HornetQ-client-glob
20:46:03,642 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-1 (Ho
20:46:03,643 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-1 (HornetQ-client-glob
```

```
20:46:03,644 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-1 (HornetQ-c
[fact 0:6:985937566:985937566:6:DEFAULT:NON_TRAIT:Event{sequenceId='buyerstoreincorrectC
20:46:03,645 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) * Buy
20:46:03,645 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,651 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,651 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) <resul
20:46:03,651 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,651 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931))     <s
20:46:03,651 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,655 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,692 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,692 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,693 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,693 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,693 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,694 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-4 (HornetQ-client-glob
20:46:03,696 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-4 (Ho
20:46:03,696 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-4 (HornetQ-client-glob
20:46:03,699 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-4 (HornetQ-c
[fact 0:7:1550870946:1550870946:7:DEFAULT:NON_TRAIT:Event{sequenceId='buyerstoreincorrec
20:46:03,701 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) * Buy
20:46:03,701 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) * Buy
20:46:03,701 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) * Buy
20:46:03,702 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-4 (HornetQ-client-global-thre
20:46:03,709 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,709 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) <resul
20:46:03,709 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,709 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931))     <s
20:46:03,709 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,713 INFO  [stdout] (Thread-4 (HornetQ-client-global-threads-1329738931)) Hibern
20:46:03,757 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,757 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,759 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,759 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,759 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,760 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-1 (HornetQ-client-glob
20:46:03,761 INFO  [class uk.ac.ncl.util.CustomWorkingMemoryEventListener] (Thread-1 (Ho
20:46:03,762 INFO  [class uk.ac.ncl.erop.RelevanceEngine] (Thread-1 (HornetQ-client-glob
20:46:03,763 INFO  [class uk.ac.ncl.util.CustomAgendaEventListener] (Thread-1 (HornetQ-c
[fact 0:8:830749851:830749851:8:DEFAULT:NON_TRAIT:Event{sequenceId='null', originator='n
20:46:03,763 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) * rese
20:46:03,764 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) * Rese
20:46:03,764 INFO  [class uk.ac.ncl.mdb.EventsMDB] (Thread-1 (HornetQ-client-global-thre
20:46:03,770 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) <?xml
20:46:03,770 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) <resul
```

```
20:46:03,770 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931))     <c
20:46:03,770 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) </resu
20:46:03,774 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1329738931)) Hibern
```

5 Implementation

Details about the technologies used in the implementation of the CCC and the client can be found in Chapter 6 of the MSC dissertation that originated this work [12]. UML class diagrams are also available.

6 Licence

The CCC is released under the Apache License, Version 2.0[13], which is available from Apache's web pages. Also, you can find a *txt* copy from our home page [14].

7 Implementation History

Table 1. Implementation history of the contract compliance checker.

Version	Date	Contributors	Key features
1.2	Feb 2018	Ioannis Sfykaris, Carlos Molina and Ellis Solaiman	Migration from jboss application server AS 7 to jboss EAP–6.4.
1.1	Aug 2012	Ioannis Sfykaris, Carlos Molina and Ellis Solaiman	Implementation of presentation and data access layer. Implementation of a client module for testing purposes.
1.0	Oct 2010	Massimo Strano	CCC logic implemented.

Acknowledgment

Carlos Molina is currently collaborating with the HAT Community Foundation under the support of Grant RG90413 NRAG/536. The implementation of version 1.2 is one of the outcomes.

Ioannis Sfykaris was partly supported by the EU Horizon 2020 project PrismaCloud (`https://prismacloud.eu`) under GA n^o 644962.

References

1. Molina-Jimenez, C., Shrivastava, S., Strano, M.: A model for checking contractual compliance of business interactions. IEEE Trans. on Service Computing **PP**(99) (2011)

2. Szabo, N.: Smart contracts: Formalizing and securing relationships on public networks. First Monday **2**(9) (September 1997)
3. Molina-Jimenez, C., Sfyrakis, I.: Deployment of the contract compliant checker: (User's Guide). Technical Report TR: no–number, School of Computing Science, Newcastle University, UK (2012)
4. Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J.: Untangling blockchain: A data processing view of blockchain systems. https://arxiv.org/abs/1708.05665 (August 2017)
5. Inc., G.: Github distributed version control system. https://github.com (2012)
6. Foundation, T.E.: Eclipse. http://www.eclipse.org (2012)
7. Corporation, O.: Mysql data base. http://www.mysql.com (2012)
8. Holzmann, G.J.: Design and Validation of Computer Protocols. Prentice Hall (1991)
9. SPIN: On-the-fly, ltl model checking with spin. http://spinroot.com (visited in Jul 2012 2012)
10. Abdelsadiq, A., Molina-Jimenez, C., Shrivastava, S.: A high–level model–checking tool for verifying service agreements. In: Proc. 6th IEEE Int'l Symposium on Service–Oriented System Engineering (SOSE'2011), IEEE Computer Society (2011) 297–304
11. Solaiman, E., Sfyrakis, I., Molina-Jimenez, C.: High level model checker based testing of electronic contracts. In Helfert, M., Muñoz, V.M., Ferguson, D., eds.: Cloud Computing and Services Science: 5th Int'l Conf, CLOSER 2015, Lisbon, Portugal, May 20-22, 2015, Revised Selected Papers. Springer-Verlag, LNCS Vol. 581 (2015) 193–215
12. Sfyrakis, I.: Implementing a contract compliance checker for monitoring contracts. http://homepages.cs.ncl.ac.uk/carlos.molina/home.formal (visited in Nov 2012 2012) MSc Dissertation Project, Aug 2012.
13. Foundation, T.A.S.: Apache license version 2.0, january. http://www.apache.org/licenses (2004)
14. Molina-Jimenez, C.: Carlos molina–jimenez home page. http://homepages.cs.ncl.ac.uk/carlos.molina (2012)