Kaniz Fatema, Christophe Debruyne, Dave Lewis, Declan O'Sullivan, John P. Morrison, and Abdullah-Al Mazed. A semi-automated methodology for extracting access control rules from the european data protection directive. In 2016 IEEE Security and Privacy Workshops, SP Workshops 2016, San Jose, CA, USA, May 22-26, 2016, pages 25–32. IEEE Computer Society, 2016.

A Semi-Automated Methodology for Extracting access control rules from the European Data Protection Directive

Kaniz Fatema, Christophe Debruyne, Dave Lewis, Declan O'Sullivan
ADAPT Centre,
Trinity College Dublin, Ireland
{first.last}@scss.tcd.ie

John P. Morrison
IC4,
University College Cork
Ireland
j.morrison@cs.ucc.ie

Abdullah-Al Mazed
Next Gen Security,
F-Secure,
Finland,
abdullah.al.mazed@f-secure.com

***Abstract*—Handling personal data in a legally compliant way is an important factor for ensuring the trustworthiness of a service provider. The EU data protection directive (EU DPD) is built in such a way that the outcomes of rules are subject to explanations, contexts with dependencies, and human interpretation. Therefore, the process of obtaining deterministic and formal rules in policy languages from the EU DPD is difficult to fully automate. To tackle this problem, we demonstrate in this paper the use of a Controlled Natural Language (CNL) to encode the rules of the EU DPD, in a manner that can be automatically converted into the policy languages XACML and PERMIS. We also show that forming machine executable rules automatically from the controlled natural language grammar not only has the benefit of ensuring the correctness of those rules but also has potential of making the overall process more efficient.**

Keywords — Legal PDP, Access Control, Rules, Conflict Resolution, EU Data Protection Directive, Controlled Natural Language.

I. Introduction

Although there are a number of legal instruments aiming to protect the personal data of individuals [2, 5-8], the enforcement of these laws in data management systems is often lacking. If the access control rules contained in these laws could be integrated in authorization infrastructures, this would arguably make the enforcement of data protection requirements more efficient and effective [9].

Prior research on the use of policy based systems to protect personal data mainly focused on obtaining and enforcing access control policies from the data subject only [10-13]. When developing a privacy preserving system, it is important to keep in mind i) the rights of the data subject, ii) the legitimate rights of the stakeholders mentioned in the law, iii) the rights of the issuer or controller of personal data and iv) the responsibilities arising from enforcing those rights. This balance of rights and responsibility is critical and other researchers have often overlooked this balance when designing policy based authorization systems to protect the privacy of personal data [11, 12]. In comparison, we have attempted in our previous research to support this balance by including independent preferences in terms of policies from all the authorities into the authorization system [14]. The system is designed to include access control and conflict resolution polices [15, 16] from different authors, possibly written in different policy languages. A one size fits all policy language is not suitable for constructing policies satisfying all types of requirements. Today we have many examples of different policy languages XACMLv2 [3], XACMLv3 [17], PERMIS [4], P3P [18], and so on and hence many different PDP implementations. For example, XACMLv2 does not support delegation of authority whilst XACMLv3 and PERMIS do. The XACML policy language cannot support state based policy rules such as separation of duties (SoD), whilst PERMIS does. The design of our system supports Policy Decision Points (PDPs) in multiple policy languages; for the demonstration that capability we chose XACML and PERMIS due to the availability of implementation for being open-sourced.

For each item of personal data, we consider the following four types of policies: provided by the law (the access control rules extracted from legislation forms a Legal PDP), provided by the data issuer (e.g. for a degree certificate the university is the issuer, whilst for a personal diary the data subject is the issuer), provided by the data subject (i.e. the individual to whom the data relates), and those of the data controller (i.e. the organization that is legally responsible for the personal data processing). When the controller's (or processor's) system receives a request to access a data item, it first retrieves all the policies related to the data item. The conflict resolution policies are prioritized in the order of law, issuer, data subject and controller, the rationale for the ordering is presented in [26]. The policy that has the highest priority and is applicable to the current request is used by the Master PDP to resolve any conflicting decisions from the access control policies of the various authors. In a previous work we demonstrated the use cases where the dynamic conflict resolution based on request context and combining obligations from different authors are not possible keeping the policies in a single PDP [16]. Therefore, we argued to keep the policies separate for different authorities in independent PDPs.

We conducted an experiment on obtaining enforceable access control rules from the EU DPD for the Legal PDP to enforce. The manual extraction of access control rules and conflict resolution rules from the EU DPD to infer legally compliant decisions was demonstrated in [1]. The drawback discovered was that the manual process was tedious, time consuming and error prone. This study aims to tackle these problems by

proposing a Controlled Natural Language (CNL) to encode the policies and propose a mapping from the CNL to policy languages. The main contributions of this paper are: firstly, introducing a Controlled Natural Language (CNL) grammar into which the natural language rules from the EU DPD can be encoded; and secondly demonstrating a process to obtain machine executable rules automatically from the CNL rules. The addition of the CNL rules in the previously presented process of extracting machine executable rules from the EU DPD [1] will reduce the time of manual transformation and eliminate the possibility of error introduced by manual extraction done by human; in other words the process will be more efficient.

The rest of this paper is structured as follows; Section II discusses related work; Section III describes the various steps of the methodology; Section IV describes the CNL grammar and the CNL encoded rules; Section V presents the conversion process of CNL rules into PDP rules; Section VI presents the implementation; Section VII presents how the implementation was evaluated; Section VIII provides the discussion and finally Section IX describes the conclusions that were drawn.

II. Related Work

Sadeh et al. [19] presented a way to leverage crowdsourcing and Natural Language Processing (NLP) techniques to semi-automatically extract key features from website's privacy policy. However, their technique is not appropriate for legal texts, which are complex in structure and ambiguous by design to maximize the scope of application. The NEURONA [20, 21] project provided a semi-automated way to determine whether some aspects of the current state of a company's personal data files did not comply with regulations. The focus of our work varies from theirs as we process the EU DPD with a structured methodology to get all possible access control rules, whereas their scope is restricted to security measurements on files to measure the compliance with requirements rather than with legal text. Travis et al. [22-25] undertook research to automate the derivation of security requirements from regulations. They applied their method to the Health Insurance Portability and Accountability Act (HIPAA) regulations. To have data access rules they considered six properties for access related activities, such as, subject, action, modality, object, target and purpose. In contrast, we aim to get enforceable access control rules that not only include elements such as subject, action, resource and environment attributes, but also the conditions that must be satisfied in order to get access decisions (grant/deny) and the obligations to execute along with the decision. Bekara et al. [37] presented a semantic information model that formalizes legal requirements. However, not all the legislative rules were possible to express with this ontology. For example, "the data subject can access the personal data if there is no legal objection" was not possible to express with this model, however it was successfully modelled in ours.

Controlled Natural Language (CNL) is well used in a wide range of areas to facilitate human-human communication and human-machine communication [30]. Several researchers have applied CNLs to privacy and access control problems. Matteucci et al. [31] used CNL to present a Data Sharing Agreement (DSA) among contracting parties to assure privacy of data exchanged on the Web, for example, by helping intelligent agents negotiating privacy requirements on behalf of human users. Ferré introduced SQUALL, a CNL for querying and updating RDF graphs [32]. Cregan et al. described a syntax to write and read ontologies in the standardized description logic language OWL to/from CNL [33]. Shi et al. implemented a user interface that enables novice users to write their own access control policies using a CNL interface [34]. Although these pieces of research show promising approaches that can be used, as yet no prior research has demonstrated the use of CNL for encoding the access control rules from the EU DPD.

III. Methodology

We adopt and extend a methodology we developed to extract access control and conflict resolution rules from the EU DPD that was first introduced in [1] and was successfully tested for the manual extraction of machine executable rules from EU DPD [26]. Here we present an improved methodology that consists of the following steps:

1. List the Legal provisions that are directly related to authorisation. This step ensures that the legal rules that are not related to access control/authorisation are eliminated from our consideration.
2. Analyse the Legal provisions obtained from step 1 with the aid of a legal expert to see if they can form enforceable access control/authorisation rules (in natural language). In this step the Legal provisions are examined one by one in order to form a set of enforceable access control rules/authorisation rules for each provision. The Legal rules that are not capable of giving automated independent decisions are discarded at this stage. The elaborate explanations can be found in [26].
3. Refine the natural language rules by grouping similar rules together and ordering them in terms of the exceptions that need to be evaluated before the ones without exceptions. For example, data subjects are allowed unconditional access to their personal data that are held by a data controller, but not if law enforcement would be jeopardised by this. Consequently the rule that concerns law enforcement must be evaluated before the rule that grants the data subject unconditional access.
4. Formalize the natural language rules manually into the form of an Access Control Rule (ACR) and a Conflict Resolution Rule (CRR) using a controlled natural language (CNL). The formalization helps to determine various subject, resource, action or environment attributes, as well as obligations.
5. Convert the controlled natural language rules into executable rules with XML based policy languages such as XACML [3] and PERMIS [4].
6. Validate the obtained Legal rules.

The conversion from the rule set into machine executable rules described in steps 4 and 5 was conducted manually in the research reported upon in [1]. In this paper, we introduce a CNL that is machine processable and map these to the XML based policy languages for which mappings have to be defined. The grammar of the CNL is specified in Section IV.

IV. The Controlled Natural Language Grammar

Controlled natural language (CNL) is a well-defined subset of natural language that uses restricting grammar and vocabulary to reduce ambiguity. CNLs can be accurately and efficiently processed by a computer [27]. The grammar that we developed for encoding the natural language into CNL is described in Augmented Backus-Naur Form (ABNF) [28] in Fig. 1. Note that the notation "|" indicates alternate values and * indicates 0 or more repetitions.

```
rule-definition=("ACR" wp rule-id wp ":" wp rule-statement
".") | ( "CRR" wp rule-id wp ":" wp crr-statement ".") ;
crr-statement = "If" wp conditions wp "then" wp "DCR=" DCR;
rule-id = STRING;
rule-statement = "If" wp conditions wp "then" wp GrantOr-
Deny wp *prep wp article wp actions *( wp prep wp ) *( wp
article wp ) *( wp ResourceType wp ) *( wp "with obliga-
tions to" wp obligations) ;
conditions = (condition wp operator wp conditions) | (con-
dition wp operator wp "(" wp conditions wp ")" wp
*(conditions)) | (condition);
condition = (article attributes wp relationalOperator wp
article attributes ) | (wp "there is" article booleanAt-
tributes) / (wp "there is no" wp booleanAttributes);
attributes = attribute | values ;
attribute = category ":" name ":" type;
category = "Subject"|"Resource"|"Action"|"Environment";
name= STRING;
type     =     "string"|"boolean"|"integer"|"double"|"time"|
"date"|"dateTime";
article= ( wp "a" wp ) | ( wp "an" wp ) | ( wp "the" wp ) |
(wp);
values= (value wp "|" wp values) | (value);
value= DQUOTE STRING DQUOTE;
relationalOperator= "is equal to" | "is "| "is not equal
to"| "is not" | "is greater than" | "is less than";
operator= "AND"|"OR";
actions = action *(wp "|" wp action);
action = word;
ResourceType = word;
prep = "to"| "on"| "at" | "for";
booleanAttributes= (booleanAttribute wp "|" wp booleanAt-
tributes) | (booleanAttribute);
booleanAttribute= category ":" name ":boolean" ;
obligations= (obligation wp "," wp obligations) | (obliga-
tion);
obligation=STRING;
DCR= "DenyOverrides" | "GrantOverrides" | "FirstApplicable"
| "SpecificOverrides"| "MajorityWins";
GrantOrDeny = "Deny" | "Grant" |"BreakTheGlass";
word=*(%x41-5A|%x61-7A|%x30-39);
STRING=                                *(%x41-5A|%x61-7A|%x30-
39|%x20|%x2D|%x27|%x91|%x92);
wp = *(%x20 | %x09 | (%x0D %x0A));
DQUOTE = %x22;
```

Fig 1. The CNL grammar for encoding the access control rules from the EU DPD

The CNL grammar allows an Access Control Rule (ACR) or a Conflict Resolution Rule (CRR) to be specified. An ACR is comprised of a set of conditions on Subjects, Actions, Resources and the Environment, an Effect (Grant/Deny/BreakTheGlass (BTG)) and an optional set of Obligations. A CRR has the same terms as an ACR except that the Effect is always a Permit, and the Obligation is to always use a specific Decision Combining Algorithm (DCA). The rule conditions are specified in terms of attributes, and attribute names can contain any combination of String. Nevertheless, the conversion has to ensure that the same attributes are used in both request contexts which is passed to the authorisation system for requesting access to data and the specified rules (otherwise matching would never occur).

Each Legal ACR rule is also converted into a matching CRR to make sure that the Legal rule gets precedence over any other authority's rules. The difference between the CRR and its corresponding ACR is that the effect of the CRR is always a Permit and the obligation always returns the DCA that is applicable. If an ACR has an effect of Deny, the corresponding DCA is DenyOverrides and if an ACR has an effect of Permit the corresponding DCA is GrantOverrides. If the ACR has an effect of BTG, the CRR's DCA is GrantOverrides, since a Grant from another PDP should not require the requester to first break the glass before gaining access.

In the next section we present some procedural steps to obtain the machine executable rules from the CNL rules so that they can be executed by a Policy Decision Point (PDP) [14].

V. Conversion of CNL Rules to PDP Rules

Since the authorization system [14, 15] we used for enforcing the Legal access control and conflict resolution policies is capable of executing policies both in XACMLv2 [3] and PERMIS [4] policy languages, we implemented the policies in both the policy languages. How XACML and PERMIS work is described in [3, 4]. Here, we only focus on the complication we faced while encoding the access control rules from the EU DPD and the steps we followed for converting the CNL rules into XACML (subsection A) and PERMIS (subsection B).

A. Conversion of CNL rules to XACMLv2

In XACML, if any of the attributes specified in the <Target> element of the <PolicySet>, <Policy> or <Rule> is missing in the request context, the evaluation of the <Target> becomes "Indeterminate" and eventually the effects of the <Rule>, <Policy> and <PolicySet> containing the target evaluates to "Indeterminate", meaning something is wrong in the request context. The XACML encoded legal rules will have a number of alternate attributes and only one set of them will be present in the request context at a time and the absence of such an alternate set of attributes should be evaluated to "NotApplicable", to mean this rule does not apply to the presented request context.

For example, let us consider the implementation of a simple Legal rule saying that the data subject can submit a policy for his/her personal data, where the data subject can be identified by either an {emailAddress}, or a {NHS Number}.[1] The requester is expected to provide any one of these sets of attributes to identify him or herself. If these identity attributes are written inside the <Target> element of the rule then they all are needed to be present in the request context for the requester to be granted access. If any of the attributes are missing from the request context, then an "Indeterminate" decision is returned. However, the requester only needs to be identified by one of these sets of attributes at a time and the other sets of attributes need not be present in the request context. Therefore, a single <Target> element is not the correct answer to encode these alternate attribute sets. Instead either three separate policies or rules with one set of identity attributes in each target element

[1] This example with two sets of attributes will be used as a running example throughout the paper. We note, however, that this is a simplified version of a typical scenario and that any arbitrary number of attribute sets are supported.

should be used, or one policy or rule with an empty target element and a <Condition> element needs to be used.

The <Condition> element represents a Boolean expression. When the <Target> element of an XACML policy evaluates to "Match" and the condition element evaluates to "True", the rule evaluates to "Effect". Or, when the <Target> element of an XACML policy evaluates to "Match" and the condition element evaluates to "False", the rule evaluates to "NotApplicable" [3]. XACML has a collection of functions that provide a powerful way to compare attribute values. The XACML function `type-at-least-one-member-of` takes two arguments that are both a bag of "type" values. It returns a Boolean value. The function evaluates to "True" if and only if at least one element of the first argument is contained in the second argument as determined by `type-is-in`. In this case the <Condition> element containing the function evaluates to "True" and eventually the rule evaluates to the "effect". Otherwise the function returns "False" and the <Condition> evaluates to "False" and consequently the rule evaluates to "NotApplicable". Hence this strategy is used to match the subject attributes. Fig. 2 presents the aforementioned rule in XACML.

```
<Rule RuleId="3" Effect="Permit">
 <Description> ACR 3: If the Subject:Email:string is equal
to the resource:DataSubject'sE-mail:string OR the Sub-
ject:NHSNumber:string is equal to the Re-
source:DataSubject'sNHSNumber:string then Grant the Submit-
Policy for PersonalData. </Description>
 <Target/>
 <Condition>
  <Apply FunctionId="X:and">
   <Apply FunctionId="X:or">
    <Apply FunctionId="X:string-at-least-one-member-of">
     <SubjectAttributeDesignator At-tributeId="E-mail"
DataType="Y#string"/>
     <ResourceAttributeDesignator At-
tributeId="DataSubjects&#x2019;E-mail"
DataType="Y#string"/>
    </Apply>
    <Apply FunctionId="X:string-at-least-one-member-of">
     <SubjectAttributeDesignator At-tributeId="NHSNumber"
DataType="Y#string"/>
     <ResourceAttributeDesignator At-
tributeId="DataSubjects&#x2019;NHSNumber"
DataType="X#string"/>
    </Apply>
   </Apply>
   <Apply FunctionId="X:any-of">
    <Function FunctionId="X:string-equal"/>
    <ActionAttributeDesignator DataType="Y#string" At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
    <AttributeValue
DataType="Y#string">SubmitPolicy</AttributeValue>
   </Apply>
   <Apply FunctionId="X:any-of">
    <Function FunctionId="X:string-equal"/>
    <ActionAttributeDesignator DataType="Y#string" At-
tributeId="ResourceType"/>
    <AttributeValue
DataType="Y#string">PersonalData</AttributeValue>
   </Apply>
  </Apply>
 </Condition>
</Rule>
```

Fig 2. Example of a Legal rule in XACML v2.[2]

The CNL rules written according to the ABNF grammar (presented in Section IV) are converted into XACML according to the following steps:

1. The initial string of a rule "ACR" indicates that the rule is an access control rule and the initial string "CRR" indicates that the rule is a CRR.
2. The ABNF grammar element <rule-id> becomes the <PolicyId> and <Rule-Id> of the XACML policy.
3. The ABNF grammar element <GrantOrDeny> becomes the <effect> of XACML rule.
4. Depending on the value of <GrantOrDeny> the rule-combining algorithm is chosen for XACML. If the value of <GrantOrDeny> is "Grant" then the rule-combining algorithm is *permit-overrides* and if the value is Deny or BreakTheGlass then rule combining algorithm is *deny-overrides.* Note: the policy-combining algorithm for Legal PDP is *first applicable*.
5. The whole rule is copied to the <Description> element of the XACML rule.
6. All the <relationaloperator> of the ABNF grammar are translated into corresponding XACML functions.
7. Each <attribute> of a <condition> element of the ABNF grammar consists of <category>, <name> and <type>. The value of <category> becomes the prefix (i.e. Subject /Resource /Action /Environment) of the AttributeDesignator of XACML, the value of <name> becomes the AttributeId and the <type> becomes the <DataType> of the AttributeDesignator of XACML.
8. The attribute <value> of ABNF becomes the <AttributeValue> of XACML.
9. The ABNF <operator> "AND" / "OR" becomes the XACML FunctionId = "and" / "or" [Note: the XACML FunctionId= "not" comes from the ABNF <relationaloperator> "is not" and "is not equal to".]
10. The <action> element of the ABNF grammar becomes the <AttributeValue> in XACML against which the <ActionAttributeDesignator> with AttributeId= action-id is matched.
11. The <obligations> of the ABNF element becomes the XACML <Obligations>.

B. *Conversion of CNL rules to PERMIS*

Here we consider the conversion of the same example of the rule mentioned in Section V.A to PERMIS. The data subject can submit a policy for his/her personal data, where the data subject can be identified by either an {emailAddress}, or a {NHS Number}. While encoding into PERMIS policies, we faced problem due to the unavailability of a way to define arbitrary subject's or resource's attributes. So the identity attributes of requesters and subjects are passed as environment attributes as presented in Fig. 3.

Similar to XACML, the CNL rules are written according to the ABNF grammar and converted into PERMIS using the following steps:

1. The initial of a rule "ACR" indicates that the rule is an access control rule and the initial "CRR" indicates that the rule is a CRR.

[2] In this figure, the X stands for "urn:oasis:names:tc:xacml:1.0:function" and Y stands for "http://www.w3.org/2001/XMLSchema".

2. The ABNF grammar element <rule-id> becomes the OID of the PERMIS policy.
3. If the value of <GrantOrDeny> is Grant then the value of DenyBased attribute of PERMIS policy is "false", otherwise it is "true".
4. All the <relationaloperator> of the ABNF grammar are translated into the corresponding PERMIS functions <EQ> (equal to), <GT> (greater than), <LT> (less than), <NOT> (not).
5. Each <attribute> of a <condition> element of the ABNF grammar consists of <category>, <name> and <type>. As no arbitrary subject/resource attributes can be defined in PERMIS all the attributes are presented as Environment attributes. The value of <name> becomes the value of Parameter and the <type> becomes the Type of the Environment attribute of PERMIS.

```
<TargetAccess>
 <RoleList>  </RoleList>
 <TargetList>
  <Target>
   <TargetDomain ID="PersonalData"/>
   <AllowedAction ID="SubmitPolicy"/>
  </Target>
 </TargetList>
 <IF>
  <OR>
   <EQ>
    <Environment Parameter="E-mail" Type="String"/>
    <Environment Parameter="DataSubjects&#x2019;E-mail"
Type="String"/>
   </EQ>
   <EQ>
    <Environment Parameter="NHSNumber" Type="String"/>
    <Environment Parameter="DataSubjects&#x2019;NHSNumber"
Type="String"/>
   </EQ>
  </OR>
 </IF>
</TargetAccess>
```

Fig 3. Example of a Legal rule in PERMIS

6. The attribute <value> of the ABNF becomes the Constant Value against which the environment attributes are compared according to the PERMIS function.
7. The ABNF <operator> of "AND" / "OR" becomes the <AND> / <OR> function of PERMIS.
8. The value of the <action> element of the ABNF grammar becomes the *Action Name* and *ID* of the PERMIS policy.
9. The <obligations> of the ABNF element becomes the PERMIS <Obligations>.

C. Conversion of CRR rules to XACMLv2 and PERMIS

All the ACRs are also converted into CRRs. While converting the ACR into CRR the <effect> of XACML is always "permit" and the <Obligation> of XACML becomes "permit-overrides" or "deny-overrides" depending on the value of <GrantOrDeny> of the ABNF (see Section IV).

The conversion of a CRR presented according to the ABNF grammar of Section IV to an XACML/PERMIS policy follows the steps similar to the conversion of an ACR. However, the new element DCR becomes an <obligation>.

In XACMLv2 the smallest element on which the <Obligation> can be applied is <Policy>. Hence in XACML the Conflict Resolution Rules are implemented as separate <Policy> elements inside a <PolicySet> element. The <Obligation> returns the DCA to use [15, 16]. In PERMIS each CRR is written as a separate Target Access Rules (TAR) with an obligation to use a DCA. As an example of conversion we use the same Legal rule used earlier (i.e. the data subject can submit a policy for his/her personal data, where the data subject can be identified by either an {emailAddress}, or a {NHS Number}) as a CRR to convert into XACML v2 and PERMIS. This is shown in Appendix 1 of [38].

VI. Implementation

For automating the generation of rules from CNL, we used only XACML, due to the availability of open access code of the parser. The conversion process is performed in two stages as shown in Fig. 4 and described next.

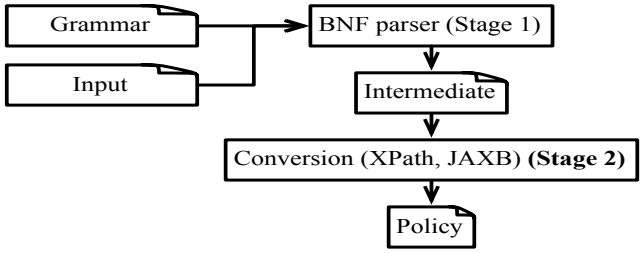

Fig 4. Automated conversion process from CNL rules to XACML rules.

```
<rule-definition>ACR
<rule-id><STRING>3</STRING></rule-id>:
<rule-statement>If
<conditions><condition>
<article>the</article>
<attributes><attribute>
<category>Environment</category>:
<name><STRING>RequestTime</STRING>
</name>:<type>date</type>
</attribute></attributes>
<relationalOperator>is less than</relationalOperator>
<attributes><attribute>
<category>Resource</category>:
<name><STRING>ValidityTime</STRING>
</name>:<type>date</type></attribute></attributes>
</condition></conditions> then
<GrantOrDeny>Deny</GrantOrDeny><article>the</article>
<actions><action><word>Access</word></action></actions>
<prep>to</prep><article>the</article>
<Resource-
Type><word>PersonalData</word></ResourceType></rule-
statement>.</rule-definition>
```

Fig 5. Example intermediate.xml produced from the input.txt

Stage1: Parsing the *input* (containing the CNL rules) according to the ABNF grammar (passed by *grammar*) produces an *intermediate.xml* file. *Intermediate.xml* is XML whose structure follows the ABNF grammar rules. Fig. 5 shows for example the intermediate.xml[3] produced from this input.txt containing "ACR 3: If the Environment:RequestTime:date is less than Resource:ValidityTime:date then Deny the Access to the PersonalData." For implementing Stage 1, we used a freely avalable general purpose tool aParse[4] to convert the CNL rules

[3] All <whitespace> (<wp>) elements have been removed from the produced output for readability.

[4] http://www.parse2.com/

into XACML.

Stage 2: Converts the intermediate.xml into policy.xml using XPath and JAVA Architecture for XML binding. A Java object representation of XACML is created first. Java Architecture for XML binding (JABX) is used to convert the Java classes into XML.

```
// Create a policy object, which will be converted to
xml using JAXB
Policy policy = factory.createPolicy();
// get ruleId using XPath
String ruleId = inputHelper.getNodeValue("//rule-
definition/rule-id/STRING/text()");
// Set ruleId to policy
Rule rule = factory.createRule();
rule.setRuleId(ruleId);
policy.setRule(rule);
// prepare marshaling to xml
JAXBContext jaxbContext = JAXBCon-
text.newInstance(Policy.class);
Marshaller jaxbMarshaller = jaxbCon-
text.createMarshaller();
// write to file
File file = new File(convertedFile);
jaxbMarshaller.marshal(policy, file);
```

Fig 6. Example source code of stage2

Fig. 6 shows some example source code of stage 2 of the XACMLConverter

VII. Evaluation

Evaluation of the implementation involved (a) validating whether the rules generated were accurate; and (b) measuring the speed of the conversion process.

A PHP implemented web service and SoapUI[5] were used for the validation tests. The XACMLConverter tool was installed on an Amazon Web Service EC2 instance of type t2.micro. This gives us a setup with variable EC2 Compute Unit (ECUs), 1 64 bit vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, Elastic Block Store (EBS) only. We have chosen Amazon Machine Image named as Amazon Linux AMI 2015.03 (HVM), SSD Volume Type - ami-e7527ed7. The Java SDK that we used for our test was java-1.7.0-openjdk.x86_64.

The aim of the validation tests of the Legal policy rules was to determine whether the generated rules derived from the EU DPD give the desired responses or not. In order to do this, we constructed a set of test cases based on determining each rule one by one. Each Legal rule is a combination of conditions and each condition consists of either an attribute-attribute pair or attribute-value pair and their relationship. Each condition can evaluate to *true* or *false*. A condition is related to other conditions by a binary operator (AND, OR). Exhaustive test cases were generated based on each condition where each condition in each rule had two test cases created for it, one where the condition was known to be true and one where the condition was known to be false. For generating the test cases a Multi-terminal binary decision diagram (MTBDD) was generated where each condition becomes a node in the binary tree [29]. For example, Fig. 7 shows the MTBDD of the test cases for the Legal rule "If the requested purpose of processing does not match with the original purpose of collection or is not for a historical purpose/statistical purpose / scientific purpose then Deny the request". This formed the CNL "If the Action:Purpose:string is not the Resource:PurposesOfCollection:string OR the Action:Purpose:string is not a "historical purpose" / "statistical purpose" / "scientific purpose" then Deny the Access to the PersonalData."

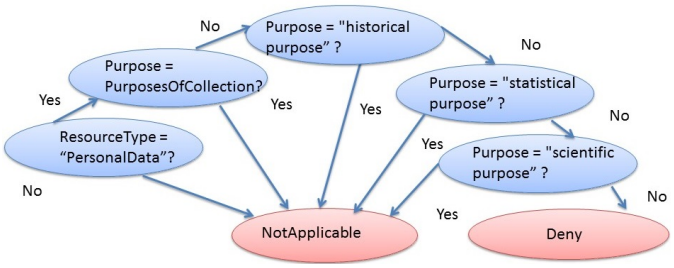

Fig 7. Presentation of rule in the form of MTBDD where the results form terminal nodes and each condition becomes a decision node.

A request context was prepared for each test case. For each request context the responses of the Legal access control and conflict resolution PDPs were compared with the desired outcomes. For performing the validation tests the system was first configured with only the Legal Conflict Resolution PDP and the decisions obtained like any normal PDP decisions. The DCA was obtained as a part of the returned obligation. Then the system was configured with only the Legal Access Control PDP to see the response. The same set of tests were conducted for it as well. More than 100 test cases were generated in total and they are fully described in [26]. Here we exemplify the test case for one rule in Appendix 3 in [38]. We observed the outcomes for all rules for the given request contexts were as expected, i.e. the obtained decisions are accurate for the given circumstances. However, since the policy is matched against the request context, any single mismatch in the attribute representation or a typo in the request context can provide an incorrect outcome. Therefore a high degree of caution was required in presenting the right attributes in the request contexts while doing the tests. In our previous manual process of writing machine readable rules required high cautiousness for errors for typo. The automatically translated machine executable rules from the CNL required no such cautiousness and reduces the human effort and time to a great extent.

To measure the average time of the conversion process, we ran the same script 10 times on the above-mentioned Amazon EC2 instance. Fig. 8 presents the results. As a sample CNL rule, we have used the same rule as demonstrated in Section V.

The average time taken to convert the selected CNL rule with two conditions (start of plot), separated by "OR", into intermediate XML is 0.1035 second and from the intermediate xml to XACML based rules is 0.3734 second, which in total comes to 0.4769 seconds for converting the CNL rule into XACML. We then observed the time taken by the tool to convert the CNL to XACML for increasing complexity of rules. To do this, we increased the number of conditions in a CNL rule as a way of incrementally increasing the complexity of a rule, and measured the time it takes to convert the CNL into XACML for various size of complex rules containing 1-30

[5] http://www.soapui.org/

conditions (separated by “OR” or “AND”).

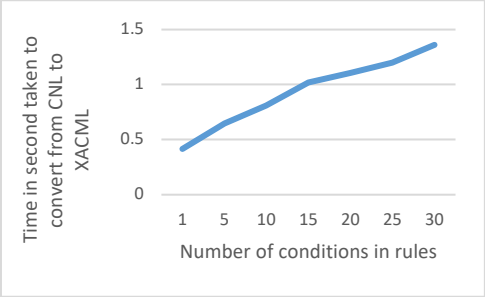

Fig 8. Processing time in second for the conversion of CNL into XACML for increasing complexity of rule

We can observe from Fig. 8 that a linear progression in processing time results when we incremented the complexity in a rule. For the current implementation of the legal rules the maximum number of conditions found was 10. In summary the evaluation indicates that if we add more complex conditions in the rules the processing time would increase linearly.

VIII. DISCUSSION

From our evaluation of the mapping of CNL to XACML and PERMIS we have concluded that the conversion from CNL to an executable policy language needs to consider not only the structure of the policy language, but also the execution process of the policy language, in order to ensure the constructed policy will return the expected decision when executed.

We have demonstrated the feasibility of transcribing policies with a CNL that will generate executable rules. The examples derived from the EU DPD were simple enough for the algorithm to scale well and indicates that the approach holds great promise. Inclusion of more complex rules and the analysis of the algorithm’s performance on such rules are part of future work. Although we have demonstrated our approach’s feasibility and promise, we have been unable as yet to compare the effectiveness and efficiency of transcribing policies in our CNL with the manual process presented in [1]. Our hypothesis however is that the process would help reduce the issues outlined earlier in the paper.

We applied our approach on 53 rules of the EU DPD. Although the addition of CNL has made the process of extraction of access control rules from EU DPD semi-automated, the complexity and nature of some of its rules does not allow the conversion of it into deterministic rules to be fully automated. From the 53 rules of the EU DPD that were considered for analysis in step 2 (since they mentioned some actions on personal data) 27 of them could contribute to the construction of enforceable authorisation rules. However, 14 rules among these 53 are found to be guidelines or instructions only and did not therefore map into authorisation rules. For example, Article 6.1.(a) states that personal data must be processed fairly and lawfully, Article 17.1 says that controller must implement appropriate technical and organisational measures to protect personal data against accidental or unlawful destruction or accidental loss. These guidelines set out the overall requirements for privacy protection. Within an operational data management system, these guidelines may need to be auditably instantiated in data protection compliance processes. Access control is one of the important aspects of privacy protection but it does not cover all such aspects. Data compliance tasks would require however extension to the CNL and their mapping to data management workflows that enforce decision making and reporting beyond access control. However, three other rules can be supported by our system design, explained in details in [26]. The remaining 9 rules are found to be too dependent on other laws or human judgement to be turned into access control rules by themselves, for example, Article 7(f) “processing of personal data for legitimate interest are allowed except where such interests are overridden by the fundamental rights and freedom of data subject” presents an extremely complex condition where the balance of interests are not feasible to be presented in an access control policy. Supporting decision-making and reporting around these broader concerns requires further work to combine access control rules and compliance decision-making and reporting obligations with enterprise governance rules and national interpretation of EU data protection law. We plan to investigate the role of CNL for supporting different stakeholders, but with the focus on the requirements for data management under the emerging EU General Data Protection Regulations (GDPR).

IX. CONCLUSIONS

Transcribing EU DPD into executable rules to support access policies has been undertaken manually in the past [1]. The process, however, has been deemed time consuming and error prone in terms of possibility of having typo [35, 36]. However, even with the limited capabilities of not having full set of automatically enforceable access control rules, the automation of Legal policy enforcement significantly reduces the effort required to ensure compliance [9]. In this paper, we have successfully demonstrated our approach that uses Control Natural Language to transcribe these policies, and in turn transform them into executable rules in PERMIS and XACML. We would argue that the adoption of a CNL not only renders the transcription process more efficient, but that the rules in CNL are more accessible to the user (in terms of usability and understanding) than XML formats. The extent that these are more accessible however does need to be investigated further with appropriate user trials. In particular, differentials in accessibility to the different stakeholder types, i.e. data subjects, issuers and controllers, may indicate that different, but consistent forms of CNLs may be required for the different cohorts.

We have implemented and evaluated our approach by converting 27 of the 53 rules in the EU DPD. Some of the rules were impossible to convert. Several requirements may be far more complex to model in practice, due to the divergences of national or sector specific laws from the European Directive or uncertainty of how these map to a given operational context. We aim to develop rules that can better accommodate this greater complexity using semantic modelling techniques including description logics in the future.

In summary, we would argue that the approach developed, demonstrated and evaluated as reported upon in this paper holds significant promise in ensuring EU DPD rules are encoded in a manner that will be efficient and ensure correct execution of the rules within systems. Our future work will investi-

gate whether these benefits can be extended to data protection compliance regimes anticipated as Europe moves from the DPD to the GDPR era. We further plan to automate the 1st step using natural language processing technique to aid legal experts to analyse the legal texts.

X. ACKNOWLEDGEMENTS

This paper is based on work funded by the EU TAS3 project, and is partially supported by the ADAPT Centre for Digital Content Technology, which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

REFERENCES

[1] K. Fatema, D.W. Chadwick and B.V. Alsenoy, "Extracting Access Control and Conflict Resolution Policies from European Data Protection Law," in *Privacy and Identity Management for Life*, Springer, Heidelberg, 2012, pp. 59-72.

[2] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.

[3] OASIS XACML 2.0. eXtensible Access Control Markup Language (XACML)Version 2.0, Oct, 2005, http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=xacml#XACML20.

[4] D. Chadwick, G. Zhao, S. Otenko, R. Laborde, L. Su, and T. A. Nguyen, "PERMIS: a modular authorization infrastructure," *Concurrency And Computation: Practice And Experience*, vol 20, issue 11, pp 1341-1357. 2008.

[5] Health Information Privacy, HIPAA 1996 privacy and Security Rules, http://www.hhs.gov/ocr/privacy/

[6] Protection of personal information in the private sector,http://www2.parl.gc.ca/HousePublications/Publication.aspx?pub=bill&doc=C-6&parl=36&ses=2&language=E&File=32#4

[7] Australian Govt. Com Law, Privacy Act 1988,http://www.comlaw.gov.au/Series/C2004A03712

[8] OECD, Privacy and Personal Data Control, http://www.oecd.org/dataoecd/30/32/37626097.pdf

[9] N. Papanikolaou, S. Pearson, and M.C. Mont. "Towards natural-language understanding and automated enforcement of privacy rules and regulations in the cloud: survey and bibliography," in *Secure and Trust Computing, Data Management, and Applications*, Springer, 2011, pp. 166-173.

[10] G. Karjoth, M. Schunter and M. Waidner, "Privacy-enabled services for enterprises," in *13th International Workshop on Database and Expert Systems Applications*, IEEE Computer Society, Washington DC, 2002, pp. 483-487.

[11] M. C. Mont, "Dealing with Privacy Obligations: Important Aspects and Technical Approaches," in International conference on trust and privacy in digital business, Zaragoza, 2004.

[12] C. A. Ardagna, L. Bussard, S. D. C. Vimercati, G. Neven, S. Paraboschi, E. Pedrini, F-S. Preiss, D. Raggett, P. Samarati, S. Trabelsi and M. Verdicchio, "PrimeLifePolicy Language," in *Workshop on Access Control Application Scenarios*, W3C, 2009.

[13] S. Trabelsi, A Njeh, L. Bussard, and G. Neven, "PPL Engine: A Symmetric Architecture for Privacy Policy Handling," in *W3C Workshop on Privacy and data usage control*, October, 2010.

[14] D. W. Chadwick and K. Fatema, "An advanced policy based authorisation infrastructure," in *Proceedings of the 5th ACM workshop on Digital identity management*, DIM'09, Chicago, Illinois, USA, 2009, pp.81-84.

[15] K. Fatema, D.W. Chadwick and S. Lievens, "A Multi Privacy Policy Enforcement System," in *Privacy and Identity* 2010, IFIP AICT 352, 2011, pp. 297–310.

[16] K. Fatema and D. Chadwick, "Resolving Policy Conflicts - Integrating Policies from Multiple Authors," in CAiSE International Workshops, Thessaloniki, Greece, 2014.

[17] OASIS XACML 3.0. eXtensible Access Control Markup Language (XACML) Version 3.0, 16 April, 2009, http://docs.oasisopen.org/xacml/3.0/xacml-3.0-core-spec-en.html.

[18] W3C: The Platform for Privacy Preferences 1.0 (P3P 1.0), Technical Report, 2002.

[19] N. Sadeh, A. Acquisti, T.D. Breaux, L.F. Cranor, A.M. McDonald, J. Reidenberg, N.A. Smith, F. Liu, N.C. Russell, F. Schaub, S. Wilson, J.T. Graves, P.G. Leon, R. Ramanath, A. Rao, "Towards Usable Privacy Policies: Semi-automatically Extracting Data Practices From Websites' Privacy Policies," *FTC PrivacyCon*, Jan 2016.

[20] N. Casellas, M. R. D. L Mozos, P.Casanovas, "Ontology-Enhanced Legal Decision-Support Tools: The NEURONA Data Protection Compliance Application," *Eleventh International Conference on Substantive Technology in Legal Education and Practice*, University of Zaragoza, July 2010.

[21] N. Casellas, J-E. Nieto, A. Merono, A. Roig, S. Torralba, M. Reyes, P. Casanovas, "Ontology Semantics for Data Privacy Compliance: the NEURONA Ontology," in *AAAI Spring Symposium Series Technical Reports (Intelligent Information Privacy Management)*, Stanford, March 2010.

[22] T. D. Breaux, A. I. Antón, "Analyzing Regulatory Rules for Privacy and Security Requirements," in *IEEE Transactions on Software Engineering, Special Issue on Software Engineering for Secure Systems (IEEE TSE)*, vol 34, Issue 1, pp 5-20, 2008.

[23] T. D. Breaux, A. I. Antón, "A Systematic Method for Acquiring Regulatory Requirements: A Frame-Based Approach," in *Proc. 6th International Workshop on Requirements for High Assurance Systems (*RHAS-6*)*, Delhi, India, Sep. 2007.

[24] T. D. Breaux, A. I. Antón, "Analyzing Goal Semantics for Rights, Permissions and Obligations," in *Proc. IEEE 13th International Requirements Engineering Conference (RE'05),* Paris, France, Aug. 2005, pp. 177-186.

[25] N. Kiyavitskaya, N. Zeni, T.D. Breaux, A.I. Antón, J.R. Cordy, L. Mich, J. Mylopoulos, "Automating the Extraction of Rights and Obligations for Regulatory Compliance," in: *proc. 27th International Conference on Conceptual Modelling (ER'08),* Barcelona, Spain, Oct. 2008, pp 154-168.

[26] K. Fatema, "Adding Privacy Protection to Policy Based Authorisation Systems," PhD thesis, University of Kent, UK, 2013.

[27] N. E. Fuchs, & R. Schwitter, "Specifying logic programs in controlled natural language," *in: arXiv preprint cmp-lg/9507009*, 1995.

[28] D. Crocker & P. Overell, "Augmented BNF for syntax specifications: ABNF," *RFC 5234*, 2005.

[29] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, M. C. Tschantz, "Verification and change-impact analysis of access-control policies," *in 27th International Conference on Software engineering*. ACM, 2005, pp. 196-205. ().

[30] A. Wyner, K. Angelov, G. Barzdins, D. Damljanovic, B. Davis, N. Fuchs & J. Sowa, "On controlled natural languages: Properties and prospects," in *Controlled Natural Language*, Springer Berlin Heidelberg, 2010, pp 281-289.

[31] I. Matteucci, M. Petrocchi, & M. L. Sbodio, "CNL4DSA: a controlled natural language for data sharing agreements," in: *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp 616-620, ACM.

[32] S. Ferré, "SQUALL: A controlled natural language for querying and updating RDF graphs," in: Controlled Natural Language, Springer, Berlin, Heidelberg, 2012, pp. 11-25.

[33] A. Cregan, R. Schwitter & T. Meyer, "Sydney OWL Syntax-towards a Controlled Natural Language Syntax for OWL 1.1," in: OWLED. Vol. 258, 2007.

[34] L Shi, and D.W. Chadwick, "A controlled natural language interface for authoring access control policies," in: *proceedings of the 2011 ACM Symposium on Applied Computing*, ACM, 2011.

[35] K. K. Waterman, "Pre-processing legal text: policy parsing and isomorphic intermediate representation," *in: AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[36] M.C. Mont, S. Pearson, S. Creese, M. Goldsmith and N. Papanikolaou. "EnCoRe: towards a conceptual model for privacy policies." *PrimeLife/IFIP Summer School 2010: Privacy and Identity Management for Life.* Helsingborg, Sweden: Springer, 2010.

[37] K. Bekara, and M. Laurent. "A semantic information model based on the privacy legislation." *IEEE Conference on Network and Information Systems Security (SAR-SSI).* IEEE, 2011, pp.1-6.

[38] Appendix, available at https://db.tt/e8hcGxsR .