T3UC 2009
Sophia-Anitpolis, France

uOttawa

L'Université canadienne
Canada's university

Testing Access Control tools

by Bernard Stepien, Amy Felty, Stan Matwin

(bernard | afelty | stan)@site.uottawa.ca

School of Information Technology and Engineering

Université d'Ottawa | University of Ottawa

www.uOttawa.ca

Motivation

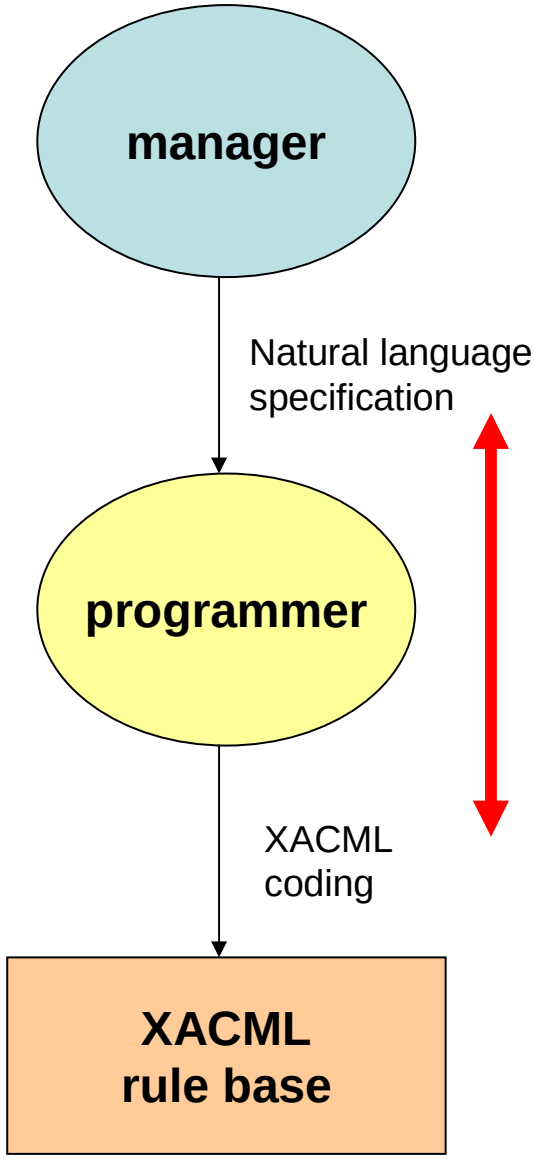
- XACML is a very precise Access Control specification language.
- XACML allows the specification of very complex rule conditions.
- However, the verbosity (XML tags and domains) makes XACML hard to read.
- Thus, it is difficult to ensure that a XACML rule performs the intended requirements.
- Thus, XACML is an ideal candidate for testing.

XACML example

Condition: “It is permitted to buy food or alcohol on a Wednesday or a Thursday if the Balance of the account is over 1000”

```
<Condition  FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
    <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="Merchandise" DataType="http://www.w3.org/2001/XMLSchema#string" />
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">food</AttributeValue>
    </Apply>
    <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="Merchandise" DataType="http://www.w3.org/2001/XMLSchema#string" />
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">alcohol</AttributeValue>
    </Apply>
  </Apply>
  <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
    <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="DayOfTheWeek" DataType="http://www.w3.org/2001/XMLSchema#string" />
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Wednesday</AttributeValue>
    </Apply>
    <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="DayOfTheWeek" DataType="http://www.w3.org/2001/XMLSchema#string" />
      </Apply>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Thursday</AttributeValue>
    </Apply>
  </Apply>
  <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than">
    <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
      <SubjectAttributeDesignator AttributeId="BalanceOfAccount" DataType="http://www.w3.org/2001/XMLSchema#integer" />
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">1000</AttributeValue>
  </Apply>
</Condition>
```

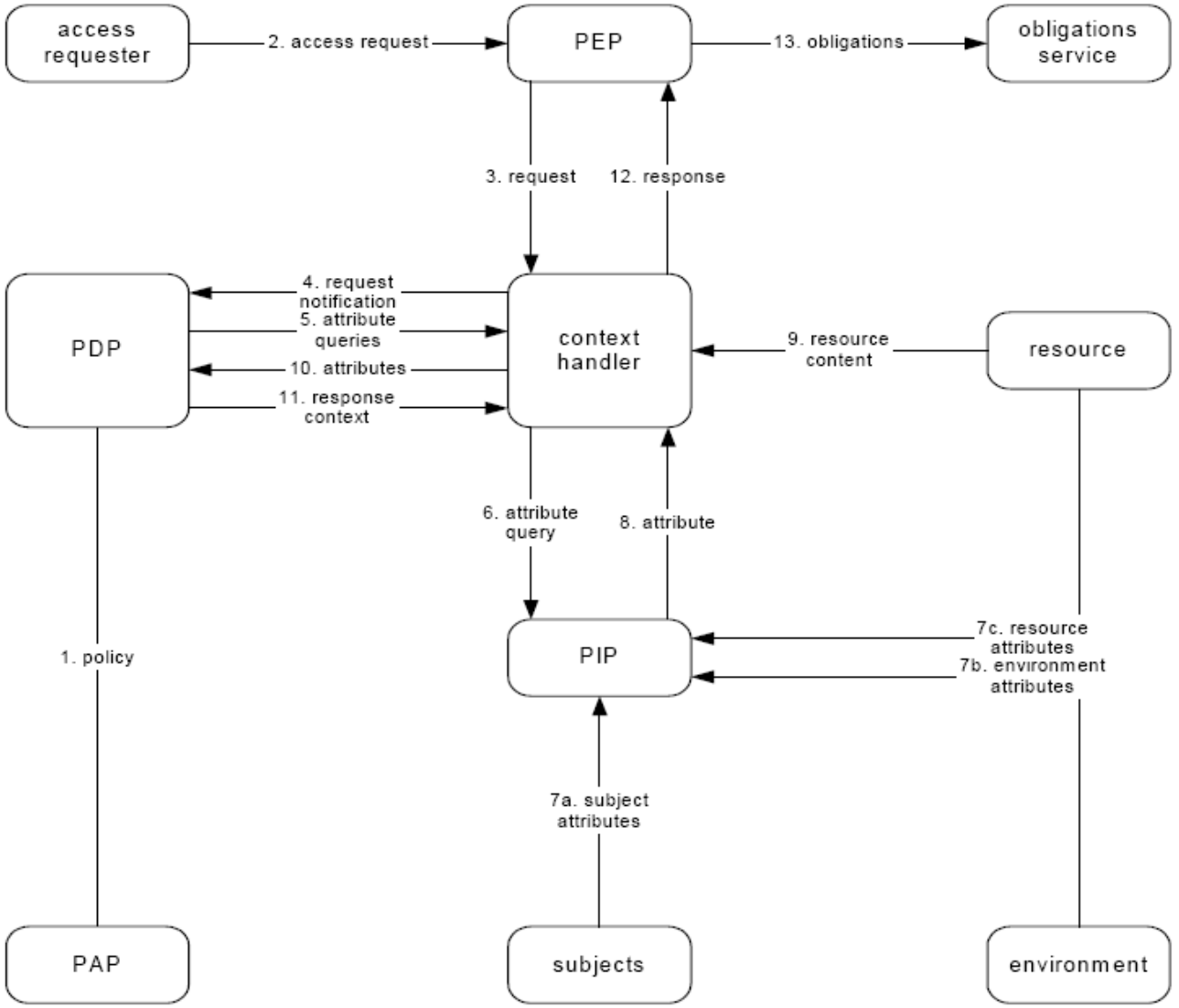
Access control management



1. **Interpretation errors**
2. **Coding errors**
3. **Verification barrier**

- Testing by reducing the complexity of XML.
- TTCN-3 as abstract test requests and responses specification.
- XACML handled by the codec

XACML Access control systems architecture



Characteristics of access control testing

- From a TTCN-3 point of view, it is extremely trivial.
- However the complexity of XACML makes it hard to specify test oracle as TTCN-3 templates.
- Thus, the real challenge is in crafting a template API that abstracts the complexity of XACML.

Categories of XACML Tools

- Access control Policy Decision Points software (**PDP**).
- Access control Policy Administration Points software (**PAP**).

1.Testing access control PDPs

- Testing a PDP server:
  - TTCN-3 test suite sends Access control requests in XML format
  - TTCN-3 test suite receives a response to grant or deny access in XML format.
- Testing a PDP software:
  - Sun microsystems open source tool (sunxacml.jar)
  - TTCN-3 uses procedure oriented communication to invoke directly methods of the software

XACML message request to PDP

```
<Request>
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:Merchandise"
               DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue>Food</AttributeValue>
    </Attribute>
    <Attribute AttributeId="DayOfTheWeek"
               DataType="http://www.w3.org/2001/XMLSchema#string"
               Issuer="admin@users.example.com">
      <AttributeValue>Thrusday</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
               DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>credit_card</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>purchase</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

- Could be handled directly by the TTCN-3 external language interface, here XML

PDP testing data types

```
type record XacmlPDPRequestType {
    charstring policySet,
    XacmlSubjectsType requestedSubjects,
    XacmlResourceType requestedResource,
    XacmlActionType requestedAction
}
```

```
type record XacmlSubjectType {
   charstring attributeId,
   SubjectValueType attributeValue
}
```

```
type union SubjectValueType {
    charstring stringValue,
    …
    DateType dateValue
}
```

```
type record of XacmlSubjectType XacmlSubjectsType;
```

PDP testing templates

- Practically as complex as XACML…

```
template XacmlPDPRequestType t_Request := {
   policySet := "complex_example_6.xml",
   requestedSubjects := {
      {attributeId := "Merchandise", attributeValue := { stringValue := "clothing"}},
      {attributeId := "DayOfTheWeek", attributeValue := { stringValue := "Tuesday"}},
      {attributeId := "BalanceOfAccount", attributeValue := {integerValue := 850}},
      {attributeId := "TimeOfTheDay", attributeValue := {timeValue := {13,0,0}}}
   },
   requestedResource := {"resource-id", "credit_card"},
   requestedAction := {"action-id", "purchase"}
}
```

Designing a templates API

Base template definition

```
template XacmlPDPRequestType t_CreditCardPurchase_Request := {
   policySet := "complex_example_6.xml",
   requestedSubjects := {},
   requestedResource := {"resource-id", "credit_card"},
   requestedAction := {"action-id", "purchase"}
}
```

Constant templates definition

```
template XacmlSubjectType t_DayIs_Tuesday := {
   attributeId := "DayOfTheWeek",
   attributeValue :=  {stringValue := "Tuesday" }
}
```

Parametric templates definition using modifies base template

```
template XacmlPDPRequestType t_PDP_request(XacmlSubjectsType subjects)
                                    modifies t_CreditCardPurchase_Request := {
   requestedSubjects := subjects
}
```

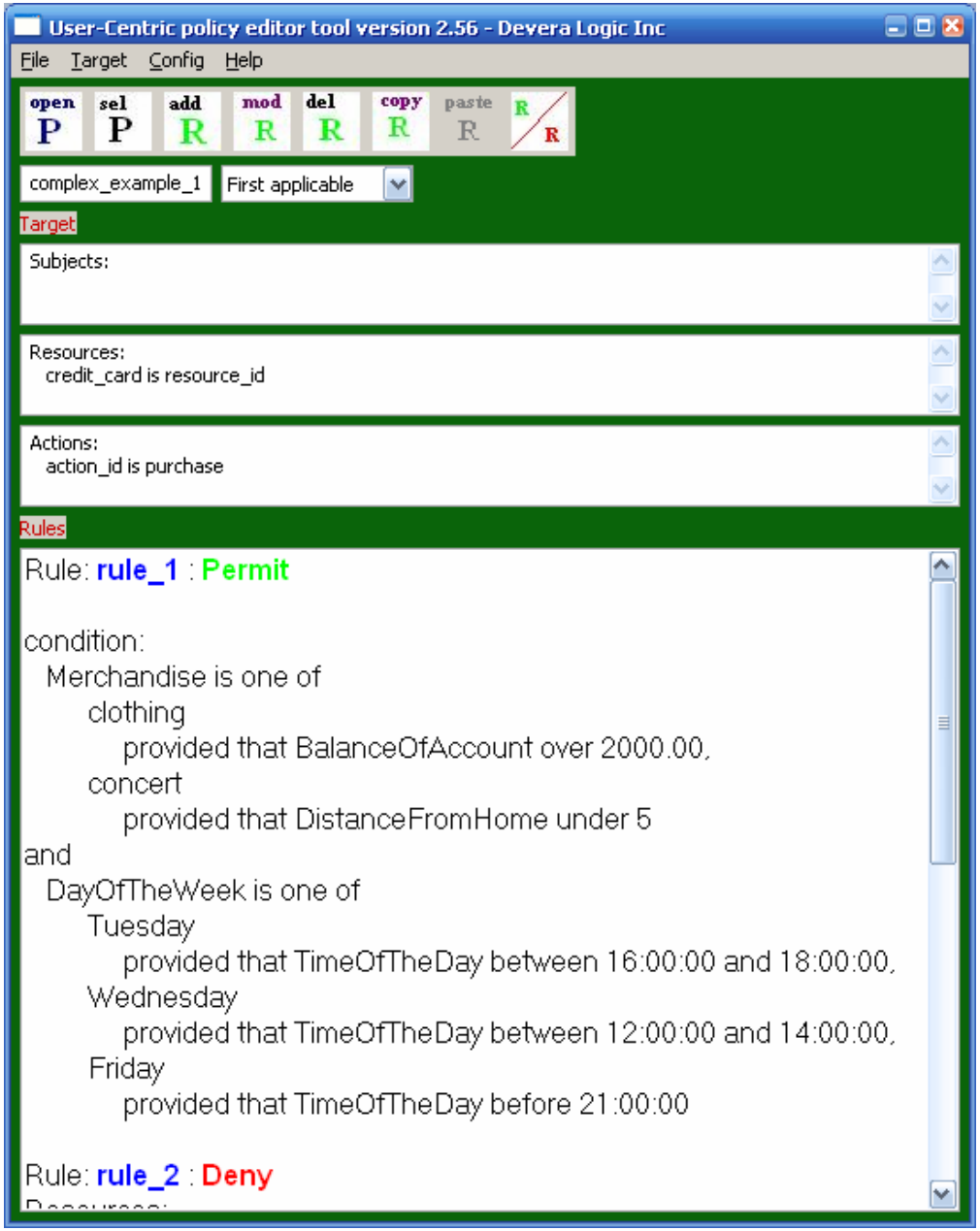
Non-technical user friendliness

```
testcase TC_msg_1() runs on MTCType system SystemType {
   var charstring response;
   map(mtc:sutPort, system:system_sutPort);

   sutPort.send(t_PDP_request({
                                          t_DayIs_Tuesday,
                                          t_MerchandiseIs_Clothing,
                                          t_BalanceOfAccount(850),
                                          t_TimeOfTheDay(13,0,0)
                                       }
                                      )
                     );

   alt{
        [] sutPort.receive("permit") -> value response {
            setverdict(pass);
        }
        [] ...
   }
}
```

2. Testing Access control PAPs

- XACML PAP by University of Ottawa
- Reads policy sets files written in XACML
- Displays XACML policies in a user friendly non XML notation.
- Allows creating new rules or editing existing rules using the user friendly notation.
- Performs conflict detection in rules.

PAP software example



Test purposes

- **Unit testing:**
  - of XACML rule translations into internal representation as a tree.
  - Translation into user-friendly notation
  - Conflict detection checking
- **Regression testing**
  - After adding new features
  - After altering algorithms

Integration testing

- While unit testing is normally simple, the three test purposes are linked and thus must be run in a strict order:
  - Test internal representation of the parsing results.
  - Test the translation into our notation.
  - Test the conflict detection algorithms.
- Thus, there is some flavor of integration testing in this project.

TTCN-3 testing approach

- Using procedure oriented communication using the tools jar file.

```
testcase parseXacmlTest_1() runs on ConflictDetectionComponent
                                system SystemConflictDetectionComponent {

  map(mtc:sutPort, system:systemSutPort);

  sutPort.call(parseRuleFile: {filename := "testing_rules_example"}, 5.0 );
  alt {
     [] sutPort.getreply(parseRuleFile: {"testing_rules_example"}
                                        value {rule_1, rule_2, rule_3, rule_4} ) {
           setverdict(pass);
     }
     [] sutPort.getreply {
          log("wrong parse tree list");
          setverdict(fail);
     }
    [] sutPort.catch(timeout) { setverdict(inconc) }
  } …
```

Test specification goals

- Make testing the XACML policy sets as easy as our user-friendly notation.
- The easier the writing of test specifications, the more confidence we gain in the results of the test.
- Increase testers productivity.

TTCN-3 representation of an access control policy rule condition

- Operators are defined recursively
- A rule condition is an operation

```
type record OperatorType {
    charstring name,
    record of OperatorType arguments optional
}

type record RuleType {
    charstring ruleName,
    charstring effect,
    OperatorType condition
}
```

TTCN-3 rules conditions oracles

```
template OperatorType condition := {
  name := "and",
  arguments := {
    {
      name := "or",
      arguments := {
        { name := "string_equal", arguments := {
            { name := "Merchandise", arguments := omit },
            { name := "food", arguments := omit }
          }
        },
        { name := "string_equal", arguments := {
            { name := "Merchandise", arguments := omit },
            { name := "alcohol", arguments := omit }
          }
        }
      }
    },
    {
      name := "or",
      arguments := {
        { name := "string_equal", arguments := {
            { name := "DayOfTheWeek", arguments := omit },
            { name := "Wednesday", arguments := omit }
          }
        },
        { name := "string_equal", arguments := {
            { name := "DayOfTheWeek", arguments := omit },
            { name := "Thursday", arguments := omit }
          }
        }
      }
    },
    { name := "integer_greater_than",
      arguments := {
        { name := "integer_one_and_only",
          arguments := {
            { name := "BalanceOfAccount",
              arguments := omit }
          }
        },
        { name := "1000", arguments := omit }
      }
    }
  }
}
```

- Potentially as cryptic as XACML itself
- However, the TTCN-3 templates are re-usable.
- This expression can then be greatly reduced

The craft of TTCN-3 templates

- Maximize template parameterization and re-usability

Defining constants

```
template OperatorType constant(charstring theConstant) := {
   name := theConstant,
   arguments := omit
}

template OperatorType food := constant("food");
```

Defining variables

```
template OperatorType merchandiseVar := {
   name := "string_one_and_only",
   arguments := { {name := "merchandise", arguments := omit } }
}
```

Defining comparison operators

```
template OperatorType merchandiseIsFood := {
   name := "string_equal",
   arguments := { merchandiseVar, food }
}
```

Simplified condition templates

```
condition := {
   name := "and",
   arguments := {
         {
            name := "or",
            arguments := {
                merchandiseIsFood,
                merchandiseIsAlcohol
            }
        },
        {
            name := "or",
            arguments := {
                dayOfTheWeekIsWednesday,
                dayOfTheWeekIsThursday
            }
         },
         balanceOfAccountGreaterThan("1000")
    }
}
```

Logical operators template representation

```
template OperatorType _and(OperatorsListType theArguments) := {
        name := "and",
        arguments := theArguments
}
```

```
template OperatorType condition3 :=
  _and({
        _or({
              merchandiseIsFood,
              merchandiseIsAlcohol
            }
        ),
        _or({
              dayOfTheWeekIsWednesday,
              dayOfTheWeekIsThursday
            }
         ),
         balanceOfAccountGreaterThan("1000")
       });
```

Note: in XACML the conjunction and disjunction operators are not binary operators. Instead they operate on lists of arguments.

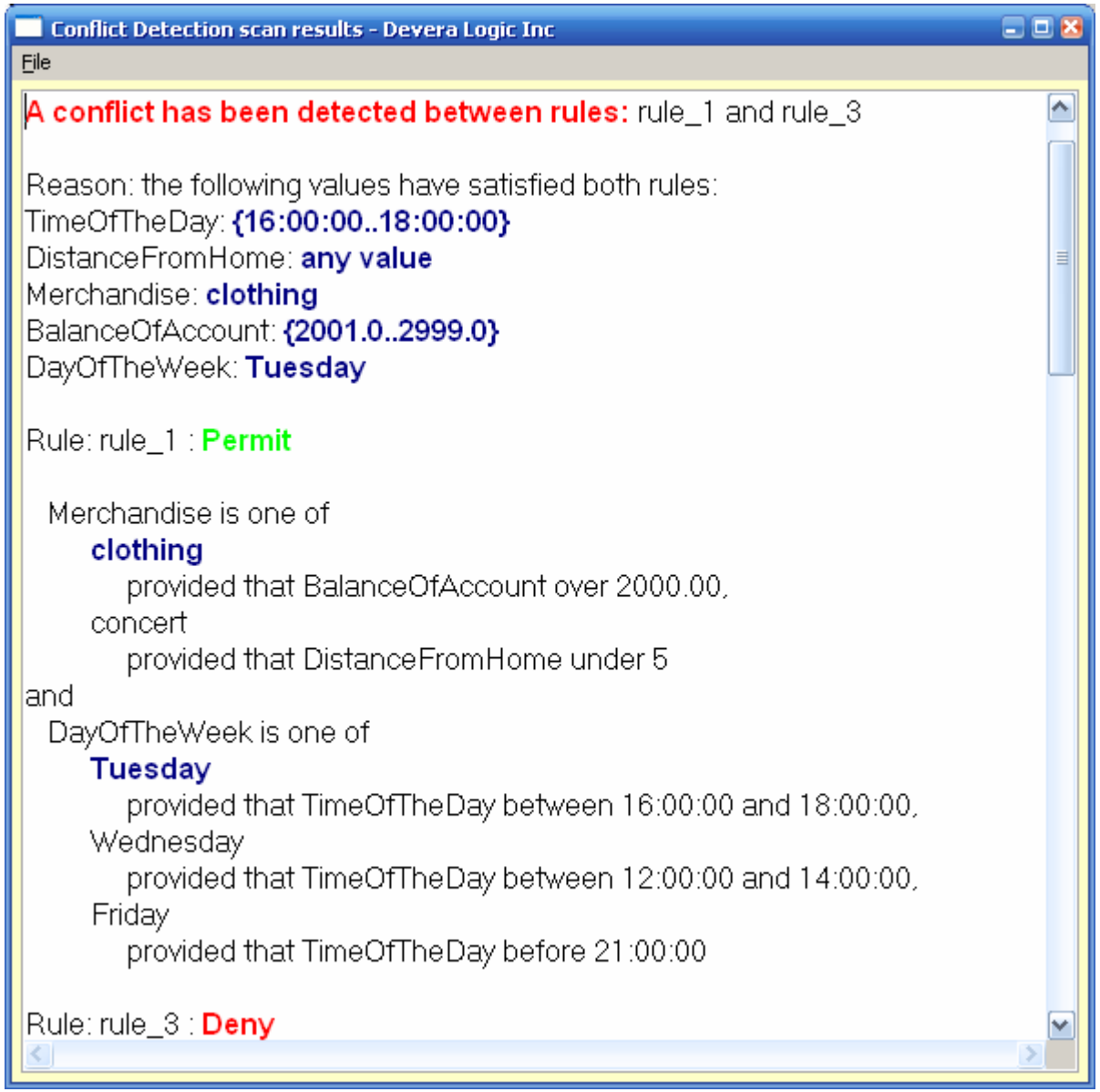
This makes the use of parametric templates possible and powerful

Testing the rendering of the tool's User friendly notation

- The test oracle is the rendering of a XACML rule condition
- Invoke the rendering
- Read the test oracle from a file
- Try to match the rendering result with the file content.

```
   Merchandise is one of food, alcohol
and
   DayOfTheWeek is one of Wednesday, Thursday
and
   BalanceOfAccount over 1000
```

Conflict detection algorithm testing



- Invoke the conflict detection algorithm.
- Try to match the parsed content with a TTCN-3 template.

Advantages of TTCN-3

- XACML rules are very verbose (long tags)
- Even small rules expressed in XACML can be hard to read.
- The TTCN-3 template specification provides an ideal abstraction that can be understood even by non-XACML experts.

Is access control testing really trivial?

- Not really.
- There is more than requests.
- Controlling environment variables makes the problem considerably more complex.
- Thus, this is an interesting research subject.

Conclusions

- TTCN-3 is very powerful
- Strong typing reduces test suite development time.
- The PDP testing data types, codec and test adapter developed in this research constitute a full framework that is re-usable for any XACML PDP testing application.
- Future users of this PDP testing framework can exclusively concentrate on the abstract layer.