A Design Phase for Data Sharing Agreements

Ilaria Matteucci, Marinella Petrocchi, Marco Sbodio, and Luca Wiegand

Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche - Pisa – Italy
&
HP Innovation Center – Torino – Italy

Presenter: Charles Morisset

Outline

- Data Sharing Agreements
- DSA LifeCycle
- DSA Authoring
- DSA Analysis
- Conclusions

Data Sharing Agreements

- Traditionally, collaborating organizations use legal contracts to regulate how data is shared
  - Complex, non standardised, ambiguous documents
  - It is difficult to translate a traditional legal contract into machine understandable data policies
- A Data Sharing Agreement (DSA) aims at being:
  - A human readable contract describing how data is shared
  - A machine processable document that can be automatically analysed and transformed into enforceable policies

DSA Format

Title

Parties

Period

Data

Policies

Date & Signatures

gives a title to the DSA

DSA Format

Title

Parties

Period

Data

Policies

Date & Signatures

defines the parties making the agreement

DSA Format

Title

Parties

Period

Data

Policies

Date & Signatures

specifies the validity period

DSA Format

Title

Parties

Period

Data

Policies

Date & Signatures

lists the data
covered by the DSA

DSA Format

Title

Parties

Period

Data

Policies

Date & Signatures

defines Authorizations, Obligations, and Prohibitions covered by the DSA

DSA Format

Title

Parties

Period

Data

Policies

Date & Signatures

contains the date and the (digital) signatures of the parties

DSA Policies Section

Authorizations: they express the actions that subjects CAN perform on objects

The family doctor can produce/read/integrate medical data of their patients

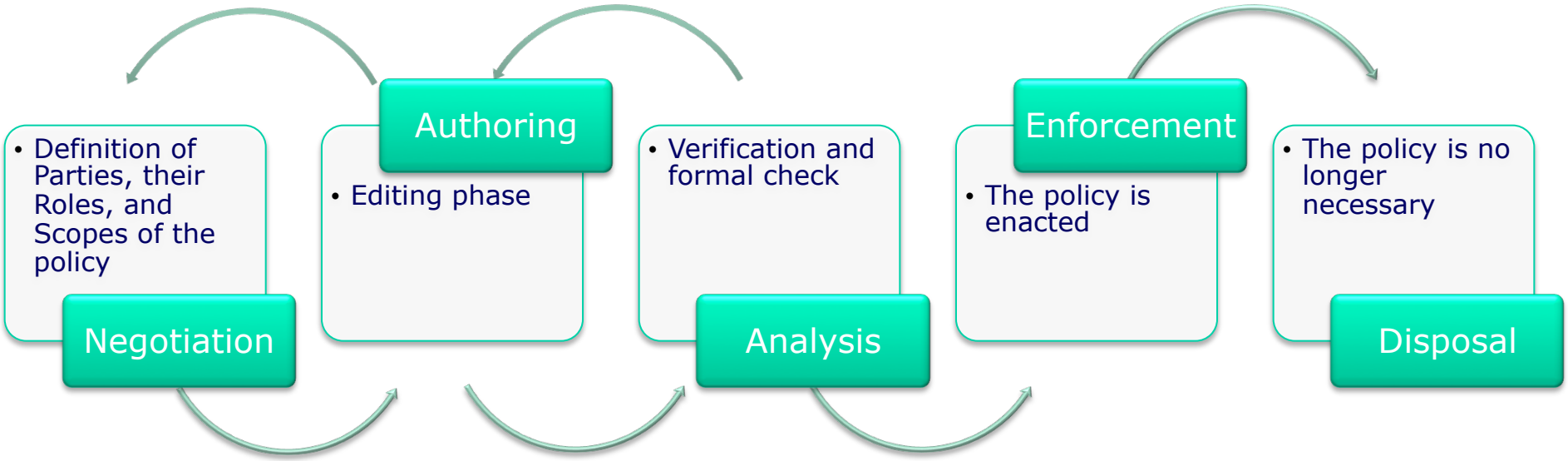
Obligations: actions that subjects MUST perform on objects

After modification of patient medical data, patient must be notified

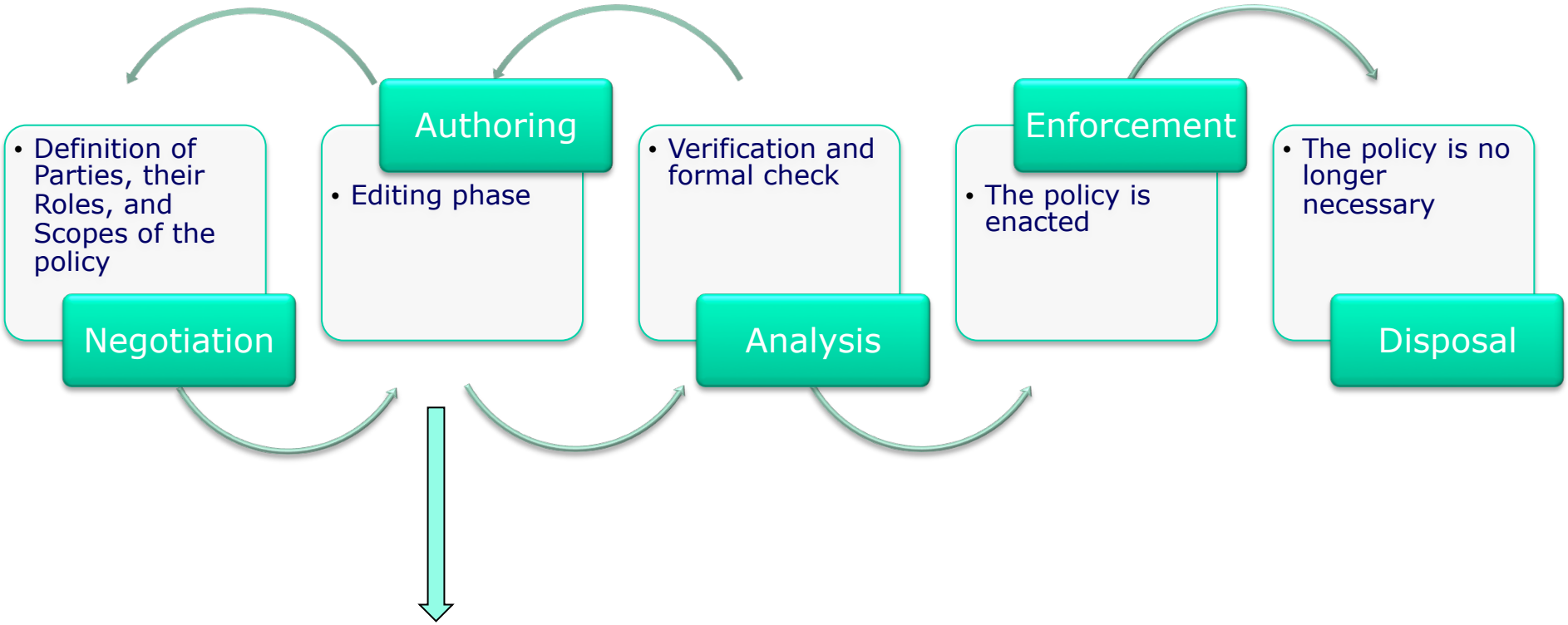
Prohibitions: actions that subjects CANNOT perform on objects

Medical data cannot be modified outside the organization in which they have been created

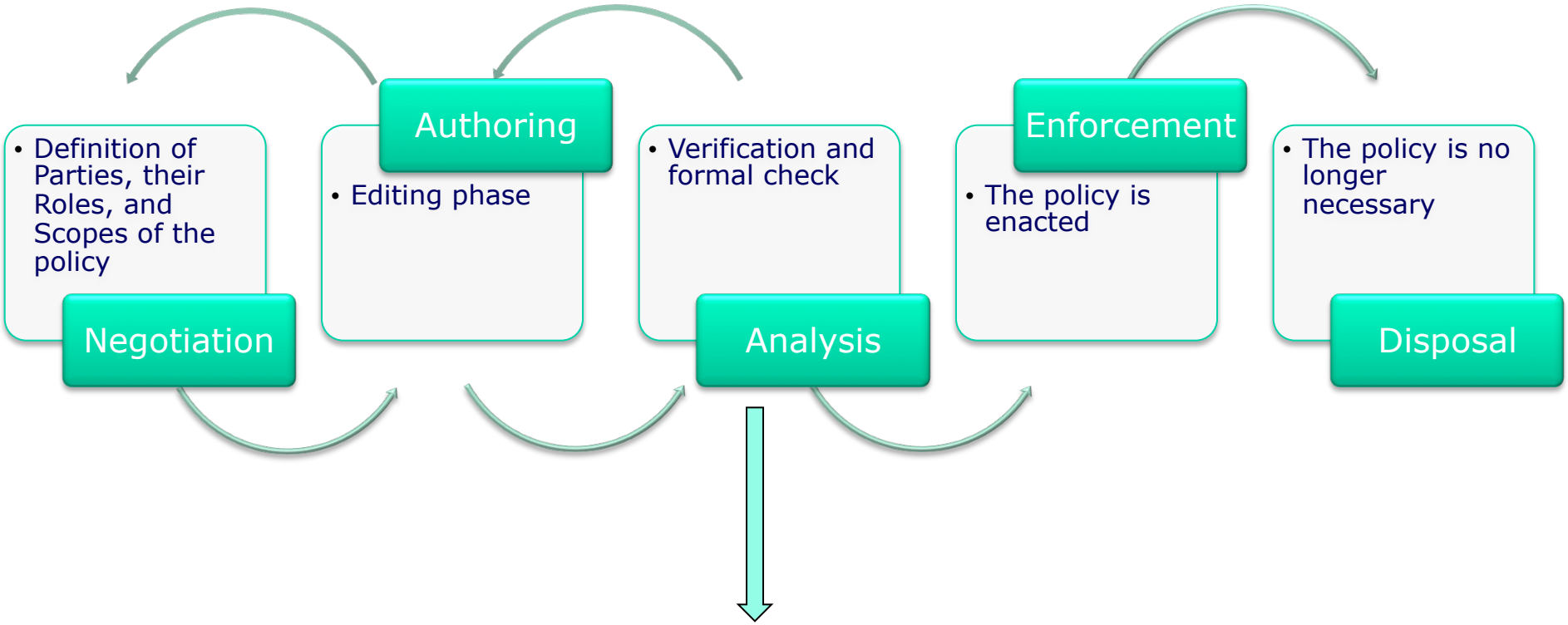
DSA LifeCycle



DSA LifeCycle

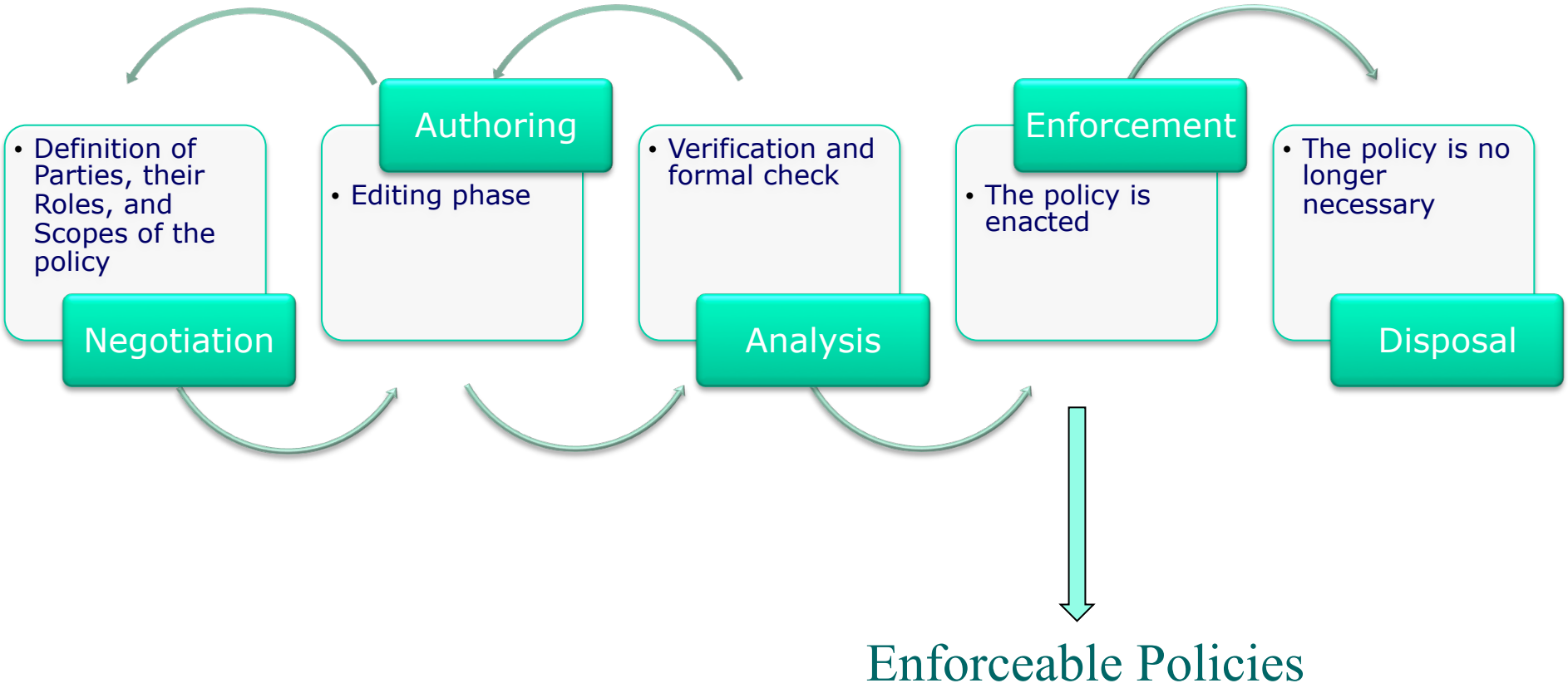


DSA LifeCycle



High-level formal language

DSA LifeCycle



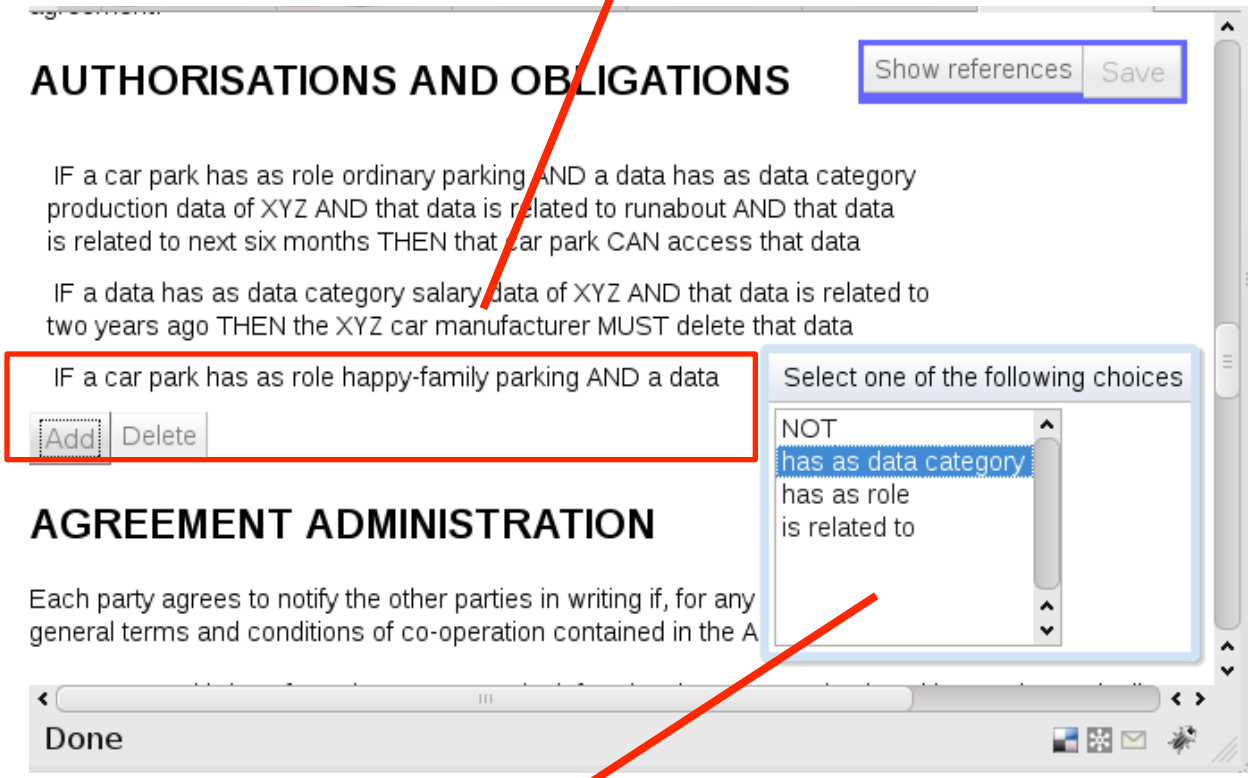
DSA Authoring

- The DSA Authoring Tool is a lightweight Web 2.0 application that:
  - Allows intuitive and interactive creation/ editing of DSAs
  - Uses controlled natural language
  - Saves DSAs in XML
- Benefits
  - Non-technical users can edit DSAs
  - XML DSAs are machine processable, and at the same time, the DSA Authoring Tool can represent them in a human readable way

The DSA Authoring Tool and related technologies are the subject of the International patent application PCT/EP2011/058303 filed by Hewlett-Packard Development Company LP

Authoring: adding a DSA statement

Statement being edited

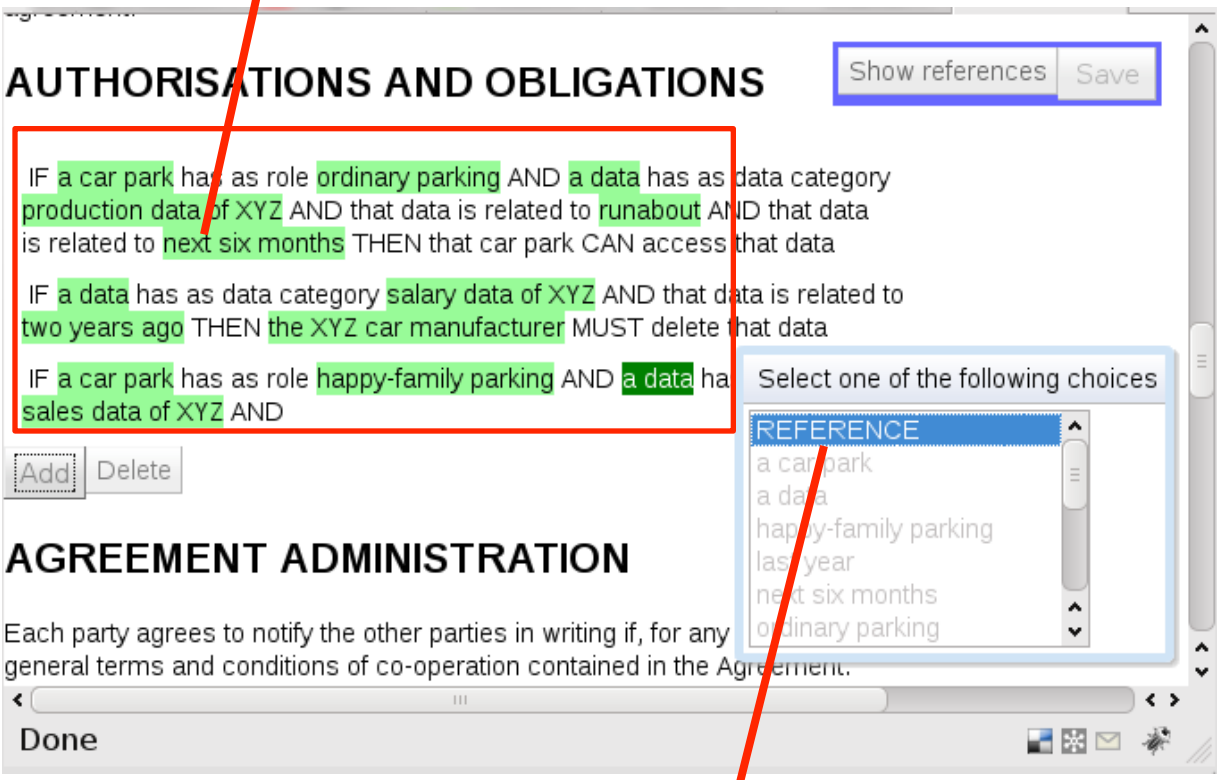


- The user can add terms from a list
- Terms are taken from a controlled vocabulary
- The content of the terms list adapts during the editing (based on previous choices)

List of terms from the controlled vocabulary

Authoring: Adding a reference

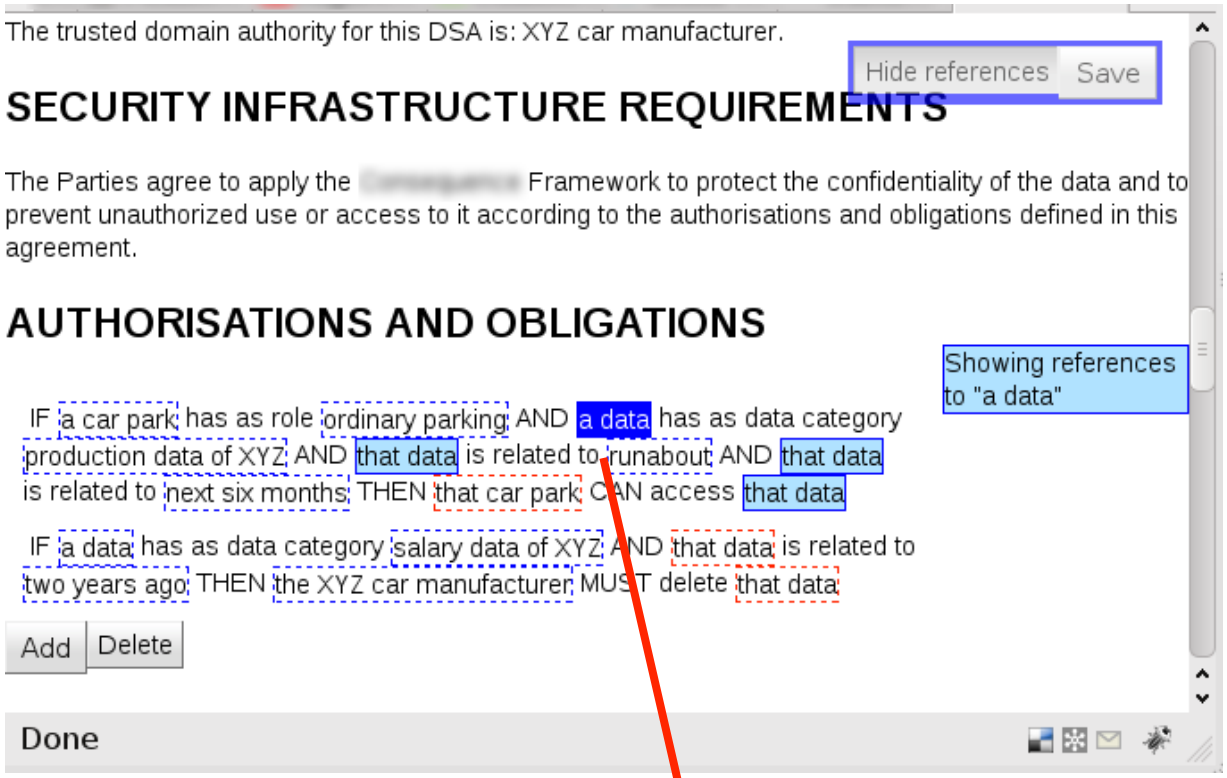
The tool highlights referenceable terms (green)



- During statement creation, the user can refer to previously used terms
- The tool highlights referenceable terms so that the user can simply clicked on the proper one

The user decides to insert a reference

Authoring: showing references



- For complex DSAs it is useful to navigate references
- The tool can help the user in understanding which is the target of a reference

Showing references to a selected item

DSA Analysis: Criticalities

1. Test the policies for concrete scenarios
   - *CAN Alice access the salary data of employees of factory X?*

DSA Analysis: Criticalities

1. Test the policies in a concrete scenario
   - *CAN Alice access the salary data of employees of factory X?*

2. Avoid the arbitrary enforcement of conflictual policies
   - *Car parks outside the European Community CAN access sale data of XYZ car manufacturer*
   - *Car parks outside the European Community CANNOT access sale data of XYZ car manufacturer*

DSA Analysis: Criticalities

1. Test the policies in a concrete scenario
   - *CAN Alice access the salary data of employees of factory X?*

2. Avoid the arbitrary enforcement of conflictual policies
   - *Car parks outside the European Community CAN access sale data of XYZ car manufacturer*
   - *Car parks outside the European Community CANNOT access sale data of XYZ car manufacturer*

First Applicable, Deny-override, Permit-override…?

DSA Analysis Architecture

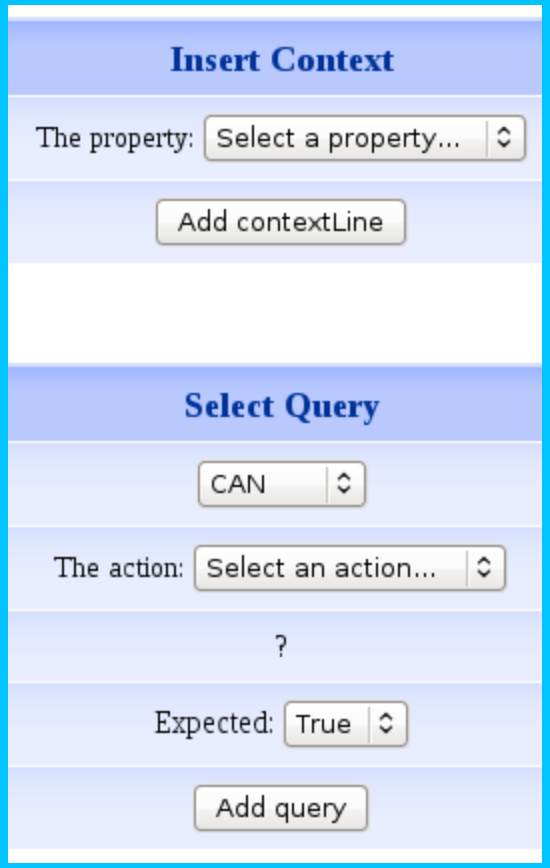
The analysis consists of two components, communicating through service calls

- The Maude analysis engine http://maude.cs.uiuc.edu
- The GUI, designed as a Web Application

http://dev4.iit.cnr.it:8080/DsaAnalyzerWebGUI-0.1/?dsaID=cars.xml

Analysis Architecture

GUI



Context=addContext()
Set(Query) = addQuery()

Maude

Internal Analysis Engine

Set(Results) =
Analyse(Policy, Context, Set(Query)

Maude

- Specification language based on Rewriting Logic
- Distributed systems specified as:
  - Algebraic data types axiomating systems states
  - Rewrite rules axiomating system's local transitions
- Executable, comes with a toolkit that allows formal reasoning on the produced specification (e.g., model checking, theorem proving capabilities are built-in)

Maude modules

- a collection of sorts and operations on them
- the information to reduce and rewrite input expressions of the Maude environment

Functional modules define equations

System modules map transitions of systems into rewrite rules:

```
Mod climate is
   sort wheatercondition .
   op sunnyday : -> wheatercondition .
   op rainyday : -> wheatercondition .
    rl [raincloud] : sunnyday => rainyday .
  endm
```

Policy specification

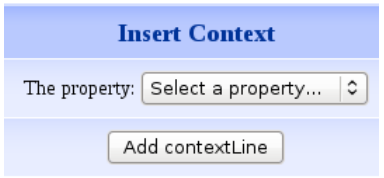
- *"CNL4DSA: a controlled natural language for Data Sharing Agreements".* SAC 2010, Privacy on the Web

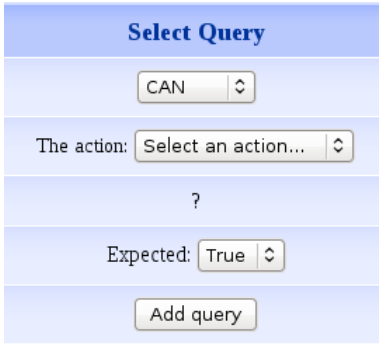
  ***If** (hasRole(user1, doctor) **and** hasDataCategory(data, medical)) **then** **CAN/MUST/CANNOT** modify(user1, data)*

- CNL4DSA has a formal foundation based on a labelled transition system. This allows for a translation to rewriting logic-based languages

- From CNL to Maude: we implement and executable specification of CNL to the Maude language, available: *www.iit.cnr.it/staff/marinella.petrocchi/template.maude*

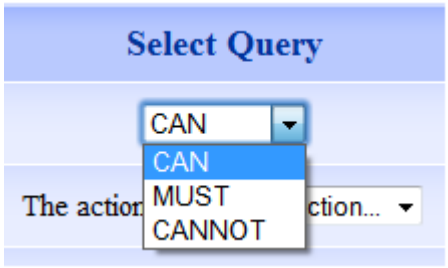
GUI



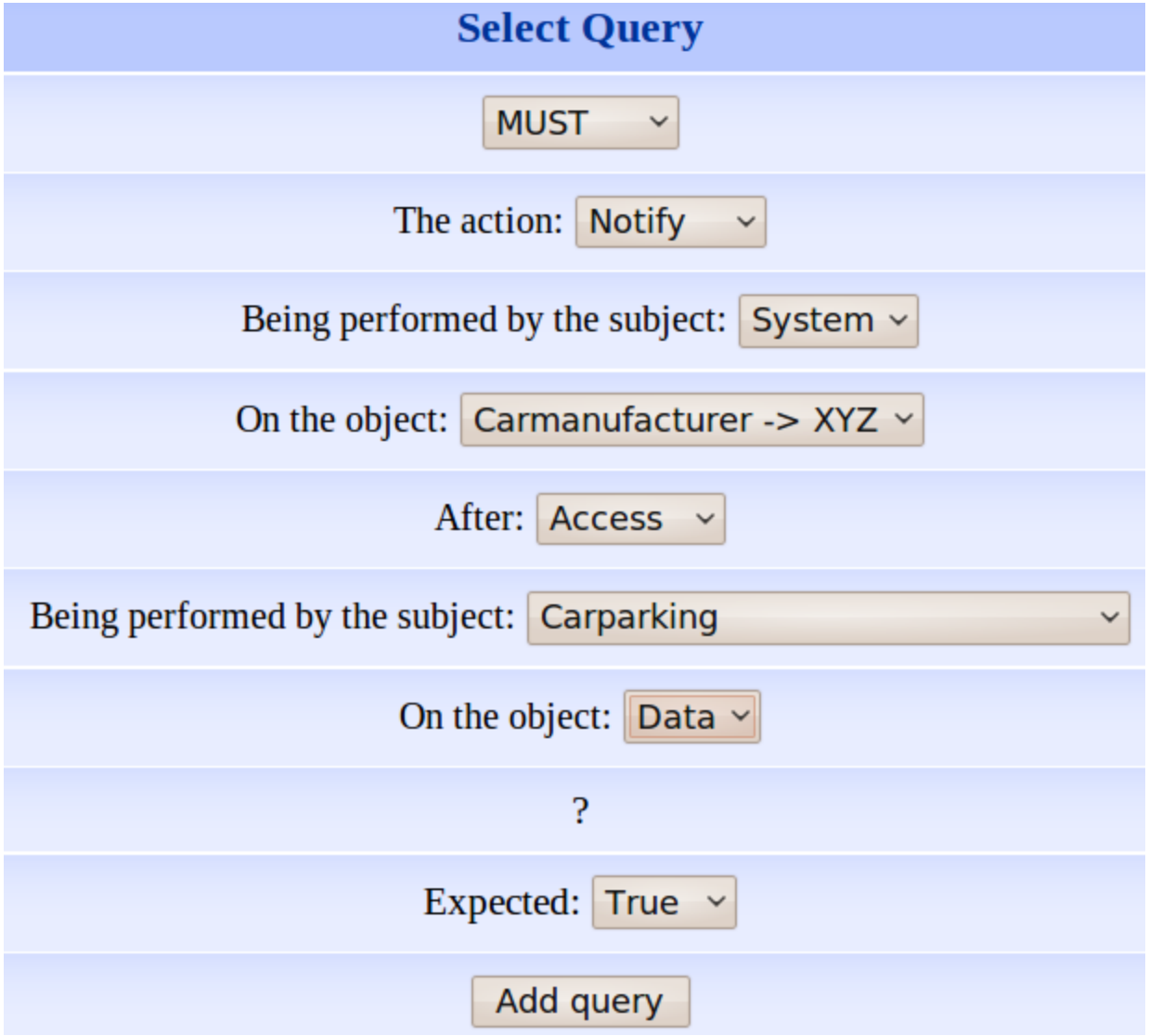


- Allow users to query the analysis engine and visualize the results
- Deployed as a Web Application
- The Maude engine exposes its functionalities as Web Services methods
- GUI retrieves policies and vocabularies from a repository (e.g., servers in the healthcare orgs that store patient data)
- Vocabularies as ontologies
- Help on line available

GUI functionalities (1): Context & Queries Selection

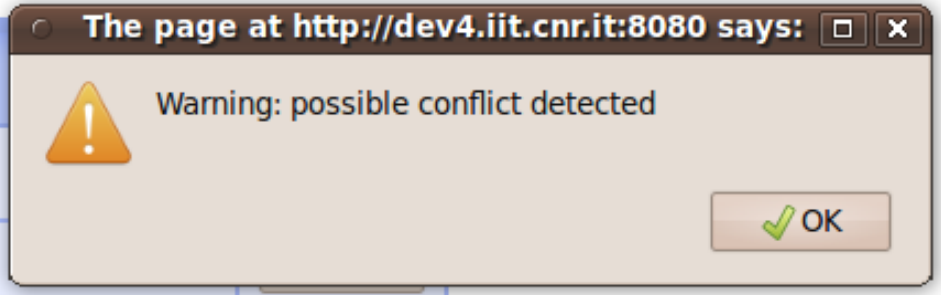


GUI functionalities (2): Queries Composition



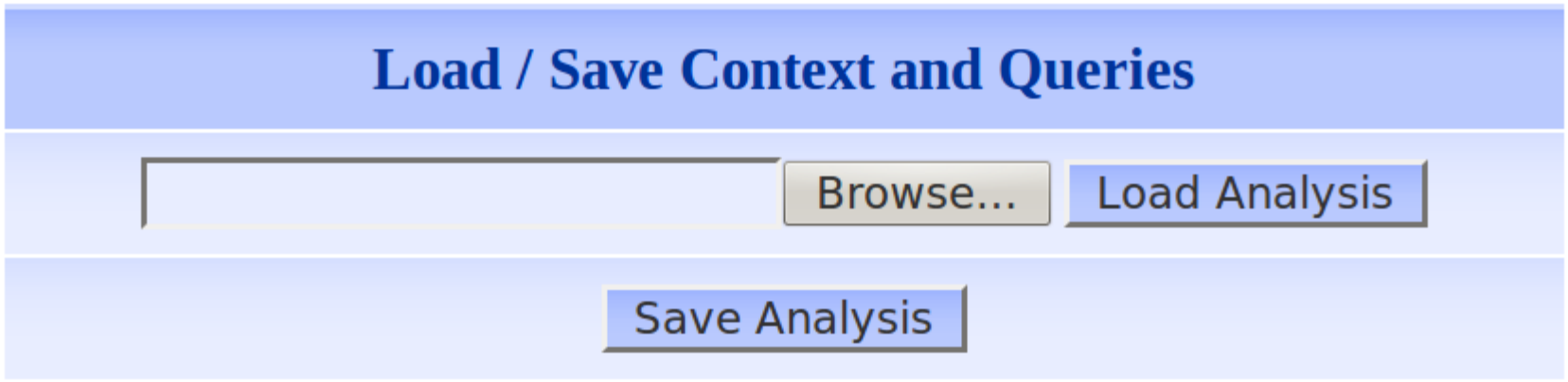
GUI functionalities (3): Conflict Detection

Query	Expected	Result
Can a user1 modify a data?	true	true
Cannot a user1 modify a data?	false	true



GUI functionalities (4)

Save/Load a Configuration



Conclusions

- (User-friendly) specification and analysis framework for a controlled data sharing

(Some) open issues:

- Runtime enforcement of data sharing policies
- Extension to the specification language (e.g., parameterised actions)
- Conflict resolution
- A deeper analysis of social aspects is needed -> usability survey