UNIVERSITÄT
KOBLENZ · LANDAU
Institut für Informatik

FB 4
Informatik

PAPEL: Syntax and Semantics for Provenance-Aware Policy Definition

Christoph Ringelstein
Steffen Staab

Nr. 4/2010

Arbeitsberichte aus dem Fachbereich Informatik

Arbeitsberichte des Fachbereichs Informatik

ISSN (Print): 1864-0346
ISSN (Online): 1864-0850

Herausgeber / Edited by:

Der Dekan:
Prof. Dr. Zöbel

Kontaktdaten der Verfasser

Christoph Ringelstein, Steffen Staab
Institut für Informatik
Fachbereich Informatik
Universität Koblenz-Landau
Universitätsstraße 1
D-56070 Koblenz
EMail: cringel@uni-koblenz.de, staab@uni-koblenz.de

PAPEL: Syntax and Semantics for Provenance-Aware Policy Definition

Christoph Ringelstein and Steffen Staab

WeST - Institute for Web Science and Technologies
University of Koblenz-Landau
Universitaetsstrasse 1, 56070 Koblenz, Germany
{cringel,staab}@uni-koblenz.de
http://west.uni-koblenz.de

Abstract. The processing of data is often restricted by contractual and legal requirements for protecting privacy and IPRs. Policies provide means to control how and by whom data is processed. Conditions of policies may depend on the previous processing of the data. However, existing policy languages do not provide means to express such conditions. In this work we present a formal model and language allowing for specifying conditions based on the history of data processing. We base the model and language on XACML.

Keywords: Context-aware processes, Distributed process execution, Process tracing

1 Introduction

Contracts (e.g. service level agreements) and laws (e.g. privacy laws) entitle customers to control the processing of their data. *Policies are statements of the goals for the behavior of a system* [7] and thus provides means to control the processing of data. However, conditions of policies frequently depend on environmental (or contextual) information. Such policies demand for controlling the process not only with respect to the actual processing step (including actors, processed data, etc.) but also with respect to the history of the processing (e.g. the number of backups that have been made, or who has encrypted the data). In this paper, we focus on policies in business processes containing conditions based on the processing history; as is common in communities dealing with such processing histories, we call the information about processing histories *provenance information*.

To accomplish this task, we need to tackle several principal problems. First, in closed environments the environmental information can be collected with various existing logging mechanisms and be accessed by means of various, often proprietary, solutions. However, as soon as processes span multiple organizations and data is transfered across organizational boundaries, the provenance information must be provided in a standardized manner. The open provenance model (OPM) [10] constitutes such an approach.

Second, the policy conditions must be able to relate to provenance information. Some policy languages allow for building policies containing conditions based on provenance information (e.g. XACML [1] or Rei [8]). However, existing policy mechanisms lack a specification of how to specify or access provenance information. Based on the open provenance model, we introduce a formal language that specifies the relation between policy condition and provenance information. We specify the language by means of an abstract syntax extending the syntactic structure of eXtensible Access Control Markup Language (XACML) and we provide a corresponding description of its execution semantics. We call our approach *PAPEL*: Provenance-Aware Policy definition and Execution Language. To show its feasibility, we implement PAPEL using Datalog.

Third, the provenance information may not be queriable, because the information may not be accessible (e.g. due to log encryption [11]). To be still able to validate compliance with the given policies, the required provenance information must stay accessible. We achieve this by introducing *attributes* and *reduced facts* in PAPEL. Both mechanisms ensure that only a minimum of confidential information is disclosed to third parties.

In this paper, we tackle these problems as follows: First, we introduce a case study, point out problematic issues and derive requirements in Section 2. Then, we discuss the foundations of PAPEL in Section 3, introduce the syntax of PAPEL in Section 4, and define the semantics in Section 5. In Section 6, we discuss the Datalog implementation of our abstract syntax. Finally, we analyze related work and its capabilities to solve the previously introduced problems in Section 7.

2 Case Study and Requirements

In this section, we present a case study depicting issues that occur with policies in distributed environments:

Sharing of Health Records: A research institute of the `University of Koblenz (ukob)` requires data about actual patients for its research. To this end, it cooperates with the hospital `Koblenz Medical Center (kmc)`. The kmc shares its patients' health records with the research institute that can access the records on a server of the hospital. The patients may choose if their data can be used for research or not. The patients' permissions are required before the hospital is allowed to share the data. If patients permit the use, they may additionally demand that their data is only used after their names have been exchanged by pseudonyms.

`Jane Doe` a patient of the `kmc` has specified a list of policies regarding her health record (`record_JD`) and the forwarding of the record to the research laboratory of `ukob`:

- **(A)** *Jane Doe allows* `kmc` *to share her record for research purposes with* `ukob`.
- **(B)** *However, she demands that her record is de-identified before it is shared.*

– **(C)** *With the de-identified record the research laboratory is allowed to do anything, beside transferring it to another organization.*

In addition to the patient's policies, the kmc has own policies for forwarding health records:

– **(D)** *The* `kmc` *demands that the sharing of health records is approved and the approval confirmed before the record is accessed by* `ukob`.

In the case study multiple issues arise. We introduce a selection of these issues in the following list, before we describe what is required to solve the issues.

– **(1)** `ukob` will access health records. To this end, the institute needs to know which actions are permitted or restricted.
– **(2)** The provenance information of a patient may contain personal information. Thus, the `kmc` wants to encrypt the provenance information before forwarding a record. At the same time the `kmc` has to ensure that the information needed to interpret the policies is still available without decryption.

Requirements

In summary we can point out the following technical and organizational requirements:

– **Requirement 'Availability'**: The policies must be available to anybody accessing the associated data (see issue (1)).
– **Requirement 'Expressiveness'**: The used policy language must be able to express permissions and restrictions (see issue (1); cf. XACML [1]).
– **Requirement 'Accessibility'**: The information required by policy conditions must be accessible even if confidential parts of the data and of the provenance information stay hidden (see issue (2)).
– **Requirement 'Updating'**: Rules are required that specify which additional information about the processing must be provided and when it must be updated (see issue (2)).

In addition to access policies such as specified in languages like XACML, we need provenance information to address these issues and thus to meet the requirements. In the following sections we present a solution consisting of three parts: first, the foundations we build on; second, an abstract syntax of our novel policy language PAPEL; and third, the semantics of this language.

3 Foundations of PAPEL

In this section we describe the foundations PAPEL is based on. As discussed above, we require to model provenance information and policies. To this end, we make use of the *Open Provenance Model* and of the *eXtensible Access Control Markup Language*. To integrate both models we introduce an abstract syntax extending the abstract syntax of XACML.

3.1 Open Provenance Model

The *Open Provenance Model (OPM)* [10] defines a process as an *action or series of actions performed on or caused by artifacts, and resulting in new artifacts.* We represent the information which is given by a graph structure in the OPM by a set of primitives. The transformation leads to provenance information represented by our abstract syntax, which is specified in Table 1. We use the `step` primitive:

```
step (Data, Actors, InvolvedAgents, Category, Purpose, ID, PIDs)
```

to express the following constituents of OPM:

- `Data` is the *artifact* processed during the execution of the processing step. In OPM an artifact is defined as an *immutable piece of state, which may have [..] a digital representation in a computer system.*
- `Actors` are the *agents* controlling the process, In OPM an agent is defined as a *contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, affecting its execution.*
- `InvolvedAgents` are *agents* involved in a process that do not trigger the processing (e.g. the receiver of a transfer or access rights).
- `ID` is the unique *identifier* of the processing step (as defined by OPM[1]).
- `PIDs` are the unique *identifiers* of the directly preceding processing steps.

The `ID` together with the `PIDs` creates a partial order of processing steps. Beside these constituents of OPM, the `step` primitive also specifies the following properties of a processing step:

- `Category` is the category of the processing step. The possible categories are defined in a domain specific ontology.
- `Purpose` is the purpose of executing the processing step. The purpose of a processing step is defined in domain specific ontologies.

Example 1: In the following we depict the provenance information of a processing step of the `Koblenz Medical Center (kmc)`: *In the depicted processing step the health record of Jane Doe is shared*[2] *for research purposes with the* `University of Koblenz (ukob)`:

```
step (record_JD, {kmc}, {ukob}, share, research, 3, {2})
```

[1] In the examples we make use of simple integers, to increase readability.

[2] In this and the following examples we assume that used concepts are defined in a domain ontology provided by kmc

3.2 eXtensible Access Control Markup Language

We use The *eXtensible Access Control Markup Language (XACML)* [1] as a starting point for our formal model. XACML is a standard defining a XML based policy framework. XACML supports three policy elements, i.e. permission (*permit*), restriction (*deny*) and obligation, that we need to fulfill the Requirement 'Expressiveness'. To generalize from XACML and to integrate with the provenance information, we introduce an abstract syntax for permission, restrictions and obligations based on XACML in Section 4 below.

XACML policies define rules by connecting a set of subjects (actors) with a set of targets (data) and by specifying the conditions of the rule. If the conditions are fulfilled, XACML rules result in a given effect, which is *permit* or *deny*. In addition, a rule may require that certain obligations are met before a permission is granted. However, this kind of obligations is expressed as part of conditions in PAPEL (see Example 3) exploiting the provenance information.

4 Syntax of PAPEL

To evaluate rules, the provenance information can be used as source of information about the previous processing. PAPEL defines an abstract syntax (see Table 1) to express provenance information and polices rules based on provenance information.

The provenance information is provided by means of logging or monitoring mechanisms (cf. [11, 12]). In PAPEL, we use primitives to define provenance information (`primitive` is the start symbol of the provenance part of the PAPEL syntax). We use the `step` primitive as introduced in Section 3 (see Example 1) and the `reduced` primitive to specify the processing provenance and the `attribute` primitive to specify attributes and their value (see Table 1). The processing steps, attribute assignments and reduced facts collected during the execution of a process build the processing history, which we define as follows:

Definition 1: *We define a history H as a conjunction of positive facts specifying processing steps, attribute assignments and reduced facts.*

Policy rules are provided by the person concerned or IPR owner. In PAPEL, we use rules to specify policies (`rule` is the start symbol of the policy part of the PAPEL syntax). We distinguish three types of rules: `permit`, `deny` and `assignment` rules. All three types may depend on a `condition` and have the parameter ID, which refers to the actual processing step. Assignment rules specify the change of an attribute value by use of the `set_attribute` primitive.

Permissions and Restrictions

We model permissions as rules specifying which kinds of processing steps are permitted. Likewise, restrictions are modeled as rules specifying which processing

PAPEL Syntax for Provenance Information:

Syntax Element	EBNF syntax
`Primitive`	`Step | ReducedFact | Attribute ;`
`Step`	`"step ("Data", "Actors", "InvolvedAgents", "Category", "Purpose", "ID", "PIDs")." ;`
`ReducedFact`	`"reduced ("Data", "(Actors | "hidden")", "(InvolvedAgents | "hidden")", "(Category | "hidden")", "(Purpose | "hidden")", "ID", "PIDs")." ;`
`Attribute`	`"attribute ("Data", "Name", "Value", "ID")." ;`

PAPEL Syntax for Policies:

Syntax Element	EBNF syntax
`Rule`	`Permission | Restriction | Assignment ;`
`Permission`	`"permit (ID) IF " Condition "." ;`
`Restriction`	`"deny (ID) IF " Condition "." ;`
`Assignment`	`"assignment (ID) IF " Condition "DO" SetAttribute | SetReducedFact "." ;`
`Condition`	`Primitive | ("(NOT" Primitive | Condition | "permit (ID)" | "deny (ID)" ")") | ("(" Primitive | Condition | "permit (ID)" | "deny (ID)" BooleanOperator Primitive | Condition | "permit (ID)" | "deny (ID)" ")") | (Step | ReducedFact "AFTER" Step | ReducedFact ")") ;`
`SetAttribute`	`"set_attribute ("Data", "Name", "Value", "ID")." ;`
`SetReducedFact`	`"set_reduced ("Data", "(Actors | "hidden")", "(InvolvedAgents | "hidden")", "(Category | "hidden")", "(Purpose | "hidden")", "ID", "PIDs")." ;`
`BooleanOperator`	`"AND" | "OR" | "XOR" ;`

Table 1. Syntax of PAPEL.

steps are prohibited. The rules consist of two parts: the name, which indicates the type of the policy (`permit` or `deny`), and the body of the rule, which defines the conditions and obligatory processing steps. Conditions are used to express dependencies between policies and environmental information and are specified after the `IF` statement (see Table 1).

Example 2: This example formalizes policy (A): *All entities are denied to transfer the data* `record_1`*, beside* `kmc` *that is allowed for transferring the data for research purposes to* ***`ukob`****:*

```
permit (ID) IF step (record_JD, {kmc}, {ukob}, transfer,
               research, ID, _).
deny (ID)   IF step (record_JD, _, _, transfer, _, ID, _) AND
               NOT permit (ID).
```

The second rule of this example denies the transfer of `record_JD`. By means of the additional part of the condition `AND NOT permit (ID)` we can define ex-

ceptions to this rule. In the example the exception is defined by the first rule and allows `kmc` to transfer the record to `ukob`.

Example 3: The following example depicts the implementation of policy (B) of the running example (see Section 2). *The patient demands that her* `record_1` *is de-identified before it is transfered:*

```
permit (ID) IF step (record_JD, _, _, transfer, _, ID, {PID}) AND
            step (record_JD, _, _, update, de-identify, PID, _).
```

By using variables (in the example `PID`), we can specify that the approval step has to be performed *directly* before the processing step, which should be permitted. If the variable is not used, any preceding approval step will fulfill the condition.

Example 4: In this example we express the policy (C) from our running example: *All processing steps that are not transfer actions will be permitted if they are performed by* `ukob`:

```
permit (ID) IF (step (_, {ukob}, _, Category, _, ID, _) AND
            NOT (Category = transfer)).
```

In the example above we make use of the logical constant `ukob`, which is defined in a domain ontology. In addition, we use the variable `Category` and unnamed variables indicated by `_`. The `_` represents another unnamed variable each time it is used. An unnamed variable matches all possible values of its type (cf. existential quantification).

Condition Statements

In PAPEL, condition statements can be composed using logical operators (`NOT`, `AND`, `OR`, `XOR` and `AFTER`) How these operators can be combined is specified in Table 1. In addition, Table 1 depicts, that parentheses are used to specify the interpretation order of complex statements. We define the semantics of the operators in Table 2.

Example 5: The following example illustrates the implementation of policy (D) from the running example. The policy is specified by means of a permission in combination with a condition: *If the approval of the access has been confirmed, permit the transfer:*

```
permit (ID) IF (step (R, {ukob}, _, access, _, ID, _) AND
            (step (R, {kmc}, _, _, confirmation, _, _) AFTER
            step (R, {kmc}, _, _, access_approval, _, _))).
```

In difference to Example 3the rule in this example uses `AFTER`. By means of the `AFTER` construct we can access the partial order of the processing steps in the history. The variable `R` assures that the approval and confirmation is given for the health record that should be accessed.

Attributes and Reduced Facts

As stated above the provenance information can be used as source of information about the previous processing. However, the provenance information may contain sensitive data (e.g. the provenance information that the cancer medication has been adjusted). Thus, some logging approaches provide security mechanisms like the encryption of the provenance information (e.g. [11]). To maintain the ability to check the compliance with a policy, a method is needed to enable access to relevant information or provide the information directly (Requirement 'Accessibility'). To this end, we introduce *attributes* as well as *reduced facts*.

Attributes can be used to specify provenance information that can be expressed by a value (e.g. de-identification status, modification counter, etc.). We specify attributes by the `attribute` primitive (see Table 1). The `attribute` primitive has four parameters, the `Data` item the attribute relates to, the `Name` of the attribute, its assigned `Value`, and the identifier `ID` of the processing step, when the assignment was performed. The value of the attribute is assigned by the actor performing the processing step. The assignment is done according to the `assignment` rule defined by the creator of the policies.

Example 6: This example illustrates the permit rule and the assignment rule required to implement policy (B) of the running example by means of an attribute `de-identified`. In difference to the implementation of policy (B) we introduced above (see Example 3), this implementation allows for de-identifying the health record at any time: *The patient demands that her record (`record_1`) is de-identified before it is transfered:*

```
permit(ID) IF (step (record_JD, _, _, transfer, _, ID, _) AND
           attribute (record_JD, de-identified, true, ID)).
```

If the executed processing step replaces all names in `record_1` with pseudonyms, set the attribute named `de-identified` *to* `true`*, and if the record is re-identified, set the attribute to* `false`*:*

```
assignment(ID) IF step (record_JD, _, _, _, de-identified, ID, _)
               DO set_attribute (record_JD, de-identified, true, ID).
assignment(ID) IF step (record_JD, _, _, _, re-identified, ID, _)
               DO set_attribute (record_JD, de-identified, false, ID).
```

A reduced fact is a description of a processing step that is reduced to the necessary and contains only the required information. Which information is hidden is specified by means of assignments that are defined by the creator of the

policy. Reduced facts are used if attributes are not expressive enough (e.g. has the last de-identification been performed by a specific actor). A reduced fact is added to the provenance information. In difference to the full description, the information of the reduced fact must not be encrypted to allow for querying the information. The adding of the reduced fact is achieved by the actor performing the processing step following assignments defined by the policy creator.

The parameters `Data`, `Actors`, `InvolvedAgents`, `Category`, and `Purpose` are replaced by the empty placeholders `hidden` as required. The parameters `ID` and `PIDs` must not be replaced. These two parameters are required to reproduce the (partial) order of processing steps.

Example 7: In the following we depict the provenance information of a processing step of the department for `nuclear medicine`: *The processing step updates the health record by adding a new cancer medication:*

```
step (record_JD, {nuclear_medicine}, {},
      update, new_cancer_medication, 3, {2})
```

If this information is not relevant to the analysis of `ukob`*, the* `kmc` *will encrypt the original information about the processing step and will provide reduced provenance information:*

```
reduced (record_JD, hidden, hidden, update, hidden, 3, {2})
```

5 Execution Semantics of PAPEL

In this section we define the semantics of PAPEL. The semantics of PAPEL specifies if execution steps of a given history violate a given set of policy rules. To this end, we define a Tarskian semantics mapping syntactic elements of PAPEL onto subsets and relations over a universe U. Based on this semantics we define when a set of policies is fulfilled with respect to a history H (see Definition 1). First we define minimal models of a history H:

Definition 2: *We define a minimal model M of the history H as a model to which no strictly smaller Herbrand model of the history H exist [9].*

A minimal model of a history is only a model of the history and of subparts of the history. Based on the definition of a minimal model of a history we define when a set of policies is fulfilled:

Definition 3: *We define that a set of policy rules R is fulfilled with respect to a history H if each minimal model M of the history H is also a model of the set of policy rules R.*

The Universe and Basic Axioms

The universe U is the set: $U = \Delta \cup A \cup X \cup \Psi \cup \Phi \cup N \cup V$, where Δ is the set of artifacts (e.g. data), A is the set of agents (including actors), X is the set of categories, Ψ is the set of purposes, Φ is the set of processing step identifiers, N is the set of attribute identifiers, and V the set of attribute values. These subsets of U are mutually disjoint.

For the following definitions be I a partial interpretation function and be $P(S)$ the power set of S. We define I to map atomic elements of PAPEL onto the (parts of) our universe U as follows: $\delta^I \in \Delta$, $\alpha^I \subseteq A$, $\beta^I \subseteq A$, $\chi^I \in X$, $\psi^I \in \Psi$, $id^I \in \Phi$, $\rho^I \subseteq \Phi$, $\mu^I \in N$, and $\nu^I \in V$. The predicate representing the processing steps *step* is interpreted as: $step^I \subseteq \Delta \times P(A) \times P(A) \times X \times \Psi \times \Phi \times P(\Phi)$ and the predicate representing the attribute assignments *attribute* is interpreted as: $attribute^I \subseteq \Delta \times N \times V \times \Phi \times P(\Phi)$.

Thereby, the partial interpretation function I must satisfy the following constraints: Each identifier id^I must clearly identify one processing step:

$$\forall id^I \in \Phi : (\delta_1^I,\ \alpha_1^I,\ \beta_1^I,\ \chi_1^I,\ \psi_1^I,\ id^I,\ \rho_1^I), (\delta_2^I,\ \alpha_2^I,\ \beta_2^I,\ \chi_2^I,\ \psi_2^I,\ id^I,\ \rho_2^I) \in step^I \Rightarrow \delta_1^I = \delta_2^I \wedge \alpha_1^I = \alpha_2^I \wedge \beta_1^I = \beta_2^I \wedge \chi_1^I = \chi_2^I \wedge \psi_1^I = \psi_2^I \wedge \rho_1^I = \rho_2^I.$$

Analogously, at each processing step specified by id^I each attribute has exactly one value at a time:

$$\forall id^I \in \Phi : (\delta^I,\ \mu^I,\ \nu_1^I,\ id^I,\ \rho_1^I), (\delta^I,\ \mu^I,\ \nu_2^I,\ id^I,\ \rho_2^I) \in attribute^I \Rightarrow \nu_1^I = \nu_2^I \wedge \rho_1^I = \rho_2^I.$$

The processing history is defined as partial order $> \subseteq \Phi \times \Phi$ of processing steps, reduced facts, and attribute assignments. A tuple $(id^I,\ pid^I)$ is element of $>$, if and only if the processing step specified by *pid* precedes the processing step specified by *id*. Before we define $>$, we define the relation $\succ \subseteq \Phi \times \Phi$ of directly preceding processing steps, reduced facts, and attribute assignments. A processing step s_{dpid} immediately precedes a processing step s_{id}, if s_{id} is the successor of s_{dpid}:

$$\succ = \{(id^I,\ dpid^I) | \exists(\delta_s^I,\ \alpha_s^I,\ \beta_s^I,\ \chi_s^I,\ \psi_s^I,\ id^I,\ \rho_s^I) \in step^I \wedge dpid^I \in \rho_s^I \vee \exists(\mu_a^I,\ \nu_a^I,\ id^I,\ \rho_a^I) \in attribute^I \wedge dpid^I \in \rho_a^I\}.$$

A processing step precedes another processing step, if a trace of consecutive processing steps exists that connects both steps:

$$> = \{(id^I,\ pid^I) | \succ\ (id^I,\ pid^I) \vee \exists mid^I :\succ (id^I, mid^I) \wedge > (mid^I, pid^I)\}.$$

In the following, we extend the interpretation function I by the interpretation of non-atomic syntactic expressions of PAPEL.

Processing Steps, Reduced Facts and Attributes

Processing Step: Be _ an unspecified parameter and be $s' = (\delta',\ \alpha',\ \beta',\ \chi',\ \psi',\ id',\ \rho')$, the *step* predicate is interpreted as follows:

$$(step(\delta,\ \alpha,\ \beta,\ \chi,\ \psi,\ id,\ \rho))^I = \begin{cases} \text{true} & \text{if } \exists s' \in step^I : (\delta^I = \delta'^I \vee \delta^I = \_) \wedge \\ & (\alpha^I = \alpha'^I \vee \alpha^I = \_) \wedge\ ..., \\ \text{false} & \text{else.} \end{cases}$$

Reduced Fact: The *reduced* relation is interpreted as follows:

$$(reduced(\delta_r,\ \alpha_r,\ \beta_r,\ \chi_r,\ \psi_r,\ id_r,\ \rho_r))^I = \begin{cases} \text{true} & \text{if } \exists(\delta_s^I,\ \alpha_s^I,\ \beta_s^I,\ \chi_s^I,\ \psi_s^I,\ id_s^I, \\ & \rho_s^I) \in step^I :\ id_r^I = id_s^I \wedge \\ & (\delta_r^I = \delta_s^I \vee \delta_r^I = hidden) \wedge\ .., \\ \text{false} & \text{else.} \end{cases}$$

Setting of Reduced Facts: Be $(\delta_s,\ \alpha_s,\ \beta_s,\ \chi_s,\ \psi_s,\ id_s,\ \rho_s) \in step$ the processing step described by the reduced fact, be $id_r = id_s$ and $\rho_r = \rho_s$, and be $\delta_r = \delta_s \vee \delta_r = hidden$, $\alpha_r = \alpha_s \vee \alpha_r = hidden$, $\beta_r = \beta_s \vee \beta_r = hidden$, $\chi_r = \chi_s \vee \chi_r = hidden$, and $\psi_r = \psi_s \vee \psi_r = hidden$. The *set_reduced* predicate is interpreted as follows:

$$(set\_reduced(\delta_r,\ \alpha_r,\ \beta_r,\ \chi_r,\ \psi_r,\ id_r,\ \rho_r))^I = \begin{cases} \text{true} & \text{if } (\delta_r^I,\ \alpha_r^I,\ \beta_r^I,\ \chi_r^I,\ \psi_r^I, \\ & id_r^I,\ \rho_r^I) \in reduced^I, \\ \text{false} & \text{else.} \end{cases}$$

Attributes: The *attribute* relation is interpreted as follows:

$$(attribute(\delta, n, v, id, \rho))^I = \begin{cases} \text{true} & \text{if } (\delta^I, n^I, v^I, id^I, \rho^I) \in attribute^I, \\ \text{true} & \text{if } (\delta^I, n^I, v^I, id^I, \rho^I) \notin attribute^I \wedge \\ & \exists(\delta^I, n^I, v^I, id_l^I, \rho_l^I) \in attribute^I \wedge > (id^I, id_l^I) \wedge \\ & \neg\exists(\delta_m^I, n_m^I, v_m^I, id_m^I, \rho_m^I) \in \Phi :> (id^I, id_m^I) \wedge \\ & > (id_m^I, id_l^I), \\ \text{false} & \text{else.} \end{cases}$$

An attribute has the value assigned in the actual step, or if no value is assigned in the actual step, it will have the value that has been assigned last.

Setting of Attributes: The *set_attribute* predicate is interpreted as follows:

$$(set\_attribute(\delta, n, v, id, \rho))^I = \begin{cases} \text{true} & \text{if } \forall fid^I : \succ (fid^I, id^I) \rightarrow (\delta^I, n^I, v^I, fid^I, \\ & \{id^I\}) \in attribute^I, \\ \text{false} & \text{else.} \end{cases}$$

The result of the interpretation of the *set_attribute* predicate will be true if the value of the attribute is updated in all directly succeeding processing steps.

Logical expressions

Above we introduced the following syntactical elements `NOT`, `AND`, `OR`, and `XOR`. Their semantics are defined in the same manner as the corresponding boolean expressions (see Table 2). For example the $not \subseteq \{true,\ false\}$ relations is defined as:

$$(not(e))^I = \begin{cases} \text{true} & \text{if } e^I = false, \\ \text{false} & \text{else.} \end{cases}$$

In addition, we introduce the syntactical element `AFTER`, which is defined as the following relation: $after^I \subseteq (step^I \cup reduced^I) \times (step^I \cup reduced^I)$, where:

$$(after(s_1, s_2))^I = \begin{cases} \text{true} & \text{if } s_1 = (.., id_1, ..) \wedge s_2 = (.., id_2, ..) \wedge id_1 > id_2, \\ \text{false} & \text{else.} \end{cases}$$

Syntax	Natural language semantics
`A`	A is true
`NOT A`	A is not true
`A AND B`	A and B are true
`A OR B`	A or B (non-exclusive or) are true
`A XOR B`	either A or B (exclusive or) is true
`B AFTER A`	first A and then B (in the given order[3]) are true
`IF Condition`	if the *Condition* is fulfilled
`DO ObligatoryActions`	the *ObligatoryActions* have to be performed

Table 2. Condition Statements.

Permission, restriction and assignment

Be L the set of functions implementing logical expressions. Permissions, restrictions, and assignments are functions $permit^I : \Phi \mapsto \{true,\ false\}$, $deny^I : \Phi \mapsto \{true,\ false\}$, and $assignment^I : \Phi \mapsto \{true,\ false\}$ as follows:

[3] The processing history can be represented as a directed graph (see Section 5). Thus, processing steps can be in order if they can be connected by a path in the graph.

$permit : \Phi \mapsto \{true,\ false\}$ is a function that will return $true$ if the processing step identified by ID is permitted. If it is not permitted, it will return $false$. Be $p_1,\ p_2,\ ..,\ p_n \in L$ the functions implementing the conditions of all rules defining permissions. The predicate $permit$ is interpreted as follows:

$$(permit(ID))^I = \begin{cases} \text{true} & \text{if } \exists\ (\delta^I,\ \alpha^I,\ \beta^I,\ \chi^I,\ \psi^I,\ id^I,\ \rho^I) \in step^I : \\ & ID^I = id^I \wedge (p_1^I \vee p_2^I \vee .. \vee p_n^I) = true \\ \text{false} & \text{else} \end{cases}$$

$deny : \Phi \mapsto \{true,\ false\}$ is a function that will return $true$ if the processing step identified by ID is *not* denied. If it is denied, it will return $false$. Be $d_1,\ d_2,\ ..,\ d_m \in L$ the functions implementing the conditions of all rules defining restrictions and be $s = (\delta,\ \alpha,\ \beta,\ \chi,\ \psi,\ id,\ \rho)$, $deny$ is interpreted as follows::

$$(deny(ID))^I = \begin{cases} \text{false} & \text{if } \exists\ s^I \in step^I : ID^I = id^I \wedge\ (d_1^I \vee d_2^I \vee .. \vee d_m^I) = true \\ \text{true} & \text{else} \end{cases}$$

$assignment : \Phi \mapsto \{true,\ false\}$ is a function that will return $true$ if the processing step identified by ID does not violate an assignment. If it violates an assignment, it will return $false$. Be $c_1,\ c_2,\ ..,\ c_k \in L$ the functions implementing the conditions of all rules defining assignments and be $a_1,\ a_2,\ ..,\ a_k$ the $set\_attribute$ and $set\_reduced$ predicates of all rules defining assignments, $assignment$ is interpreted as follows:

$$(assignment(ID))^I = \begin{cases} \text{true} & \text{if } \exists\ s^I \in step^I : ID^I\ =\ id^I \wedge \\ & ((\neg c_1^I \vee o_1^I) \wedge (\neg c_2^I \vee o_2^I) \wedge .. \wedge (\neg c_k^I \vee o_k^I)) = true \\ \text{false} & \text{else} \end{cases}$$

Fulfilling Policies

Policies will be fulfilled with respect to a history if each minimal model of a history is also a model of the policies (see Definition 3). This definition in combination with the PAPEL semantics have the following conclusion: A processing history will fulfill a set of policy rules, if all processing steps in the history are permitted with respect to the permissions ($permit^I$) and not prohibited with respect to the restrictions ($deny^I$) and if no assignment ($assignment^I$) has been violated.

6 Datalog Implementation of PAPEL

Implementing PAPEL, we have demonstrated its feasibility. In the implementation the provenance information is provided as database. To access the provenance information we make use of the database query language Datalog. We have

chosen Datalog to provide a general implementation with a formal grounding. In our implementation, we use facts to specify the provenance information and rules to query it.

To validate a set of policies the following preparation steps are required: (1) All Datalog rules required to specify syntactical elements are added to the database; (2) A Datalog fact is generated for each processing step, which is described by the provenance information. The facts are added to the database; (3) The Datalog rules specifying the policies are added to the database; (4) A Datalog fact specifying the processing step that should be performed is added to the database; and (5) For each attribute Datalog facts for each processing step are generated and added to the database.

Then, the database is queried by means of a rule, which specifies the query if the processing step identified by the given ID will be allowed or not.

7 Related Work

Many policy languages exist that are applicable for our purpose. Thus, we based PAPEL on these existing languages. We have decided to use the main policy elements 'restriction' (deny) and 'permission' (permit), as defined in the *eXtensible Access Control Markup Language (XACML)* [1]. Polices, which are based on complex environmental knowledge such as provenance information, can not be specified using XACML. With our policy language we extend the expressiveness of XACML conditions to cover information about the processing history of data.

As a foundation for more complex policy conditions we used the work of Kagal et al. [8]. They present a domain-centric and entity-centric policy language named Rei. Rei provides a mechanism for modeling of speech acts and delegation of policies. The conditions in Rei are specified by means of Prolog. However, Rei does not define how to model conditions based on provenance information. We extend the work of Rei by defining a formalism for conditions based on provenance information and by providing a model-theoretical semantics.

Other work which is related to our work is the *Web Services Policy 1.5 - Framework (WS-Policy)* [2] that provides a model and syntax to specify policies about entities in a Web services-based system. As WS-Policy is used to specify the policies about entities, e.g. Web services, and not of the processed data, it is not in the target of the WS-Policy framework to model conditions based on provenance information.

In Table 3, we give an overview of further related work in the field of policy languages. The Table compares the following properties of of policy languages: Are policies directly linked to a piece of data? Does the policy language allow for expressing dataflow polices and/or access control policies? Do policy conditions allow for relating to provenance information (processing traces)? Does the policy language allow for protecting confidential provenance information. Finally, we compare if the language has a syntax definition, a formal semantics and an implementation.

Property \ Approach	PAPEL	XACML [1]	EPAL [4]	WS-Policy [2]	Rei [8]	XrML [13]	Casandra [5]	e-Wallet [6]	IFAudit [3]
Linked to Pieces of Data	✓	-	-	-	-	✓	-	✓	-
Dataflow Policies	✓	-	-	-	-	-	-	-	✓
Access Control Policies	✓	✓	✓	(1)	✓	✓	✓	✓	-
Provenance Information	✓	-	-	-	-	-	-	-	✓
Protection of Provenance Information	✓	-	-	-	-	-	-	-	-
Syntax Definition	✓	✓	✓	✓	✓	✓	✓	✓	✓
Formal Semantics	✓	-	✓	-	(✓)	-	✓	(2)	✓
Implementation	✓	✓	✓	✓	✓	✓	✓	✓	✓

(1) WS-Policy provides data usage policies.

(2) e-Wallet provides a formal policy processing algorithm.

Table 3. Related Work.

However, all of them provide additional aspects of policy languages that are not in our focus. Some policy approaches provide methods to enforce policy compliance in closed environments, like organizations (cf. [4]) or data silos (cf. [6]). Other languages define policies of actors (cf. [2]) not resources. Finally some languages consider credentials to gain access rights (cf. [13, 5]), support roles and role delegation (cf. [5]), or provide algorithms to identify violated policies (cf. [3]).

8 Conclusion and Future Work

Existing policy languages provide means to control how and by whom data is processed. The conditions of policies may depend on the previous processing of the data. Such policies demand for controlling the process with respect to the history of the processing. However, existing policy languages do not specify how to access the provenance information about the previous processing. In this work we have introduced PAPEL a provenance-aware policy definition and execution language motivated by XACML. PAPEL allows for policy conditions to relate to provenance information.

We have presented the abstract syntax of PAPEL, which extends the syntactic structures of XACML and OPM. In addition, we have provided a corresponding description of its execution semantics and we have sketched an implementation of PAPEL using Datalog.

In addition we have discussed how data protection can hamper the interpretation of policy conditions. We have presented means to validate compliance with given policies even if the required provenance information is confidential. At the same time the confidential methods of PAPEL ensure that only a minimum of confidential information is disclosed to third parties.

Acknowledgment

The basic idea of this work has been motivated by discussion with Prof. Marianne Winslett and her working group. This work has been supported by the European projects Where eGovernment meets the eSociety (WeGov, FP7-248512) and Knowledge Sharing and Reuse across Media (X-Media, FP6-26978) funded by the Information Society Technologies (IST) 6th and 7th Framework Programme.

References

1. eXtensible Access Control Markup Language (XACML) Version 2.0. Oasis standard, OASIS, February 2005.
2. Web Services Policy 1.5 - Framework. W3c recommendation, W3C, September 2007.
3. Rafael Accorsi and Claus Wonnemann. Auditing workflow executions against dataflow policies. In *BIS 2010: Proceedings of the 13th International Conference on Business Information Systems*, 2010.
4. P Ashley, S Hada, G Karjoth, C Powers, and M Schunter. Enterprise Privacy Authorization Language (EPAL 1.2). Submission to w3c, W3C, November 2003.
5. Moritz Y. Becker and Peter Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, page 159, Washington, DC, USA, 2004. IEEE Computer Society.
6. Fabien L. Gandon and Norman M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *J. Web Sem.*, 1(3):241–260, 2004.
7. Heather M. Hinton and E. Stewart Lee. The compatibility of policies. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 258–269, New York, NY, USA, 1994. ACM.
8. Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. *IEEE International Workshop on Policies for Distributed Systems and Networks*, 0:63–75, 2003.
9. John Wylie Lloyd. *Foundations of Logic Programming*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1993.
10. Luc Moreau, Juliana Freire, Joe Futrelle, Robert Mcgrath, Jim Myers, and Patrick Paulson. The open provenance model: An overview. *Provenance and Annotation of Data and Processes*, 5272:323–326, 2008.
11. C. Ringelstein and S. Staab. Logging in Distributed Workflows. In *Proceedings of the Workshop on Privacy Enforcement and Accountability with Semantics*, Busan, South-Korea, 2007.
12. Christoph Ringelstein and Steffen Staab. Dialog: Distributed auditing logs. In *IEEE International Conference on Web Services*, pages 429–436, Los Angeles, CA, USA, 2009. IEEE Computer Society.
13. Xin Wang, Guillermo Lao, Thomas DeMartini, Hari Reddy, Mai Nguyen, and Edgar Valenzuela. Xrml – extensible rights markup language. In *XMLSEC '02: Proceedings of the 2002 ACM workshop on XML security*, pages 71–79, New York, NY, USA, 2002. ACM.

Bisher erschienen

Arbeitsberichte aus dem Fachbereich Informatik

(http://www.uni-koblenz.de/fb4/publikationen/arbeitsberichte)

Christoph Ringelstein, Steffen Staab, PAPEL: Syntax and Semantics for Provenance-Aware Policy Definition, Arbeitsberichte aus dem Fachbereich Informatik 4/2010

Nadine Lindermann, Sylvia Valcárcel, Harald F.O. von Kortzfleisch, Ein Stufenmodell für kollaborative offene Innovationsprozesse in Netzwerken kleiner und mittlerer Unternehmen mit Web 2.0, Arbeitsberichte aus dem Fachbereich Informatik 3/2010

Maria Wimmer, Dagmar Lück-Schneider, Uwe Brinkhoff, Erich Schweighofer, Siegfried Kaiser, Andreas Wieber, Fachtagung Verwaltungsinformatik FTVI Fachtagung Rechtsinformatik FTRI 2010, Arbeitsberichte aus dem Fachbereich Informatik 2/2010

Max Braun, Ansgar Scherp, Steffen Staab, Collaborative Creation of Semantic Points of Interest as Linked Data on the Mobile Phone, Arbeitsberichte aus dem Fachbereich Informatik 1/2010

Marc Santos, Einsatz von „Shared In-situ Problem Solving“ Annotationen in kollaborativen Lern- und Arbeitsszenarien, Arbeitsberichte aus dem Fachbereich Informatik 20/2009

Carsten Saathoff, Ansgar Scherp, Unlocking the Semantics of Multimedia Presentations in the Web with the Multimedia Metadata Ontology, Arbeitsberichte aus dem Fachbereich Informatik 19/2009

Christoph Kahle, Mario Schaarschmidt, Harald F.O. von Kortzfleisch, Open Innovation: Kundenintegration am Beispiel von IPTV, Arbeitsberichte aus dem Fachbereich Informatik 18/2009

Dietrich Paulus, Lutz Priese, Peter Decker, Frank Schmitt, Pose-Tracking Forschungsbericht, Arbeitsberichte aus dem Fachbereich Informatik 17/2009

Andreas Fuhr, Tassilo Horn, Andreas Winter, Model-Driven Software Migration Extending SOMA, Arbeitsberichte aus dem Fachbereich Informatik 16/2009

Eckhard Großmann, Sascha Strauß, Tassilo Horn, Volker Riediger, Abbildung von grUML nach XSD soamig, Arbeitsberichte aus dem Fachbereich Informatik 15/2009

Kerstin Falkowski, Jürgen Ebert, The STOR Component System Interim Report, Arbeitsberichte aus dem Fachbereicht Informatik 14/2009

Sebastian Magnus, Markus Maron, An Empirical Study to Evaluate the Location of Advertisement Panels by Using a Mobile Marketing Tool, Arbeitsberichte aus dem Fachbereich Informatik 13/2009

Sebastian Magnus, Markus Maron, Konzept einer Public Key Infrastruktur in iCity, Arbeitsberichte aus dem Fachbereich Informatik 12/2009

Sebastian Magnus, Markus Maron, A Public Key Infrastructure in Ambient Information and Transaction Systems, Arbeitsberichte aus dem Fachbereich Informatik 11/2009

Ammar Mohammed, Ulrich Furbach, Multi-agent systems: Modeling and Virification using Hybrid Automata, Arbeitsberichte aus dem Fachbereich Informatik 10/2009

Andreas Sprotte, Performance Measurement auf der Basis von Kennzahlen aus betrieblichen Anwendungssystemen: Entwurf eines kennzahlengestützten Informationssystems für einen Logistikdienstleister, Arbeitsberichte aus dem Fachbereich Informatik 9/2009

Gwendolin Garbe, Tobias Hausen, Process Commodities: Entwicklung eines Reifegradmodells als Basis für Outsourcingentscheidungen, Arbeitsberichte aus dem Fachbereich Informatik 8/2009

Petra Schubert et. al., Open-Source-Software für das Enterprise Resource Planning, Arbeitsberichte aus dem Fachbereich Informatik 7/2009

Ammar Mohammed, Frieder Stolzenburg, Using Constraint Logic Programming for Modeling and Verifying Hierarchical Hybrid Automata, Arbeitsberichte aus dem Fachbereich Informatik 6/2009

Tobias Kippert, Anastasia Meletiadou, Rüdiger Grimm, Entwurf eines Common Criteria-Schutzprofils für Router zur Abwehr von Online-Überwachung, Arbeitsberichte aus dem Fachbereich Informatik 5/2009

Hannes Schwarz, Jürgen Ebert, Andreas Winter, Graph-based Traceability – A Comprehensive Approach. Arbeitsberichte aus dem Fachbereich Informatik 4/2009

Anastasia Meletiadou, Simone Müller, Rüdiger Grimm, Anforderungsanalyse für Risk-Management-Informationssysteme (RMIS), Arbeitsberichte aus dem Fachbereich Informatik 3/2009

Ansgar Scherp, Thomas Franz, Carsten Saathoff, Steffen Staab, A Model of Events based on a Foundational Ontology, Arbeitsberichte aus dem Fachbereich Informatik 2/2009

Frank Bohdanovicz, Harald Dickel, Christoph Steigner, Avoidance of Routing Loops, Arbeitsberichte aus dem Fachbereich Informatik 1/2009

Stefan Ameling, Stephan Wirth, Dietrich Paulus, Methods for Polyp Detection in Colonoscopy Videos: A Review, Arbeitsberichte aus dem Fachbereich Informatik 14/2008

Tassilo Horn, Jürgen Ebert, Ein Referenzschema für die Sprachen der IEC 61131-3, Arbeitsberichte aus dem Fachbereich Informatik 13/2008

Thomas Franz, Ansgar Scherp, Steffen Staab, Does a Semantic Web Facilitate Your Daily Tasks?, Arbeitsberichte aus dem Fachbereich Informatik 12/2008

Norbert Frick, Künftige Anfordeungen an ERP-Systeme: Deutsche Anbieter im Fokus, Arbeitsberichte aus dem Fachbereicht Informatik 11/2008

Jürgen Ebert, Rüdiger Grimm, Alexander Hug, Lehramtsbezogene Bachelor- und Masterstudiengänge im Fach Informatik an der Universität Koblenz-Landau, Campus Koblenz, Arbeitsberichte aus dem Fachbereich Informatik 10/2008

Mario Schaarschmidt, Harald von Kortzfleisch, Social Networking Platforms as Creativity Fostering Systems: Research Model and Exploratory Study, Arbeitsberichte aus dem Fachbereich Informatik 9/2008

Bernhard Schueler, Sergej Sizov, Steffen Staab, Querying for Meta Knowledge, Arbeitsberichte aus dem Fachbereich Informatik 8/2008

Stefan Stein, Entwicklung einer Architektur für komplexe kontextbezogene Dienste im mobilen Umfeld, Arbeitsberichte aus dem Fachbereich Informatik 7/2008

Matthias Bohnen, Lina Brühl, Sebastian Bzdak, RoboCup 2008 Mixed Reality League Team Description, Arbeitsberichte aus dem Fachbereich Informatik 6/2008

Bernhard Beckert, Reiner Hähnle, Tests and Proofs: Papers Presented at the Second International Conference, TAP 2008, Prato, Italy, April 2008, Arbeitsberichte aus dem Fachbereich Informatik 5/2008

Klaas Dellschaft, Steffen Staab, Unterstützung und Dokumentation kollaborativer Entwurfs- und Entscheidungsprozesse, Arbeitsberichte aus dem Fachbereich Informatik 4/2008

Rüdiger Grimm: IT-Sicherheitsmodelle, Arbeitsberichte aus dem Fachbereich Informatik 3/2008

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik 2/2008

Markus Maron, Kevin Read, Michael Schulze: CAMPUS NEWS – Artificial Intelligence Methods Combined for an Intelligent Information Network, Arbeitsberichte aus dem Fachbereich Informatik 1/2008

Lutz Priese,Frank Schmitt, Patrick Sturm, Haojun Wang: BMBF-Verbundprojekt 3D-RETISEG Abschlussbericht des Labors Bilderkennen der Universität Koblenz-Landau, Arbeitsberichte aus dem Fachbereich Informatik 26/2007

Stephan Philippi, Alexander Pinl: Proceedings 14. Workshop 20.-21. September 2007 Algorithmen und Werkzeuge für Petrinetze, Arbeitsberichte aus dem Fachbereich Informatik 25/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS – an Intelligent Bluetooth-based Mobile Information Network, Arbeitsberichte aus dem Fachbereich Informatik 24/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS - an Information Network for Pervasive Universities, Arbeitsberichte aus dem Fachbereich Informatik 23/2007

Lutz Priese: Finite Automata on Unranked and Unordered DAGs Extented Version, Arbeitsberichte aus dem Fachbereich Informatik 22/2007

Mario Schaarschmidt, Harald F.O. von Kortzfleisch: Modularität als alternative Technologie- und Innovationsstrategie, Arbeitsberichte aus dem Fachbereich Informatik 21/2007

Kurt Lautenbach, Alexander Pinl: Probability Propagation Nets, Arbeitsberichte aus dem Fachbereich Informatik 20/2007

Rüdiger Grimm, Farid Mehr, Anastasia Meletiadou, Daniel Pähler, Ilka Uerz: SOA-Security, Arbeitsberichte aus dem Fachbereich Informatik 19/2007

Christoph Wernhard: Tableaux Between Proving, Projection and Compilation, Arbeitsberichte aus dem Fachbereich Informatik 18/2007

Ulrich Furbach, Claudia Obermaier: Knowledge Compilation for Description Logics, Arbeitsberichte aus dem Fachbereich Informatik 17/2007

Fernando Silva Parreiras, Steffen Staab, Andreas Winter: TwoUse: Integrating UML Models and OWL Ontologies, Arbeitsberichte aus dem Fachbereich Informatik 16/2007

Rüdiger Grimm, Anastasia Meletiadou: Rollenbasierte Zugriffskontrolle (RBAC) im Gesundheitswesen, Arbeitsberichte aud dem Fachbereich Informatik 15/2007

Ulrich Furbach, Jan Murray, Falk Schmidsberger, Frieder Stolzenburg: Hybrid Multiagent Systems with Timed Synchronization-Specification and Model Checking, Arbeitsberichte aus dem Fachbereich Informatik 14/2007

Björn Pelzer, Christoph Wernhard: System Description:“E-KRHyper“, Arbeitsberichte aus dem Fachbereich Informatik, 13/2007

Ulrich Furbach, Peter Baumgartner, Björn Pelzer: Hyper Tableaux with Equality, Arbeitsberichte aus dem Fachbereich Informatik, 12/2007

Ulrich Furbach, Markus Maron, Kevin Read: Location based Informationsystems, Arbeitsberichte aus dem Fachbereich Informatik, 11/2007

Philipp Schaer, Marco Thum: State-of-the-Art: Interaktion in erweiterten Realitäten, Arbeitsberichte aus dem Fachbereich Informatik, 10/2007

Ulrich Furbach, Claudia Obermaier: Applications of Automated Reasoning, Arbeitsberichte aus dem Fachbereich Informatik, 9/2007

Jürgen Ebert, Kerstin Falkowski: A First Proposal for an Overall Structure of an Enhanced Reality Framework, Arbeitsberichte aus dem Fachbereich Informatik, 8/2007

Lutz Priese, Frank Schmitt, Paul Lemke: Automatische See-Through Kalibrierung, Arbeitsberichte aus dem Fachbereich Informatik, 7/2007

Rüdiger Grimm, Robert Krimmer, Nils Meißner, Kai Reinhard, Melanie Volkamer, Marcel Weinand, Jörg Helbach: Security Requirements for Non-political Internet Voting, Arbeitsberichte aus dem Fachbereich Informatik, 6/2007

Daniel Bildhauer, Volker Riediger, Hannes Schwarz, Sascha Strauß, „grUML – Eine UML-basierte Modellierungssprache für T-Graphen“, Arbeitsberichte aus dem Fachbereich Informatik, 5/2007

Richard Arndt, Steffen Staab, Raphaël Troncy, Lynda Hardman: Adding Formal Semantics to MPEG-7: Designing a Well Founded Multimedia Ontology for the Web, Arbeitsberichte aus dem Fachbereich Informatik, 4/2007

Simon Schenk, Steffen Staab: Networked RDF Graphs, Arbeitsberichte aus dem Fachbereich Informatik, 3/2007

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik, 2/2007

Anastasia Meletiadou, J. Felix Hampe: Begriffsbestimmung und erwartete Trends im IT-Risk-Management, Arbeitsberichte aus dem Fachbereich Informatik, 1/2007

„Gelbe Reihe“
(http://www.uni-koblenz.de/fb4/publikationen/gelbereihe)

Lutz Priese: Some Examples of Semi-rational and Non-semi-rational DAG Languages. Extended Version, Fachberichte Informatik 3-2006

Kurt Lautenbach, Stephan Philippi, and Alexander Pinl: Bayesian Networks and Petri Nets, Fachberichte Informatik 2-2006

Rainer Gimnich and Andreas Winter: Workshop Software-Reengineering und Services, Fachberichte Informatik 1-2006

Kurt Lautenbach and Alexander Pinl: Probability Propagation in Petri Nets, Fachberichte Informatik 16-2005

Rainer Gimnich, Uwe Kaiser, and Andreas Winter: 2. Workshop "Reengineering Prozesse" – Software Migration, Fachberichte Informatik 15-2005

Jan Murray, Frieder Stolzenburg, and Toshiaki Arai: Hybrid State Machines with Timed Synchronization for Multi-Robot System Specification, Fachberichte Informatik 14-2005

Reinhold Letz: FTP 2005 – Fifth International Workshop on First-Order Theorem Proving, Fachberichte Informatik 13-2005

Bernhard Beckert: TABLEAUX 2005 – Position Papers and Tutorial Descriptions, Fachberichte Informatik 12-2005

Dietrich Paulus and Detlev Droege: Mixed-reality as a challenge to image understanding and artificial intelligence, Fachberichte Informatik 11-2005

Jürgen Sauer: 19. Workshop Planen, Scheduling und Konfigurieren / Entwerfen, Fachberichte Informatik 10-2005

Pascal Hitzler, Carsten Lutz, and Gerd Stumme: Foundational Aspects of Ontologies, Fachberichte Informatik 9-2005

Joachim Baumeister and Dietmar Seipel: Knowledge Engineering and Software Engineering, Fachberichte Informatik 8-2005

Benno Stein and Sven Meier zu Eißen: Proceedings of the Second International Workshop on Text-Based Information Retrieval, Fachberichte Informatik 7-2005

Andreas Winter and Jürgen Ebert: Metamodel-driven Service Interoperability, Fachberichte Informatik 6-2005

Joschka Boedecker, Norbert Michael Mayer, Masaki Ogino, Rodrigo da Silva Guerra, Masaaki Kikuchi, and Minoru Asada: Getting closer: How Simulation and Humanoid League can benefit from each other, Fachberichte Informatik 5-2005

Torsten Gipp and Jürgen Ebert: Web Engineering does profit from a Functional Approach, Fachberichte Informatik 4-2005

Oliver Obst, Anita Maas, and Joschka Boedecker: HTN Planning for Flexible Coordination Of Multiagent Team Behavior, Fachberichte Informatik 3-2005

Andreas von Hessling, Thomas Kleemann, and Alex Sinner: Semantic User Profiles and their Applications in a Mobile Environment, Fachberichte Informatik 2-2005

Heni Ben Amor and Achim Rettinger: Intelligent Exploration for Genetic Algorithms – Using Self-Organizing Maps in Evolutionary Computation, Fachberichte Informatik 1-2005