INF FAKULTÄT FÜR INFORMATIK

SPONSORED BY THE
Federal Ministry of Education and Research

Dagstuhl Seminar on Principles of Provenance

Toward Provenance as Cross-cutting Concern

Martin Schäler

School of Computer Science, University of Magdeburg, Germany

The work has been funded in part by the German Federal Ministry of Education and Science (BMBF) through the Research Program under Contract No. FKZ: 13N10817.

SPONSORED BY THE Federal Ministry of Education and Research

Agenda

I. Motivation: Flexibility and the illusion of control

II. Our vision: Provenance as cross-cutting concern

III. What is provenance? (For us)

IV. Provenance framework: Covering a broad variety of approaches

V. First prototype HSQL: The benefit of modern software engineering approaches

VI. Future work

SPONSORED BY THE Federal Ministry of Education and Research

I. Motivation

Provenance gained much attention in the recent past.

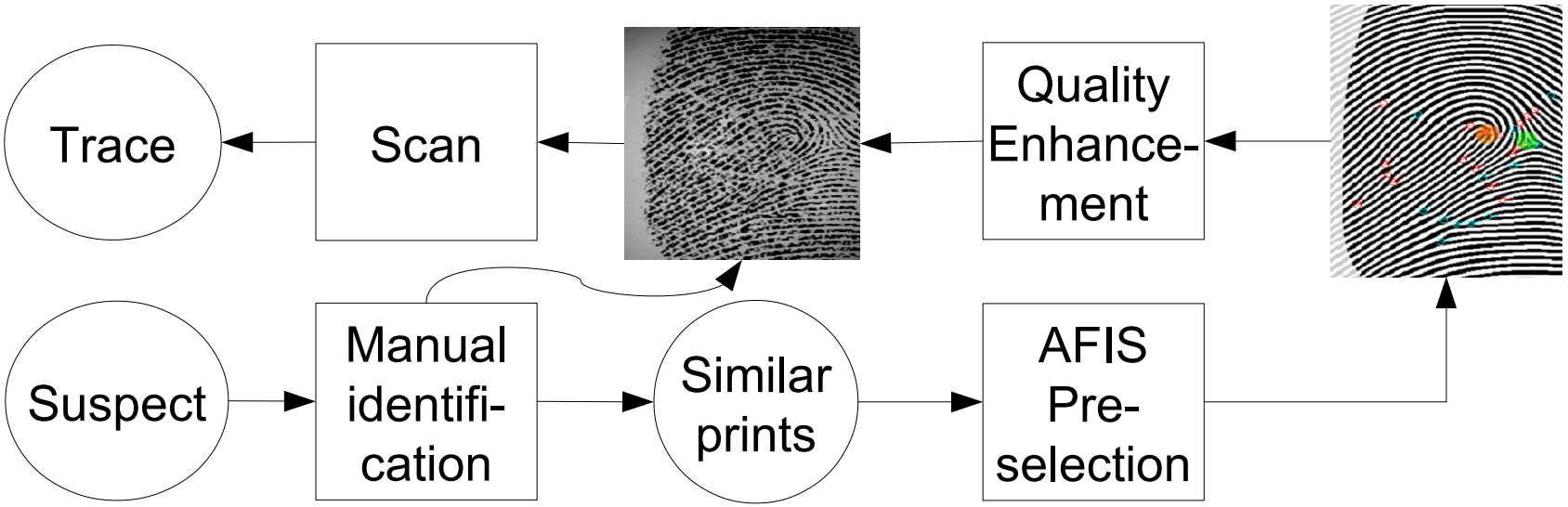
Current provenance capturing

- Database formalisms (Why & Where [1], How [2])
- Domain-specific all-in-one solutions
  - Kepler [3], Karma 2 [4], Panda [5], Taverna 2 [6] etc.

Problems: Inflexible and costly

- Complex IT-infrastructure → Creating domain-specific solutions (and linking them) is inflexible, costly, or even impossible.

Motivating example: DigiDak workflow



- Distributed IT-landscape: Illusion of control
- Privacy issues (missing infos)
- Inter-system provenance
- Fragmentation: Don't need to capture and show all the infos

SPONSORED BY THE Federal Ministry of Education and Research

II. Our vision: Provenance a cross-cutting concern

In an **ideal** world there would be ...

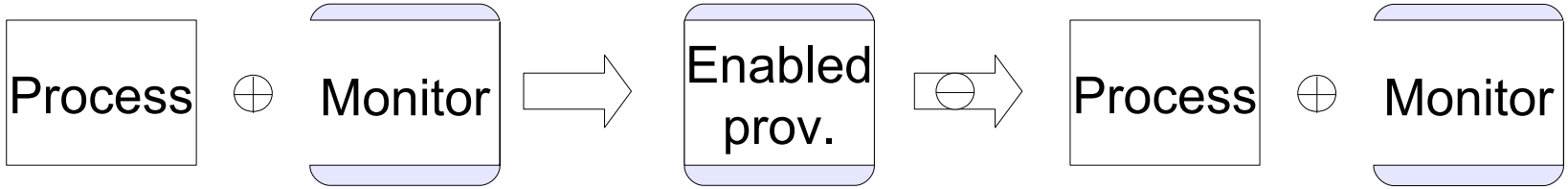
(1) One piece of software (monitor) capturing provenance for all applications → Re-use

(2) No change of original software necessary

(3) Tailored & adaptive provenance capturing

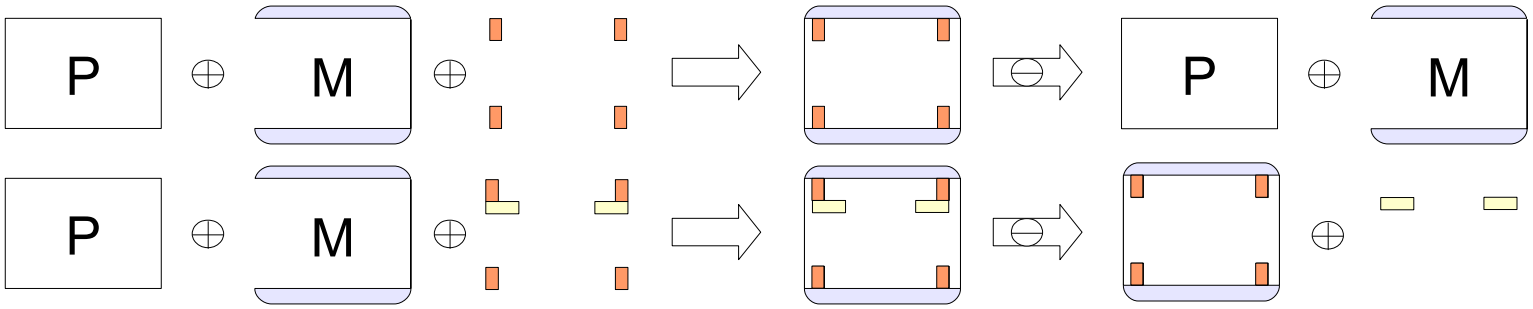
(4) Inter-system, privacy-preserving, and reliable provenance

(5) Minimal overhead (performance & storage)



Our vision – now more realistic

Issue	Goal	ToDo
(1) Re-use of concepts	Re-use of general concepts (Framework)	Identify commonalities in provenance capturing
(2) Invasive	Minimal invasive	Identify and evaluate different approaches
(3) Tailored implementation	Configurations (build-time variability)	Implementation based on (1)
(4) Trust	Configurations	Implementation based on (1)
(5) Overhead	Minimization	As for (2)



III. What is provenance? (For us)

- Term used in many communities
  - Databases [1,2]
  - (scientific) workflows [3,4,5,6]
  - Source code ownership [7]
  - ...

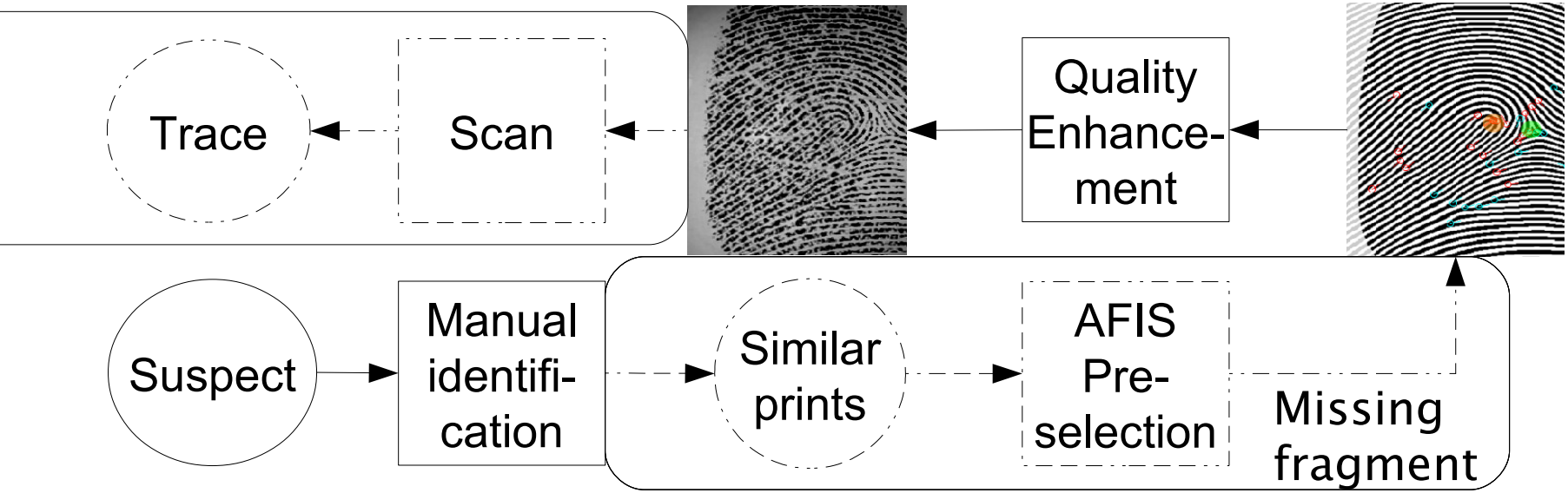
→ No clear definition.

- What is provenance used for?
  (1) Explaining data origin & derivation history
  (2) Computation of subsequent results (propagation)

What is provenance – Identifying commonalities

- General properties
  - Unchangeable [8]
  - Fragmentary [9,10,11] (granularity)
  - Uncertainty [12,13,14]

} Dimensions of provenance



What is provenance? – Dimensions of provenance

Dimensions form a hierarchical framework [15]

Fragmentation

- **Workflow:** OPM [16], similar to user views [17]
- **Existence:**
  - Recomputation: Black box similar to [18]
  - Grey box: Lineage [19]/ Endogenous input [20]
  - Paths – Why/(How)
- **Value origin:** Where

Uncertainty

- Simple annotations
- Integrity (also for linking)
- Authenticity

Building the re-computation entry

Using the structure of an (object-oriented) program

```
public class foo {
    private int memOne;
    private foo2 mem2;//never used

    RetType[] somefunc(int arg1){
        RetType[] ret = new RetType[arg1];
        memOne++;
        for (int i=0;i<arg1;i++)
            ret[i] = new RetType(memOne);
        return ret;
    }
}
```

- On entry (method call)
  - CalledBy
  - Input: arguments + members
  - CurrentValues: arguments

On leave (after return)

- Output: return value & modified Complex artifacts
- Map every `ret[i]` to `ret`
- Not changed members

This is also done for the constructor calls of every `ret[i]` → no total black box

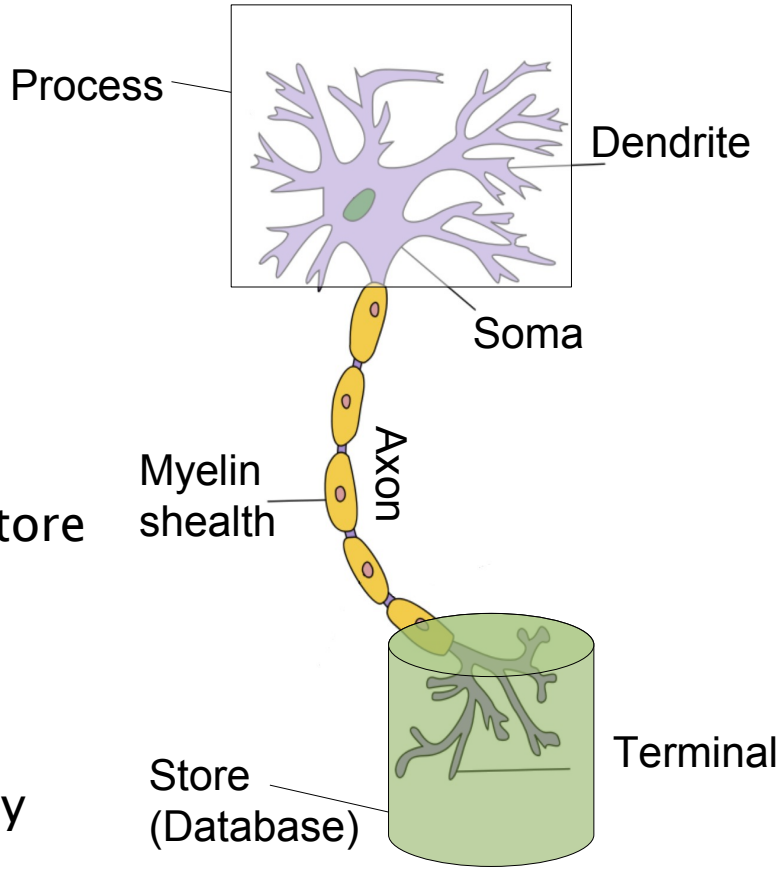
Problem: Invisible input

IV. Implementation – General re-use architecture

- Inspired by neurons: Re-use of single parts

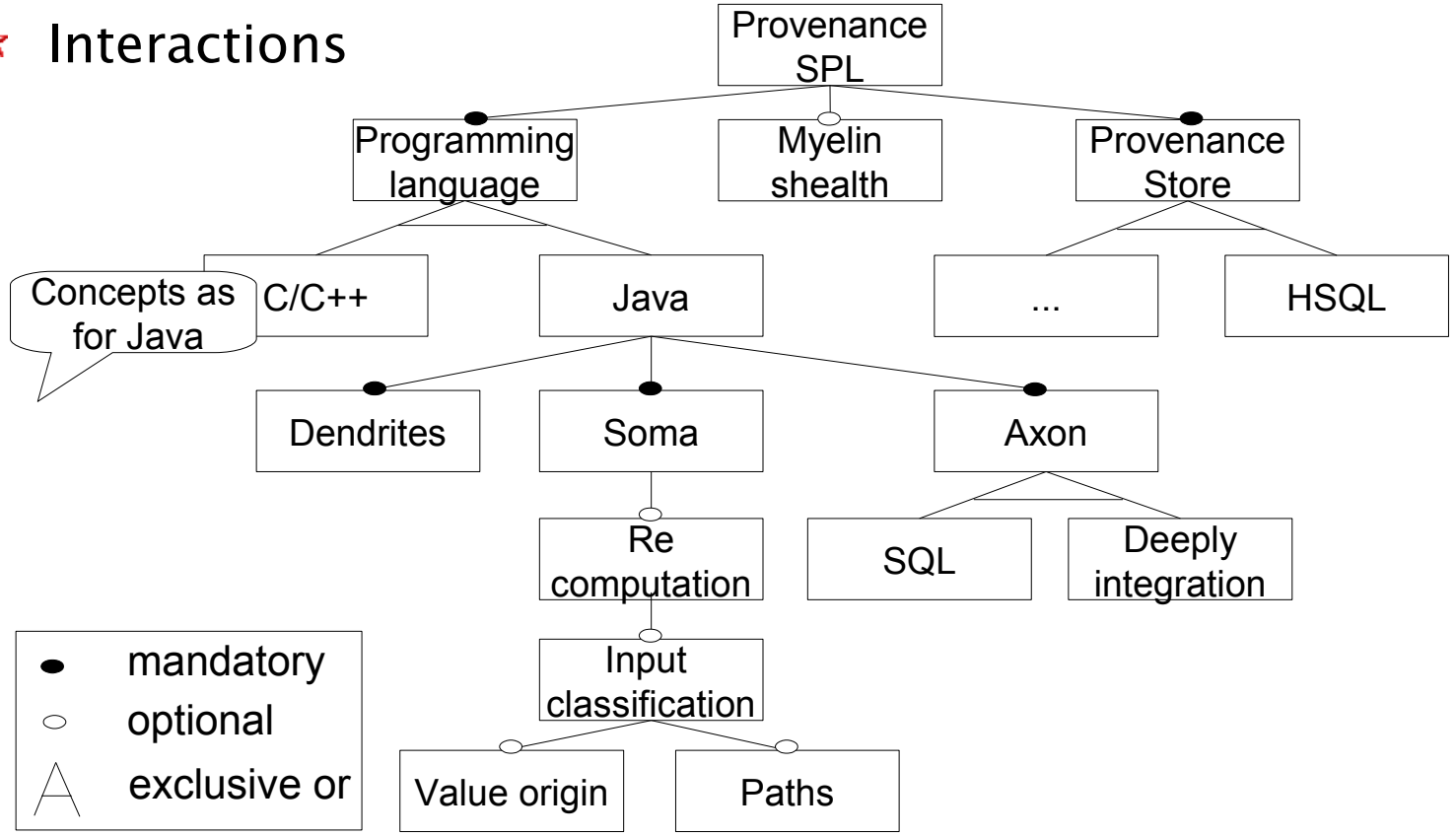
Increased re-use

- Dendrite
  - Collecting provenance
  - Mapping infos to soma types
  - Application specific (invasive parts)
- Soma
  - Mapping infos to axon type
  - Language specific
- Axon – Transport to provenance store
- Terminal – Mapping to store types
  - SQL or
  - Deeply integrated
- Myelin shealth – Ensuring reliability



Implementation Provenance Framework

- Application-specific dendrites
- Interactions



Implementing dendrites – Possible techniques

- Reference implementation
  - Maximum invasive
  - Maximum effort
  - Default performance
  - Default granularity
  - Minimum re-use
- Promising approaches
  - Annotation-based
  - Aspect-oriented progr. (AOP)
- Possibly to consider as well
  - Feature-oriented progr. (FOP)
  - Composition (Superimposition)

Reference: Using ifs

```
if (PROV_SOMA){
    //do something
}
//original code
if (PROV_SOMA && not PROV_LINEAGE{
    //do something different
}
```

Annotations (with colors)

```
//original code
someMethodCall();
//(invasive) provenance part
//do something
//again the original code
```

Aspects

```
//similar to triggers
pointcut someName : (whenToFire);

after() someName {
    //do something
}
```

SPONSORED BY THE Federal Ministry of Education and Research

V. First prototype – Where to start?

- Programming languages requires tools → Java
  - Annotation based: CIDE [21]
  - AOP: AspectJ [22]
  - FOP: Fuji [23]
  - Superimposition: FeatureHouse [24]
- Complex application
  - Represent different kinds of application → DBMS
  - Documentation etc. → HyperSQL 2.2.6 (cur. 2.2.8)
- Other nice points: Re-use
  - Re-use as provenance storage (small axons)
  - Transaction management
  - Different table types

V. First prototype – Dendrite implementation

- Reference approach
  - Permanently modifies source
  - Changes visibility of fields
  - Expression based

```
private Result getSingleResult(Session session, int maxRows) {

    int[]                limits    = getLimits(session, maxRows);
    //XXX @BEGIN Provenance
    if (PROV_RECOMPUTATION)
    {
        de.ovgu.provenance.Monitors.addInputTables(this, limits[2]);
    }
    //END Provenance
```

- Annotation based
  - Permanent modification, but less invasive
    → Automatically create original system
  - Changes visibility of fields
  - Non expression-based
    → more flexible

```
private Result getSingleResult(Session session, int maxRows) {

    int[]                limits    = getLimits(session, maxRows);
    de.ovgu.provenance.Monitors.addInputTables(this, limits[2]);
    Re"[QualifiedName] - Preview not supported."buildResult(session, limits[2]);
    Ro                                          r.getNavigator();
       Features:
    if - Recomputation_source
```

V. First prototype – Implementation examples

AspectJ

- No source modification
- Separation in invasive parts and mapping
- Re-use advices
- Pointcuts can match groups of functions
- Pointcut needs entry-points (e.g., function calss) → less flexible
- ... and some bugs in AspectJ & respective Eclipse plug-ins

```
pointcut re() : execution (* foo.*(..));

before() : re(){
    System.out.println("Entered");
    String functionName     = thisJoinPointStaticPart.
    Object[] args           = thisJoinPoint.getArgs();
    //save values
    long localID = id++;
    activeFunc = localID;
}

after() returning(test.RetType[] R) : re(){
    System.out.println("Leaving");
    activeFunc--;
    Object[] args           = thisJoinPoint.getArgs();
    //compare args
    //add Re and changed complex args to return types
}
```

SPONSORED BY THE Federal Ministry of Education and Research

V. First prototype – Reliability (myelin shealth)

- Simple annotations: Additional key-value pairs for the re-computation entry (requires additional dendrites)
- Integrity – Cryptographic hash sums of primitive values (e.g., tables or tuples)
  - MD5, SHA-family
  - Problem granularity
- Authenticity – Digital signatures
- Privacy preserving provenance linking
  - Use hash sums (be aware of collisions)
  - Keep sums after deletion
- Special case multimedia data – Invertible watermarks [25]

V. First prototype – Current status HSQL prototype

- Reference approach (IF's)
  - Fragmentation: Lineage (tuple level)
  - No myelin shealth
- Using CIDE – Same status as reference approach
- AspectJ – General aspect (Recomputation aspect)
  - Tracking *all* arguments from the function call & current status of *all* class members
  - Problems with call of static methods or fields
  - No thread support
  - Possible scenario: Track provenance from SQL to HDD-access

SPONSORED BY THE Federal Ministry of Education and Research

VI. Future work

- Finishing first prototype
  - Esp. myelin shealth
  - Documentation + Open source access
- Evaluation for each approach (Dendrites)
  - Effort to implement/invasiveness
  - First *isolated* overhead determination
- Additional applications (simulation of complex systems)
  - Postgres
  - Set up complex infrastructures
  - Real-scenarios and workloads
- Severe question: What is the same?

Thank you for your attention.

SPONSORED BY THE Federal Ministry of Education and Research

Literature

[1] Buneman et al. Why and where: A characterization of data provenance. In ICDT 2001.

[2] Green et al. Provenance semirings. In PODS 2007.

[3] Altintas et al. Provenance collection support in the kepler scientific workflow system. In Provenance and Annotation of Data 2006.

[4] Simmhan et al. Provenance management for data-driven workflows. Int'l J. Web Service Res. 5, (2) 2008.

[5] Ikeda and Widom. Panda: A system for provenance and data. IEEE Data Eng. Bull. 33 (3) 2010.

[6] J. Sroka et al. A formal semantics for the Taverna 2 workflow model. J. Comput. Syst. Sci., 76(6) 2010.

[7] Davies et al. Software bertillonage: Finding the provenance of an entity. In MSR 2011.

[8] Cheney et al. Provenance: A future history. In OOPSLA 2009.

[9] Braun et al. Securing provenance. In USENIX Workshop on Hot Topics in Security 2008.

[10] Acar et al. A graph model of data and workflow provenance. In TaPP 2010.

SPONSORED BY THE Federal Ministry of Education and Research

Literature

[11] Cheney et al. Provenance in databases: Why, how, and where. Foundations and Trends in Databases 1 (4) 2009.

[12] Lu et al. Secure provenance: The essential of bread and butter of data forensics in cloud computing. In ASIACCS 2010.

[13] Lyle and Martin. Trusted computing and provenance: Better together. In TaPP 2010.

[14] McDaniel et al. Towards a secure and efficient system for end-to-end provenance. In TaPP 2010.

[15] Schäler et al. A Hierarchical Framework for Provenance Based on Fragmentation and Uncertainty. Tech. Rep. 01-2012 University of Magdeburg

[16] Moreau et al. Open Provenance Model, 2007.

[17] Biton et al. Querying and managing provenance through user views in scientific workflows. In ICDT 2008.

[18] Amsterdamer et al. Putting lipstick on pig: Enabling database-style workflow provenance. In PVLDB Endow. 5 (4) 2011.

[19] Cui et al. Tracing the lineage of view data in a warehousing environment. In TODS 25 (2) 2000.

Literature

[20] Meliou et al. Causality in databases. IEEE Data Eng. Bull 33 (3) 2010.

[21] CIDE available at: http://wwwiti.cs.uni-magdeburg.de/iti_db/research/cide/

[22] AspectJ available at: http://www.eclipse.org/aspectj/

[23] Fuji available at: http://fosd.de/fuji

[24] Feature house available at: http://www.fosd.de/fh

[25] Schäler et al. Reliable provenance for multimedia data using invertible fragile watermarks. In BNCOD 2011.

Figures

Slide 5&9 SfinGe – Synthetic Fingerprint Generator, University of Bologna, Version 4.0 (Demo), 2009

Slide 11 http://upload.wikimedia.org/wikipedia/commons/thumb/b/bc/Neuron_Hand-tuned.svg/800px-Neuron_Hand-tuned.svg.png