A LOGIC FILE SYSTEM

YOANN PADIOLEAU and OLIVIER RIDOUX

IRISA / University of Rennes

{padiolea,ridoux}@irisa.fr

http://www.irisa.fr/LIS

USENIX ANNUAL TECHNICAL CONFERENCE, 2003

Plan

- Motivations

  (to combine navigation and querying in a file system)

- Specification (ls = ?, mv = ?, . . .)

- Implementation (data structures and algorithm)

- Evaluation (time and space)

- Related works

  (file systems and alternative organizations)

- Conclusion and further works

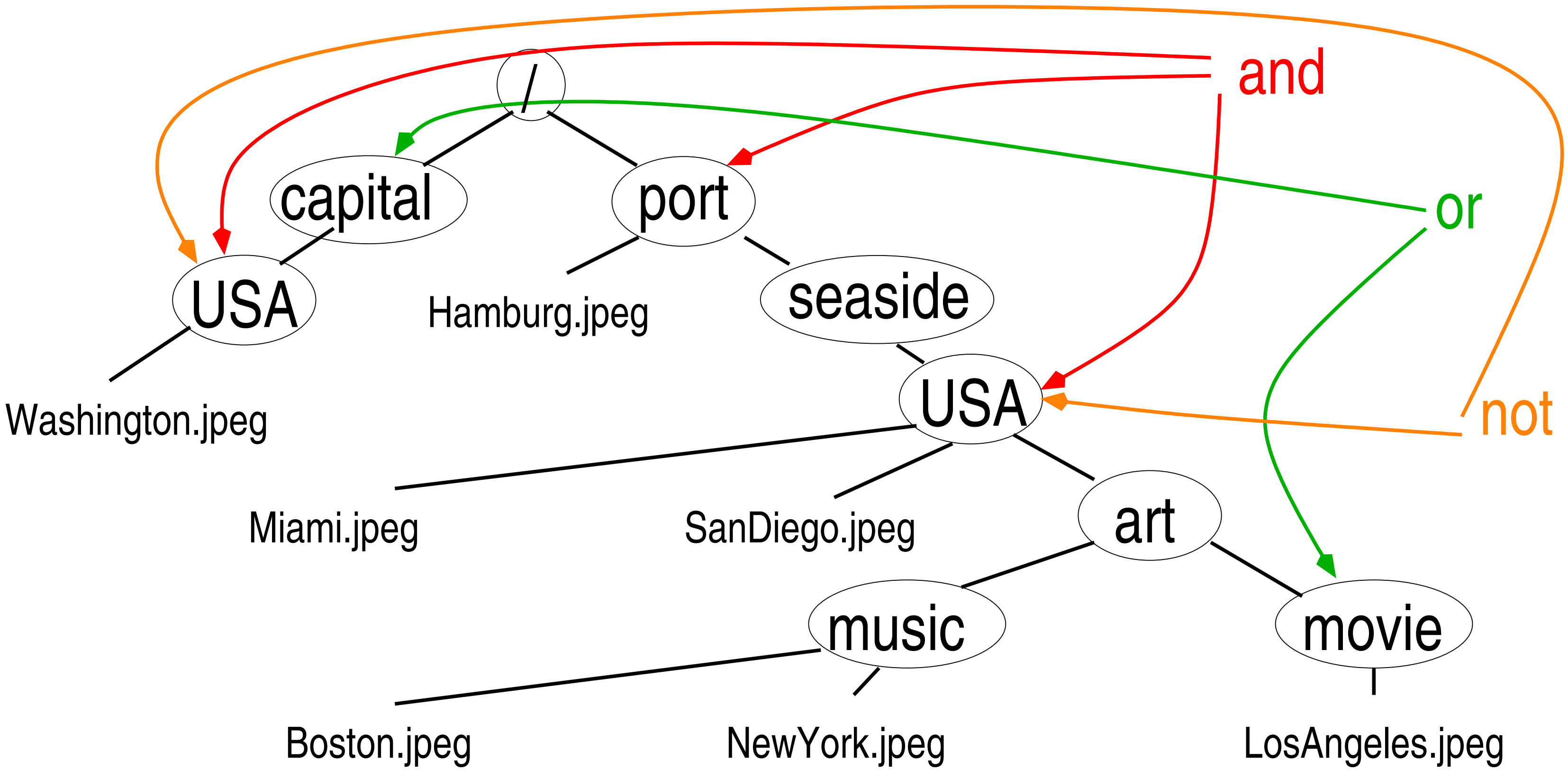
  (refine logic and improve performance)

A toy example

to represent a collection of city maps

- a collection of cities — Boston, Hamburg (Germany), Los Angeles, Miami, New York, San Diego, Washington
- a collection of descriptive attributes —
  - to be a port, on the seaside,
  - to be in the USA,
  - a capital,
  - to be a art city, for music, or movie

A hierarchical organization



a meaning for cd USA, cd <u>not</u> USA, cd capital <u>or</u> movie, and cd port <u>and</u> USA?

Boolean organization

or and not

	art	music	movie	port	seaside	USA	capital
Boston.jpeg	x	x		x	x	x	
LosAngeles.jpeg	x		x	x	x	x	
Miami.jpeg				x	x	x	
Hamburg.jpeg				x			
SanDiego.jpeg				x	x	x	
Washington.jpeg						x	x
NewYork.jpeg	x	x		x	x	x	

good for cd not USA and cd capital or movie

but, not progressive enough for cd USA and cd port and USA

Observations

- hierarchical organizations are rigid (one path per object)
- navigation is easy to understand (cd a ; ls x)

- boolean organizations are flexible (many queries yield the same answers)
- the relation between queries and answers is difficult to control (precision and recall)

$\longrightarrow$ merge navigation (as in hierarchical organizations)
and querying (as in boolean organizations)
in a file system

(every tool benefits of it: from shells to multimedia players)

Specifications

based on a previous work on Logic Information Systems [Ferré&Ridoux, DOOD'2000]

(hence the name LISFS)

Important notions (1)

LISFS content

- a logic — $A \models B$

  deduction rules (ex. $a \wedge b \models a$

  and axioms and *music* $\models$ *art*)

- information — an attachment d (description) of a logic formula p (path) to every file o (object) of a collection $\mathcal{F}$ (files)

  p expresses the property of o

  (ex. $d($*SanDiego.jpeg*$) =$ *port* $\wedge$ *seaside* $\wedge$ *USA*)

Important notions (2)

querying LISFS

paths are formulas

- extension — given a path p, the set all files that satisfy this property

$$\textit{ext}(p) = \{o \in \mathcal{F} \mid d(o) \models p\}$$

$$\textit{ext}(p) \quad \approx \quad \text{ls} \ \ \text{-R} \ \ p$$

$\textit{LosAngeles.jpeg} \in \textit{ext}(\textit{art})$ because

$$d(\textit{LosAngeles.jpeg}) = \textit{movie} \wedge \textit{port} \wedge \textit{USA} \models \textit{movie} \models \textit{art}$$

paths denote directories that denote extensions

(working directory $\equiv$ working query)

Important notions (3)

navigating LISFS

- subdirectories — given a directory O, every directory O' such that $O' \subsetneq O$

$$\textit{Dirs}(p) = max_{\models}\{p' \in \mathcal{L} \mid \emptyset \subsetneq \textit{ext}(p \wedge p') \subsetneq \textit{ext}(p)\}$$

(p' refines p)

only largest subdirectories are relevant to navigation

(most relevant hints)

- to be a file of a directory — given a path p, to be in $\textit{ext}(p)$, and in the extension of no subdirectory

$$\textit{Files}(p) = \textit{ext}(p) - \bigcup_{p' \in \textit{Dirs}(p)} \textit{ext}(p')$$

$$\textit{Files}(p) \bigcup \textit{Dirs}(p) \qquad \approx \qquad \text{ls} \quad p$$

LISFS organizations

$movie \models art \qquad music \models art \qquad \ldots$

and

Dirs

Files

	art	music	movie	port	seaside	USA	capital
Boston.jpeg	x	x		x	x	x	
LosAngeles.jpeg	x		x	x	x	x	
Miami.jpeg				x	x	x	
Hamburg.jpeg				x			
SanDiego.jpeg				x	x	x	
Washington.jpeg						x	x
NewYork.jpeg	x	x		x	x	x	

A LISFS scenario

mounting

% mount /dev/lisfs /lisfs/ ; cd /lisfs/

taxonomy

% mkdir art ; cd art ; mkdir music ; mkdir movie ; . . .

(adds $\textit{music} \models \textit{art}, \ldots$)

context

% cd seaside/USA/

% cjpeg /local/maps/Boston.ppm $>$ Boston.jpeg

($d(\textit{Boston.jpeg}) = \textit{seaside} \wedge \textit{USA}$)

updating

% mv Boston.jpeg music/ ($d(\textit{Boston.jpeg}) = \textit{music} \wedge \textit{seaside} \wedge \textit{USA}$)

navigating and querying

% ls port / USA $\longrightarrow$ art/ Miami.jpeg SanDiego.jpeg

% ls USA $\longrightarrow$ art/ port/ capital/

% ls ! USA $\longrightarrow$ Hamburg.jpeg

Other semantic features

- extrinsic/intrinsic properties —
  user gives extrinsic properties (e.g., interesting, correct, . . .)
  LISFS gives intrinsic properties
  (e.g., size:1024, owner:pad, . . .)
  LISFS can use user-defined transducers (LISFS plug-ins)
  (e.g., JPEG $\rightarrow$ resolution:640x480, . . .)
- views — to focus on a range of properties
- a security model — see article

Implementation

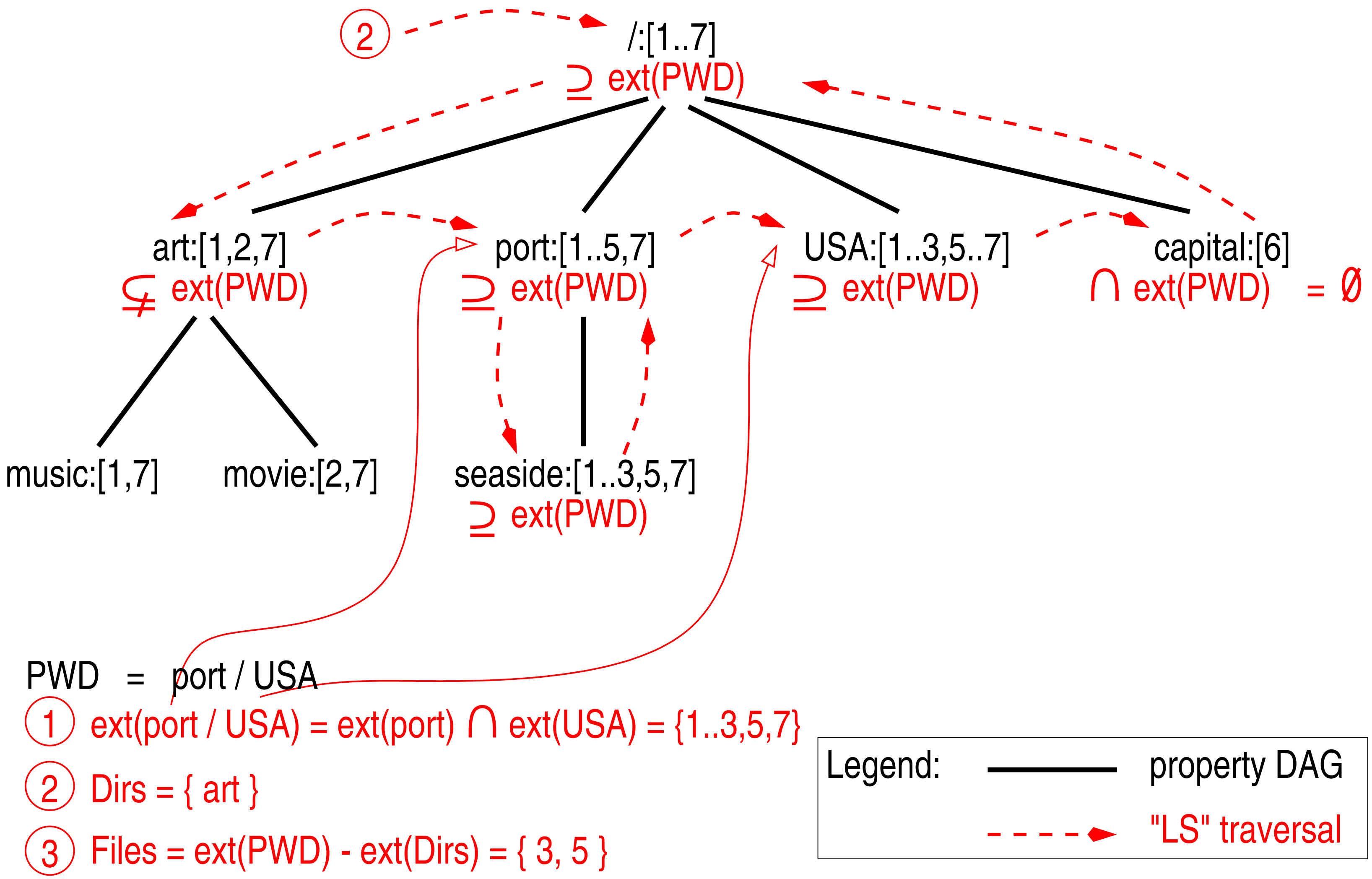
to implement the specification at a reasonable cost

basic principles (to avoid calling $\models$)

- to represent relation d as a table and inverted table on disk
- to represent on disk a directed acyclic graph (DAG) of the properties (a taxonomy)
- to attach extensions to every vertex in the property DAG

  (extensions are computed when updating LISFS content)

Computing *Files* and *Dirs*



Evaluation

- software — a user-level implementation, using PerlFS
- platform — Linux kernel 2.4, with a 2Ghz Pentium 4, 750Mb RAM, and a 40 Gb IDE disk.

Benchmarks data

all files have intrinsic system properties

(size, last modification time, owner, . . .)

- Andrew benchmark — the modified Andrew benchmark ($\times$ 10)

  (e.g., cd function:GXfind)

- MP3 files — with properties extracted from meta-data, and subjective properties (e.g., cd excellent/disco ; ls artist)

- Man pages — with keywords extracted from section "apropos"

  (e.g., cd change ; ls $\rightarrow$ directory, owner, . . .

  cd owner ; ls $\rightarrow$ chown.1)

Synthesis of evaluation

- disk space —

  space overhead: 20 % for small files, and 0.20 % for large files

  space overhead per file: $\approx$ 2 to 5 Kb

  (naive marshalling, 50 attributes per file)

- cpu time —

  creation time ratio LISFS / EXT2: $\approx$ 4 to 34

  (transducer parsing, 50 attributes per file)

total time ratio LISFS / EXT2: $\approx$ 2 to 5

$\longrightarrow$ compatible with an interactive usage

Related works

- SFS [Gifford et al.], HAC [Gopal & Manber], BeFS [Giampaolo], Nebula [Bowman et. al], . . .

  $\longrightarrow$ no navigation in result of arbitrary query

  (no computation of relevant subdirectories)

- formal concept analysis [Ganter & Wille, Lindig]

  intension, extension, subconcept ordering

- information retrieval *co-occurrence lists, term suggestions, relevant informations, significant keywords, . . .*

  (mainly application level and visual interfaces)

  $\longrightarrow$ no file system (no genericity)

Conclusions

- a running alternative to hierarchical file systems
- a formally defined integration of query and navigation
- a generic service

  (many types of files: JPEG, MP3, programs, text, . . .
  and associated descriptions)
- a security model
- encouraging performances
- availability: http://www.irisa.fr/LIS

  (Logic Information Systems)

Further works

- improve performances (especially file creation)
- integrate a theorem-prover (to express complex $\models$)
- query/navigation inside files (e.g., cd usenix-2003.tex@
  cd section:3/!comment
  emacs usenix-2003.tex)

Semantics of LISFS operations

- readdir(path) — lists $\mathit{Files}(\mathit{path}) \bigcup \mathit{Dirs}(\mathit{path})$ (ls path)
- lookup(name,path) — checks $\mathit{name} \in \mathit{Files}(\mathit{path}) \bigcup \mathit{Dirs}(\mathit{path})$
- create(name,path) — adds $d(\mathit{name}) = \mathit{path}$ (touch path/name)
- mkdir(name,path) — adds $\mathit{name} \models \mathit{path}$ (mkdir path/name)
- file operations — as usual (open, read, write, ...)
- ...

	Andrew	MP3	Man	remarks
total number/size of files	860/10 Mb	633/1772 Mb	11502/246 Mb	
total size of LISFS tables	2 Mb	3.1 Mb	43.3 Mb	
average number of attributes per file (intrinsic/extrinsic)	26/23	36/20	21/24	≈ 50
total number of attributes	1686	3730	43442	
average file size	11.6 Kb	2799 Kb	21.4 Kb	
space overhead (per cent)	20 %	0.17 %	17.6 %	
average space overhead per file	2.3 Kb	4.9 Kb	3.7 Kb	≈ 2 to 5 Kb
average space overhead per attribute	1.2 Kb	0.84 Kb	1 Kb	≈ 1 Kb
average space overhead per attribute of file	47 bytes	87 bytes	84 bytes	≈ 80 bytes
remarks	many repeated attributes	large files	many files	

	Andrew	MP3	Man
remarks	many repeated attributes	large files	many files
average number of attributes per file	≈ 50		
space overhead (per cent)	20 %	0.17 %	17.6 %
average space overhead per file	≈ 2 to 5 Kb		
average space overhead per attribute	≈ 1 Kb		
average space overhead per attribute of file	47	≈ 80 bytes	

Time

	Ext2	PerlFS	LISFS (transducer off)	LISFS (transducer on)	remarks
Mkdir	0.217s	0.986s	1.823s	3.703s	
Copy	1.359s	5.943s	13.212s	46.296s	creation
Scan	2.506s	5.141s	5.348s	6.638s	
Read	3.548s	11.510s	11.119s	12.333s	
Make	16.896s	28.384s	36.182s	46.260s	compilation & creation
Total	24.526s	51.964s	67.684s	115.230s	
MP3	2min28s	4min30s	5min	5min30s	creation & copy
Man	22min	29min	44min	85min	indexing & creation

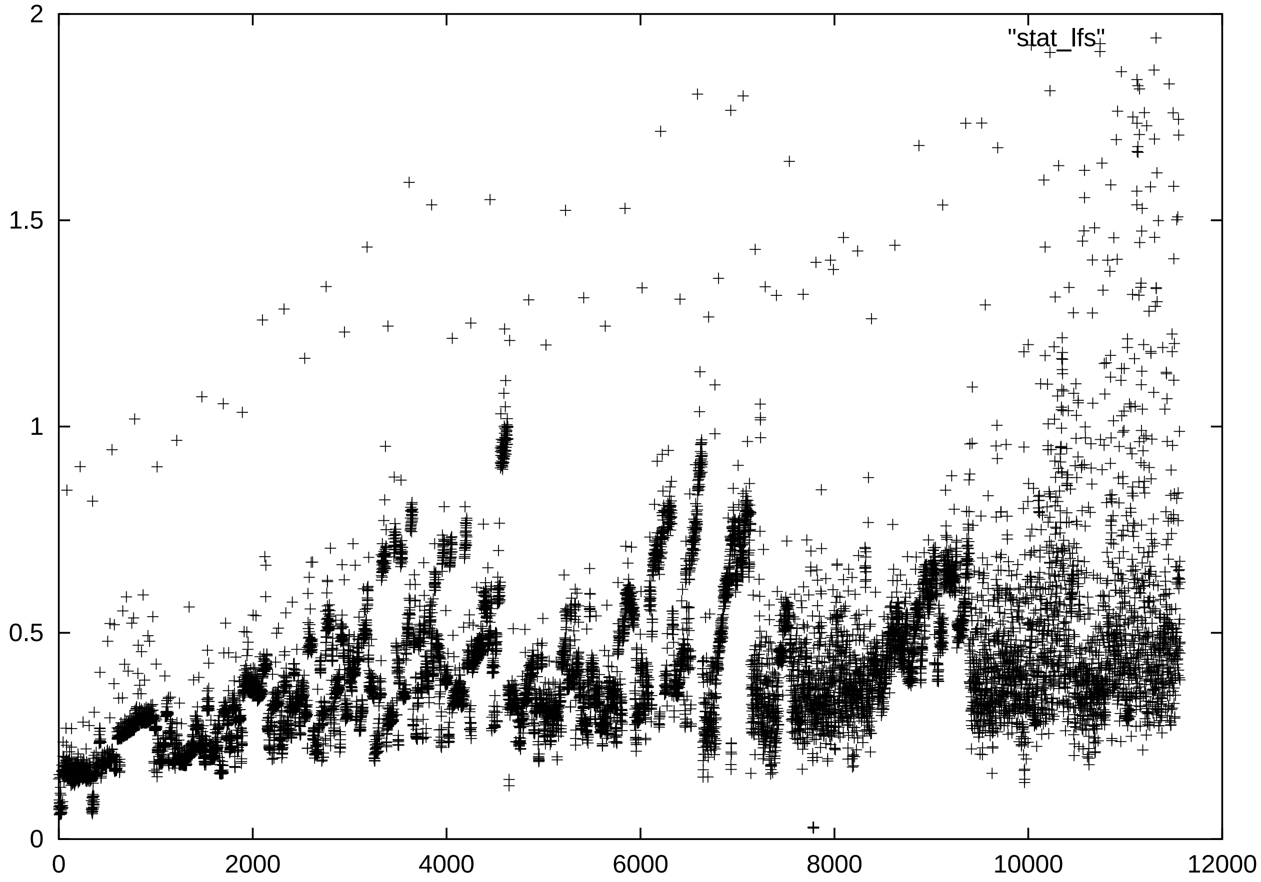
Time ratios

	Ext2	PerlFS	LISFS (transducer off)	LISFS (transducer on)	remarks
Mkdir	1	4.5	8.4	17	
Copy	1	4.37	9.7	34	creation
Scan	1	2.05	2.13	2.65	
Read	1	3.24	3.13	3.48	
Make	1	1.68	2.14	2.74	compilation & creation
Total	1	2.12	2.76	4.7	
MP3	1	1.8	2	2.2	creation & copy
Man	1	1.32	2	3.86	indexing & creation

Creation times

11502 man pages

(sec)



Related works (file systems)

- SFS, Gifford et al. — only content-based (virtual) directories

  (no means to move a file into a virtual directory)

- HAC, Gopal & Manber — virtual directories are made real

  (can move a file where it does not belong)

- BeFS, Giampaolo — non-standard interface

- Nebula, Bowman et. al — a hierarchy of views

  (no real query/navigation integration)

⟶ no navigation in result of arbitrary query

(no computation of relevant subdirectories)

Related works (non-hierarchical file systems)

- formal concept analysis, Ganter & Wille, Lindig — intension, extension, subconcept ordering

- information retrieval — *co-occurrence lists, term suggestions, relevant informations, significant keywords, . . .*

(mainly application level and visual interfaces)

$\longrightarrow$ no file system (no genericity)