A Secure Autonomous Document Architecture for Enterprise Digital Right Management

Manuel Munier

LIUPPA
Université de Pau et des Pays de l'Adour
Mont de Marsan, France
manuel.munier@univ-pau.fr

SITIS 2011

November 28 - December 1, 2011

Dijon, France

This paper

- Information system security is currently one of the most important goals for enterprises
- The problem becomes even more difficult if a user wants to "checkout" a document from the information system
  - e.g. *to work offline or to distribute the document to other people outside the organization*

$\Rightarrow$ Problem: how to ensure the security of the document once it has left the information system ?

This paper

- We use an object oriented approach to encapsulate within the document itself some security components (access control, usage control, traceability,. . .)

$\Rightarrow$ The "intelligent" document self-manages its own security

$\Rightarrow$ We defined[1] a secure autonomous document architecture for Enterprise Digital Right Management

[1] project FLUOR, ANR-SESUR 2008-2011
http://fluor.no-ip.fr/index.php

Table of contents

Context of Information Sharing

- Information sharing ?
  - collaborative work for **enterprises**: reports, medical records, tender documents, whole project as bulk document,...
  - documents can go outside the company where they have been designed (export from IS)... and return (import updated documents)
  - we have to control how partners use the documents
    - access control (of course...)
    - usage control (cf. obligations)
      e.g. user has to read a section before writing his review
    - traceability (cf. metadata, auditing,...)

  $\Rightarrow$ **D**igital **R**ight **M**anagement approach with user licenses

  $\rightarrow$ **E**nterprise-**DRM**

Context of Information Sharing

Document security enforced on server side

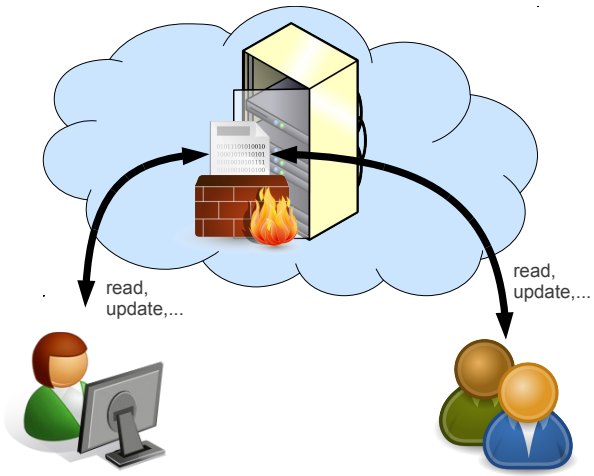
- "Classic" DRM architectures
  - server ciphers the digital document & build user license
  - client side viewer deciphers the document according to rights found in the license

  $\Rightarrow$ well suited for multimedia documents
    - content providers & read-only viewer clients
    - the document is created once and never changes
    - security policy remains the same

Context of Information Sharing

Document security enforced on server side

- E-DRM architectures
  - documents are not "static"
    $\Rightarrow$ updates, item deletion, new data,...
  - security policy may change during the document lifecycle



$\Rightarrow$ client application has to contact the server to check access & usage rights for user actions

- server can also provide audit facilities
  $\rightarrow$ *traceability allows to control how information is used & to demonstrate that it has been used as defined in the security policy*
- off-line use by leasing the document for a finite period of time

e.g. Adobe LiveCycle Policy Server

Context of Information Sharing

Specific needs

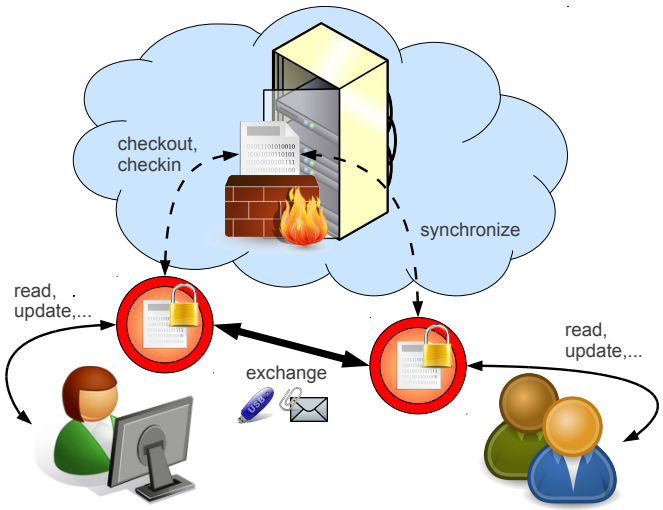
- Our specific needs
  - users can update shared documents ($\neq$ multimedia DRM)
  - multi-site enterprises, virtual enterprises, nomadic users
    - $\rightarrow$ *using a centralized site for the exchanges is seen as a constraint*
  - usability with legacy applications: email attachment, USB flash drive, share resource on a WebDAV server,...
    - $\rightarrow$ *users could exchange docs without having to connect to a server*

$\Rightarrow$ "Classic" centralized architectures do not suit these needs

Context of Information Sharing

Object oriented approach

- OO approach to encapsulate
  - **data**: content of the document itself
  - security control **components**: access control, usage control, traceability & metadata, collaborative work management,...



$\Rightarrow$ autonomous document self-manages its security

$\rightarrow$ *such a document is a kind of information system on its own*

Context of Information Sharing

Object oriented approach

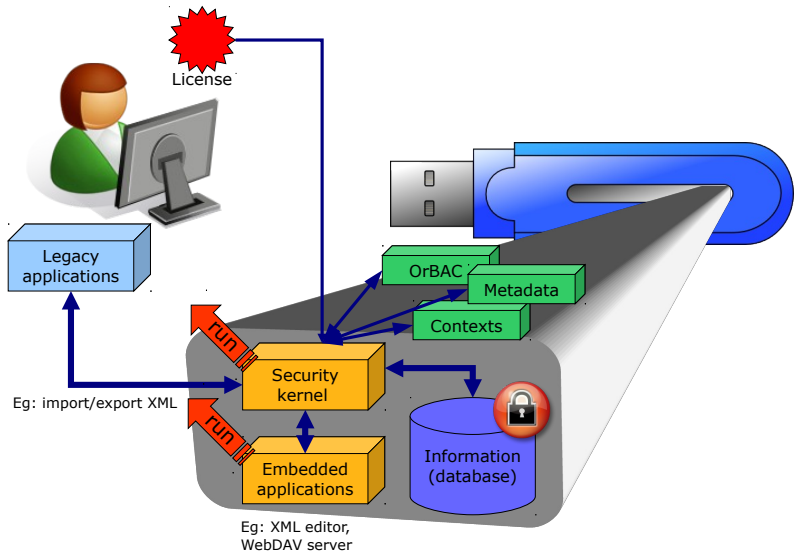
- How to "use" such a document ?
  - when "opening" the document, the user should provide her/his license
  - security control components are configured according to security rules contained in the user license
    → *permissions, obligations, metadata required,...*
  - they check all the accesses to information (embedded IS)

⋮

  - user can forward the document to another user (who handles the document according to her/his own license)
    → *no need to publish the amended doc on the server*

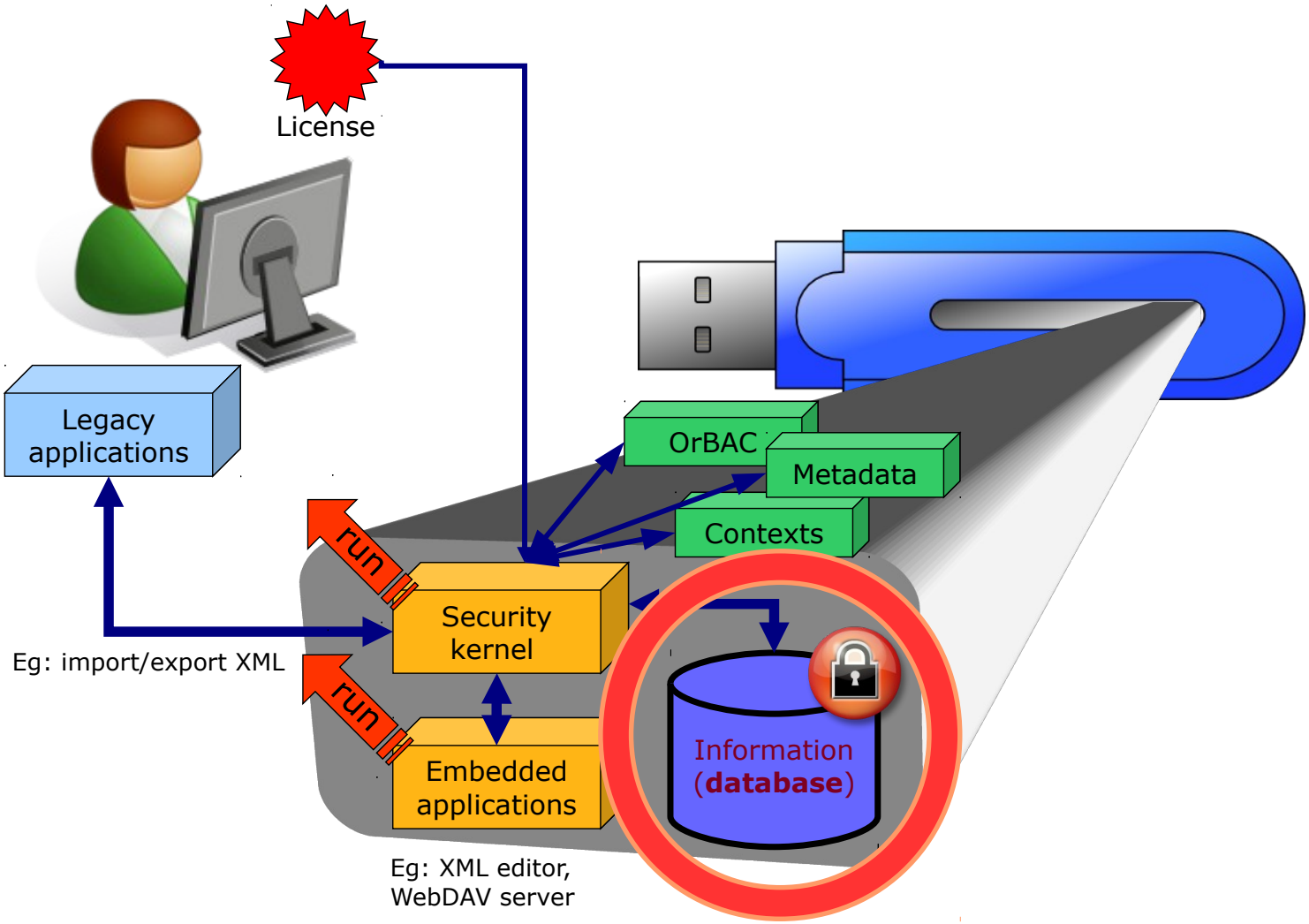
Intelligent Documents
Overall architecture

- Main components
  - embedded database
    - → *contents of the document, metadata*
  - security kernel & security modules
    - → *enforce the security policy*
    - → *monitor all actions on the doc*
  - embedded applications & services
    - → *dedicated tools*
    - → *export/import mechanisms*
  - user license
    - → *permissions, prohibitions, obligations*
    - → *metadata to be collected*



Intelligent Documents

Embedded database



Intelligent Documents
Embedded database

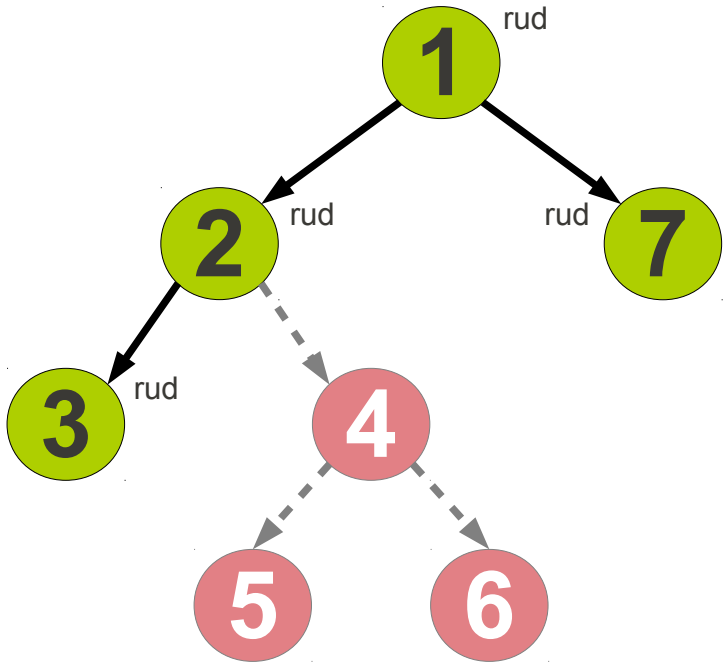
- In previous work we defined a new data model for embedded information system
  - multi-view approach to ensure both confidentialty & integrity
  - formal model to store data & calculate views
  - mapping of user actions to "low level" actions

- Dilemma privacy vs. integrity
  - → **Confidentiality**: How to prevent the disclosure of information to unauthorized individuals (or systems)
    - *breach of access control: someone can perform actions without the proper permissions*
    - *system behavior allows one to deduce the existence of hidden information*
  - → **Integrity**: How to avoid data to be modified without authorization
    - *someone accidentally (or with malicious intent) modifies/deletes data by side effects of a legitimate action*

Intelligent Documents

Embedded database - Example: removing nodes in data tree

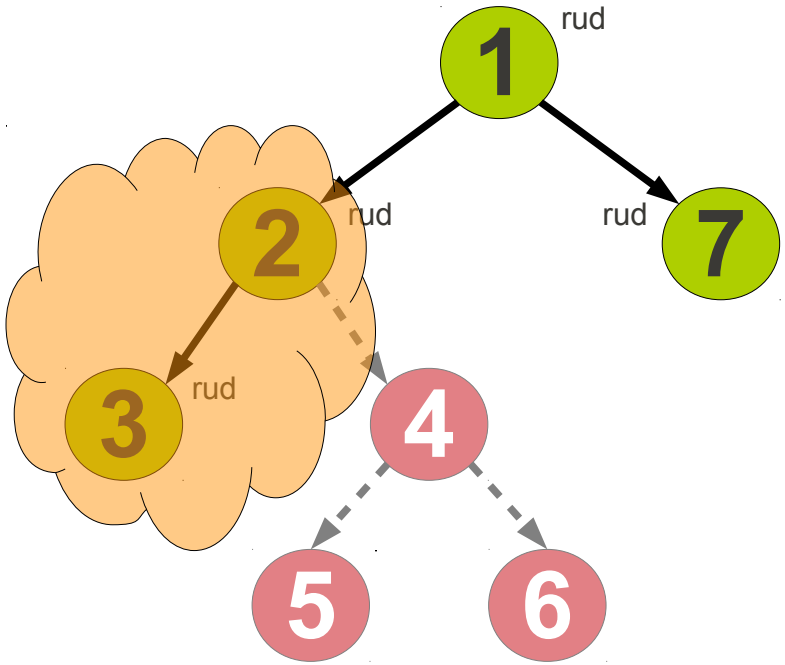
- User can access nodes 1,2,3,7 with permissions **r**ead, **u**pdate and **d**elete
- He's not aware of nodes 4,5,6
- What happens if he decides to delete the node 2 ?



Intelligent Documents

Embedded database - Example: removing nodes in data tree

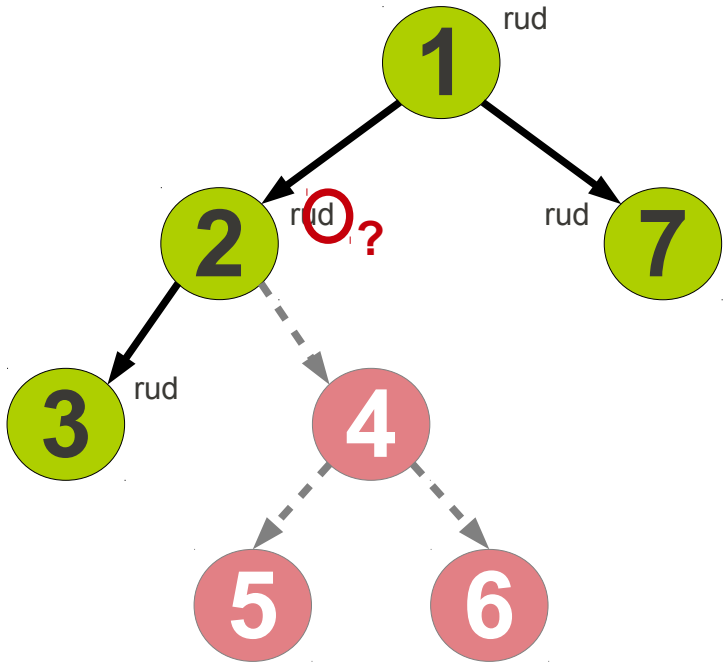
- If the system accepts to remove nodes 2 and 3, what happens for node 4 ?
- Breach of integrity: node 4 is no longer attached to the tree



Intelligent Documents

Embedded database - Example: removing nodes in data tree

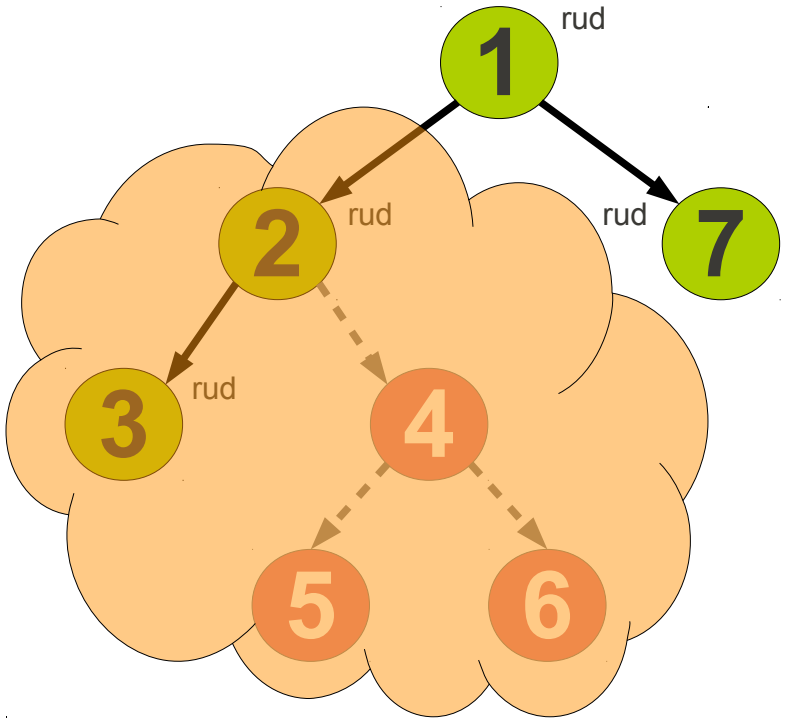
- User is not allowed to delete node 4 (and its descendants)
- If the system refuses to remove nodes 2 and 3 to preserve the integrity of the data, then user can deduce the existence of hidden information (nodes 4,5,6)



Intelligent Documents

Embedded database - Example: removing nodes in data tree

- If the system decides to remove nodes 4,5,6 to preserve the integrity, then user deleted unauthorized data (by side effects)



Intelligent Documents

Embedded database - Multi-view approach

- We decouple *"what the user sees"* from *"what is stored"*
  - versions & relationships
    - *at the data store layer, all versions of each object are kept with their own relationships*
    - *data are not independent of each other ⇒ semantic relationships can denote various kinds of associations:*

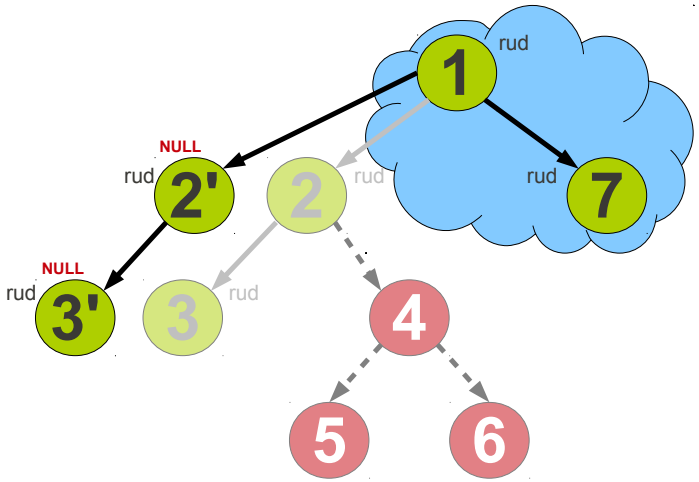
      `tree` *(structural relation like "father/child" or "container/content")*

      `use` *(semantic relation like "a program uses a library", e.g.* `#include`*)*
  - computation of views
    - *a user has only a partial view of data contained in the store*
  - mapping of user actions
    - *user actions (on user view) have to be translated into basic actions (on the data store): create new versions, update relationships,. . .*

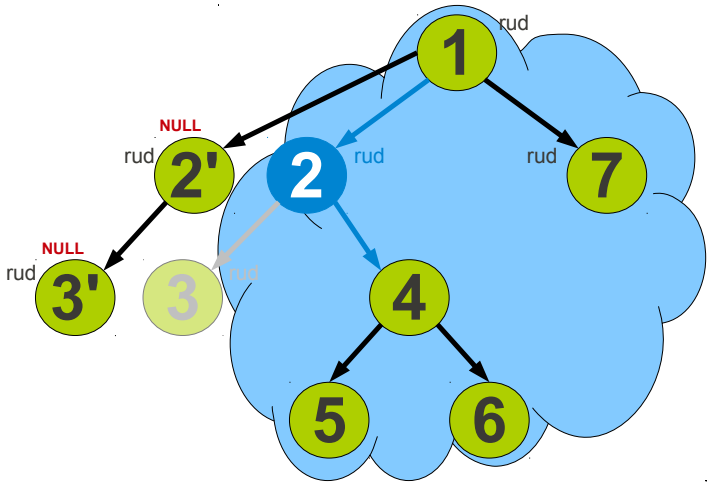
Intelligent Documents

Embedded database - Multi-view approach

1. User Anna can't access nodes 4,5,6
   - After removing nodes 2,3 her view only contains nodes 1,7
   - Node 2' is the new version of node 2; value "NULL" indicates this node has been deleted and should no longer appear in Anna's view



2. User Bob can access nodes all nodes
   - Anna deleted nodes 2,3
   - Bob's view still contains node 2 to preserve the integrity of relationships between nodes 1,2,4

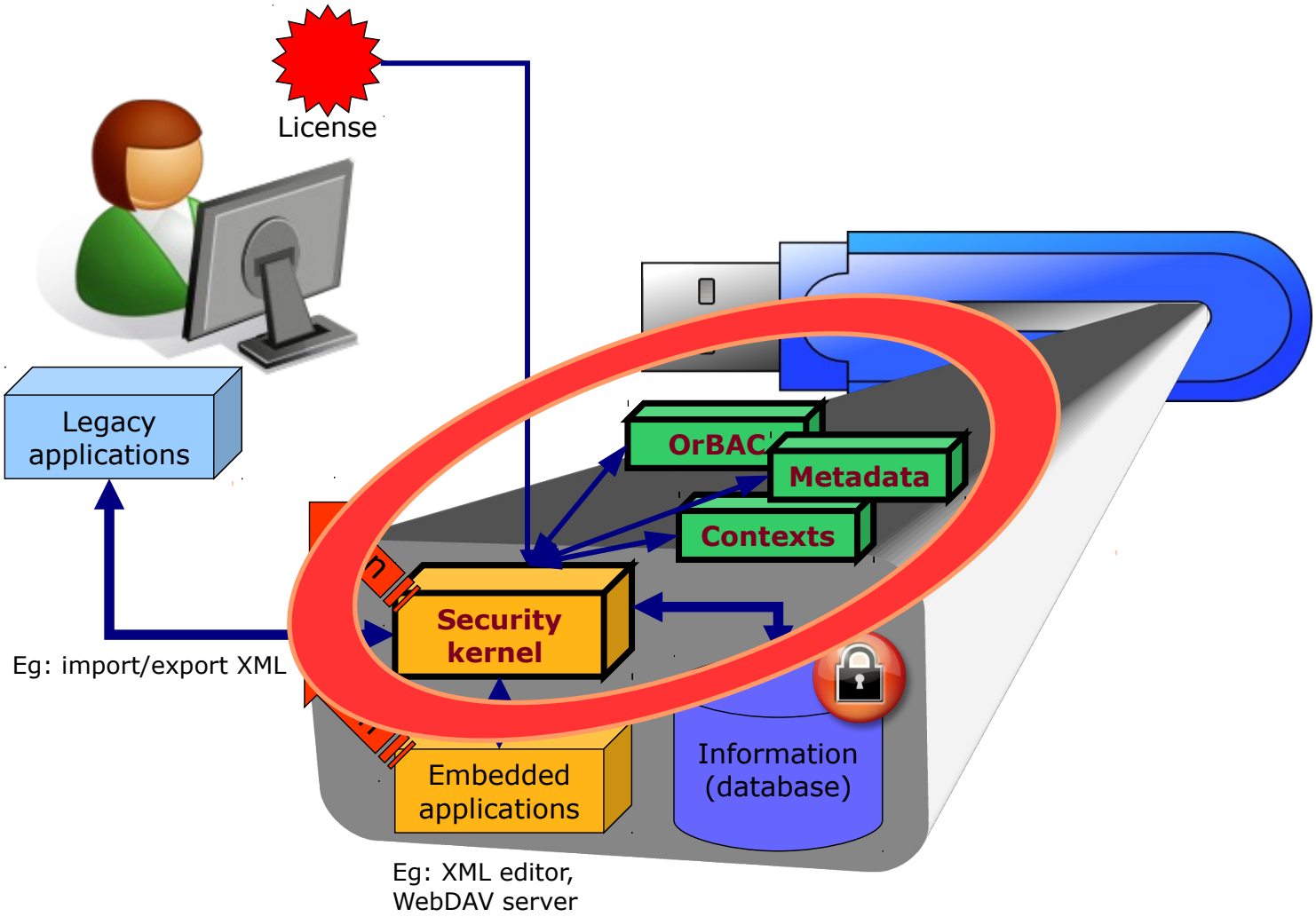


Intelligent Documents
Embedded database

- Benefits of this model
  - user actions have the intended effect on her/his view
  - system preserves the integrity of data (e.g. relationships between nodes)
- Embedding database within the intelligent document
  - nodes can be tagged with metadata
  - database is ciphered so that only the security kernel can access its content

Intelligent Documents

Security kernel & security modules



Intelligent Documents

Security kernel & security modules

- The **security kernel** is the core of our architecture
  - it is the document interface with the outside world
  - all the actions performed by the users to handle the document have to be done through the security kernel
- To enforce the security policy, the security kernel relies on various **security modules** dedicated to specific tasks
  - those responsible of **accepting or rejecting** user actions
    e.g. *access & usage control*
  - those collecting and attaching **metadata** to the actions
    e.g. *who performed this action, from which IP, at what time, with which application, in which context,...*
  - those **calculating new information** as actions go along
    e.g. *trustworthiness indicator, collaborative work management,...*

Intelligent Documents
Security kernel & security modules

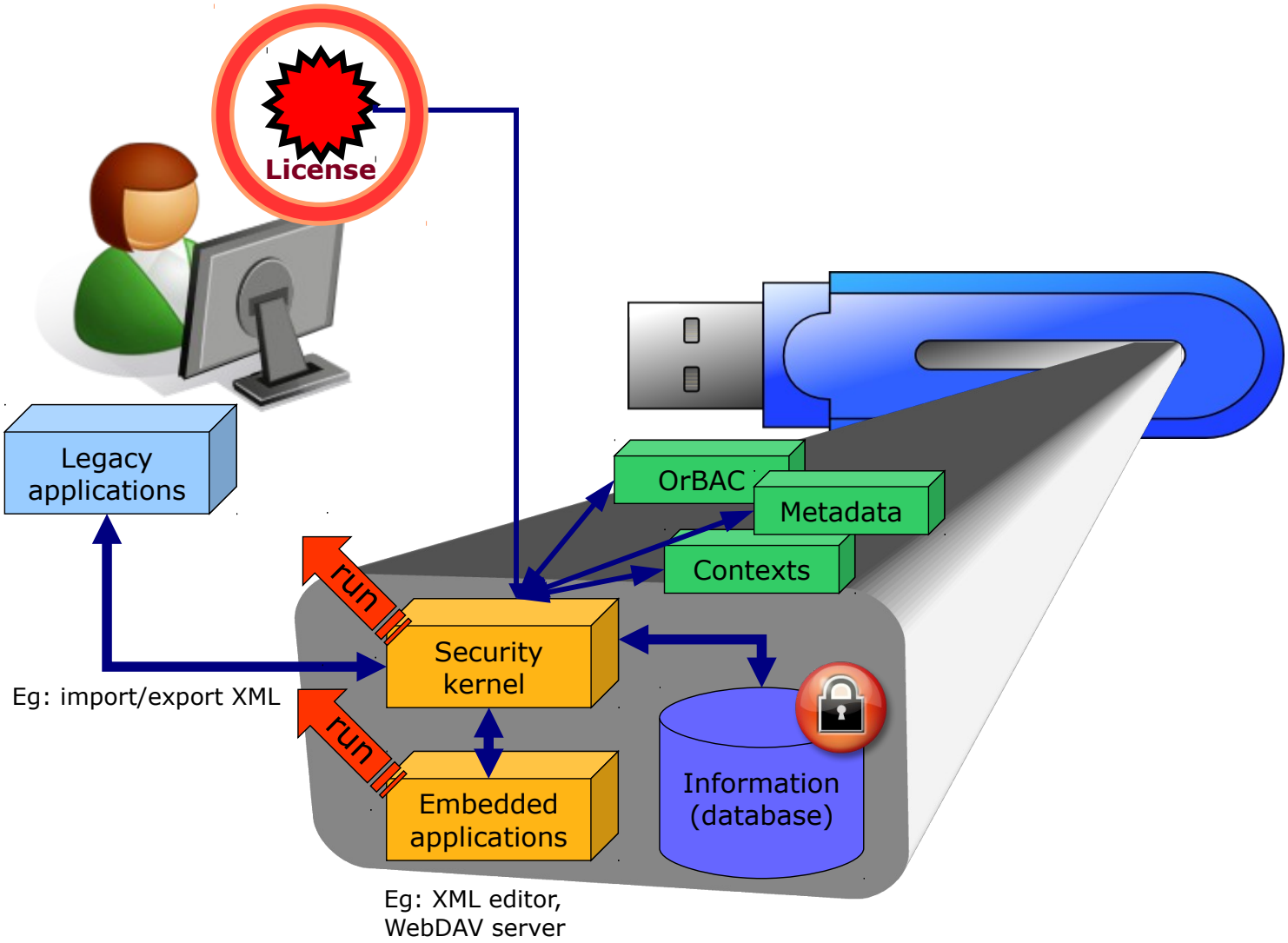
- When the user requires the execution of an action, the security kernel performs control in two stages
  1. validate the action
     - the kernel requests each security module to validate the action
       → *some modules will add information to this action (e.g. metadata)*
       → *others will indeed accept/reject the action (e.g. access control)*
  2. process the action
     - basic operations implementing this action are then performed on the data warehouse
     - the security kernel broadcasts this action a second time to each security module so they can achieve the associated processing
       → *logging (e.g. access control, usage control)*
       → *adding metadata to nodes of the embedded database*
       → *computation of additional information (e.g. trustworthiness management, collaborative work management)*

Intelligent Documents
Security kernel & security modules

- Security modules we already developed
  1. access & usage control
     - we use the OrBAC model
       → *permissions, prohibitions, obligations*
       → *security rules can be dynamic, i.e. depending on the context*
  2. context management
     - we can control context activation in the OrBAC model
     - how to check conditions from the context definition ?
       → *direct access to the host system (e.g. a global clock)*
       → *metadata carried by the actions*
  3. metadata collection
     - put metadata on actions & nodes of the embedded database

Intelligent Documents

License contents



Intelligent Documents

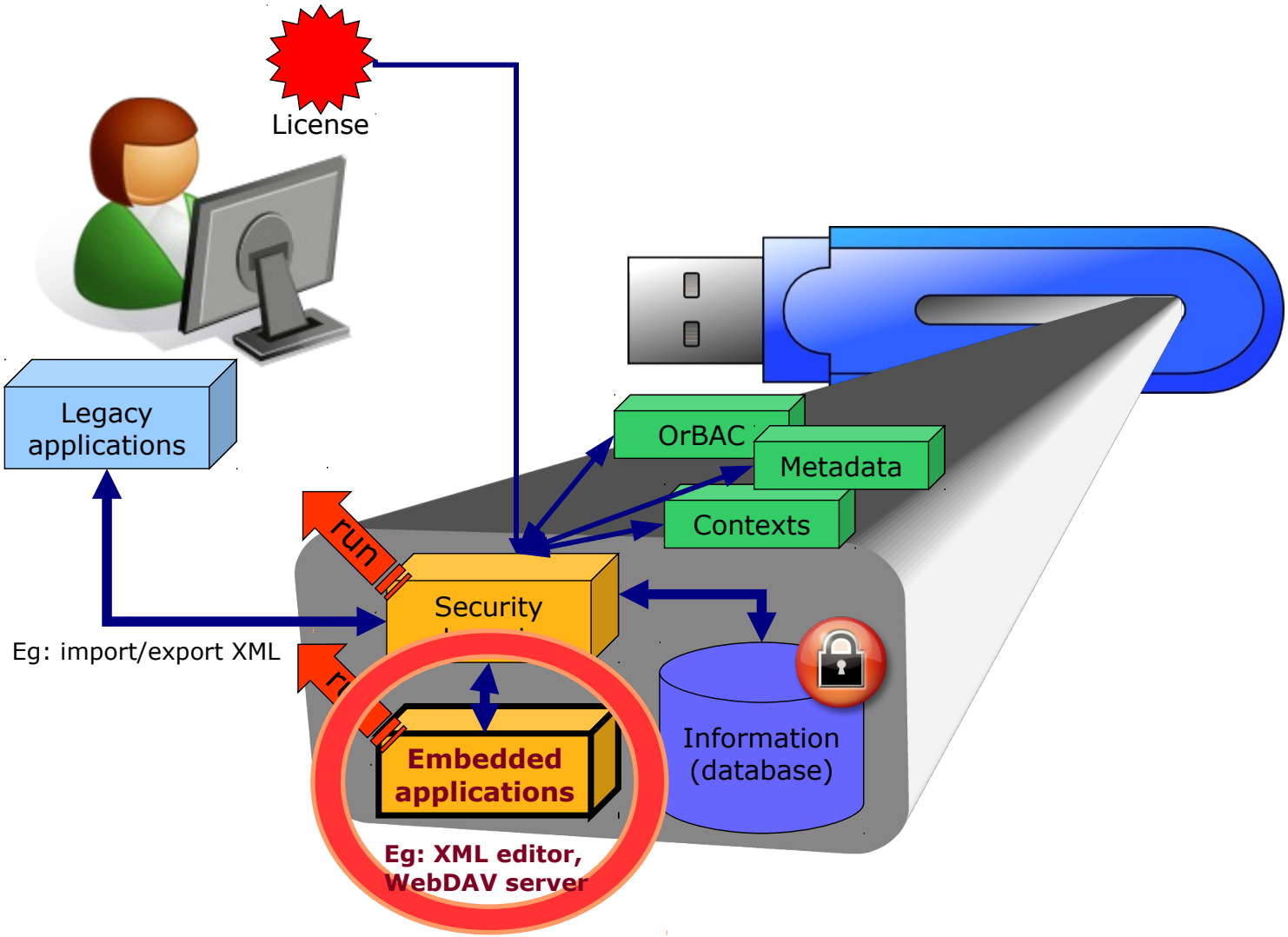
License contents

- The license contains many information:
  - identity of the server that issued the license (the licensor)
  - data about the user to which the license is granted (the licensee)
  - all the information needed to configure the various security modules
    - → for now, OrBAC security rules (with contexts)
    - → *(later) which (and how) metadata should be collected ?*
    - → *(later) what triggers must be deployed to manage contexts ?*
    - → *(later) what information can be automatically computed ? (e.g. trustworthiness indicator)*

  ⇒ standards like XrML or ODRL do not suit our future needs

Intelligent Documents

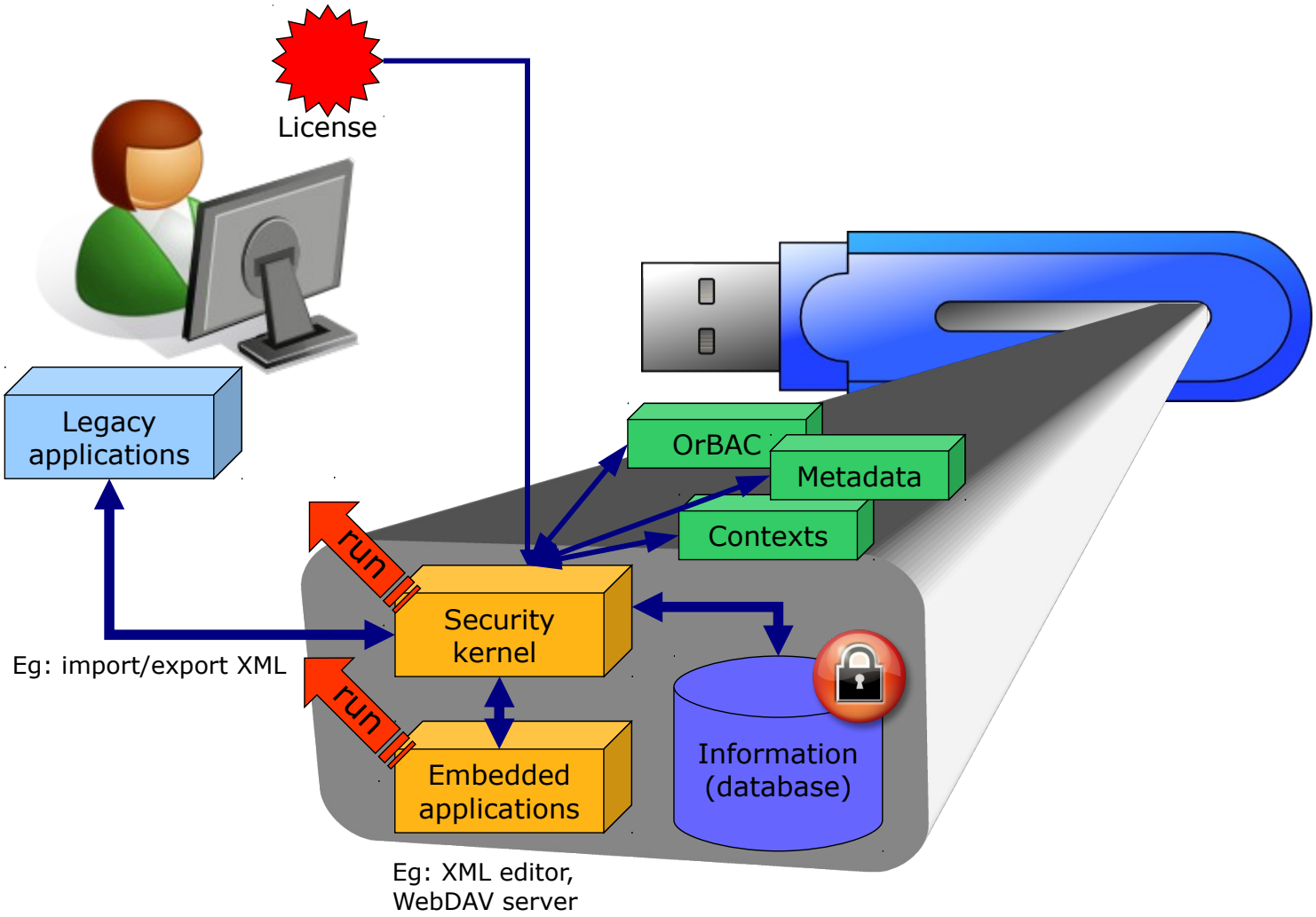
Embedded applications & services



Intelligent Documents
Embedded applications & services

- How to interact with the document ?
  - **export & import** mechanisms (XML for example) to manipulate information through existing applications
    - → *filters at the security kernel level to format information when exporting (checkout) and to interpret them when importing (checkin)*
  - **plugins** developed for existing applications
    - → *the plugin can then talk directly with the security kernel to interact at the nodes and relationships level (finer granularity)*
  - use of **services and/or dedicated applications embedded** in the secure document
    - e.g. *after starting the different security components, the document can automatically start running a WebDAV server to present the information as a tree of files/directories*
    - → *access to information can then be made from traditional applications through a WebDAV client*

Intelligent Documents
Summary



Platform Implementation

- Intelligent document ≡ decentralized IS
  - ⇒ it must bring together on the same "support"
    - a **database** (contents of the document, metadata,...)
    - several **executables** (security kernel, security modules, embedded services & applications)

- Actual implementation
  - an easy solution: a USB flash drive that represents the document and can be exchanged (physically) between users

  NB: U3 technology ⇒ USB flash drive ≡ "mobile desktop", but...
    - available only in the Microsoft Windows environment
    - SanDisk no longer supports this technologie...

  ⇒ **standard USB flash drives** with an autorun configuration to launch **Java** programs

Platform Implementation

- Embedded database
  - use of our prototype of secure versioned repository (**SeVeRe**)
  - model extension: support for operations on groups of objects

  $\Rightarrow$ *users can store structured documents like XML (where every node is represented by an object) and manipulate them via routines in the checkout/checkin style at the level of a whole document or as part of the document (and not node by node)*

- Platform tested in the **FLUOR** project[2]
  - *convergence du contrôle de* **FL***ux et d'***U***sage dans les* **OR***ganisations*

  $\rightarrow$ collaborative work based on intelligent documents embedding a small information system built from our model
  - http://fluor.no-ip.fr/index.php

[2] work supported by the French ministry for research under the ANR-SESUR 2008-2011 project FLUOR

Platform Implementation

- Security concerns
  - **Java** ⇒ document can run on various OS (MS Windows, Linux, Android,...)
  - **ciphering** to protect embedded database, license contents,...
- Future works
  - **risk analysis**
    - e.g. *ISO/IEC 27005:2011 information security risk management*
      - → *decentralized IS: benefits, but also new vulnerabilities...*
  - intelligent document as a **single file** (e.g. JAR archive)
    - → *more user friendly: 1 file on a USB flash drive, 1 email attachment,...*
  - use of **secure USB flash drives**
    - → *1 public area (USB mass storage device) and 1 private area (memory chip of the smart card)*
    - → *some components could run directly on the chip ⇒ greater security*

Conclusion & Perspectives

- Conclusion
  - E-DRM architecture using autonomous documents
    - → users do not need to be connected all the time to the server
      only for checkout/checkin/synchronize operations
    - → users can exchange docs without going through the server
      e.g. email attachment, USB flash drive
    - → documents can carry dedicated applications & services
      e.g. service to present document contents as a filesystem, business applications,. . .
  - enterprise context
    - documents are structured and complex
    - working documents ⇒ users can update the contents
    - relations between the partners are well defined ⇒ security policy definition

Conclusion & Perspectives

- Perspectives
  - legal issues, privacy concerns
    - which (and how) metadata can be collected ?
    - what information can be automatically computed ?

    $\Rightarrow$ the contents of the license gives the terms of use of the document that the user must agree
  - risk management
    - autonomous documents $\Rightarrow$ distributed information system

    $\rightarrow$ advantages & disadvantages, new vulnerabilities

    $\rightarrow$ ISO/IEC 2700x risk analysis

Conclusion & Perspectives

- Perspectives
  - programming issues
    - implement new security modules
      e.g. trustworthiness management, collaborative work management
    - provide synchronization facilities
      by merging embedded database
    - specify license contents
      metadata definition, audit specification,. . .
  - new technologies
    - secure USB flash drives
      e.g. smartcards for medical records
    - cloud storage
      secure autonomous documents self-manage their security

Manuel Munier

A Secure Autonomous Document Architecture for Enterprise Digital Right Management

Thank you for your attention.

manuel.munier@univ-pau.fr