FORTH
Institute of Computer Science

Dependency Management for Digital Preservation

Yannis Tzitzikas

FORTH-ICS and University of Crete

Last update: August 16, 2011

*Motivation for this work was the **CASPAR** EU IP project (2006-2009). The work will be continued in the context of the **APARSEN NoE** EU project (2011-2014) and **SCIDIP-ES** data infrastructure project (2011-2014).*

scidip-es

FORTH
Institute of Computer Science

Notice

This is a draft collection of slides from various presentations that I have given in the context of the CASPAR project or other conferences (the corresponding papers are referenced at the bibliographic section). I apologize for the various incoherencies and inconsistencies. I will try to keep improving this collection of slides

Acknowledgements

Many thanks to several “CASPARtners” for the various discussions we had during the CASPAR project, and to Yannis Marketakis for carrying out the associated implementation tasks.

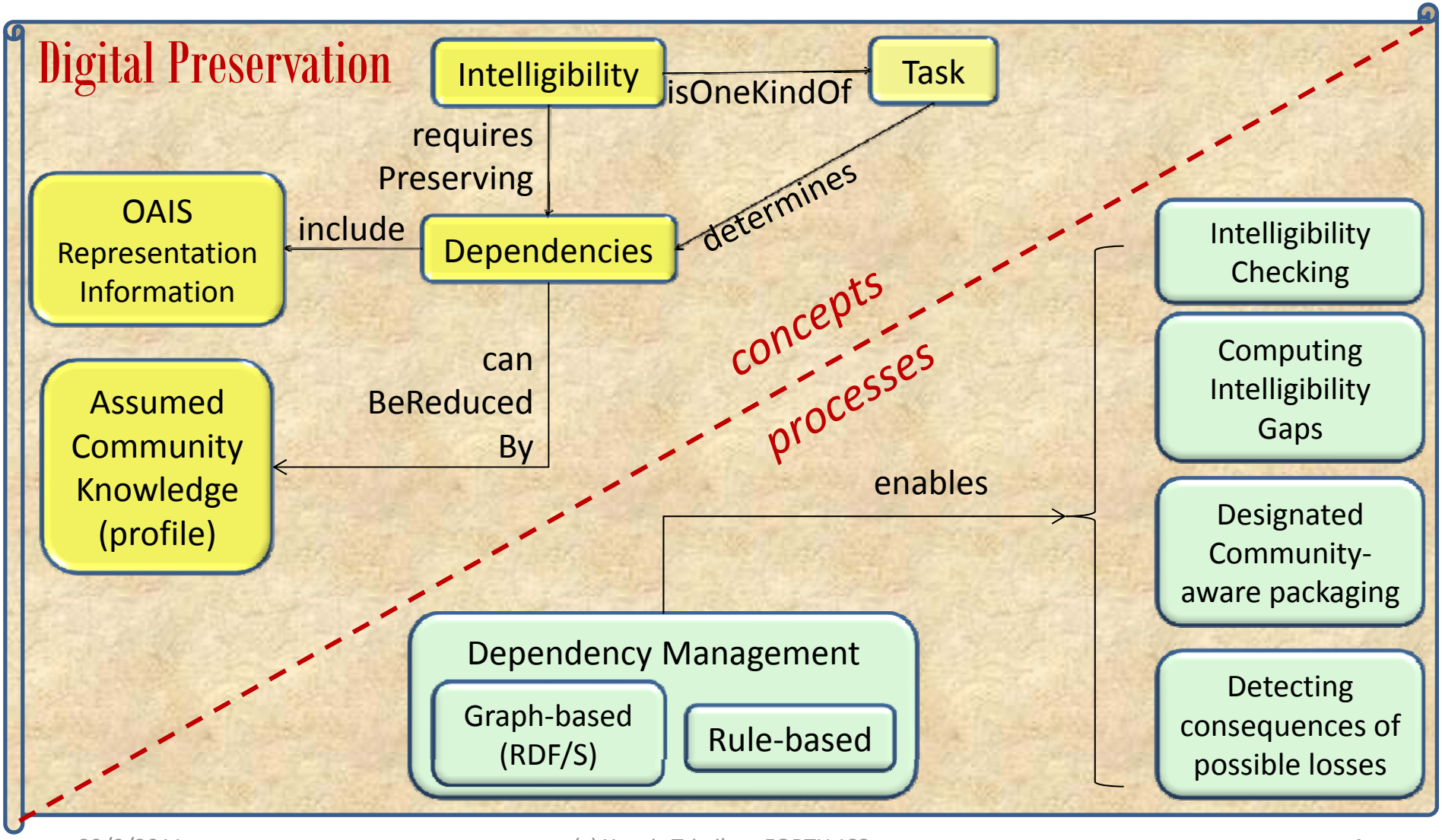
FORTH
Institute of Computer Science

Outline

- Context and Motivation
  - Perspectives of Digital Preservation
  - The problem (of intelligibility) in the physical world and the digital world
- **Dependency Management** for Digital Preservation
  - Key ideas and notions
    - module, dependency, community profile, intelligibility gap
  - Some Difficulties
  - A Graph-based Model
  - A Rule-based Model
  - How to apply (methodology)
- An Example with Scientific Data
- Back to the Digital Preservation Context
- Related Tools/Applications
  - Available software components and tools (the CASPAR project experience)
- A draft typology of approaches
- Related Publications/Tutorials/Tools
- Concluding remarks and future work/research

FORTH
Institute of Computer Science

informal concept map of this presentation



FORTH
Institute of Computer Science

CONTEXT AND MOTIVATION

FORTH
Institute of Computer Science

Motivation: Digital Preservation

- Modern society and economy is increasingly dependent on a deluge of only digitally available information. The world produces around **2 exabytes** (2^{60}) of unique information **per year** (90% of which is digital and with a 50% annual growth rate)

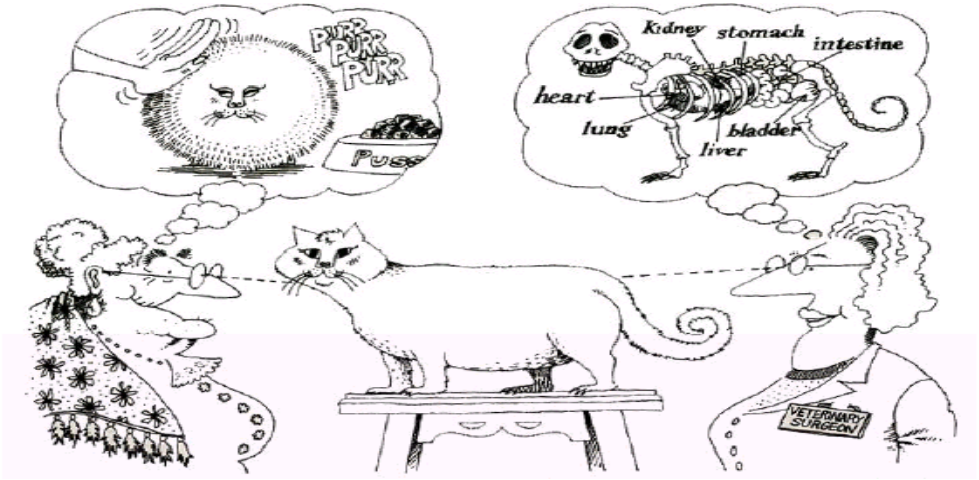
However

- Hardware, software, formats, real world knowledge changes.
- "***Everything flows, nothing stands still***" [Heraclitus]
- The preservation of digital information within an unstable and rapidly evolving technological (and social) environment is a challenging problem of prominent importance.

Therefore

- ***Digital information has to be preserved not only against hardware and software technology changes, but also against changes in the knowledge of the community.***
- **There is a need for services that help archivists in checking whether the archived digital artifacts remain *intelligible* and *functional*, in *identifying hazards* and the *consequences* of probable losses or *obsolescence risks*.**

Aspects of Preservation



But what should we preserve?

- For sure we have to preserve the **bits** of the digital objects

We should also try to preserve their

- accessibility
- integrity
- authenticity
- provenance
- intelligibility (by human or artificial actors)

We will focus on this aspect

FORTH
Institute of Computer Science

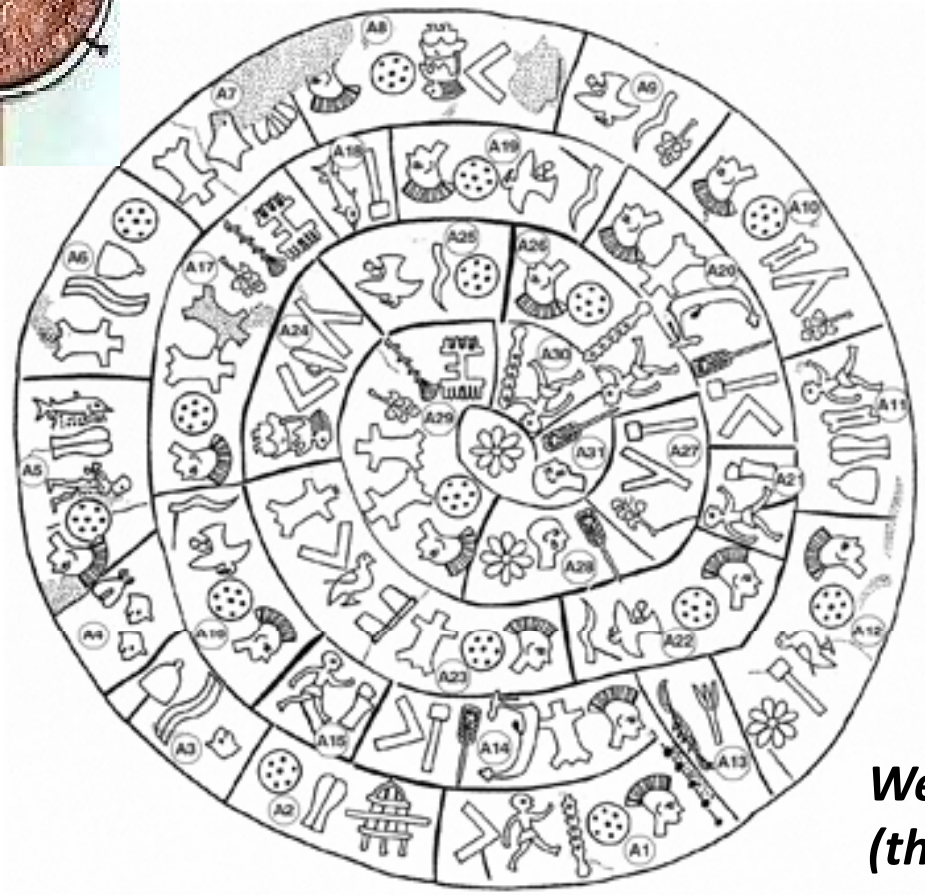
Preservation strategies and .. related words

- Digital Archaeology
- Interoperability with the Future
- Migration
- Refreshing
- Normalization
- Canonicalization
- Encapsulation
- OAIS
- Technology Preservation
- Virtualization
- Emulation
- UVC (Universal Virtual Computer)
- Reliance to Standards

FORTH
Institute of Computer Science

The problem (of intelligibility) in the physical world

Phaistos disk (dated to 1700 BC)



We still cannot understand it
(the <u>meaning</u> has not been preserved)

FORTH
Institute of Computer Science

The problem (of intelligibility) in the digital world

How can we be sure that in the future one would be able to understand this byte stream?

1001101100001101110110111011100101111100111

08909711010110 5115

&#89&#97&#110&#110&#105&#115

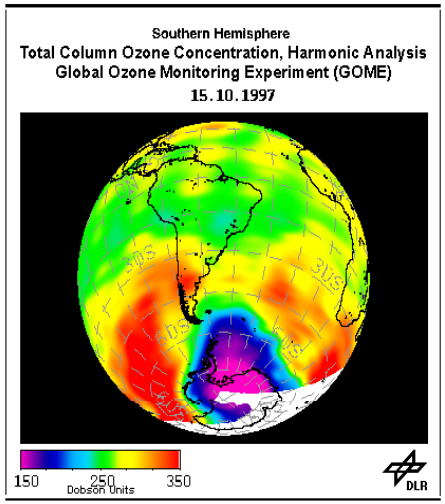
It is "**Yannis**" in ASCII

How we will preserve the meaning of digital objects?

FORTH
Institute of Computer Science

(cont)

- Even if we know that the previous bit stream consists of ASCII symbols, and we know what ASCII is, we may also need to preserve extra information.
- Example:



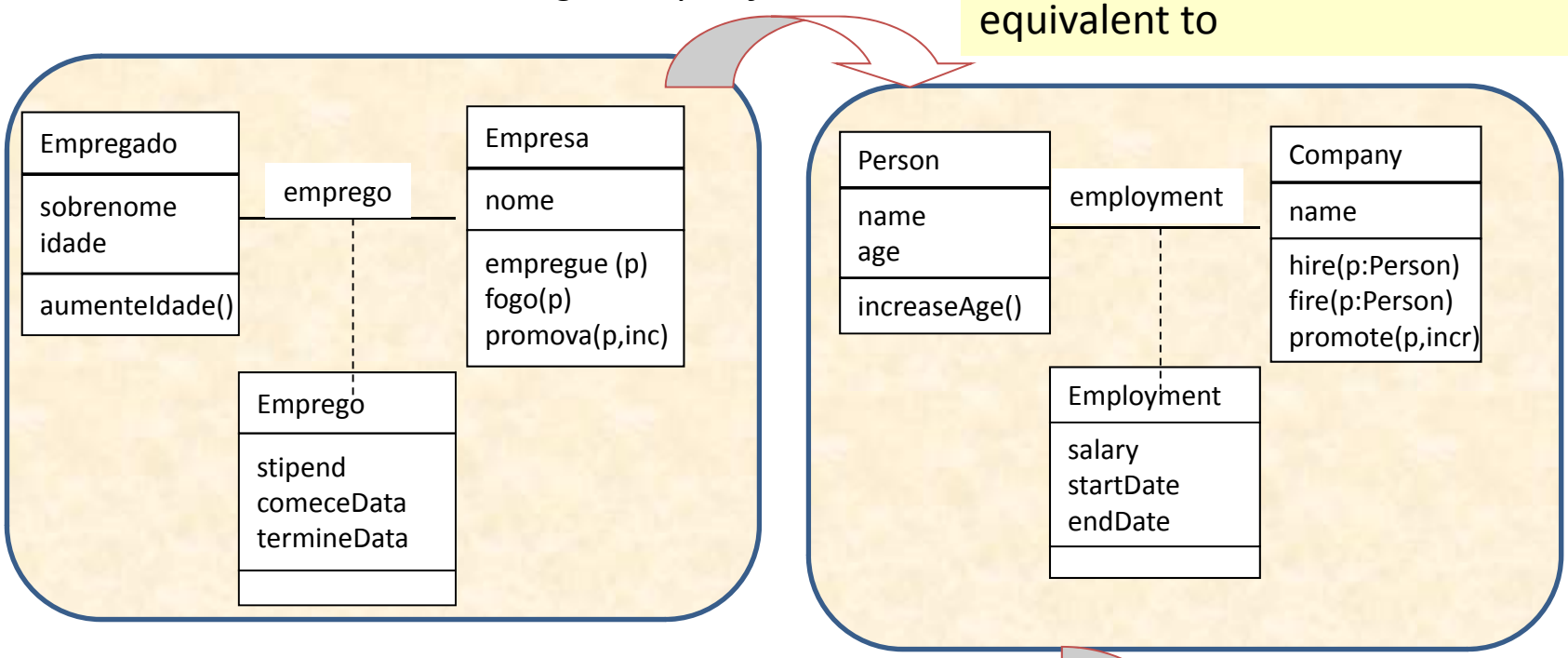
- It is not enough to preserve the rendering of this figure.
  - We also need to be able to understand (in the future) *what color signifies in this image*
- Another example follows.

FORTH
Institute of Computer Science

Digital Objects
The necessity of everyday knowledge

I know UML but what this diagram specifies?

If I knew Spanish then that would be equivalent to



- *A person cannot start a job before his/her birth*
- *A promotion cannot lower the salary of an employee*

⇒ ***Now I can develop the system or I can guess how the existing system operates***

Plus everyday knowledge

FORTH
Institute of Computer Science

DEPENDENCY MANAGEMENT FOR DIGITAL PRESERVATION

The idea in brief (*what, why, how, where*)
Benefits (from dependency management)
Examples of dependencies
Relationship to OAIS (Rep Info, Packaging)
The notion of “Intelligibility Gap” (and its role in archiving/dissemination)
Preservation Information Systems

FORTH
Institute of Computer Science

Dependency Management for Digital Preservation (*What, Why, How, Where*)

- To maintain a digital object in a ***usable state*** in the future we must have access to all the modules (e.g., software, hardware, tacit or external knowledge etc) that it depends on.
- Dependencies: a set of requirements (prerequisites) for something to be useful (readable, understandable, viewable, executable, …).
  - For example, the dependencies of this presentation include: *Computer system, properly setup , Powerpoint software, Overhead projector, Knowledge of English language, Proper eye functioning, …*

Our Approach

- **What**: We model the dependencies of digital objects, as well as some tacit community *assumptions.*
- **Why**: In this way we can identify how much dependencies we need, identify gaps, detect risks, etc. Applications in *packaging* (archival or dissemination), *quality control*, *risk detection*, etc.
- **How**: By modeling and managing the dependencies of digital objects and the assumed (designated) community knowledge
  - modeling : there are various choices, here we will see a **graph-based** and a **rule-based** model
  - management requirements: *extracting, storing, querying, updating*
- **Where:** At a packaging approach or at a semantic repository approach.

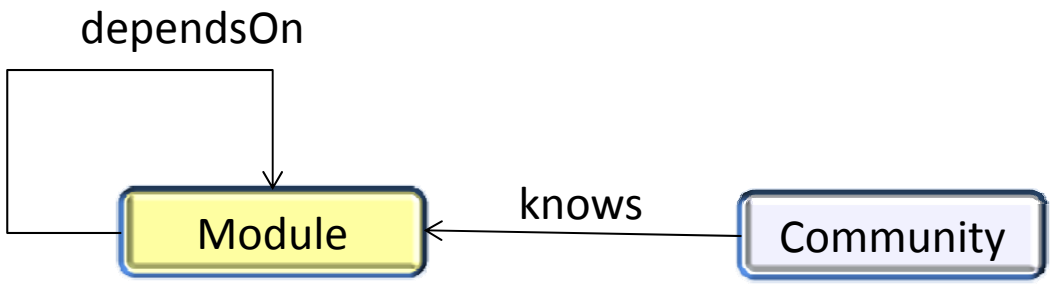
FORTH
Institute of Computer Science

Benefits (from dependency management)

- Assists in making our (inevitable) assumptions explicit and manageable
- Helps identifying (and reducing) how much dependencies we have to maintain
- Allows deriving (archival or dissemination) packages in a way based on the Designated Community
- Enables some form of quality control and risk detection

FORTH
Institute of Computer Science

Modules, Dependencies and Community Knowledge



Module

We adopt a very general definition. A module could be:

- a piece of software/ hardware module.
- a knowledge model expressed explicitly and formally (e.g. an Ontology)
- a knowledge model not expressed explicitly (e.g. GreekLanguage)

(the only constraint is that modules need to have a unique identity)

Dependency

A module t *dependsOn* t', written t>t', if t requires t'

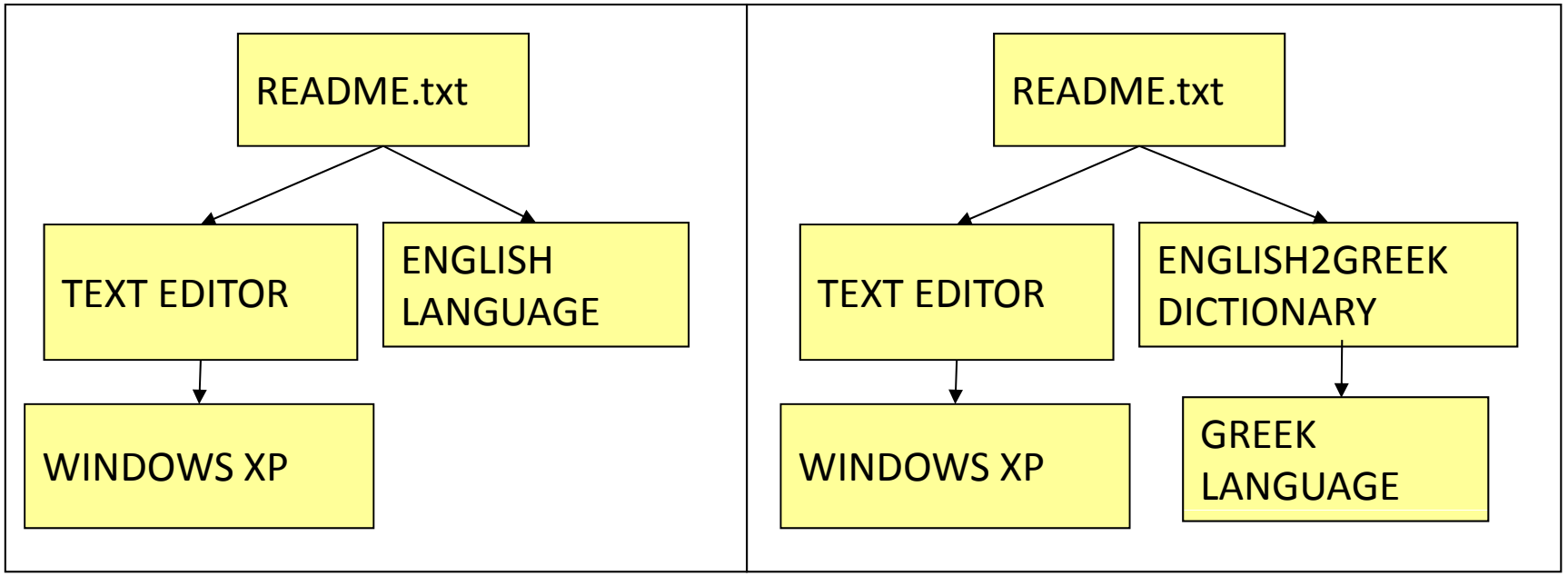
The meaning of a dependency t > t' is that t cannot function/be understood/managed without the existence of t'

Note: We can model the RI (Representation Information) requirements of OAIS as dependencies between modules.

FORTH
Institute of Computer Science

Modules and **Dependencies:** Examples

dependsOn →



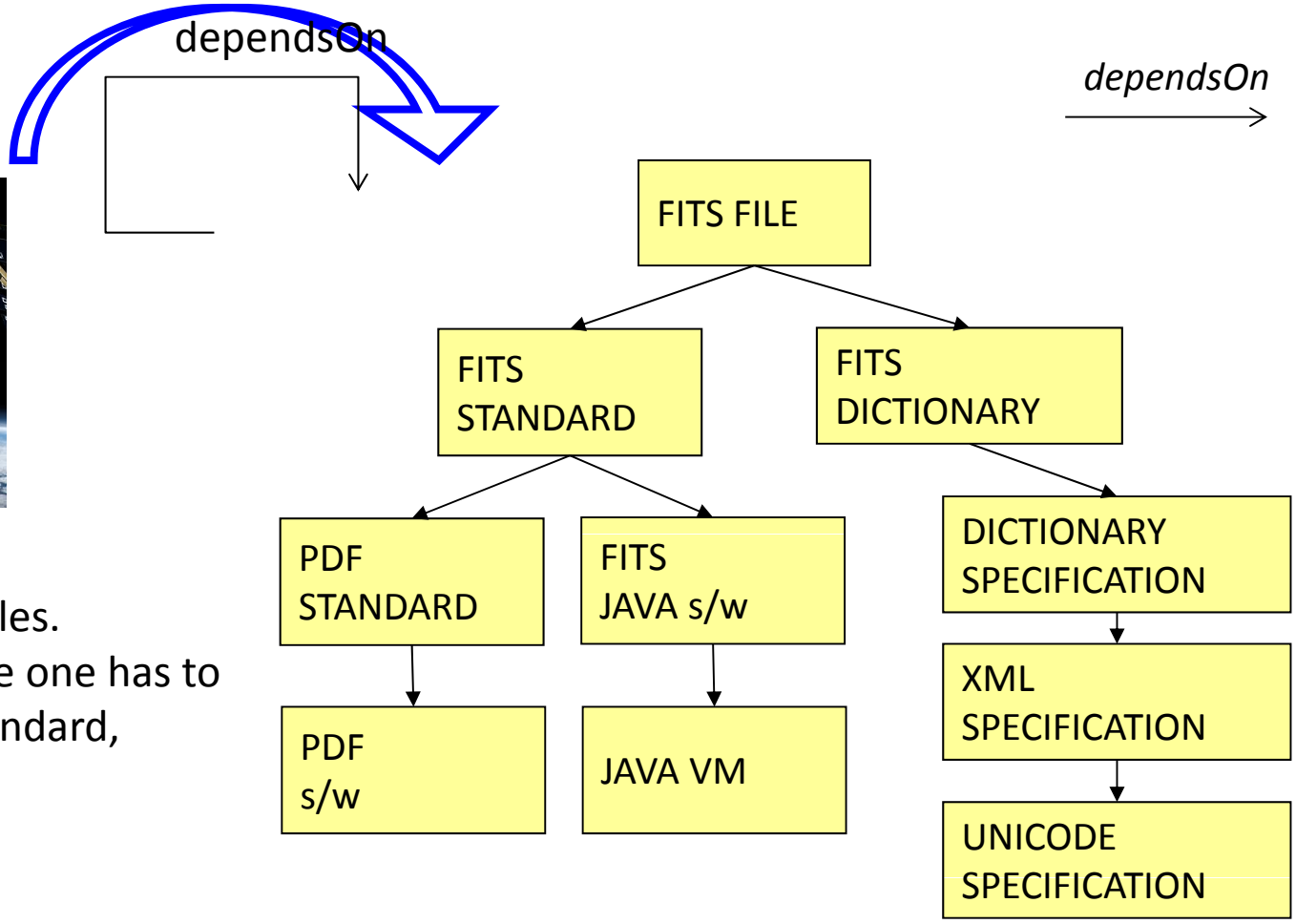
(a) (b)

FORTH
Institute of Computer Science

Modules and **Dependencies:** Examples

Scientific Data

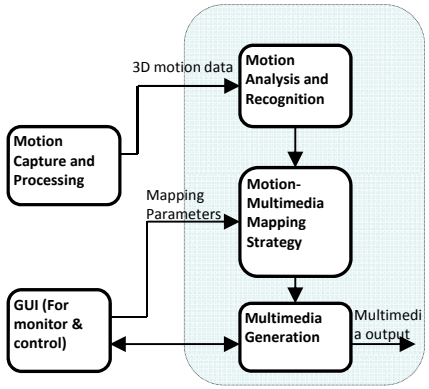
Astronomers use FITS files.
To understand a FITS file one has to understand the FITS standard, and so on.

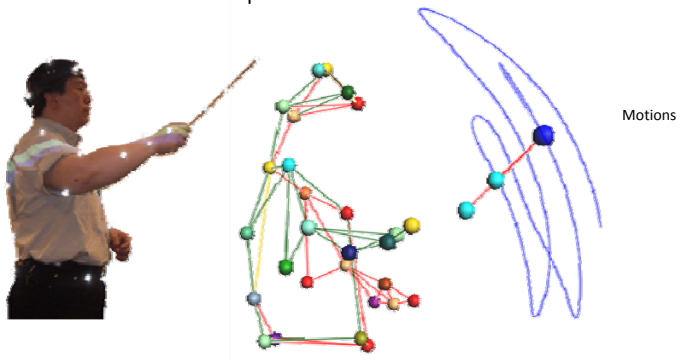


FORTH
Institute of Computer Science

Modules and **Dependencies:** Examples

Performing Arts Data





dependsOn →

MULTIMEDIA PERFORMANCE DATA

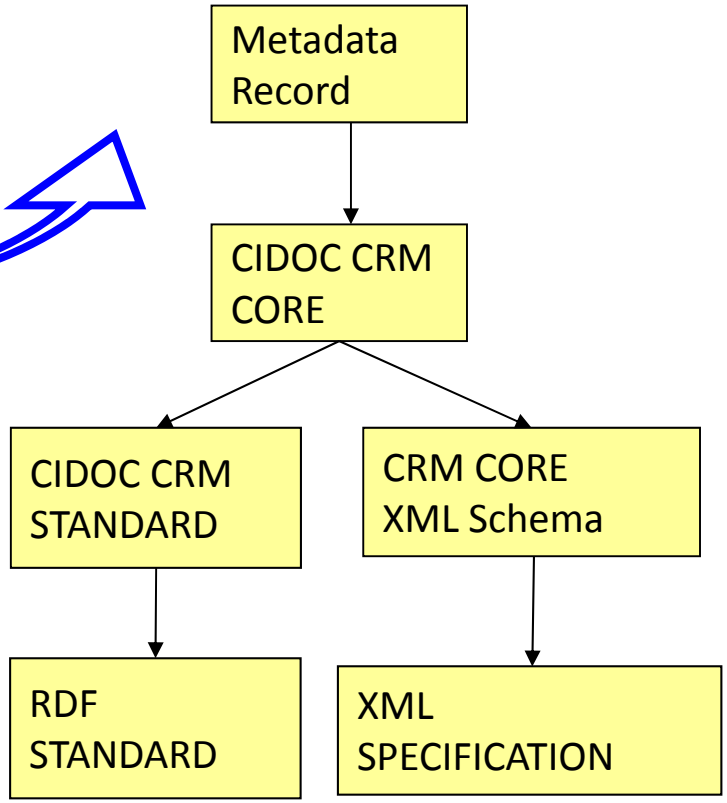
C3D	DirectX	MAX/MSP
3D motion data files	*3D scene data files*	*motion to music mapping strategy*

FORTH
Institute of Computer Science

Modules and **Dependencies:** Examples

Cultural Data

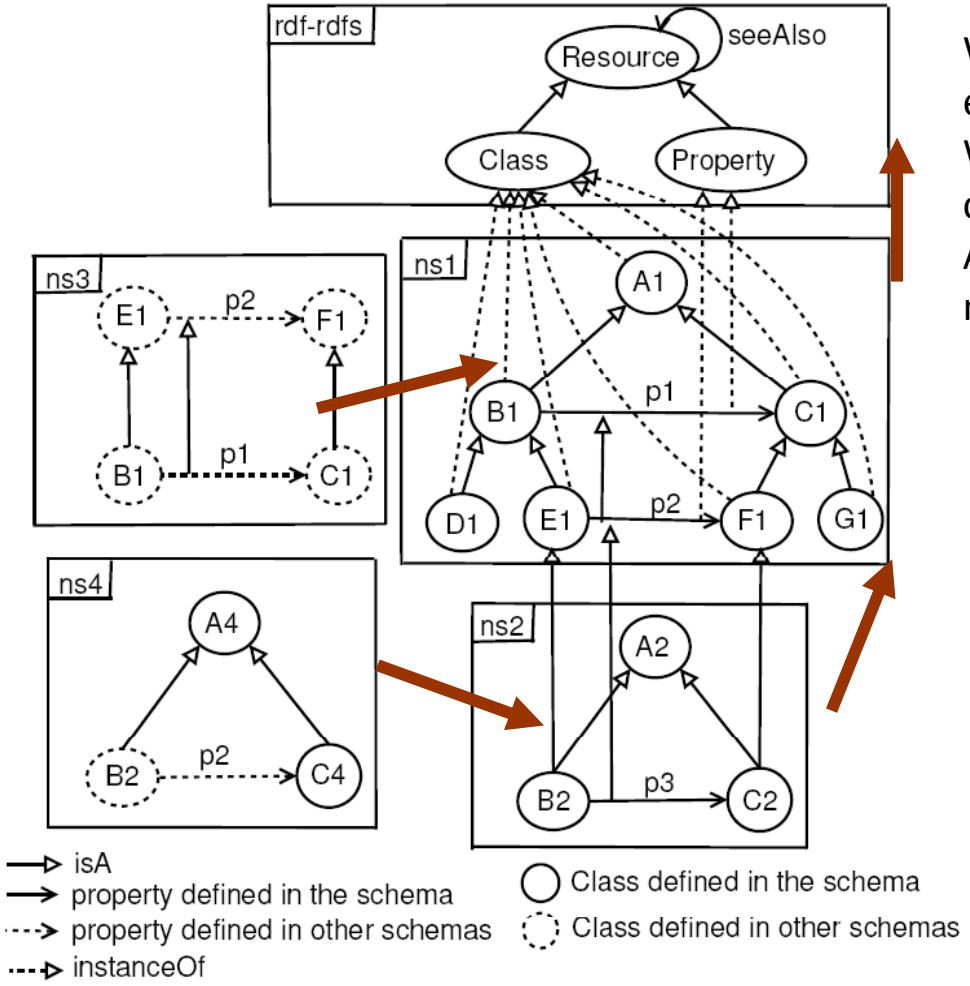
dependsOn →



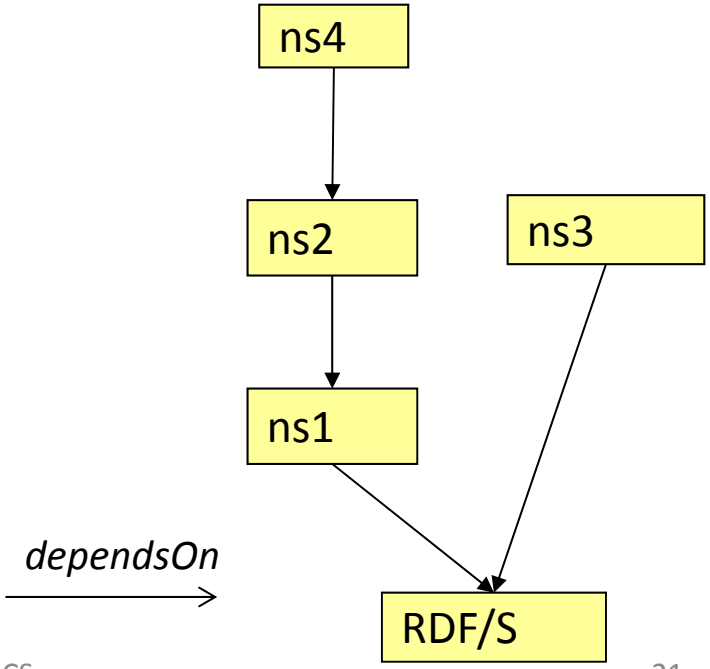
FORTH
Institute of Computer Science

Modules and **Dependencies:** Examples

Semantic Web Data

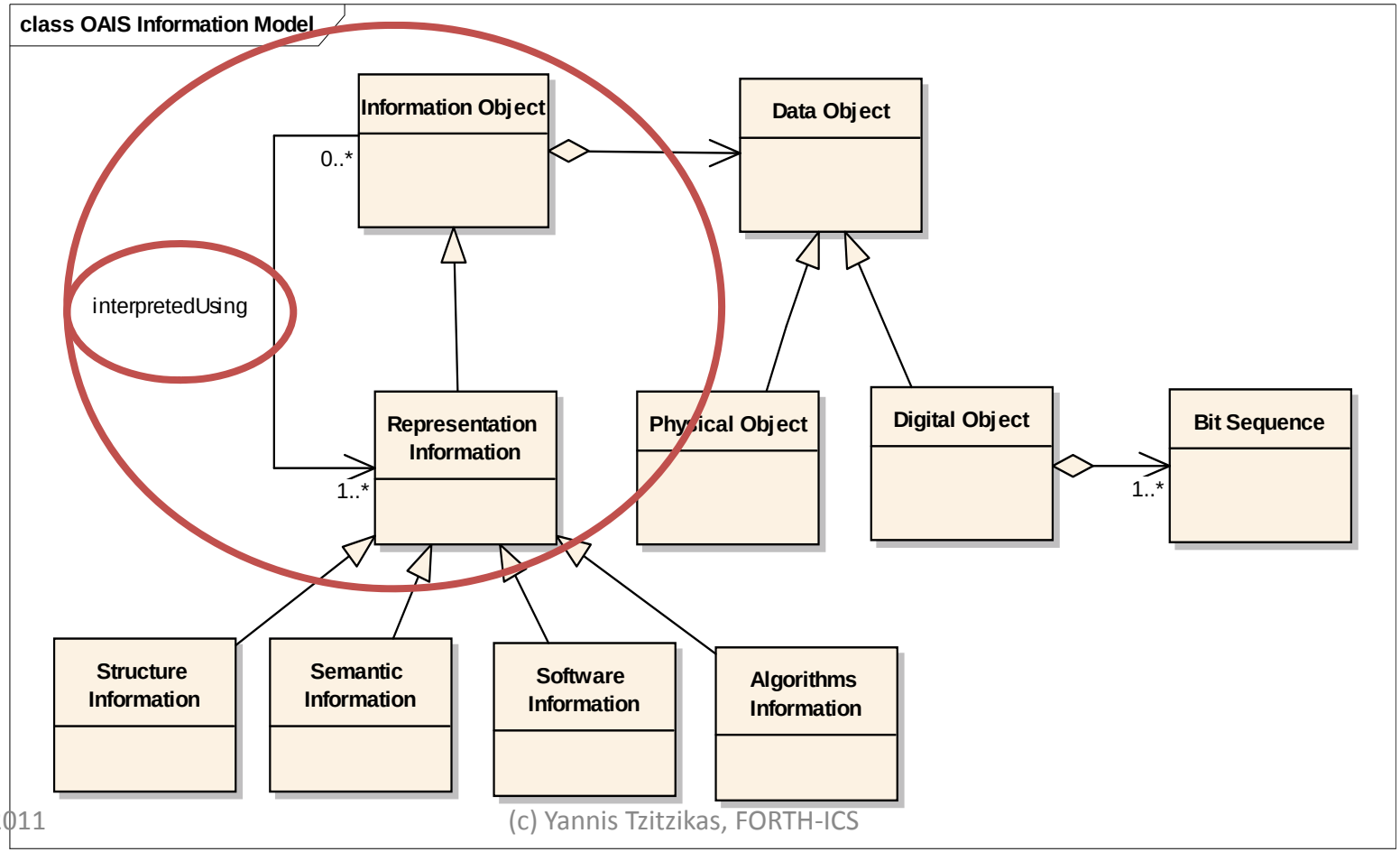


We encounter the same pattern also in formally expressed knowledge. For instance, in the Semantic Web one schema may reuse or specialize elements coming from other schemas.
Also in this case we have a dependency graph (over namespaces this time).



Modules and Dependencies and OAIS

- According to **OAIS** (Open Archival Information System – ISO 14721:2003), metadata are distinguished to various categories. One very important is that of ***Representation Information:*** aims at enabling the conversion of a collection of bits to something useful



FORTH
Institute of Computer Science

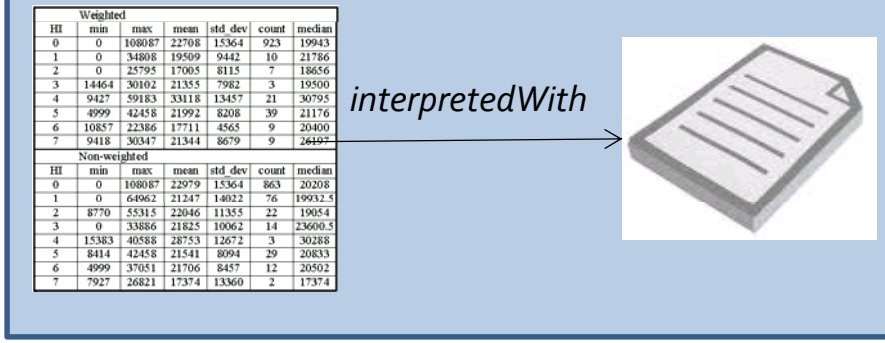
OAIS RepInfo

- According to OAIS, metadata are distinguished to various categories.
- One very important is that of ***Representation Information (RepInfo)***
  - **RepInfo aims at enabling the conversion of a collection of bits to something useful**

data object + *rep. info* =

Weighted						
HI	min	max	mean	std_dev	count	median
0	0	108087	22708	15364	923	19943
1	0	34808	19509	9442	10	21786
2	0	25795	17005	8115	7	18656
3	14464	30102	21355	7982	3	19500
4	9427	59183	33118	13457	21	30795
5	4999	42458	21992	8208	39	21176
6	10857	22386	17711	4565	9	20400
7	9418	30347	21344	8679	9	26197
Non-weighted						
HI	min	max	mean	std_dev	count	median
0	0	108087	22979	15364	863	20208
1	0	64962	21247	14022	76	19932.5
2	8770	55315	22046	11355	22	19054
3	0	33886	21825	10062	14	23600.5
4	15383	40588	28753	12672	3	30288
5	8414	42458	21541	8094	29	20833
6	4999	37051	21706	8457	12	20502
7	7927	26821	17374	13360	2	17374

Information Object



There are specific packaging formats for packaging together the data object and its RepInfo (e.g. XFDU)

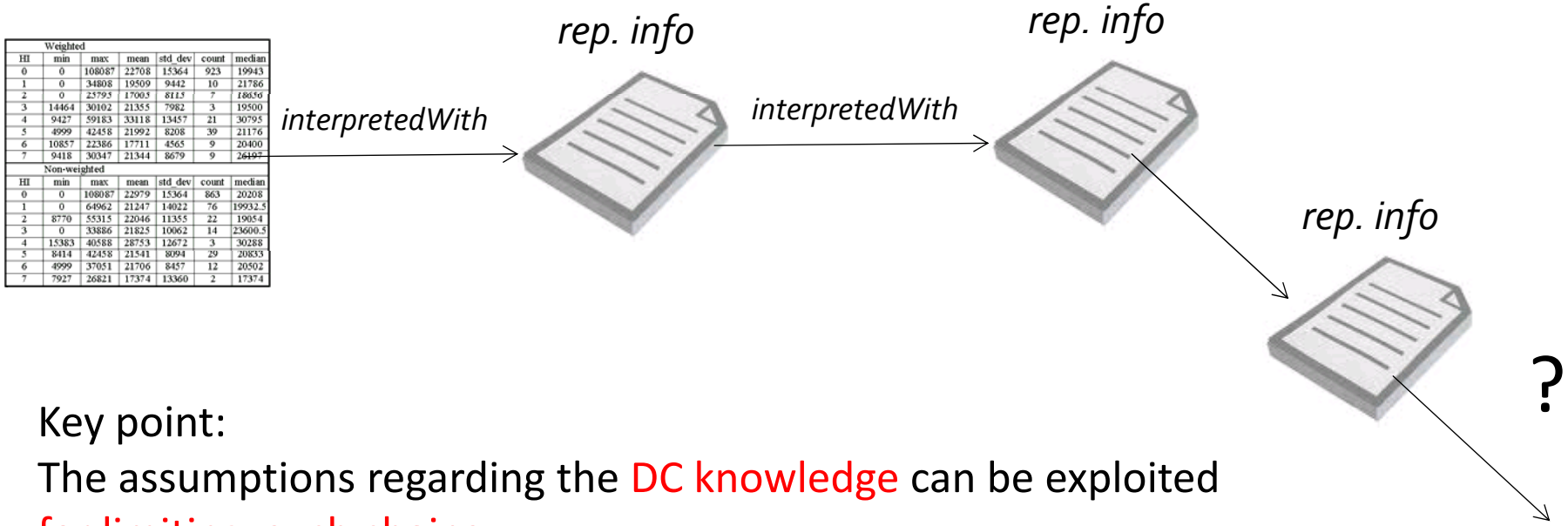
Important questions:

- ***What kind (and how much) representation information do we need?***
- ***How this depends on the Designated community?***

FORTH
Institute of Computer Science

(cont)

RepInfo is itself a digital object, so it may also need other RepInfo, and so on.



Weighted

HI	min	max	mean	std_dev	count	median
0	0	108087	22708	15364	923	19943
1	0	34808	19509	9442	10	21786
2	0	25795	17005	8115	7	18856
3	14464	30102	21355	7982	3	19500
4	9427	59183	33118	13457	21	30795
5	4999	42458	21992	8208	39	21176
6	10857	22386	17711	4565	9	20400
7	9418	30347	21344	8679	9	26197

Non-weighted

HI	min	max	mean	std_dev	count	median
0	0	108087	22979	15364	863	20208
1	0	64962	21247	14022	76	19932.5
2	8770	55315	22046	11355	22	19054
3	0	33886	21825	10062	14	23600.5
4	15383	40588	28753	12672	3	30288
5	8414	42458	21541	8094	29	20833
6	4999	37051	21706	8457	12	20502
7	7927	26821	17374	13360	2	17374

Key point:
The assumptions regarding the DC knowledge can be exploited for limiting such chains.

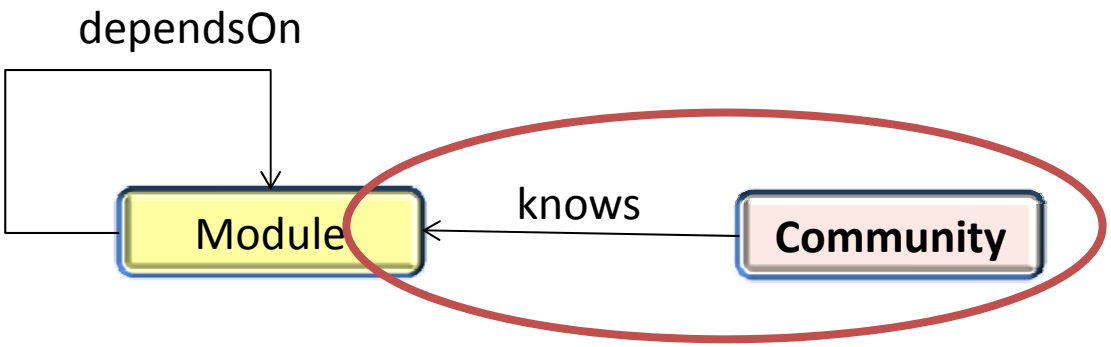
FORTH
Institute of Computer Science

Important Questions

- *What kind (and how much) representation information do we need?*
- *How this depends on the Designated community?*

FORTH
Institute of Computer Science

Modules, Dependencies and Community Knowledge



Community

Each actor or community u can be characterized by a ***profile*** that contains those modules that are assumed to be available or known to u.

FORTH
Institute of Computer Science

Intelligibility and *Intelligibility Gap*

- Intelligibility
  - Definition (dictionary)
    - **1.** Capable of being understood: *an intelligible set of directions.*
    - **2.** Capable of being apprehended by the intellect alone.

- Intelligibility Gap
  - Our Definition:
    - ***The smallest set of extra modules that u needs to have in order to understand a module t.***
  - Notation
    - **Gap(t,u):** The intelligibility gap between a user **u** with profile **Tu** and a module **t**

FORTH
Institute of Computer Science

Intelligibility Gap and Archiving/Dissemination

This means that:

→ if we want to preserve (archive) a digital object t for a community with profile Tu, then we need to get and store only Gap(t,Tu) plus an id that denotes Tu.

→if we want to deliver an object t to an actor with profile Tu, then the only extra modules that we should deliver to him in order to return him something intelligible, is the set Gap(t,Tu).

Note that OAIS defines:

AIP: Archival Information Package

DIP: Dissemination Information Package

Clearly, we can exploit our approach for the formation of AIPs/DIPs

(the packaging approach actually corresponds to the *encapsulation* preservation strategy)

FORTH
Institute of Computer Science

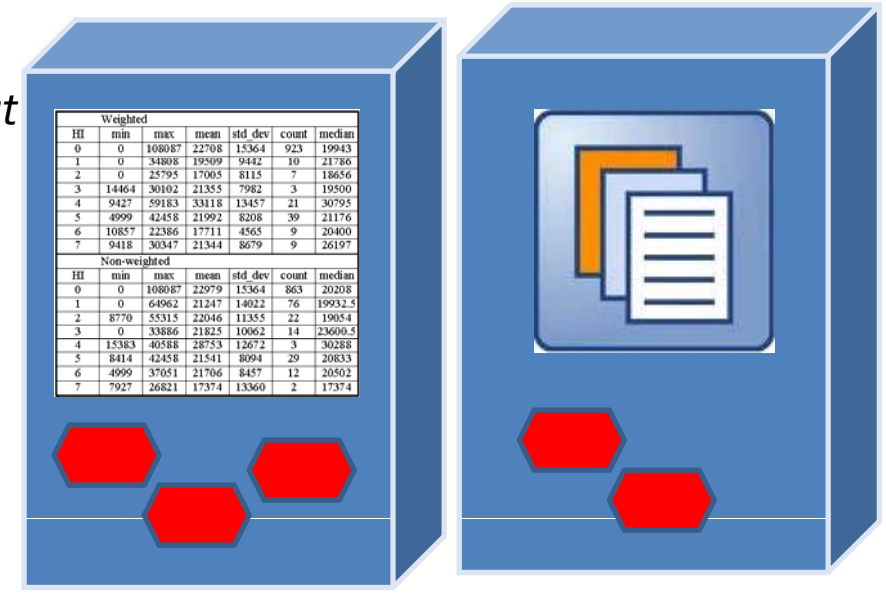
Dependencies and Packaging

digital files

packages

Weighted						
HI	min	max	mean	std_dev	count	median
0	0	108087	22708	15364	923	19943
1	0	34808	19509	9442	10	21786
2	0	25795	17005	8115	7	18656
3	14464	30102	21355	7982	3	19500
4	9427	59183	33118	13457	21	30795
5	4999	42458	21992	8208	39	21176
6	10857	22386	17711	4565	9	20400
7	9418	30347	21344	8679	9	26197
Non-weighted						
HI	min	max	mean	std_dev	count	median
0	0	108087	22979	15364	863	20208
1	0	64962	21247	14022	76	19932.5
2	8770	55315	22046	11355	22	19054
3	0	33886	21825	10062	14	23600.5
4	15383	40588	28753	12672	3	30288
5	8414	42458	21541	8094	29	20833
6	4999	37051	21706	8457	12	20502
7	7927	26821	17374	13360	2	17374

assumptions about the designated community



extra modules which are required for understanding a digital object

The extra modules that should be included in a package should be based on:

(a) The dependencies of the object (based on the task(s) we want to preserve)

(b) The assumptions regarding the knowledge of the designated community

FORTH
Institute of Computer Science

Preservation Information Systems (PIS)

(that supports dependency management)

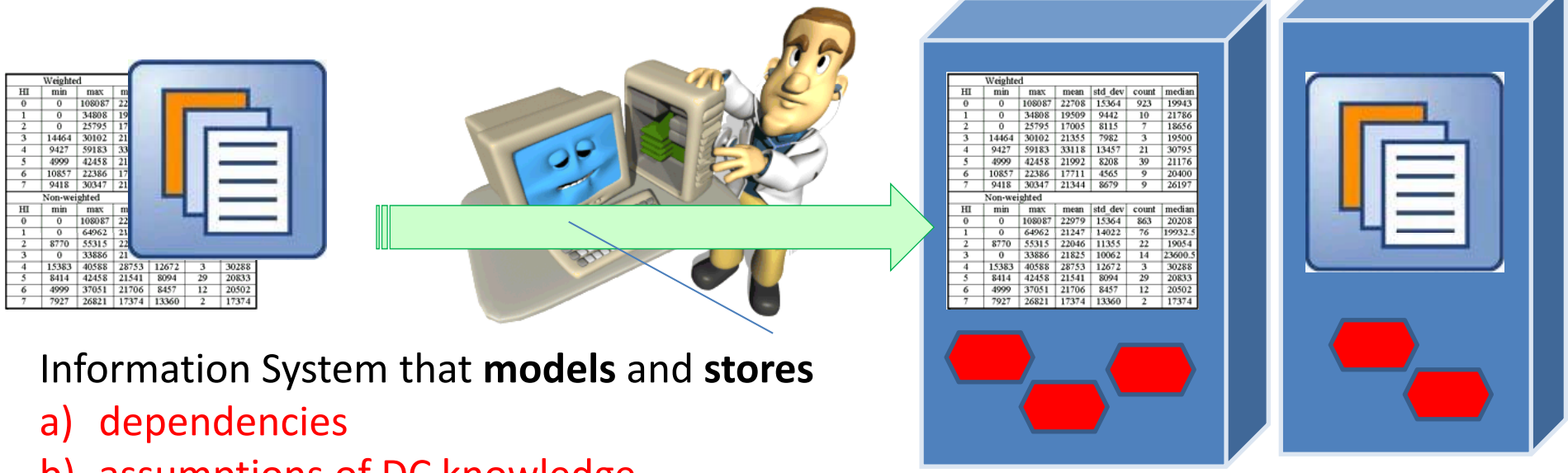
FORTH
Institute of Computer Science

Preservation Information Systems (PIS)

(that supports dependency management)

digital files

packages



Weighted						
HI	min	max	mean	std dev	count	median
0	0	108087	22708	15364	923	19943
1	0	34808	19509	9442	10	21786
2	0	25795	17005	8115	7	18656
3	14464	30102	21355	7982	3	19500
4	9427	59183	33118	13457	21	30795
5	4999	42458	21992	8208	39	21176
6	10857	22386	17711	4565	9	20400
7	9418	30347	21344	8679	9	26197
Non-weighted						
HI	min	max	mean	std dev	count	median
0	0	108087	22979	15364	863	20208
1	0	64962	21247	14022	76	19932.5
2	8770	55315	22046	11355	22	19054
3	0	33886	21825	10062	14	23600.5
4	15383	40588	28753	12672	3	30288
5	8414	42458	21541	8094	29	20833
6	4999	37051	21706	8457	12	20502
7	7927	26821	17374	13360	2	17374

Information System that **models** and **stores**

a) dependencies

b) assumptions of DC knowledge

It provides a number of **services** for

- *checking the intelligibility* of packages (e.g. checking the *performability* of tasks on objects)
- *identifying risks* (e.g. if something gets lost)

FORTH
Institute of Computer Science

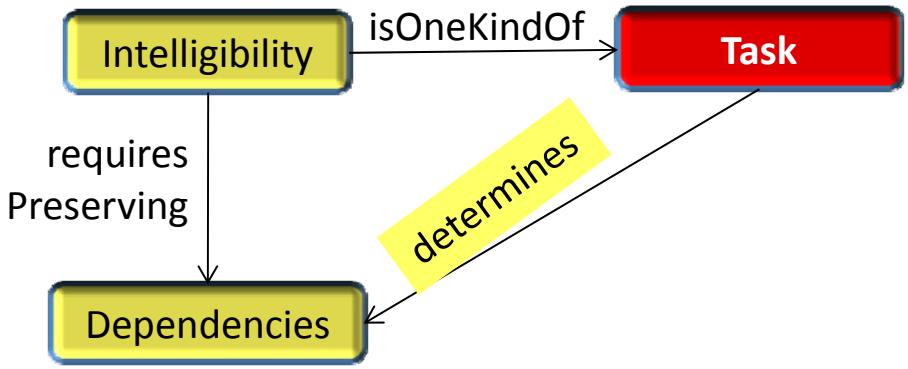
(cont)

Policies that could be adopted by a PIS:

- Input (ingestion) Policy
  - The **input** (e.g. data objects to be archived) **should be intelligible by the system**
- Output Policy
  - The **output** (e.g. returned answers) **should be intelligible by the recipients**
- ***The notion of DC profile can be used as gnomon in these policies. E.g.***
  - Answers are accompanied by their *closure, i.e. their full set of dependencies* (delivered in one shot or gradually), or
  - The user first registers his profile(s) and then the answers which are given to that user are accompanied only by their *intelligibility gaps* (wrt the registered profile).

FORTH
Institute of Computer Science

ONE DIFFICULTY OF
MODELING AND RECORDING
DEPENDENCIES
AND AN APPROACH TO
TACKLE IT (TASK-BASED
DEPENDENCIES)



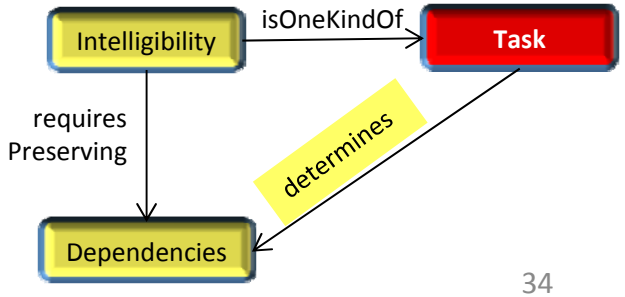
FORTH
Institute of Computer Science

One Difficulty of Modeling and Recording Dependencies

- There is not any objective method to specify what the dependencies of a digital object are!
- OAIS distinguishes two main types of dependencies *structure* and *semantics*.
  - The first can be expressed as an EAST file, the second as a DESDL file (as we shall see in the section "example with scientific data")
- However this is not very helpful, nor complete.

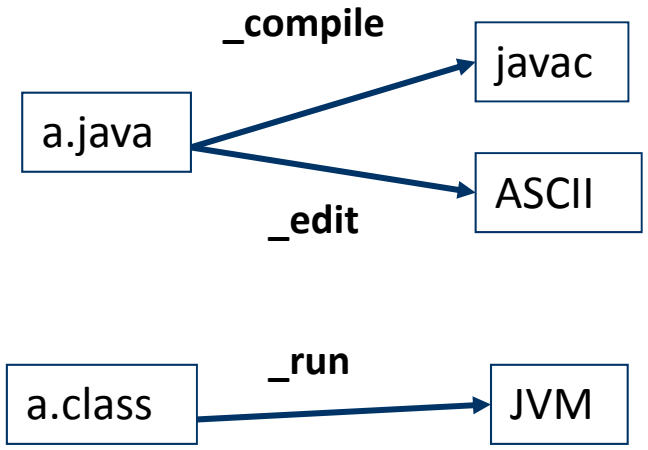
- In general, it is the **goal** (task) that determines the dependencies of a digital object.
- → For this reason we introduce what we call

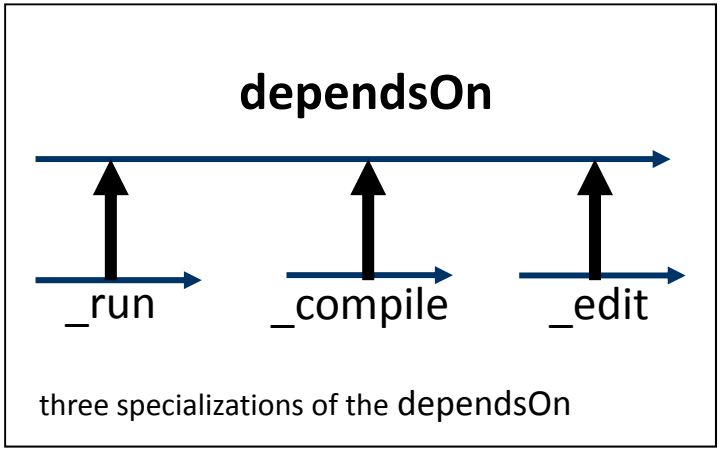
  task-dependency



FORTH
Institute of Computer Science

Task-based Dependencies (one example)





Modeling three task-based dependencies as refinements of the *dependOn* relation

FORTH
Institute of Computer Science

Task-based dependencies and intelligibility-gaps

- Now the notion of gap can be *parametric* (with the types of dependencies to be considered)
- However the way various dependency types *interact* is more complex, making more difficult the specification of the method that yields the intelligibility gaps for a given set of dependency types.

FORTH
Institute of Computer Science

ONE DIFFICULTY OF COMPUTING THE *CLOSURE* (THE SET OF ALL DEPENDENCIES)

FORTH
Institute of Computer Science

Example: Type Dependencies

- Each object depends on its type (e.g. format type).
- However objects with the *same type* can have *different dependencies*.
- If its type is not intelligible then we cannot identify the dependencies of one particular object

FORTH
Institute of Computer Science

Example: Type Dependencies (cont)

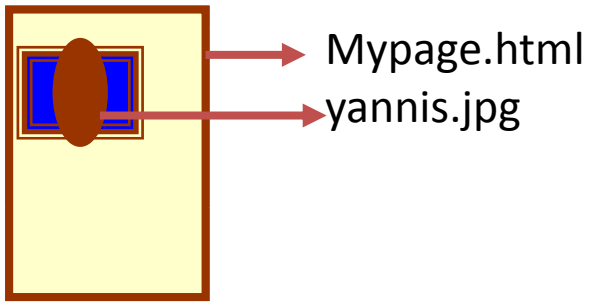
- Consider my home Web page **`mypage.html`** that includes my photo, say **`yannis.jpg`**
- The extension of the filename gives us a hint about the type of the digital object, so we may write

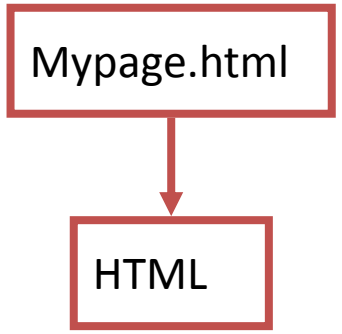
  type(mypage.html) =HTML.
- In general we can assume that: for every **o** it holds:

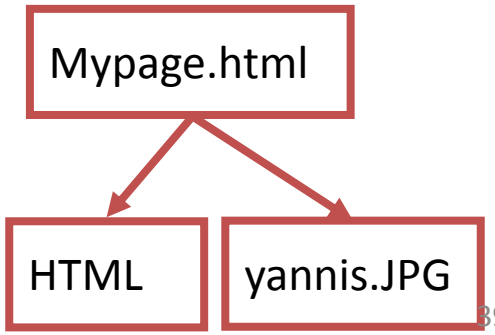
  o > type(o) i.e. o depends on its type
- in our case we have the dependency relationship:

  mypage.html > HTML.
- However **only if the type HTML is intelligible** (and we can parse it) we can realize the dependency

  mypage.html > yannis.jpg.
- Without a parser we cannot deduce the dependency **mypage.html > yannis.JPG** (and subsequently **yannis.JPG > JPG).**







- ***So we may be unable to compute the set of all dependencies***
- ***We may be able to compute only one part of the direct dependencies***

FORTH
Institute of Computer Science

WHERE AND *HOW* TO STORE THE DEPENDENCIES

FORTH
Institute of Computer Science

Where and *how* to store the dependencies

- Special case of the general question: *where to store the metadata of an object?*
- Two main approaches
  - A/ **Packaging approach**: put the object plus its dependencies into a package (*encapsulation preservation strategy*)
    - How to represent a dependency in the package?
      - By its identity (e.g. URI)?
      - By putting into the package the entire object ?
  - B/ **Repository approach** (keep all in an information system with integrity constraints)
- Discussion
  - A/ Packaging approach
    - **Plus**: objects are moved with their dependencies
    - **Minus**: to change/update dependencies we have to change a lot of packages
  - B/ Repository approach
    - **Plus**: coherent network, referential integrity
    - **Minus**: need to have and maintain a system

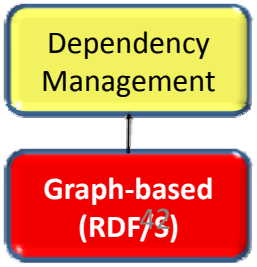
FORTH
Institute of Computer Science

A GRAPH-BASED MODEL (FOR DEPENDENCY MANAGEMENT)

Main Points

An indicative instance level modeling

Refining the model (in an O-O manner)



FORTH
Institute of Computer Science

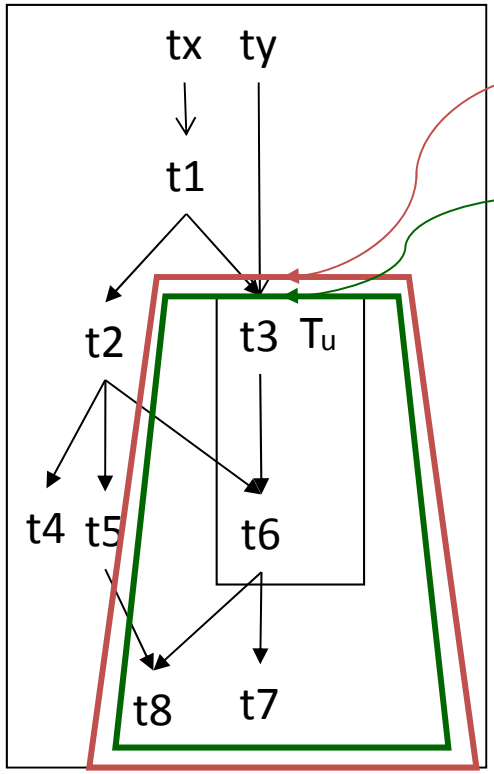
A Graph-based model for Dependency Management

- Main Points
  - Modules and dependencies actually form a *dependency graph*
  - We can extend/refine the graph according to object-oriented principles to tackle the particular requirements at hand (module hierarchies, task-based dependencies, etc)

- This approach was followed in CASPAR. An example is the GapManager system (described later on)

FORTH
Institute of Computer Science

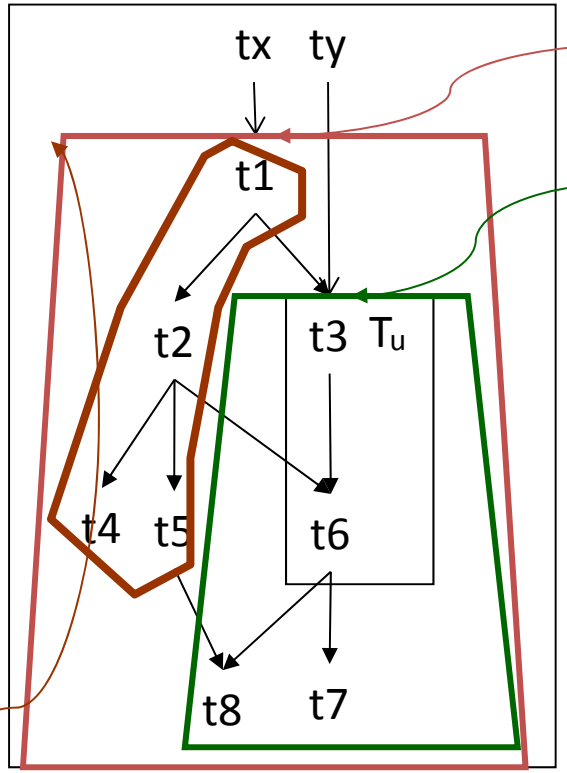
Indicative instance level modeling
(dependency graph, profile, intelligibility gap)



Requirements of ty

Closure of a profile Tu

Gap(ty,u)= ∅



Reqs of tx

Closure of Tu

Gap(tx,u)= {t1, t2, t4, t5}

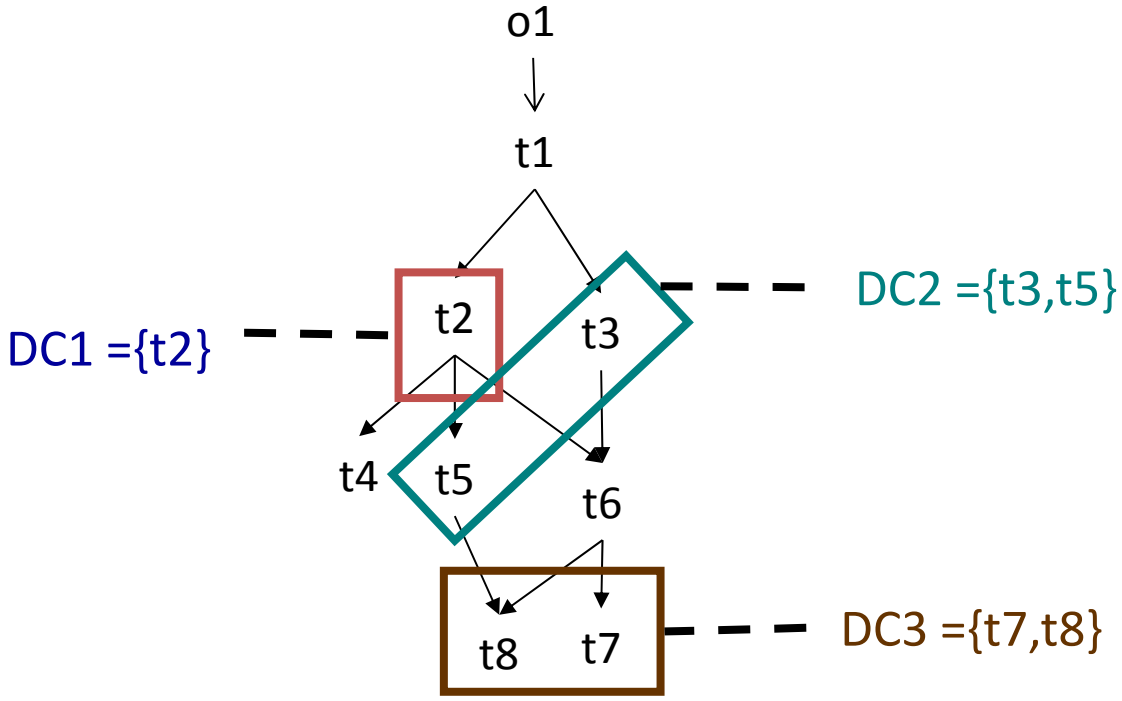
In this example we assume that the dependencies are *transitive.*

Packaging Example

Exploiting DC Profiles for constructing the "right" (intelligible and redundancy free) AIPs

Exploiting Designated Community profiles for deriving different AIPs and DIPs for different DCommunities

(If the dependencies are available this could be done automatically)



AIP of o1 wrt DC1

Object	= o1
DCprofile	= DC1
deps	= {t1,t3}

AIP of o1 wrt DC2

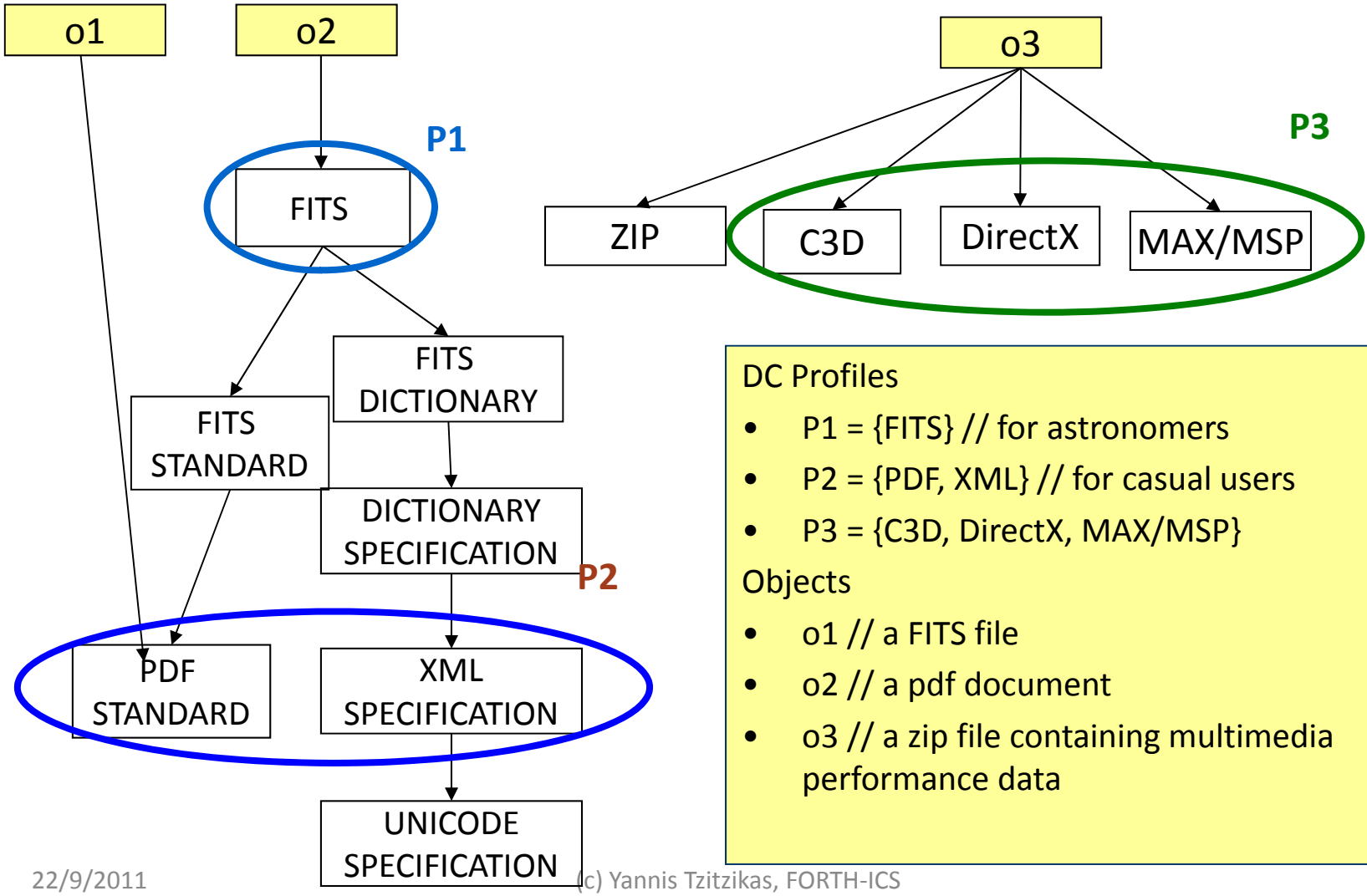
Object	= o1
DCprofile	= DC2
deps	= {t1,t2,t4}

AIP of o1 wrt DC3

Object	= o1
DCprofile	= DC3
deps	= {t1,t2,t3,t4,t5,t6}

FORTH
Institute of Computer Science

Example



DC Profiles

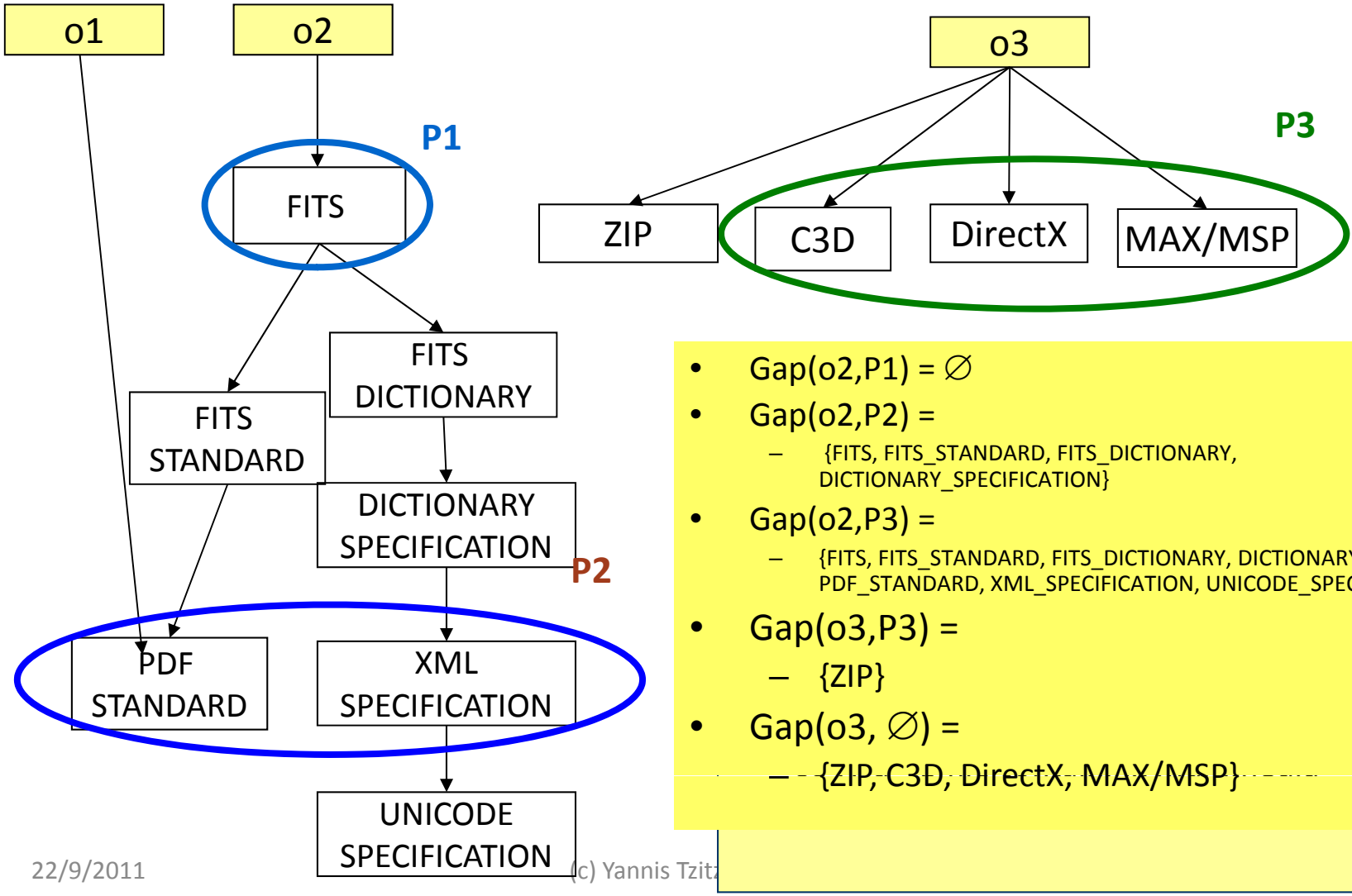
- P1 = {FITS} // for astronomers
- P2 = {PDF, XML} // for casual users
- P3 = {C3D, DirectX, MAX/MSP}

Objects

- o1 // a FITS file
- o2 // a pdf document
- o3 // a zip file containing multimedia performance data

FORTH
Institute of Computer Science

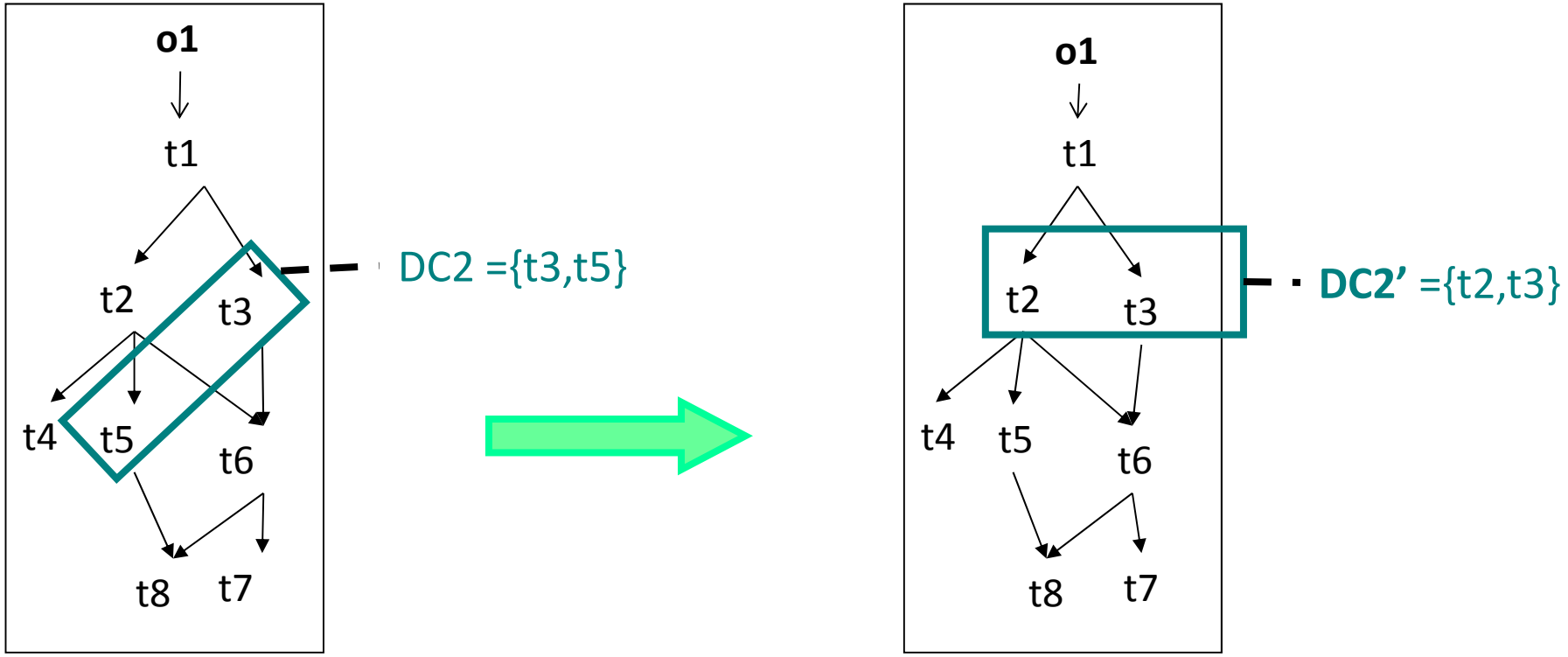
Example (cont)



- Gap(o2,P1) = ∅
- Gap(o2,P2) =
  - {FITS, FITS_STANDARD, FITS_DICTIONARY, DICTIONARY_SPECIFICATION}
- Gap(o2,P3) =
  - {FITS, FITS_STANDARD, FITS_DICTIONARY, DICTIONARY_SPECIFICATION, PDF_STANDARD, XML_SPECIFICATION, UNICODE_SPECIFICATION}
- Gap(o3,P3) =
  - {ZIP}
- Gap(o3, ∅) =
  - {ZIP, C3D, DirectX, MAX/MSP}

Example of repackaging

(when community knowledge evolves)



AIP of o1 wrt DC2

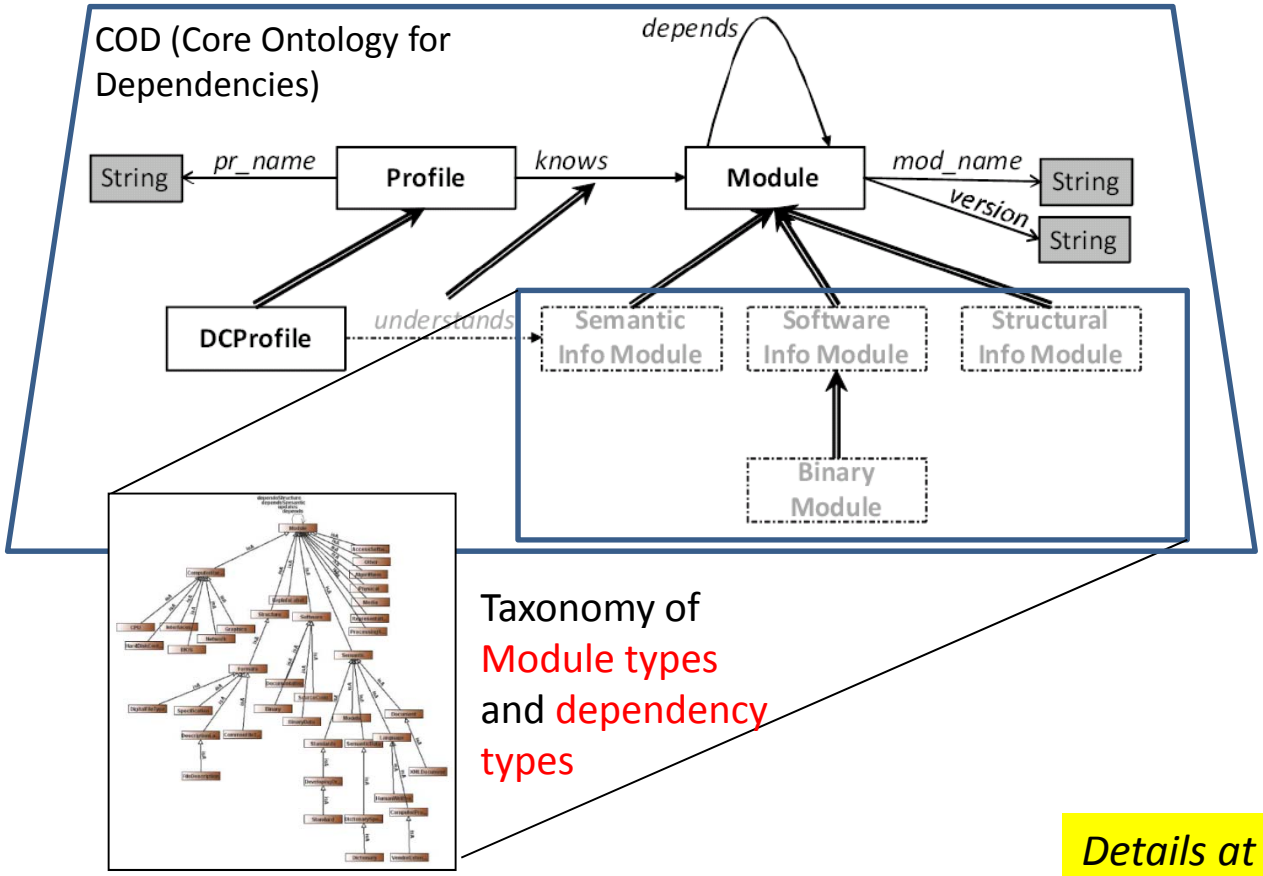
Object	= o1
DCprofile	= DC2
deps	= {t1,t2,t4}

AIP of o1 wrt **DC2'**

Object	= o1
DCprofile	= DC2
deps	= {t1}

FORTH
Institute of Computer Science

Refining the Model (in an O-O manner)



Taxonomy of Module types and dependency types

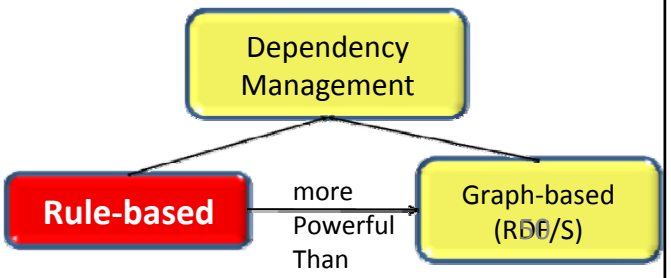
Representation framework: RDF/S

Details at the IJDL'2009 paper

FORTH
Institute of Computer Science

A RULE-BASED MODEL (FOR DEPENDENCY MANAGEMENT)

Motivation and Requirements
An Example from Software Engineering
Architecture and Implementation Approaches



FORTH
Institute of Computer Science

A Rule-based Model: Motivation

Objective

- To offer a more flexible and more expressive method to model dependencies that supports
  - task-based dependencies
    - plus ability to define their properties (e.g. *transitivity)*
  - disjunction
    - e.g. we want to be able to express that one txt file depends on an editor (no matter it is notepad, wordpad, or vi).

FORTH
Institute of Computer Science

Motivation: Requirements

Modeling Requirements

- **Task hierarchies**: Task-type hierarchies offer extra flexibility and also reduce the number of rules that have to be defined.
- **Properties of Dependencies**: Some dependencies are transitive, some are not. We should be able to define the properties of each kind of dependencies.

Functional Requirements

- **Task-Performability** *Checking*
  - To perform a task we have to perform several subtasks and fulfill associated requirements for carrying out these tasks (i.e. have the necessary modules available).
- **Risk** *Detection*
  - The removal of a software module would affect the performability of the tasks that depend on it. So we must be able to identify the tasks that are going to be affected by such removals.
- *Identification* of missing modules:
  - Ability to find the modules that which are needed (or missing) to perform a task.

FORTH
Institute of Computer Science

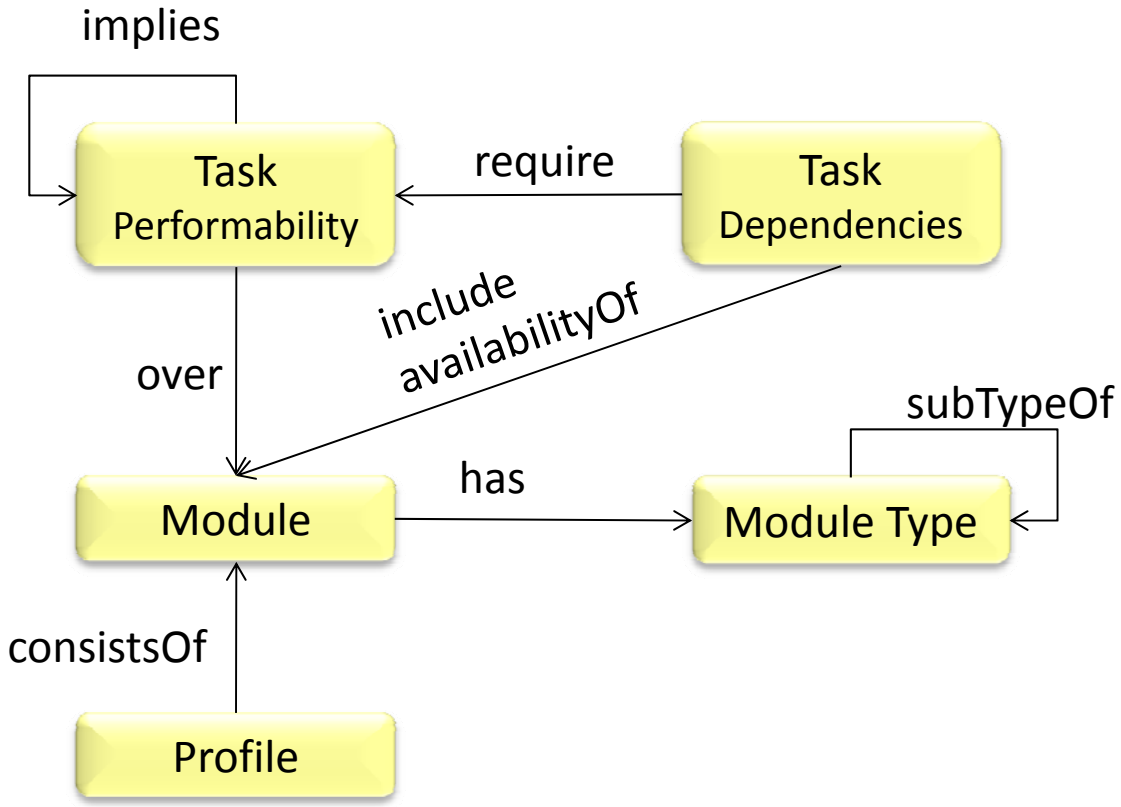
The Proposed Approach

- Use **facts** and **rules** to model
  - digital objects
  - tasks
  - dependencies
  - DC knowledge
- Use **query answering** or **reasoning** to realize the PIS functionalities

FORTH
Institute of Computer Science

The key notions of the proposed approach

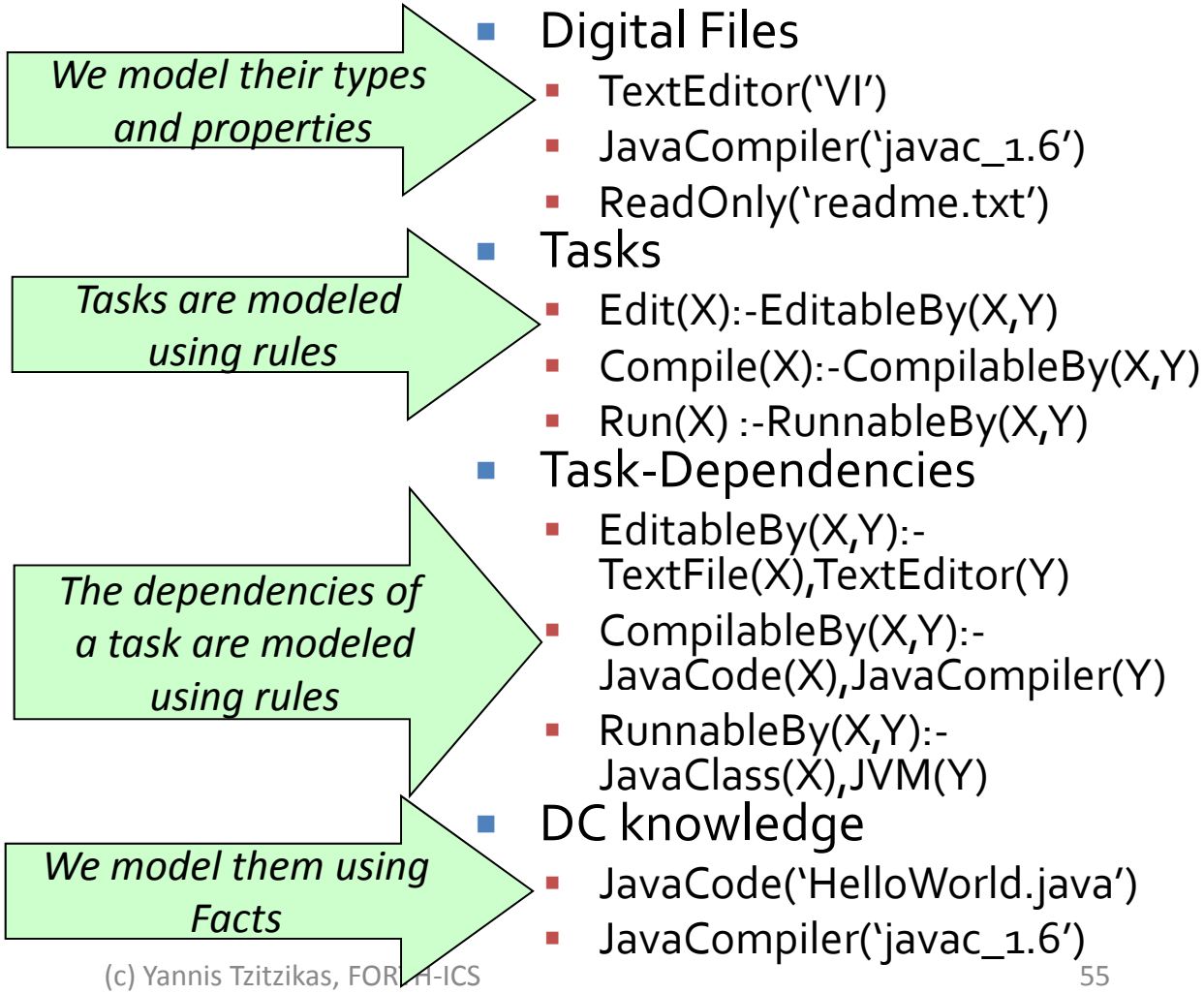
- The following diagram depicts some basic notions in the form of a structural conceptual diagram. However the rule-based modeling cannot be represented precisely as a graph.



FORTH
Institute of Computer Science

An Example from the Software domain

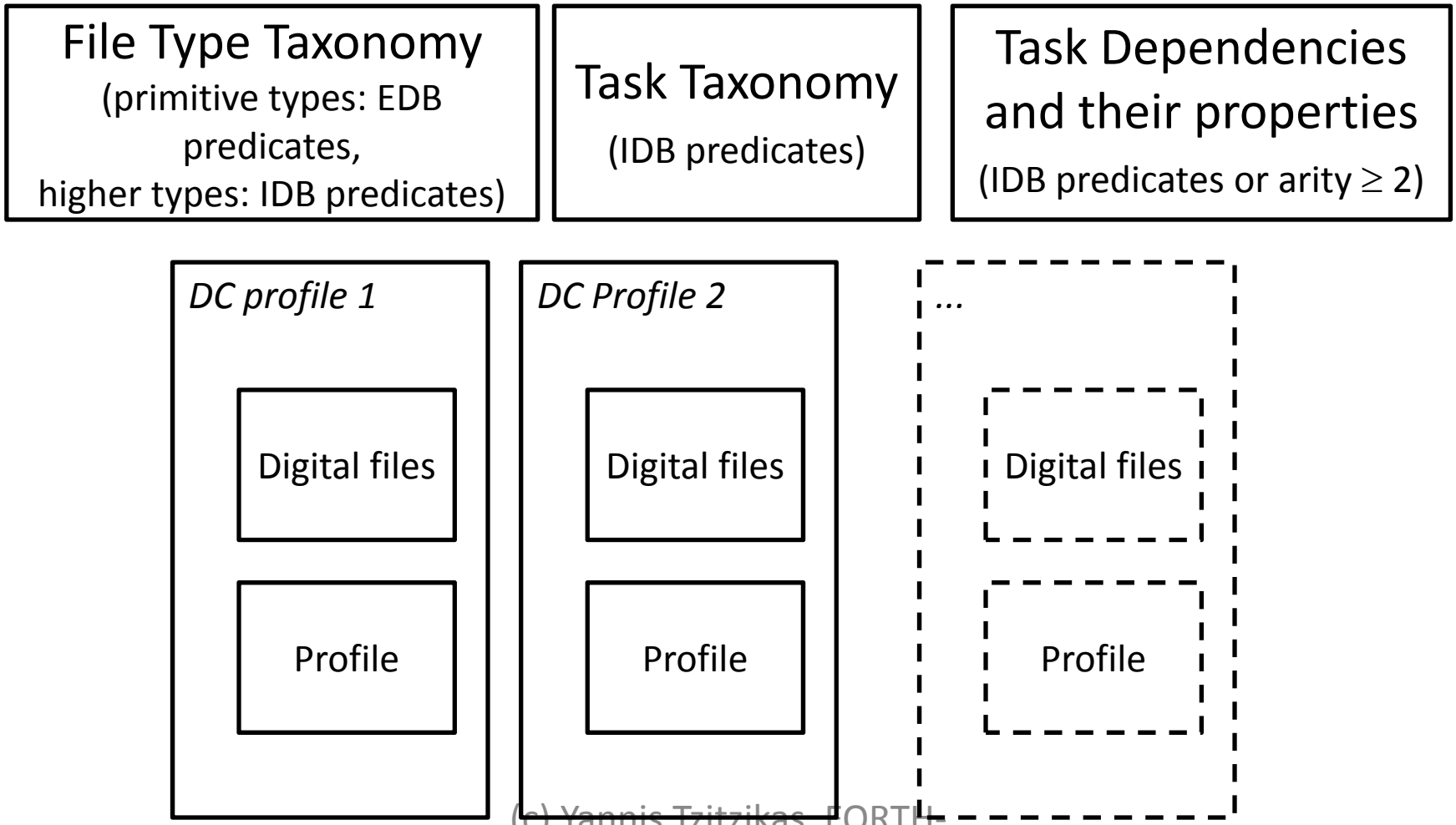
- Digital Files
  - Files containing java **code**, **compilers**, **editors** etc.
- Tasks
  - **Edit**
  - **Compile**
  - **Run**
- Task-Dependencies
  - **Edit**: existence of editors
  - **Compile**: existence of imported files, compilers
  - **Run**: existence of java virtual machine
- DC knowledge
  - The **files** that are available to one particular user



- Digital Files
  - TextEditor('VI')
  - JavaCompiler('javac_1.6')
  - ReadOnly('readme.txt')
- Tasks
  - Edit(X):-EditableBy(X,Y)
  - Compile(X):-CompilableBy(X,Y)
  - Run(X) :-RunnableBy(X,Y)
- Task-Dependencies
  - EditableBy(X,Y):-TextFile(X),TextEditor(Y)
  - CompilableBy(X,Y):-JavaCode(X),JavaCompiler(Y)
  - RunnableBy(X,Y):-JavaClass(X),JVM(Y)
- DC knowledge
  - JavaCode('HelloWorld.java')
  - JavaCompiler('javac_1.6')

FORTH
Institute of Computer Science

Knowledge Architecture



FORTH
Institute of Computer Science

(cont)

- ***Check* Compilability** of `HelloWorld.java` (*Task Performability*)
  - Relies on Datalog Query answering over the profiles of the user
    - **Compile(X)** will contain the files that can be compiled from the user
      - If the user can compile it the answer will contain HelloWorld.java
- **Remove `javac`** (*Risk Detection*)
  - Which files will be affected if a user removes javac ?
    - A ← Compile(X)
    - Remove javac (i.e. in Prolog we could use *retract*)
    - B ← Compile(X)
    - The affected modules are in the set A\B
- ***How to* compile the file `HelloWorld.java`** (*Computation of Gaps*)
  - Find the possible modules that are needed to compile the file
  - *Abductive reasoning* could aid this task
    - For the given set of rules abductive reasoning will result all the JavaCompiler facts

FORTH
Institute of Computer Science

Rule-based Approach: Implementation Approaches

	Approach		
	Prolog	**DBMS**	**SemanticWeb**
Module Predicates	-	Relational table	Class
Module Type Hierarchies	Rules	Relational table	subClassOf
Modules (instances)	Facts	Tuples	Class instances
Profile Predicates	-	Relational table	Class
Profiles (Instances)	Facts	Tuples	Class Instances
Task Predicates	Rules	IDB predicates	Predicates appearing in rules
Task Type Hierarchies	Rules	Datalog rules, or IsA if ORDBMS	subClassOf
Performability	Rules	Datalog queries (recursive SQL)	rules

FORTH
Institute of Computer Science

Related Work (on dependency management)

- In comparison to the CASPAR project approach (i.e. the graph-based model), with rules we can handle disjunction and define the properties of dependencies (e.g. transitive or not)
- There are many dependency-management systems but only a few focus on task-based dependencies
  - [BelguidoumDagnat'2007]: Proposes a static deployment system for ensuring the success of two tasks: *installation* and *deinstallation* of a software component
  - [FranchMaiden'2003]: Models a system architecture in terms of dependencies between components (it defines four types of dependencies: *goal*, *soft goal*, *task* and *resource*).
    - However these approaches are less flexible and extensible (in terms of task and dependency modeling) than our approach.

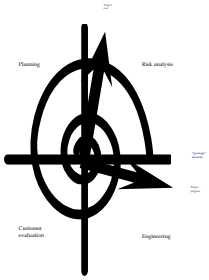
More on the SETN'2010 paper

FORTH
Institute of Computer Science

Some Notes on the Rule-based approach

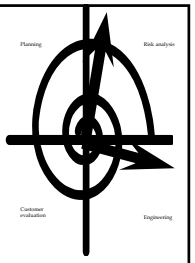
- More expressive and flexible, however *intelligibility gaps* are not unique (due to disjunction) implying that:
  - algorithmically they are less straightforward to derive
  - several possible intelligible packages for the same object
- Appropriate for the "repository approach" (recall the "where" aspect of dependency management which concerns the place where deps/metadata are stored).
- The maintenance of rules is more complex than in the graph-based model.

FORTH
Institute of Computer Science



METHODOLOGY
(FOR CAPTURING AND EXPLOITING DEPENDENCIES)

FORTH
Institute of Computer Science



Methodology for *modeling*, *capturing* and *managing* dependencies

Step 1) Identify desired **tasks** and objectives

Step 2) Model the identified **tasks** and their **dependency types**. If tasks can be hierarchically organized, then this should be done.

Step 3) Specialize the **Core Ontology of Dependencies** (or the **rule-based modeling)** according to the results of the previous step.

Step 4) **Capture the dependencies** of the digital objects of the archive. This can be done manually, automatically or semi-automatically. Tools like *PreScan* (described later on) can aid this task.

Step 5) Customize, use and **exploit the dependency services** according to the needs. For instance the intelligibility-related services can be articulated with *monitoring* and *notification* services.

Step 6) **Evaluate** the services in **real tasks** and **curate** accordingly the repository (return to Step 1)

FORTH
Institute of Computer Science

examples and discussion

- **Step 1** strongly depends on the nature of the digital objects and the tasks that we want to perform on them. For instance suppose we have a set of java files, 2D and 3D images. Common goals for java files include the ability to *compile and edit, while for images they include* the ability to render them on screen. The latter could be specialized to two goals, say *2Drendering and 3Drendering.*
- At **Step 2**, we would define five dependency types, say, *compile, edit, render, render2D, render3D, and two* subtype relationships: *render2D ⊑ render and render3D ⊑ render.*
- ***Step 3*** *we would extend COD, i.e. we would* define *compile ⊑ depends, edit ⊑ depends, and render ⊑ depends.*
- *The last step (**Step 6**) concerns evaluation* and curation. For instance, suppose the model fails for one particular module, i.e. the consumer of the package is unable to understand the delivered module. Such situations indicate that the recording of dependencies is not complete. For example, suppose a user who cannot run a software component, although the computed gap is empty. This can happen if the component has an additional dependency which has not been recorded. A corrective action would be to add this dependency. Analogously, if a user cannot understand a particular research paper this is probably because the paper uses concepts or symbols the user cannot understand. These concepts and symbols are actually dependencies which should be recorded. Synopsizing, **empirical testing** is a useful guide for defining and enriching the dependency graph.

FORTH
Institute of Computer Science

How to apply/adopt at your organization/archive

Some hints

1] Understand the dependency management approach and its potential benefits

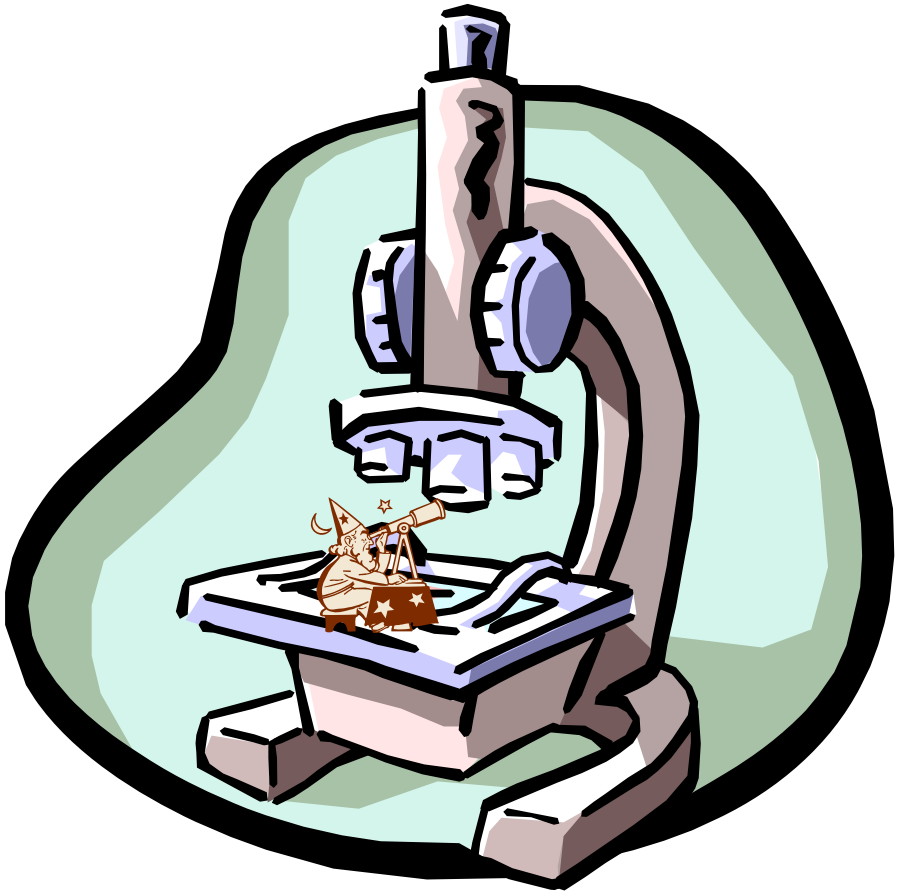
2] Consider the business processes which are already in place in your archive and identify the steps at which the dependency management approach can be employed and the expected benefits

3] If [2] is not applicable, consider establishing new processes that involve dependency management and motivate them clearly

(cont)

- Digital preservation is probably an endless-process (as physical preservation is)

FORTH
Institute of Computer Science



AN EXAMPLE WITH SCIENTIFIC DATA

Languages for
- Syntax description (EAST),
- Semantic description (DEDSL)
- Packaging (XFDU) and
- Semantic Web Languages (RDF/S)

```
25.130    35.325   30.2
25.100    35.161   28.9
25.180    35.333   29.3
```

FORTH
Institute of Computer Science

An example (ASCII file with measurements)

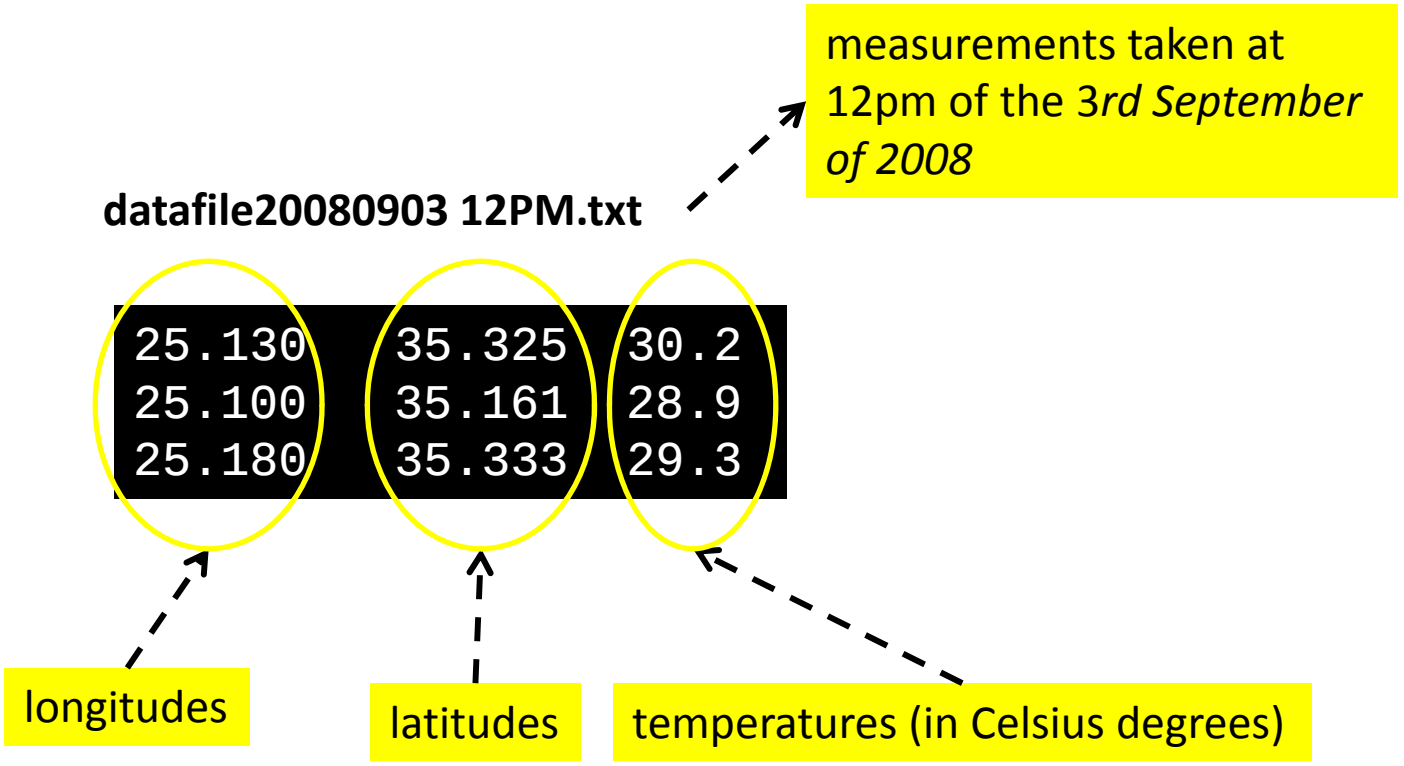
datafile20080903 12PM.txt

```
25.130   35.325  30.2
25.100   35.161  28.9
25.180   35.333  29.3
```

Suppose we want to preserve digital files in ASCII containing temperature measurements from various places on earth. Each file comprises an arbitrary number of lines where each line contains three numerical values *longitude*, *latitude* and measured *temperature* (in Celsius degrees).

Each line corresponds to the temperature at the coordinate-specified area as it was measured at a certain point in time. The time of measurement is hardwired in the name of the file, e.g. a file named **datafile20080903 12PM.txt** contains measurements taken at 12pm of the 3*rd September of 2008.*

(cont)



- We have to express these information explicitly

FORTH
Institute of Computer Science

(cont)

```
25.130    35.325   30.2
25.100    35.161   28.9
25.180    35.333   29.3
```

In the next slides we shall see in brief:

[A] An approach based on EAST, DEDSL and XFDU

syntax description → semantics description → packaging

[B] An approach based on "Semantic [(Web)]" languages

[C] A discussion of possible ways to apply the dependency management approach

FORTH
Institute of Computer Science

(cont)
[A] An approach based on EAST and DEDSL

Plan: syntax description → semantics description → packaging

Describing the **Syntax**

To preserve the **structure** of the file datafile20080903 12PM.txt we could make use of the **EAST (Enhanced Ada SubseT)** language.

Each *Data Description Record* (DDR), according to that language, consists of two packages,

- one for the ***logical*** description
  - it includes a logical description of all the described components, their size in bits as well as their location within the set of the described data.
- one for the ***physical*** description
  - it includes a representation of some basic types defined in the logical description and depend on the machine that generates these data: the organization of arrays (i.e. first-index-first, last-index-first) and the bit organization on the medium (high-order-first or low-order-first for big-endian or little endian representation respectively).

These two packages are mandatory even if the content of the physical part is empty.

FORTH
Institute of Computer Science

(cont)

```
package logical_datafileX_description is

type HORIZONTAL_COORDINATE is range -90.00 .. 90.00
for HORIZONTAL_COORDINATE'size 64; --bits

type VERTICAL_COORDINATE is range -180.00 .. 180.00
for VERTICAL_COORDINATE'size 64;

type TEMPERATURE_TYPE is range -100.0 .. 200.0
for TEMPERATURE_TYPE'size 16;

type MEASUREMENT_TUPLE is record
    LONGITUDE:VERTICAL_COORDINATE
    LATITUDE:HORIZONTAL_COORDINATE
    MEASURED_TEMPERATURE:TEMPERATURE_TYPE
end record;
for MEASUREMENT_TUPLE'size use 144;

type MEASUREMENT_BLOCK is array(1..1000) of MEASUREMENT_TUPLE;
for MEASUREMENT_BLOCK'size use 144000;

SOURCE_DATA:MEASUREMENT_BLOCK

end logical_datafileX_description;

package physical_datafileX_description is

end physical_datafileX_description;
```

Fig. 8 An example of an EAST description

Example of a DDR describing datafileX.txt.

We defined three different types one for each column of a data file (longitude, latitude , temperature), since each column represents a different kind of data.

The distinction of longitude and latitude is only made because of their different upper and lower bounds.

FORTH
Institute of Computer Science

(cont)

Plan: syntax description → semantics description → packaging

Describing the **Semantics**

We can define semantic descriptions for the entities (longitude, latitude and temperature) of the file datafileX.txt aiming at preserving the *meaning* (clarifying the interpretation) of the terms "longitude" "latitude" and "temperature".

One approach is to use the **DEDSL (Data Entity Dictionary Specification Language)** language.

We will see an example of a DEDSL description for datafileX.txt according to the **implementation of DEDSL using XML**.

Note: If we have another file with the same kind of information, we could reuse the same semantic descriptions. So semantic descriptions can be considered as reusable modules. These modules can contain abstract data descriptions to which concrete descriptions may refer.

FORTH
Institute of Computer Science

cont

```
<?xml version="1.0" encoding=="UTF-8"?>

<DATA_ENTITY_DICTIONARY>
    <DICTIONARY_IDENTIFICATION>
        <DICTIONARY_NAME CASE_SENSITIVITY="NOT_CASE_SENSITIVE">datafileX Dictionary
        </DICTIONARY_NAME>
    </DICTIONARY_IDENTIFICATION>

    <DATA_ENTITY_DEFINITION CLASS="DATA_FIELD" NAME="LONGITUDE">
        <DEFINITIONAL_PART>
            <DEFINITION>
                It represents the longitude for some certain coordinates. Longitudes east of
                use of plus (+) symbol while longitudes west of Greenwich shall be designated
            </DEFINITION>
            <SHORT_DEFINITION>Longitude</SHORT_DEFINITION>
            <UNITS>deg</UNITS>
        </DEFINITIONAL_PART>
        <REPRESENTATIONAL_PART DATA_TYPE="REAL">
            <RANGE MIN="-180.00" MAX="+180.00"/>
        </REPRESENTATIONAL_PART>
    </DATA_ENTITY_DEFINITION>
    <DATA_ENTITY_DEFINITION CLASS="DATA_FIELD" NAME="LATITUDE">
        <DEFINITIONAL_PART>
            <DEFINIITION>
                It represents the latitude for some certain coordinates.  Latitudes north of
```

```
    </DATA_ENTITY_DEFINITION>
    <DATA_ENTITY_DEFINITION CLASS="DATA_FIELD" NAME="LATITUDE">
        <DEFINITIONAL_PART>
            <DEFINIITION>
                It represents the latitude for some certain coordinates.  Latitude
                use of plus (+) symbol while latitudes south of the Equator will b
            </DEFINITION>
            <SHORT_DEFINITION>Latitude</SHORT_DEFINITION>
            <UNITS>deg</UNITS>
        </DEFINITIONAL_PART>
        <REPRESENTATIONAL_PART DATA_TYPE="REAL">
            <RANGE MIN="-90.00" MAX="+90.00"/>
        </REPRESENTATIONAL_PART>
    </DATA_ENTITY_DEFINITION>
    <DATA_ENTITY_DEFINITION CLASS="DATA_FIELD" NAME="TEMPERATURE">
        <DEFINITIONAL_PART>
            <DEFINITION>
                It represent the temperature in the area with
                the specific coordinates.
            </DEFINITION>
            <SHORT_DEFINITION>Temperature</SHORT_DEFINITION>
            <UNITS>Celsius degrees</UNITS>
        </DEFINITIONAL_PART>
        <REPRESENTATIONAL_PART DATA_TYPE="REAL">
            <RANGE MIN="-100.0" MAX="200.0"/>
        </REPRESENTATIONAL_PART>
    </DATA_ENTITY_DEFINITION>
</DATA_ENTITY_DICTIONARY>
```

Fig. 9 An example of a DEDSL description

FORTH
Institute of Computer Science

cont.

Plan: syntax description → semantics description → packaging

Packaging

- Now suppose that we want to preserve and archive a number of datafiles with such measurements. Packaging formats can be used for preparing a package that contains the data files **plus** their EAST and DEDSL descriptions.
- We can satisfy such packaging requirements using **XFDU (XML Formated Data Unit (XFDU)** which is a standard file format developed by CCSDS (Consultative Committee for Space Data Systems) for packaging and conveying scientific data, aiming at facilitating information transfer and archiving.

FORTH
Institute of Computer Science

(cont)

- The benefits of adopting a packaging approach (like that of XFDU) is that we can also add various information about the components of the package.
- For example suppose we would like to include information about the user that took the temperatures for each file, as well as the GPS (Global Positioning System) and the thermometer characteristics or the satellite information (if the samplings were made from space). We can easily add the above information using XFDU since we just have to add the necessary information in the package. For instance, we could describe such information using CIDOC CRM ontology. Such descriptions could be expressed in XML format (e.g. CIDOC CRM CORE) or as an RDF/XML file (the RDF-case will be described later on). In both cases the file could be included in the package.
- It is obvious that the major benefit from the use of XFDU is that we can package together heterogenous modules (java programs, datafiles, GPS info, provenance data) and deliver them to the user, or archive them, as a single (ideally selfdescribing) unit.

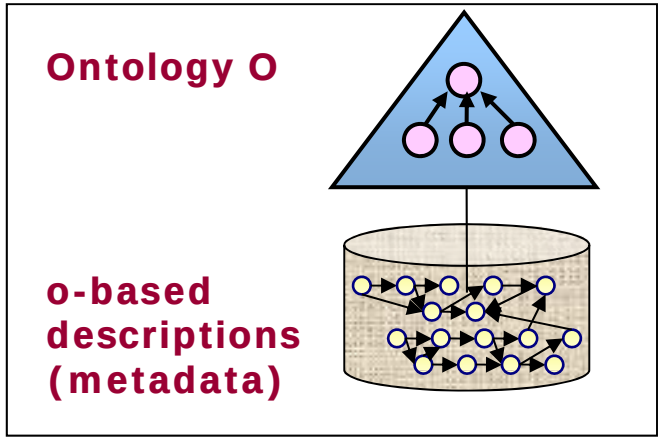
FORTH
Institute of Computer Science

(cont)

[B] An approach based on "Semantic [(Web)]" languages

Here we describe an alternative approach that relies on Semantic Web (SW) languages.

An ontology can be used to describe (syntactically and semantically) the form of the data and ontology-based descriptions can instantiate this ontology. These instantiations are actually the data themselves.

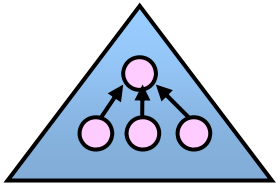


FORTH
Institute of Computer Science

cont.

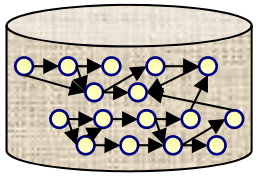
```
<?xml version='1.0'?>
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3c.org/2000/01/rdf-schema#">

    <rdfs:Class rdf:ID="Sample"/>
    <rdf:Property rdf:ID="Longitude">
        <rdfs:domain rdf:resource="#Sample"/>
        <rdfs:range rdf:resources="&rdfs;Literal"/>
    </rdf/Property>
    <rdf:Property rdf:ID="Latitude">
        <rdfs:domain rdf:resource="#Sample"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </rdf:Property>
    <rdf:Property rdf:ID="Temperature">
        <rdfs:domain rdf:resource="#Sample"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </rdfs:Property>
</rdf:RDF>
```



indicative ontology for our running example expressed in RDF/S XML.

```
<temperature:Sample
      rdf:about="sampingId20080903_12PM_35.233_25.343>
    <temperature:Longitude= "25.130"/>
    <temperature:Latitude= "35.325"/>
    <temperature:Temperature= "30.2"/>
</temperature:Sample>
```



The contents of the data files expressed wrt the ontology (in RDF/S XML)

FORTH
Institute of Computer Science

(cont)

Some **benefits** of the SW languages (like RDF/S)

- **The ontology is enough.** When creating a new data file there is no need to create its DEDSL or EAST description every time. Instead we have to represent the data using the syntax specified by the ontology.
- **Evolution**. Whenever we want to preserve more data types we can extend that ontology.
  - For example, past data files can contain only the longitude, the latitude and the temperature, while current ones may contain also the name of each location, or the thermometer used for the measurement. In such cases two different kinds of DEDSL/EAST descriptions have to be created and used. In the SW approach, we just have to extend the top ontology.

FORTH
Institute of Computer Science

Tightly Coupled Descriptions. In SW languages data and their descriptions are tightly coupled. In contrast, EAST/DEDSL-descriptions are represented as separate files and for this reason packaging formats are important. On the other hand in RDF a data file would itself define the data type of each data element in the file. To clarify this point consider the following line from our running example: 25.130 35.325 30.2. This line does not provide any information regarding what 25.130 might be, it could be either the longitude, the latitude or the temperature. On the other hand its RDF representation would be:

```
<temperature:Sample
rdf:about="sampingId20080903_12PM_35.233_25.343>
<temperature:Longitude= "25.130"/>
<temperature:Latitude= "35.325"/>
<temperature:Temperature= "30.2"/>
</temperature:Sample>
```

This part of RDF can be understood without the existence of any other (EAST/DEDSL) file.

FORTH
Institute of Computer Science

(cont)

- In brief, the benefits of adopting SW technologies is that they provide standard and semantically interpretable exchange formats, and that the knowledge artifacts expressed in these languages lend themselves to reuse and extension.

- Regarding storage, the SW approach could be based on
  - a set of RDF/XML files
  - a single RDF-repository (triple-store) where all ontologies and ontology-based description form a coherent semantic network

FORTH
Institute of Computer Science

[C] Discussion of possible ways to apply the dependency management approach

Main point

- It can capture both [A] and [B] approaches, as well as "mixed" approaches

FORTH
Institute of Computer Science

[C] Discussion of possible ways to apply the dependency management approach **(cont)**

- Capturing the [A] approach:
  - The dependency mgmt can determine what to put in a package (not just EAST and DEDSL, or it many not require to include both)
  - For instance if the DC knows the semantics (of temperature, longitude etc) then the DEDSL description is not necessary to be included.
  - If the real world task requires another software that reads these files and creates maps, then the package will contain information for that software too.
    - This will be the result of the Step 1 of the methodology

FORTH
Institute of Computer Science

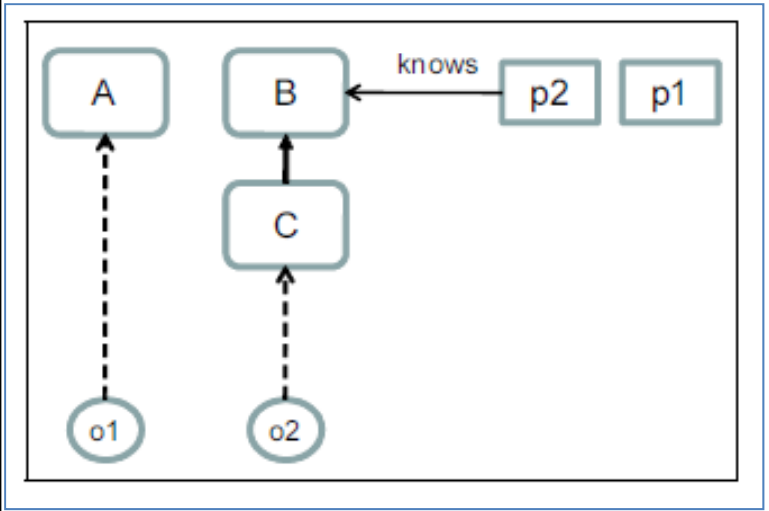
(cont)

- Capturing the [B] approach:
  - The dependency mgmt can determine what SW artifacts are required to be placed in a package to form a “complete” KB (with no dangling references).
  - For instance if we want to deliver a particular file with measurements, say A.rdf, for a community that knows RDF/S, then the package will contain also the ontology. If the community does not know RDF/S then also its description will be included. If another community knows the ontology, only A.rdf will be delivered.
  - An additional example (with dependency-related ontologies) follows:

FORTH
Institute of Computer Science

(cont)

Example (i): Dependency-related knowledge artifacts



Consider two "chunks of structured information" (e.g. two RDF/S files) *o1* and *o2 where o1 is expressed with respect* to an ontology *A, while o2 are expressed* with respect to an ontology *C.*

Now consider a particular community *p1 that is not familiar with any of* these ontologies, and a community *p2 that is familiar with ontology B and suppose* that *C is a specialization of B (i.e. it reuses and extends* elements of *B), as shown in the figure.*

We can define the *gap between an object o and* a community profile *p, denoted by gap(o, p), as the set* of ontologies that a member of the *p community needs* to understand in order to understand the contents of *o. In our case:*

dgap(o1, p1) = A

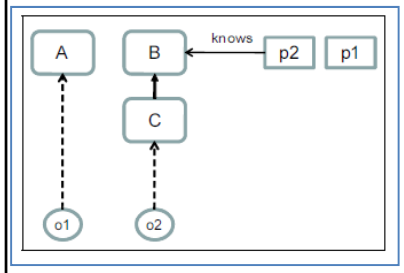
dgap(o2, p1) = C ∪ B

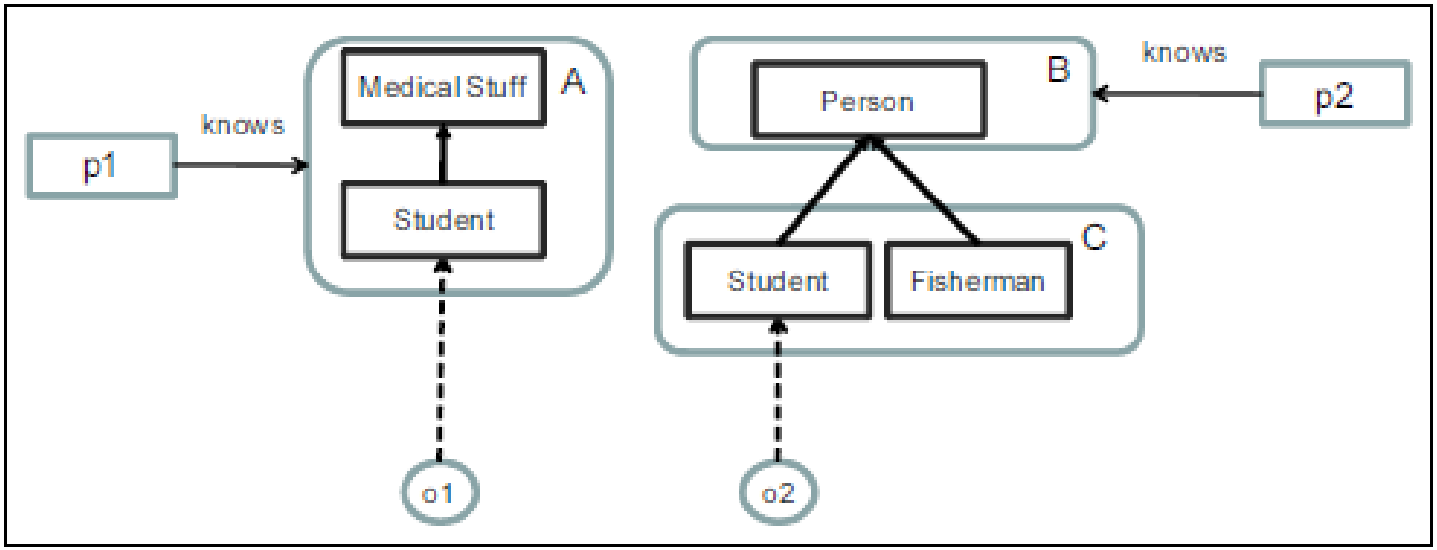
dgap(o1, p2) = A

dgap(o2, p2) = C

(cont)
Example (ii): Dependency-related fine-grained knowledge artifacts

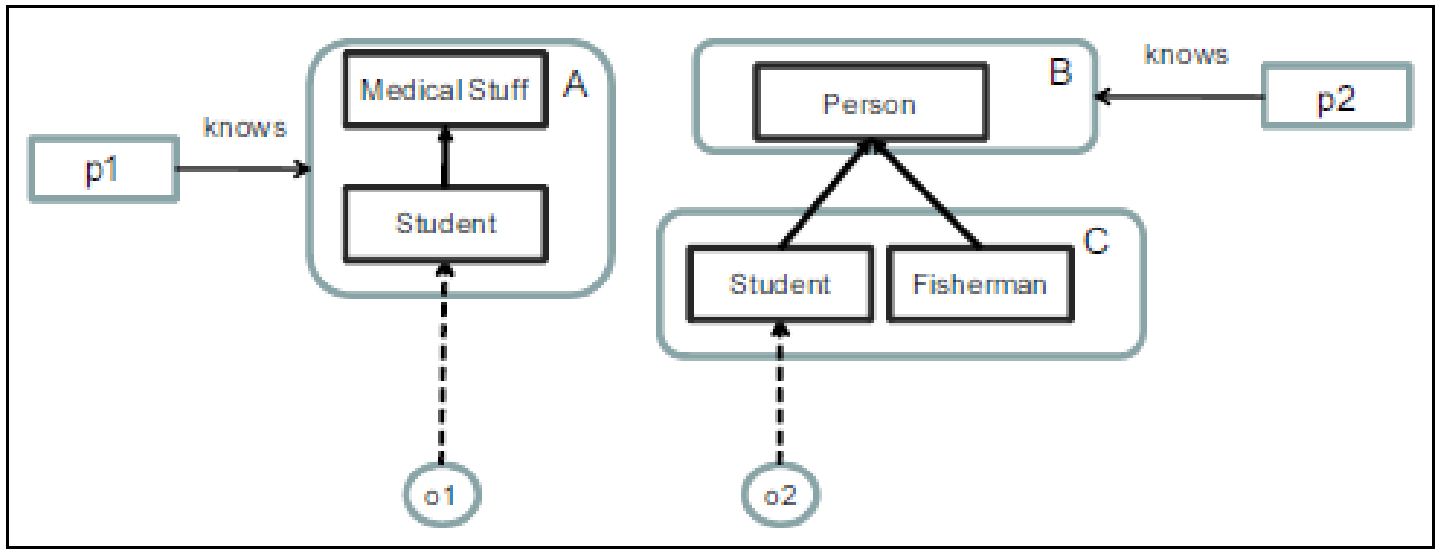
Consider the case where:





- This example demonstrates more fine grained gaps. The key observation is that *instanceOf and isA relations are special kinds of dependencies, actually* they carry more meaning than a plain dependency relation. This means that an *isA hierarchy could be construed* as a dependency graph in our framework (where each subclass depends on its superclasses).

(cont).



- This means that we have the dependencies
  - *o2 < instanceOf Student* and *Student < isA Person.*
- *Under this* perspective,
  - *Gap(o2, p2) = {Student}.*
- *It follows that* according to this view, profiles, as well as intelligibility gaps, can contain all kinds of RDF elements.

FORTH
Institute of Computer Science

(cont)

Another variation

- So far we have considered intelligibility gaps that comprise sets of modules. The dependency types that participate to the computation of gap could also be returned as they convey extra meaning which could be exploited and recorded (e.g. in the manifest file of XFDU).
- For instance, we can define gaps as sets of paths where a path is a sequence of (*depT ype, module) pairs, or RDF* triples of the form (*subject, predicate, object), indicating* the specializations of the ontology that are required.
- In our example, where *Gap(o2, p2) = {Student}, a more informative gap would* be
  - *IGap(o2, p2) = {instOf Student subclassOf Person}.*
- In a triple form we could write:
  - *IGap(o2, p2) = {{o2 instOf Student}, {Student subclassOf Person}}*

FORTH
Institute of Computer Science

[C] Discussion of possible ways to apply the dependency management approach **(cont)**

- Capturing “mixed” approaches:
  - We can define an approach (through appropriate module types and dependency types) that mixes both approaches .

FORTH
Institute of Computer Science

End of Example> Some Remarks

- Although the original file contained plain ASCII text, we have seen that we have to keep more information in order to preserve the meaning of the contents and exemplify various assumptions (e.g. filenames assumptions, degrees in Celcius)
  - We cannot automate everything
- We have seen that there are various approaches and languages for describing syntax and semantics
- The dependency management approach is more general and can capture previous approaches.
- It can be used for managing more systematically the dependencies that other approaches tacitly assume.
- However, it requires deciding
  - what a module is (desired granularity)
  - what the desired/interesting tasks and their dependency types are
  - what to assume that the DC knows (necessary for avoiding to model everything)

FORTH
Institute of Computer Science

BACK TO THE DIGITAL PRESERVATION CONTEXT

FORTH
Institute of Computer Science

Emulation/ Migration/Virtualization and Dependency Management

- Migration. We can say that it changes the dependencies (a digital object which depends on an old format, now it depends on a newer format).

- Emulation: Probably it does not change the dependencies. An emulator just makes available an old module (actually by emulating its behavior).

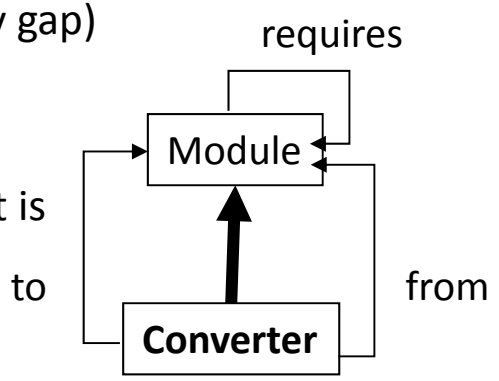
- Virtualization. Virtualization approaches, like the UVC (Universal Virtual Computer) approach, also require *Representation Information* (the typical dependency type of OAIS), as the specification of a UVC has to be recorded and it may change over time. This means that the adoption of virtualization approaches just reduces the number of dependencies that we may have (however it does not vanish them).
  - In the same spirit, MDA (Model Driven Architecture) and OMG standards aim at reducing the dependencies: instead of having a specification of an information system that depends on a number of different technologies, MDA proposes a specification that depends only on OMG standards.

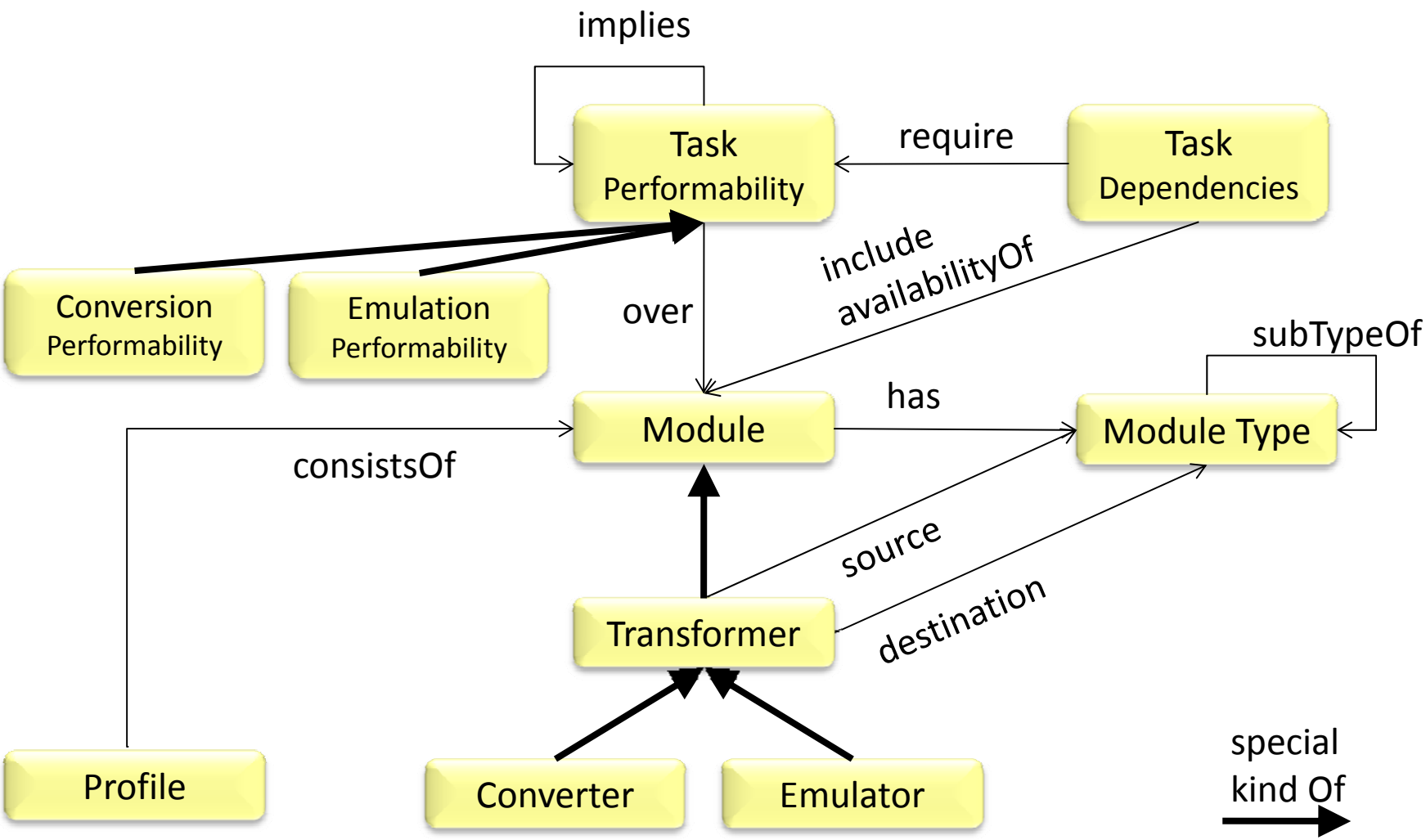
FORTH
Institute of Computer Science

(cont)

Dependency Management: One related direction for future research:

- General question:
  - *How to enrich the dependency management model with **Converters** (for capturing migration) and **Emulators** (how to model them)?*
- Requirements
  - Ability to model explicitly converters and emulators (and analyze them from a dependency point of view)
  - Exploit them while computing intelligibility gaps (for example, a sequence of conversion may be possible to vanish an intelligibility gap)
- Status
  - Work in progress.
  - The next slide shows the basic notions of a rule-based model (that is under construction) that supports converters and emulators.



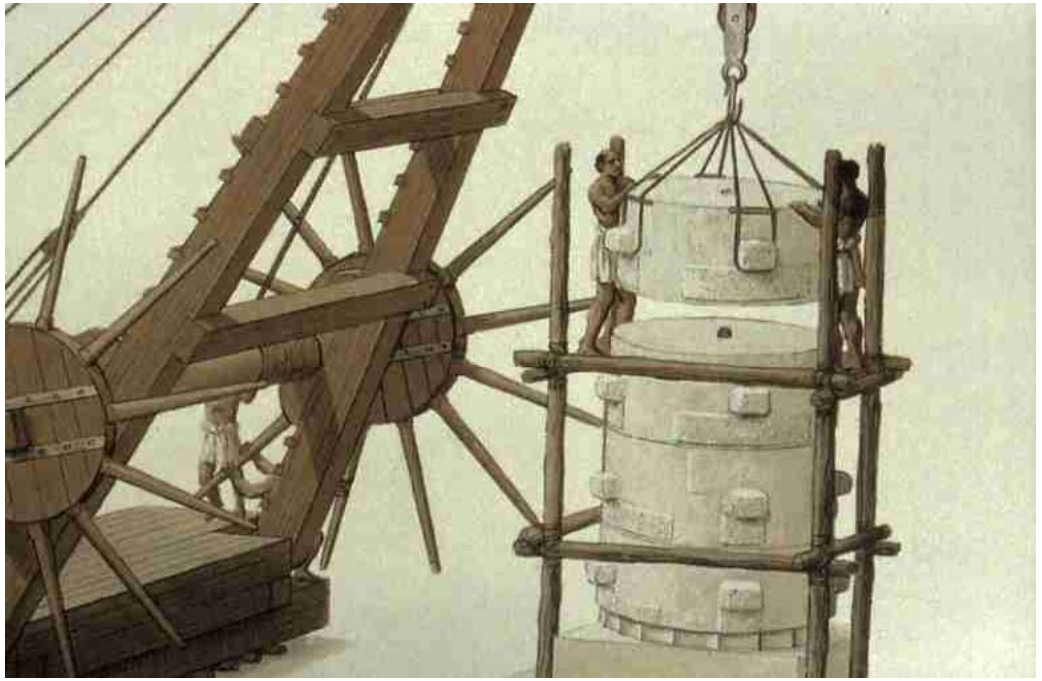

implies
Task Performability
require
Task Dependencies
Conversion Performability
Emulation Performability
over
include availabilityOf
subTypeOf
has
Module
Module Type
consistsOf
source
destination
Transformer
Profile
Converter
Emulator
special kind Of

FORTH
Institute of Computer Science

Other Remarks

- Provenance Modeling
  - We do not focus on modeling or logging the particular workflows or derivation chains of the digital artifacts, e.g. using conceptual models like OPM (Open Provenance Model) or CRM Dig (extension of CIDOC CRM, see J. DAPD 2010 paper).
  - We focus only the dependencies amongst artifacts for carrying out the desired tasks.
    - Obviously, this view is less space consuming than logging or describing processes or process models. For instance, for software preservation, we do not have to record the particular compiler that was used for the derivation of an executable (and its compilation time), we just care to have at our disposal an appropriate compiler for future use.
  - We do not model the process, hence we do not capture dependencies of the form: `digitalObjectX` *`dependsOn`* `ProcessY`
    - However, if a detailed process (or provenance) model is available, then the dependency model can be considered as a read-only "subview" of that model.

FORTH
Institute of Computer Science



RELATED TOOLS/APPLICATIONS

GapManager (modeling, use cases, indicative screens and demos, scalability, datasets)

PreScan

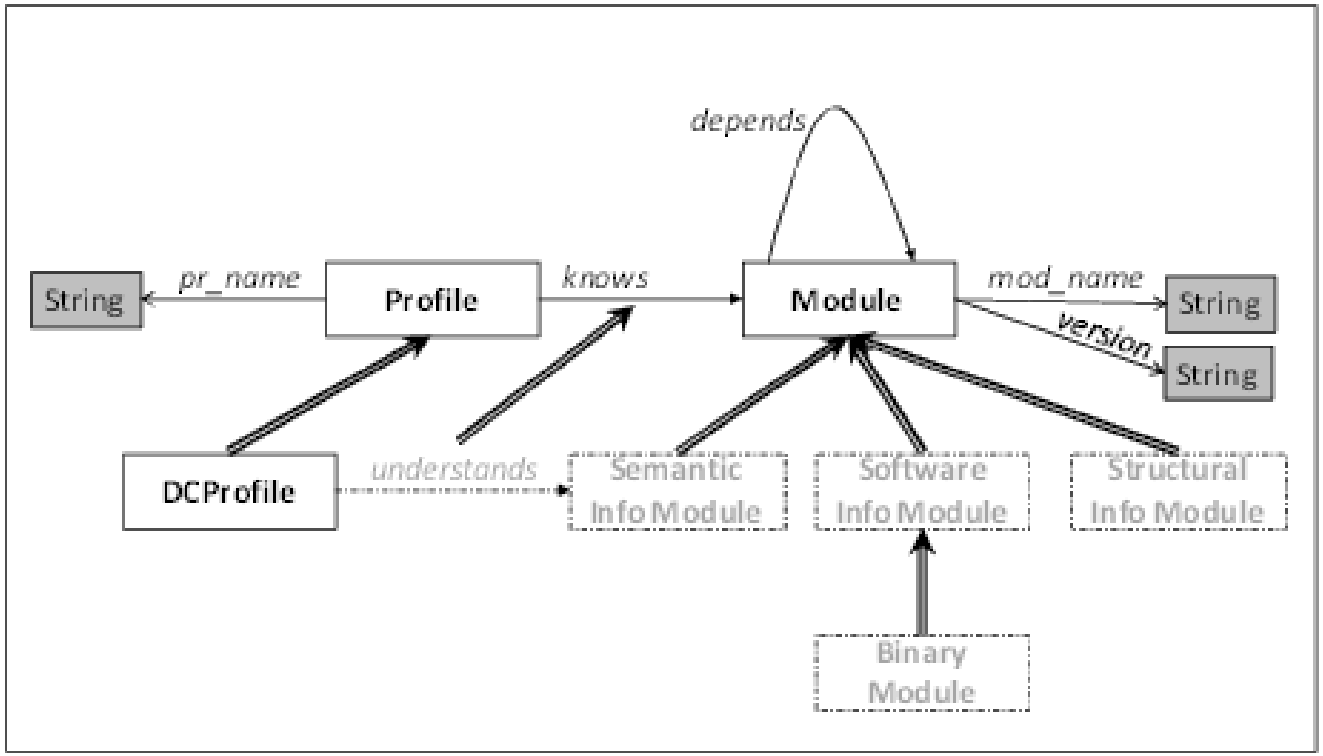
FORTH
Institute of Computer Science

GapManager

- **GapManager** in one of the key components of the CASPAR EU Project
- Can aid several tasks related to the preservation of intelligibility of digital objects. It actually provides the functionality of the Graph-based dependency management model and it is implemented using Semantic Web technologies.
- Software
  - As an API (in Java)
  - As a full Web-based application
- Outline of next slides
  - Modeling, GUI of Web Application, Scalability, Datasets, Links to extra material

FORTH
Institute of Computer Science

GapManager: Core ontology for dependencies



Also in RDF/S

FORTH
Institute of Computer Science

Gap Manager: Typology of Modules

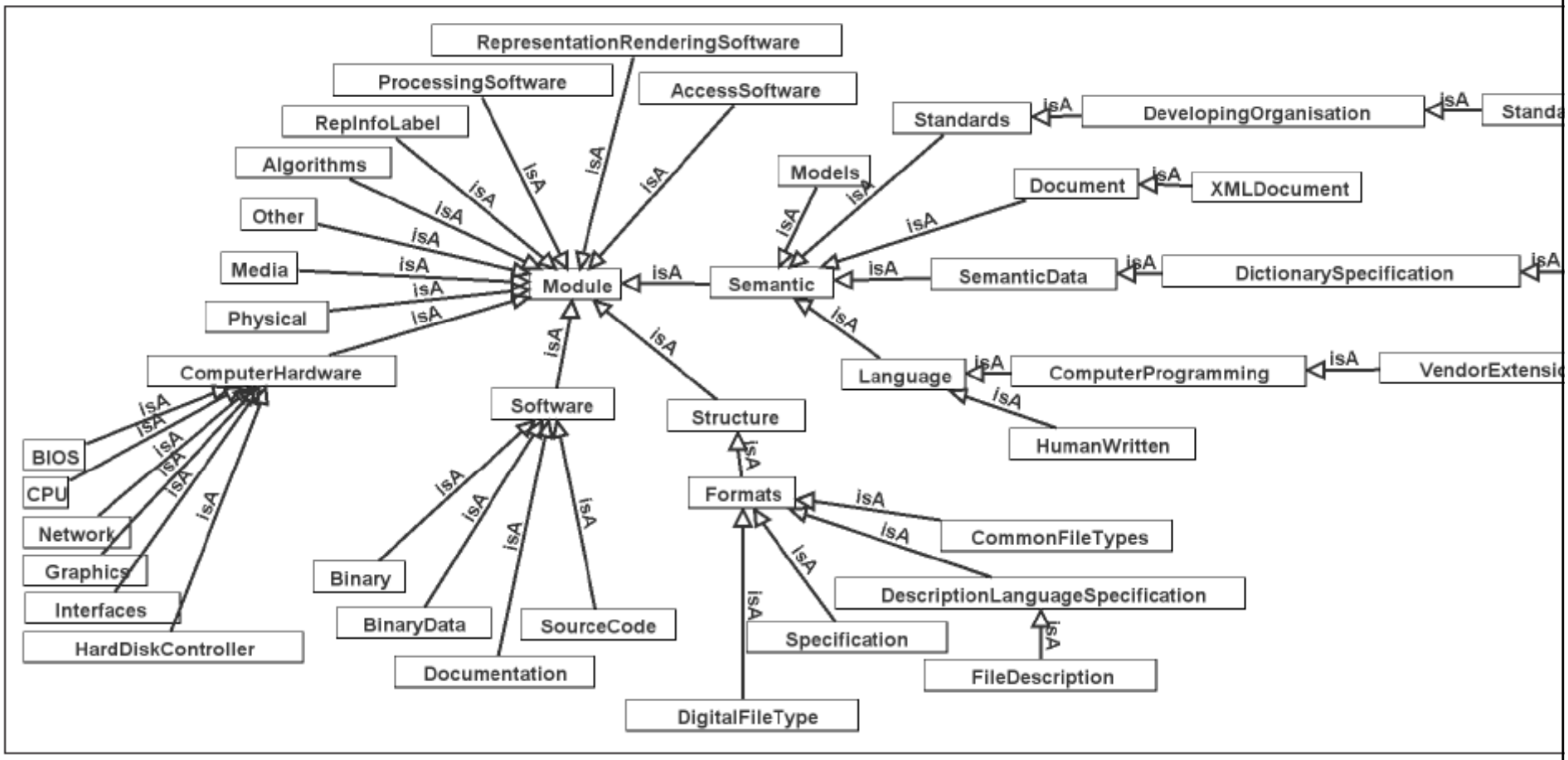
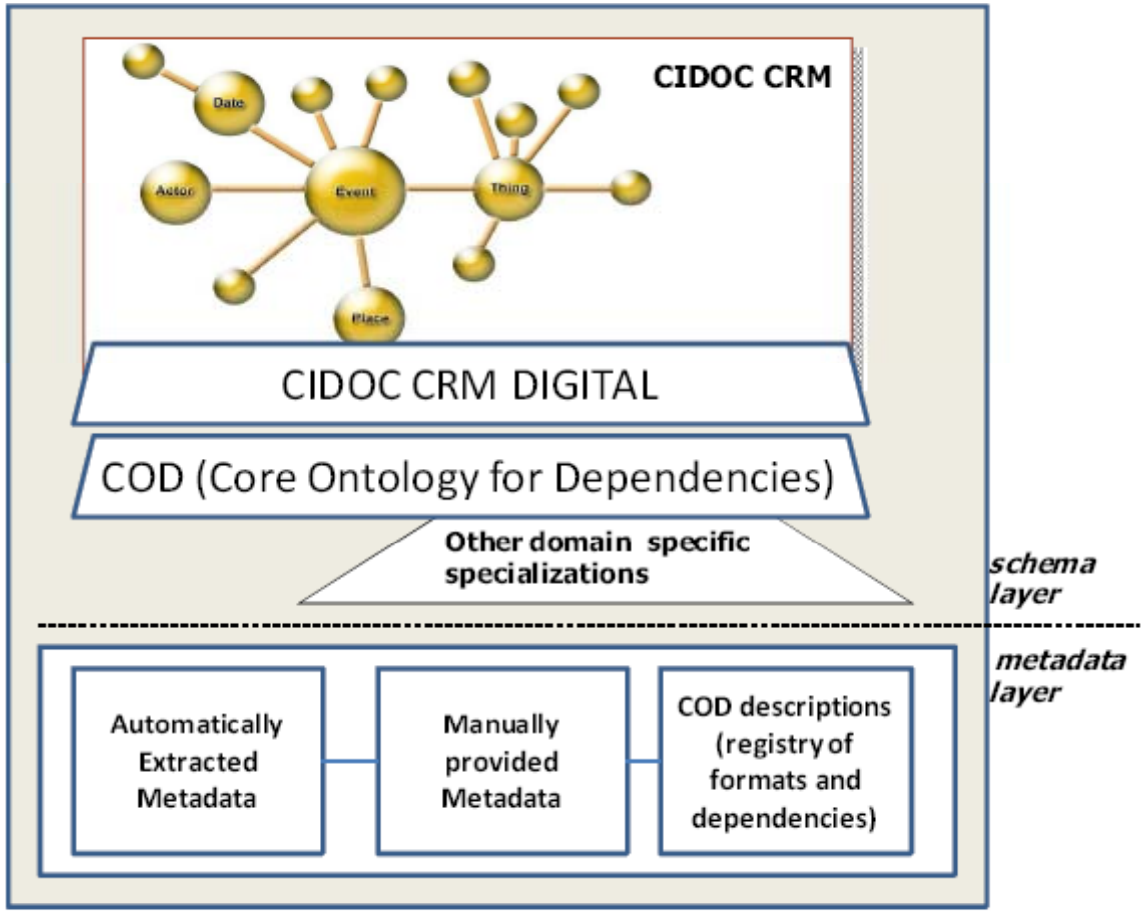


Fig. 4 Typology of Modules

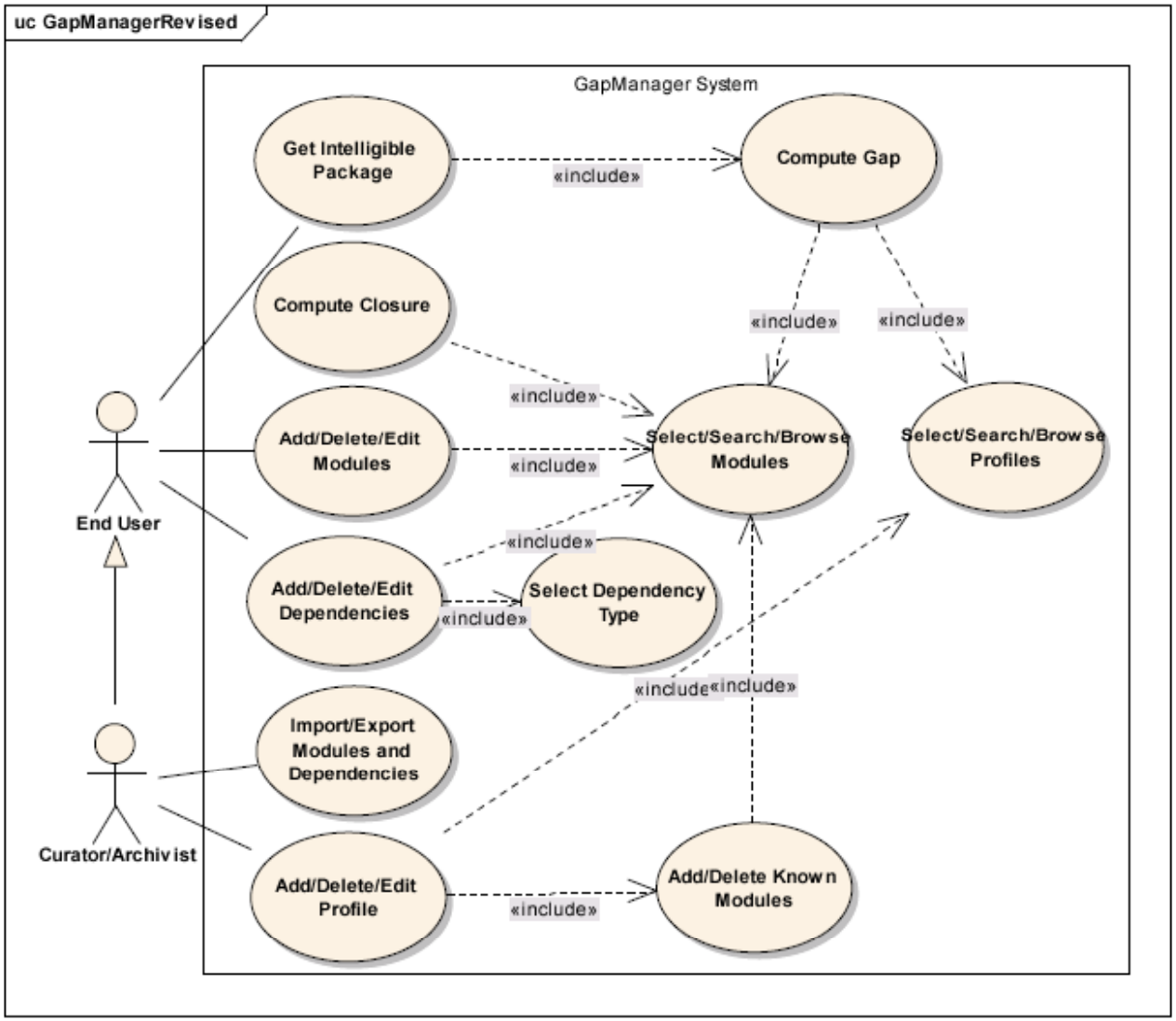
FORTH
Institute of Computer Science

GapManager: Ontology Architecture



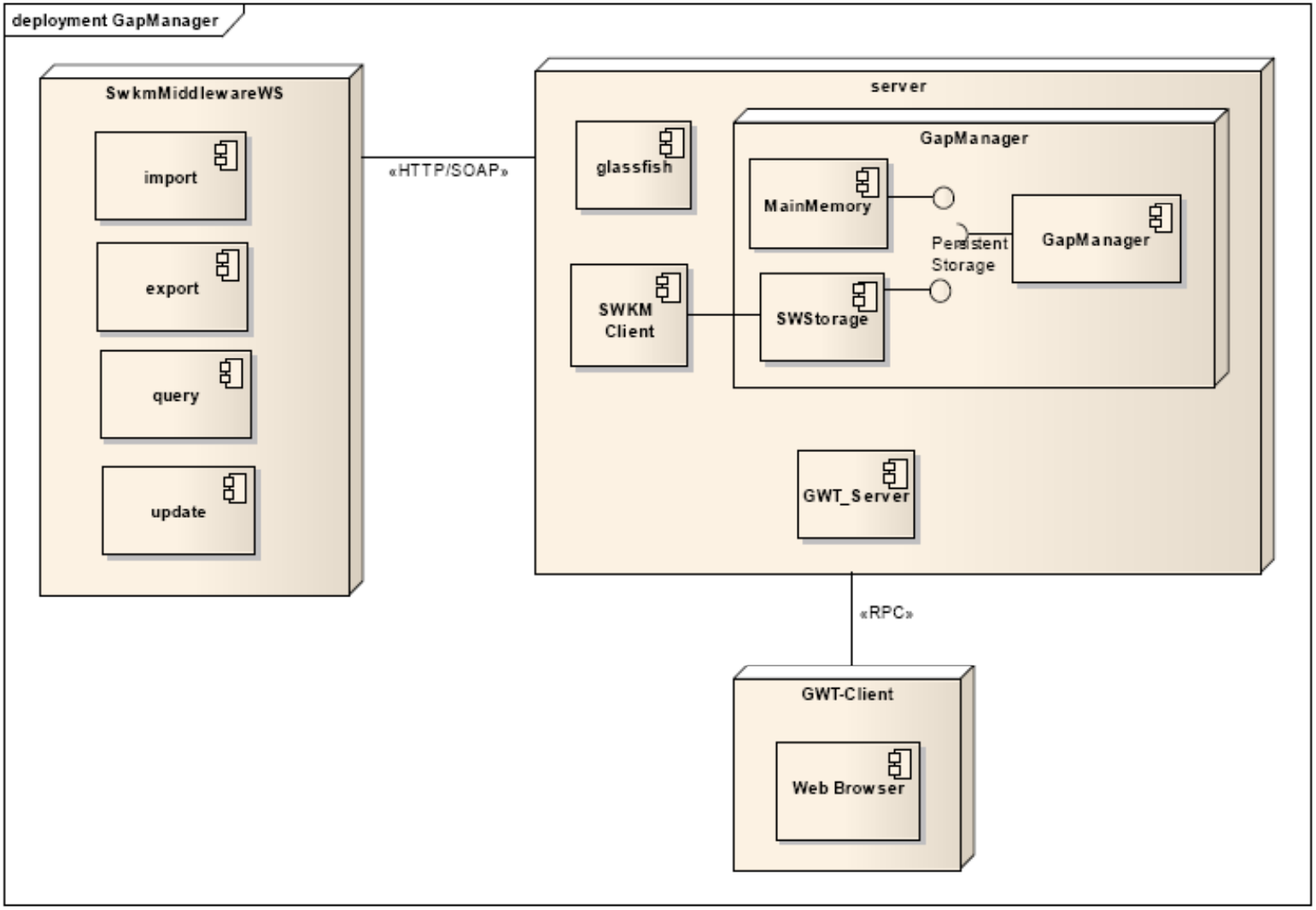
FORTH
Institute of Computer Science

GapManager: Use Case Diagram



FORTH
Institute of Computer Science

GapManager: Component & Deployment Diagram



FORTH
Institute of Computer Science

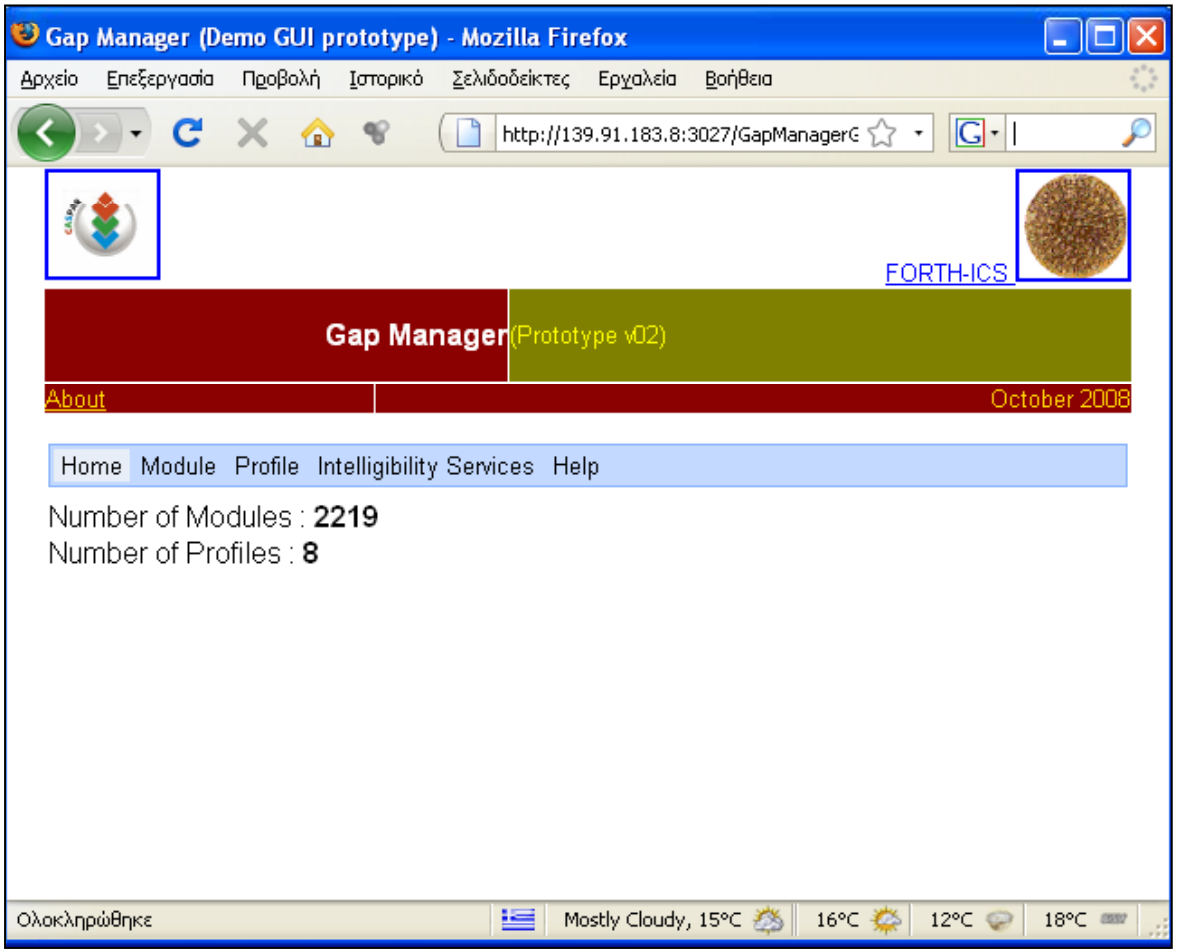
GapManager: Java API, WS, versions

- GapManager Web Page
  - http://athena.ics.forth.gr:9090/Applications/GapManager/
- GapManager API
  - http://139.91.183.8:3026/hudson/job/Caspar%20Gap%20Manager/javadoc/
- GapManager Web Application
  - There are two web applications of GapManager:
    - One using a main memory for storing everyting
    - One using SWKM (Semantic Web Knowledge Middleware)
  - http://athena.ics.forth.gr:9090/Applications/GapManager/Pages/onlineDeployments.html

GapManager/Web Application Demo

Main Functionality

- Add/Delete/Edit
  - Modules
  - Dependencies
  - Profiles
- Gap Computation
- Closure Computation
- Related Profiles
- Import data
  - RDF Format
  - Proprietary (txt)
- Export data
  - RDF Format
  - Proprietary (txt)

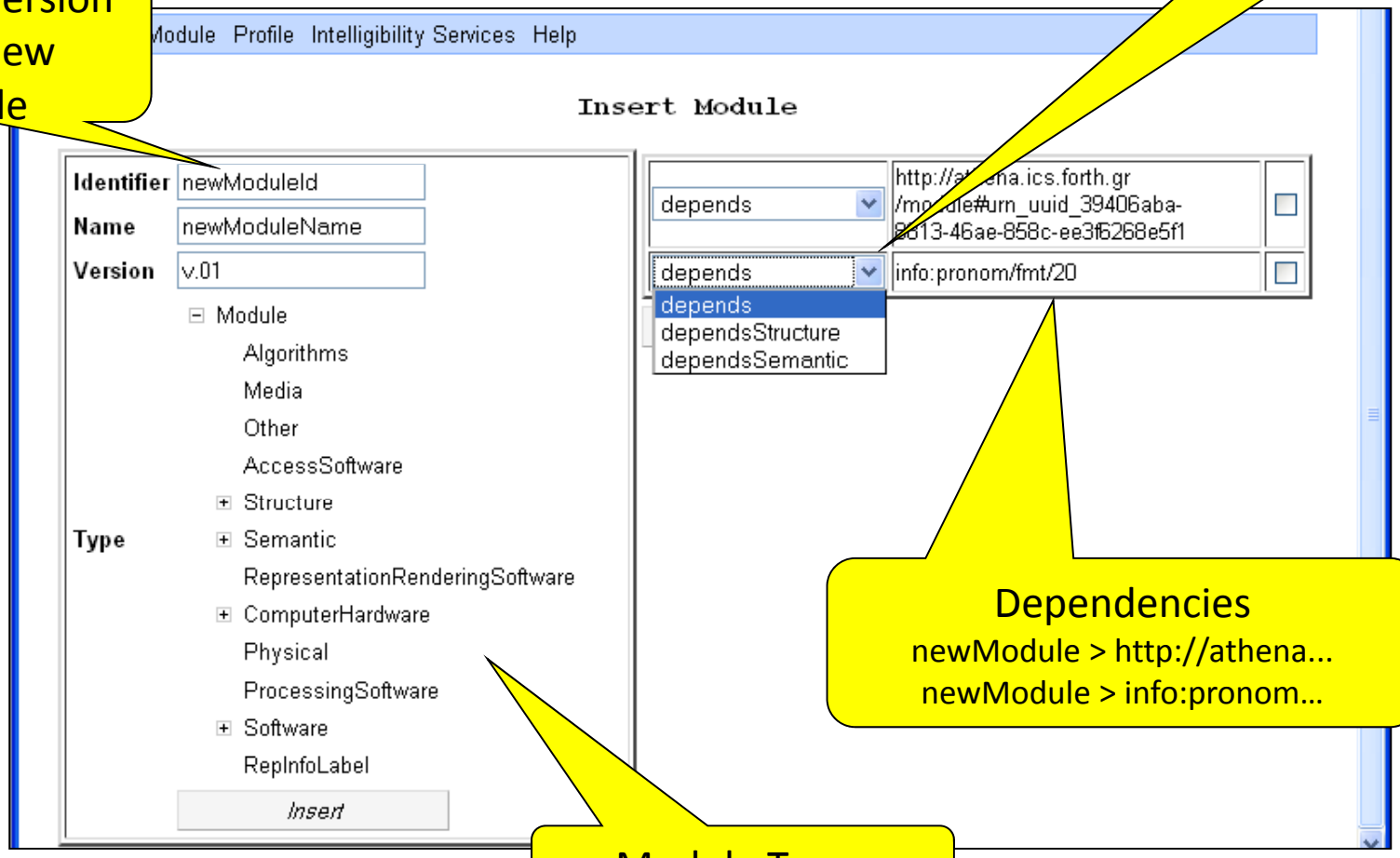


FORTH
Institute of Computer Science

GapManager
Demo - Add new Module

Id, name, version of the new Module

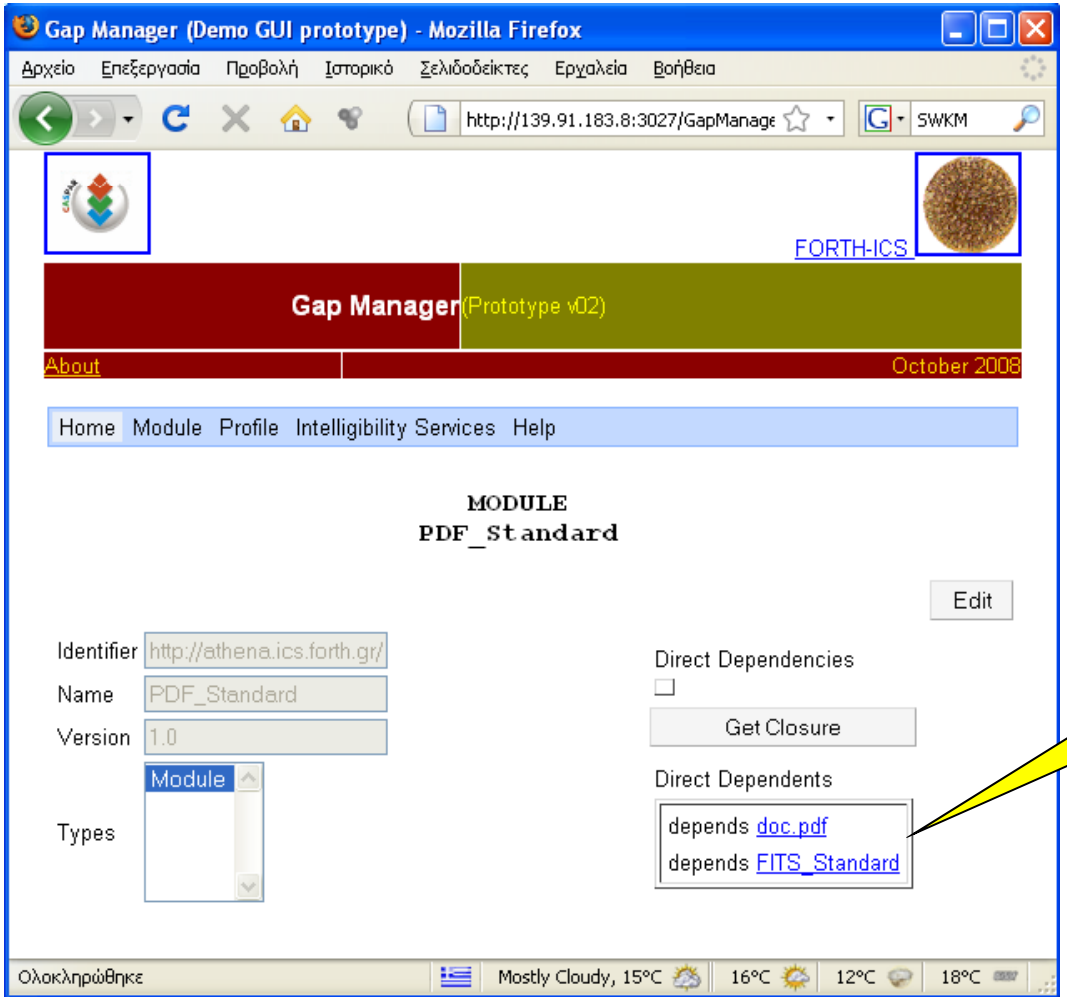
Dependency Types



Dependencies
newModule > http://athena...
newModule > info:pronom...

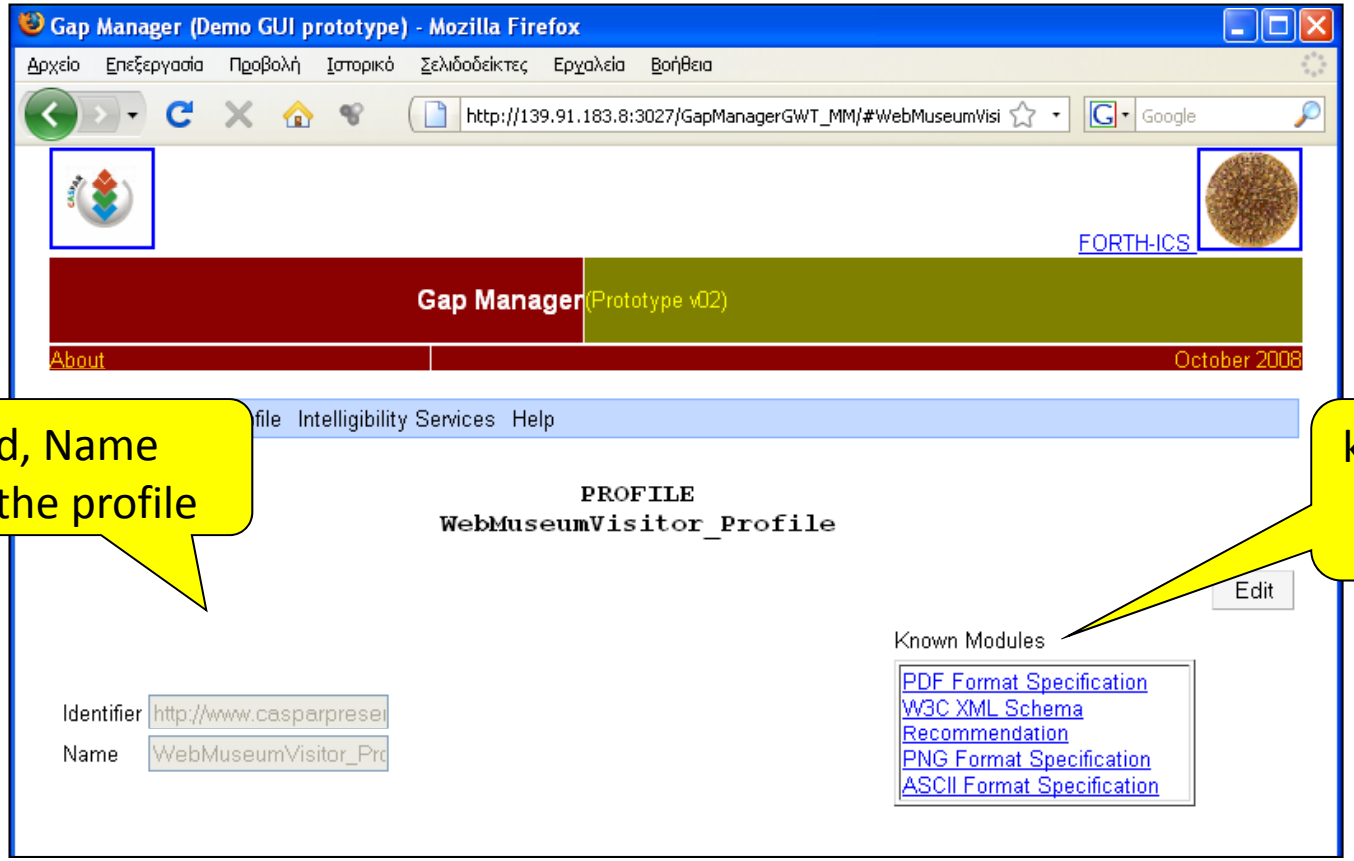
Module Types

GapManager
Demo – Module view

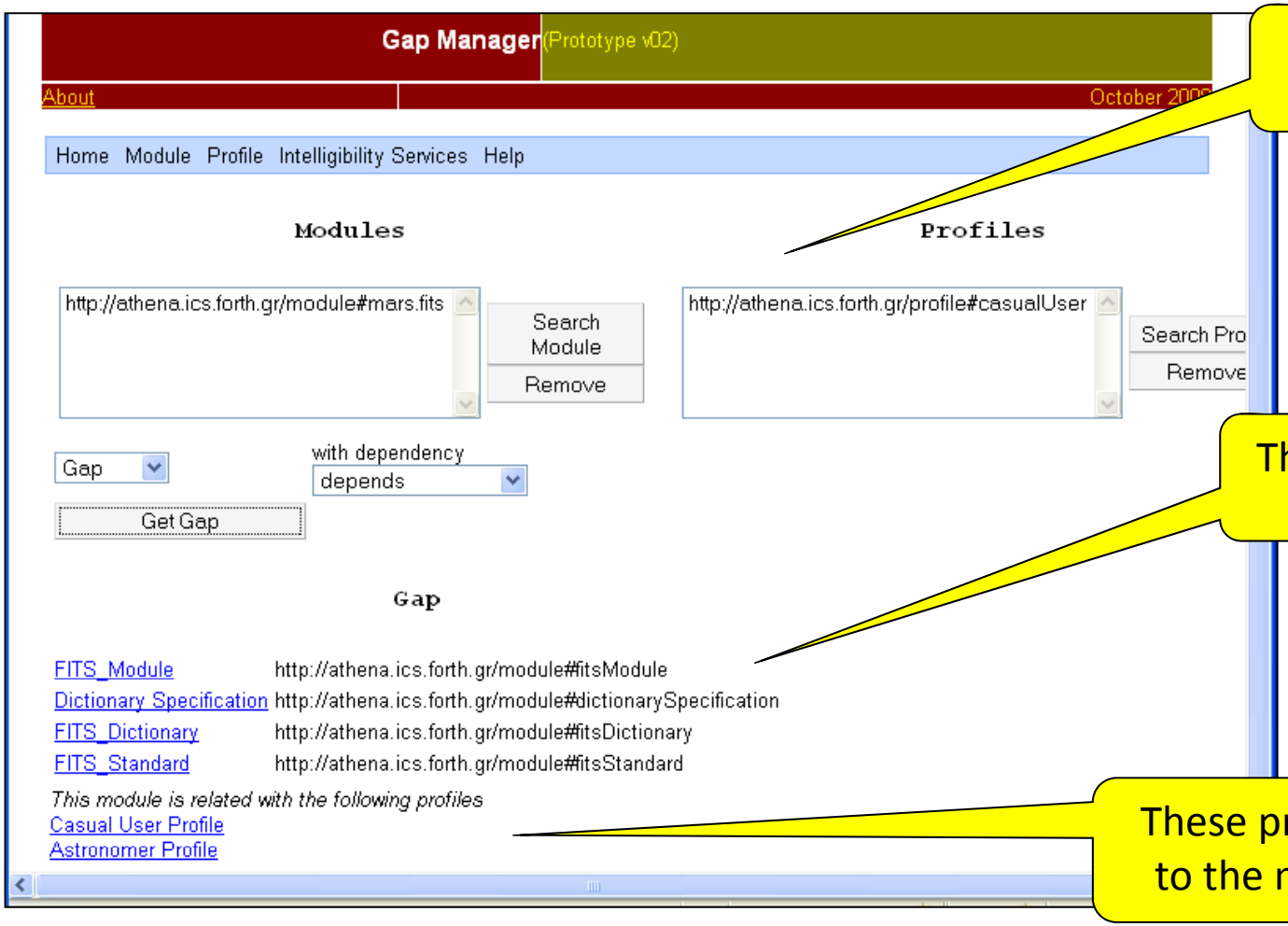


These modules are in danger if PDF_Standard becomes obsolete

GapManager
Demo – Profile view



GapManager
Demo – Gap view



Selected profile (s)

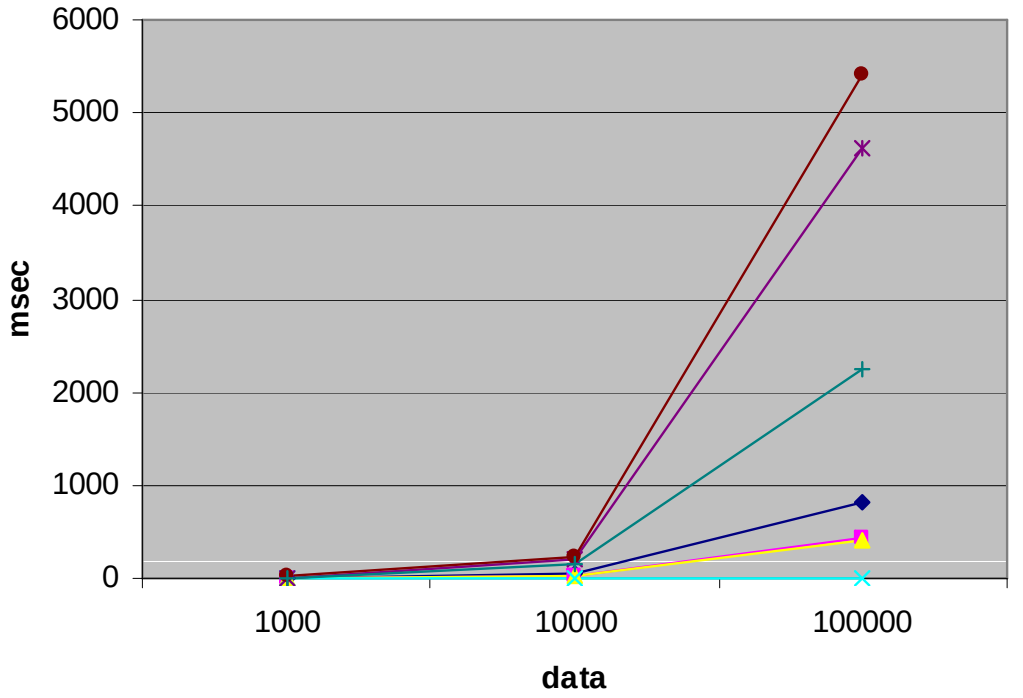
This is the Gap

These profiles are related to the module **Mars.fits**

FORTH
Institute of Computer Science

GapManager

Evaluation of Scalability



- The evaluation was made using Main Memory Implementation (everything in memory not in a SWKM which is a QL-accessed RDF triple store)
- An implementation over SWKM costs much more (mainly due to the cost of parsing the results of the QL answers)

Dependency Graph Size

N	10^3	10^4	10^5
Avg Depth	9	10	10
Avg Closure	261	2174	17227
Max Depth	23	24	25
Max Closure	675	5930	50838

FORTH
Institute of Computer Science

Gap Manager and Datasets used (during CASPAR)

The instantiations of the **Core Ontology** of Gap Manager contains more than 2.200 instances

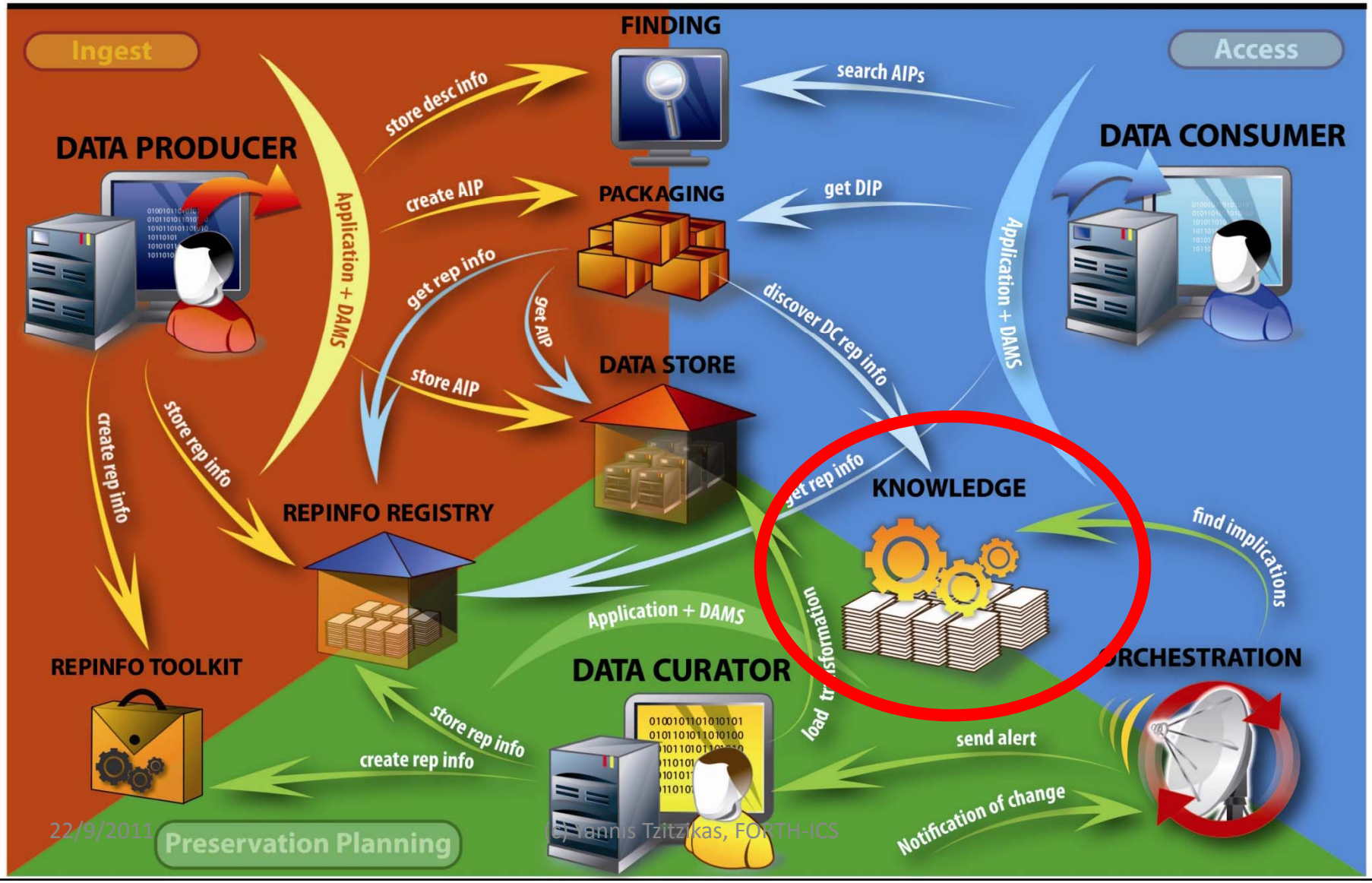
- exported from the PRONOM registry (of file format)
- from the CASPAR Registry (by SFTC)
  - more than 1.300 (last export: Oct 14, 2008)

- Therefore it can be considered as an extensible semantic registry of data formats

Several deployments of the system in the context of the CASPAR project

FORTH
Institute of Computer Science

One of the CASPAR suite of components



FORTH
Institute of Computer Science

Gap Manager/ Extra Material

- Gap Manager
  - URL: http://athena.ics.forth.gr:9090/Applications/GapManager/
  - Video: http://wiki.casparpreserves.eu/pub/Main/2103GapManager/GapManagerDemo2.wmv
- Publications
  - <<see the bibliographic section>>

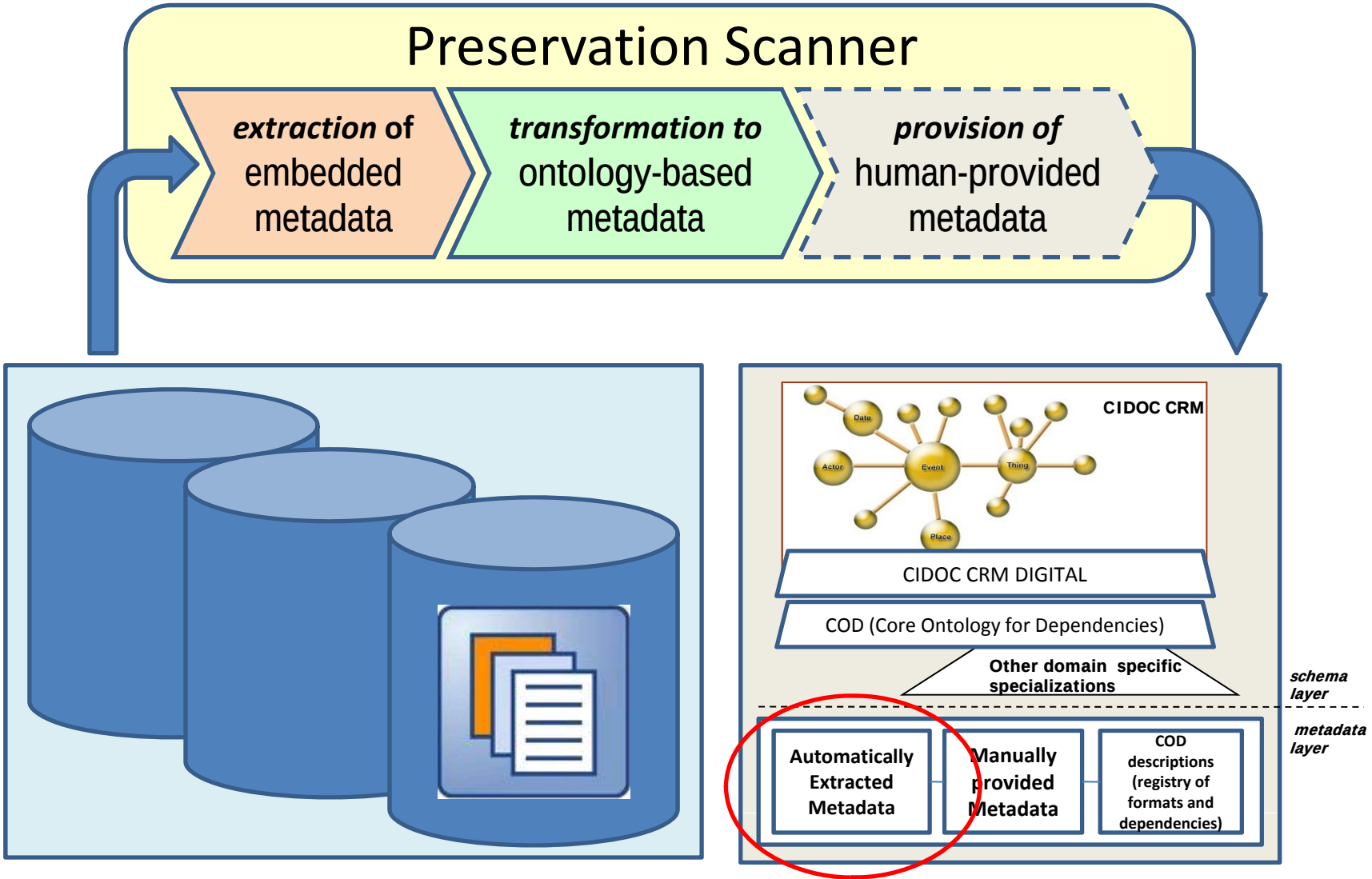
FORTH
Institute of Computer Science

PreScan

- Tool that aids the extraction and transformation of the various metadata embedded in plain files stored in a file system.
- It can actually aid the **ingestion**

FORTH
Institute of Computer Science

PreScan: Automatic Creation of Ontological Metadata (from embedded metadata)

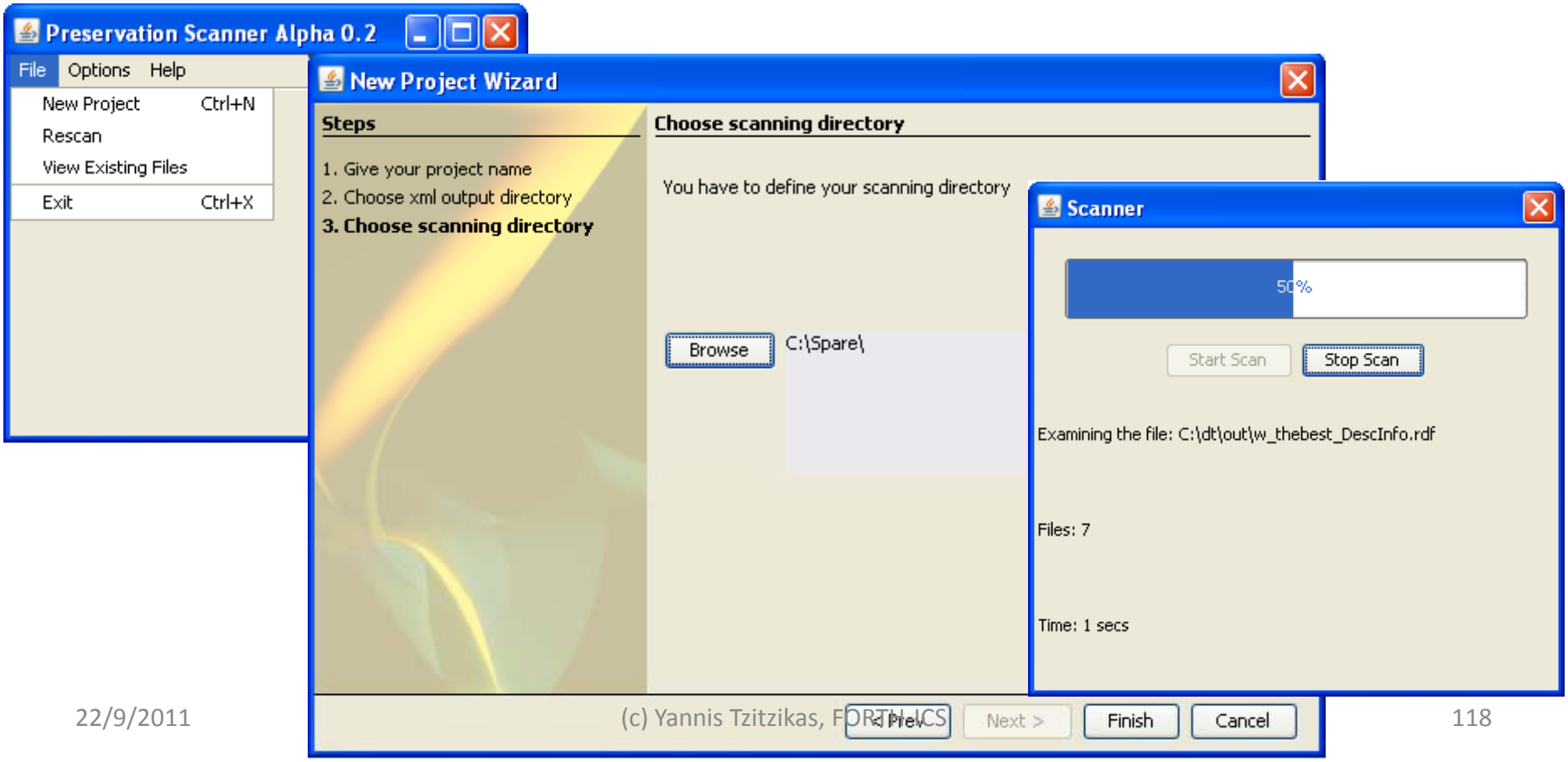


File System
(with texts, documents, images etc)

Semantic Repository of Metadata

PreScan (cont)

- It acts like the scanner of an AntiVirus program
  - The user defines the folders that should be scanned.



A SORT OF SYNOPSIS

FORTH
Institute of Computer Science

Typology of Approaches

(draft and incomplete)

What (tasks) to preserve

- General structure/semantics (e.g. in a file type, or format type, basis)
- particular (domain/application specific) tasks and external (or tacit) knowledge

Representation Framework (of Dependencies):

- None/Implicit
- Plain dependency graph
- Structurally O-O
- Rule-based

Storage (of Dependencies):

- Packages
- (Query-able) System

Ingestion (of Dependencies):

- Manual
- Semi-automatic
- Automatic

Services

- Intelligibility Checking
- Intelligibility Gap Computation
- Community-aware packaging
- Consequences of possible losses

FORTH
Institute of Computer Science

Typology of Approaches (incomplete)

What (tasks) to preserve
General structure/semantics (e.g. in format basis)
Particular (domain/application specific) tasks and external (or tacit) knowledge

Representation Framework (of Dependencies)
None/Implicit
Plain graph-based
Structurally O-O (and extensible), e.g. RDF/S
Rule-based

Storage Place (of Dependencies)
In a package (with the original object)
System with integrity constraints (and Query-able)

Ingestion (of Dependencies)
Manual
Semi-automatic
Automatic

Services
Intelligibility Checking
Intelligibility Gap Computation
Community-aware packaging
Consequences of possible losses

CONCLUDING REMARKS

FORTH
Institute of Computer Science

Concluding Remarks

- Intelligibility (task performability in general) is an important notion of preservation.
- We formalized this notion on the basis of dependencies. The notion of **dependency** is ubiquitous and **dependency management** is an important requirement that is subject of research in several (old and new emerged) areas, from software engineering to ontology engineering
  - A modern digital information preservation system should be **generic**, i.e. able to preserve heterogeneous digital objects which may have different interpretation of the notion of dependency.
- Contribution of the **graph**-based model:
  - Formalization of Intelligibility in terms of modules and dependencies
  - Formalization of community knowledge through the notion of DC profile
  - Definition of Intelligibility Gaps
    - Specify what has to be recorded/delivered to achieve intelligibility
  - Intelligibility-aware processes (for a Preservation Information System)
    - The notion of profile can serve as the gnomon for deciding/achieving:
      - **representation information adequacy** (during input)
      - **intelligibility** (during output **and for** archiving).

FORTH
Institute of Computer Science

Concluding Remarks (cont.)

Contributions of the **rule**-based approach

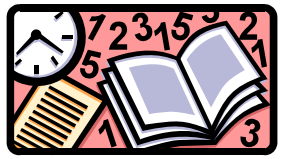
- More flexible and expressive than the graph-based approach
- We showed how rules can be employed for tackling the requirements
  - Previous attempts did not support disjunction and could not specify the properties of dependencies
  - However (as expected) the intelligibility gaps and not unique and the maintenance of rules is less straightforward
- We described how the proposed approach can be implemented using various technologies

FORTH
Institute of Computer Science

Further Work & Research

- Regarding the Graph and Rule-based model
  - Implement a **rule-based prototype**
    - Elaborate on the various open issues (abductive reasoning)
  - Study the effects of **changes** (on modules, dependencies) and **notification** services
    - Some work has been done for the graph-based model (see IJDL'09 paper)
  - Elaborate on the various difficulties (closure, dependency types)
- Extensions of the general model
  - Extend the framework with **Converters** (for tackling migration/evolution).
  - Study **composite modules** and dependencies of **different granularity**
  - Study **properties of dependency relations** that are practically useful (transitivity, acyclicity, …)
  - **Relax the notion of identify** of modules(incorporate the notion of similarity and the notion of Diff)
  - Investigate the benefits of more expressive representation frameworks (e.g. fuzzy/probabilistic)
  - Social perspective

FORTH
Institute of Computer Science



Related Publications

Intelligibility

latest results

- Y. Marketakis and Y. Tzitzikas, Dependency Management for Digital Preservation using Semantic Web Technologies, International Journal on Digital Libraries (IJDL), 10(4), 2009 (2010)
- Yannis Tzitzikas, Yannis Marketakis and Grigoris Antoniou,Task-based Dependency Management for the Preservation of Digital Objects using Rules, Procs of 6th Hellenic Conference on Artificial Intelligence, SETN'2010, Athens, Greece, May 2010.
- Y. Tzitzikas, "Dependency Management for the Preservation of Digital Information", 18th International Conference on Database and Expert Systems Applications, DEXA'2007, Regensburg, Germany, Sep. 2007
- Y. Tzitzikas and G. Flouris, "Mind the (Intelligibility) Gap", 11th European Conference on Research and Advanced Technology for Digital Libraries, ECDL'2007, Budapest, Hungary, September 2007
- Y. Tzitzikas, Dependency Management for the Preservation of Digital Information, International Conference PV'2007 (Ensuring the Long-Term Preservation and Value Adding to Scientific and Technical Data), Oberpfaffenhofen/Munich, Germany, October 2007

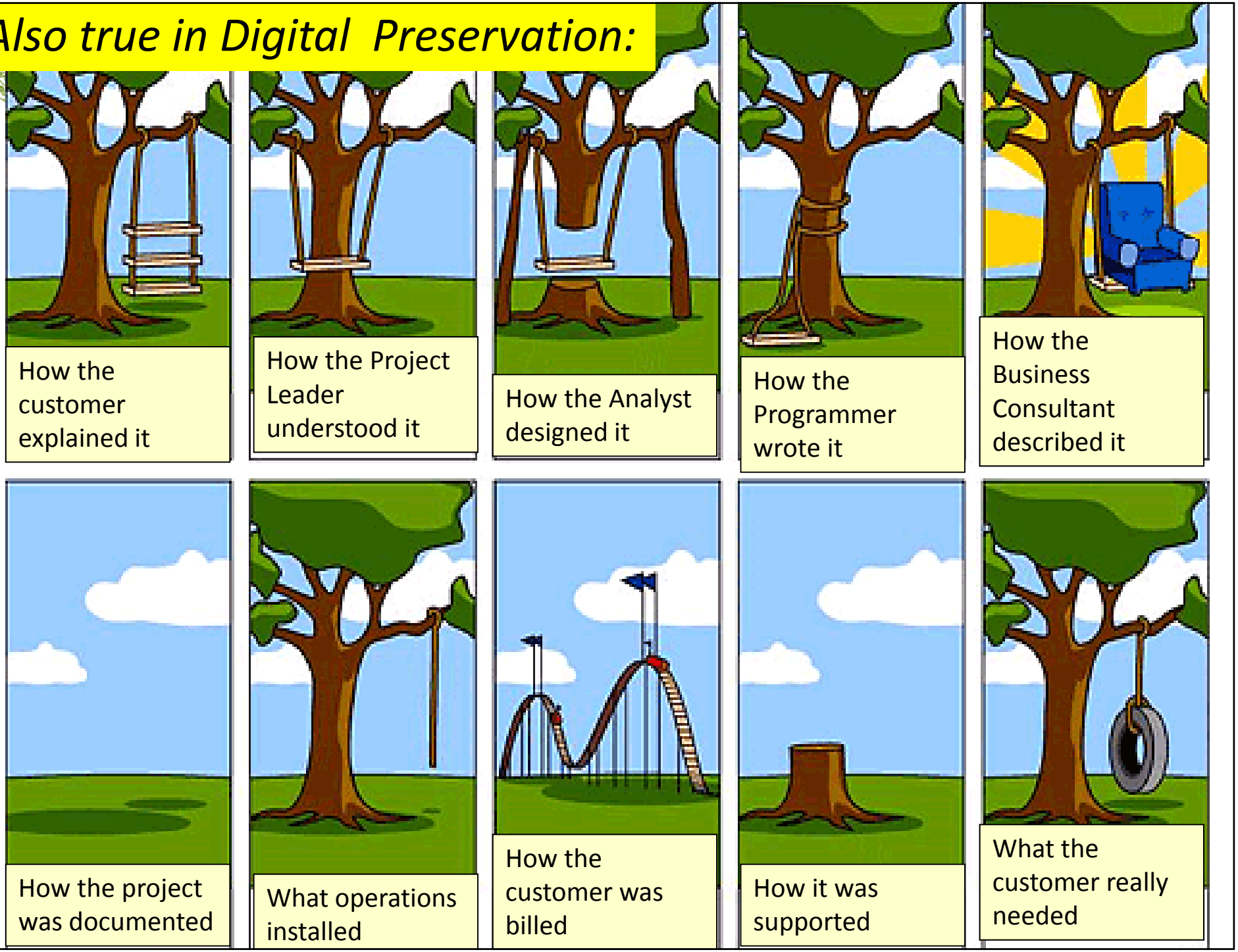
FORTH
Institute of Computer Science

Related Links

- **CASPAR** project (**C**ultural, **A**rtistic and **S**cientific knowledge for **P**reservation, **A**ccess and **R**etrieval) FP6 Integrated Project (April 2006 – September 2009)

- *Digital preservation is an endless-process which poses a number of challenging questions and research problems.*
- *However, it seems to me that in most cases we have to carry out a detailed requirements analysis and specification phase, while trying to avoid :*

Also true in Digital Preservation:



For questions/comments please send me an email.

Coordinates at: www.ics.forth.gr/~tzitzik

thanks for your attention