

Ontology based CBR with jCOLIBRI

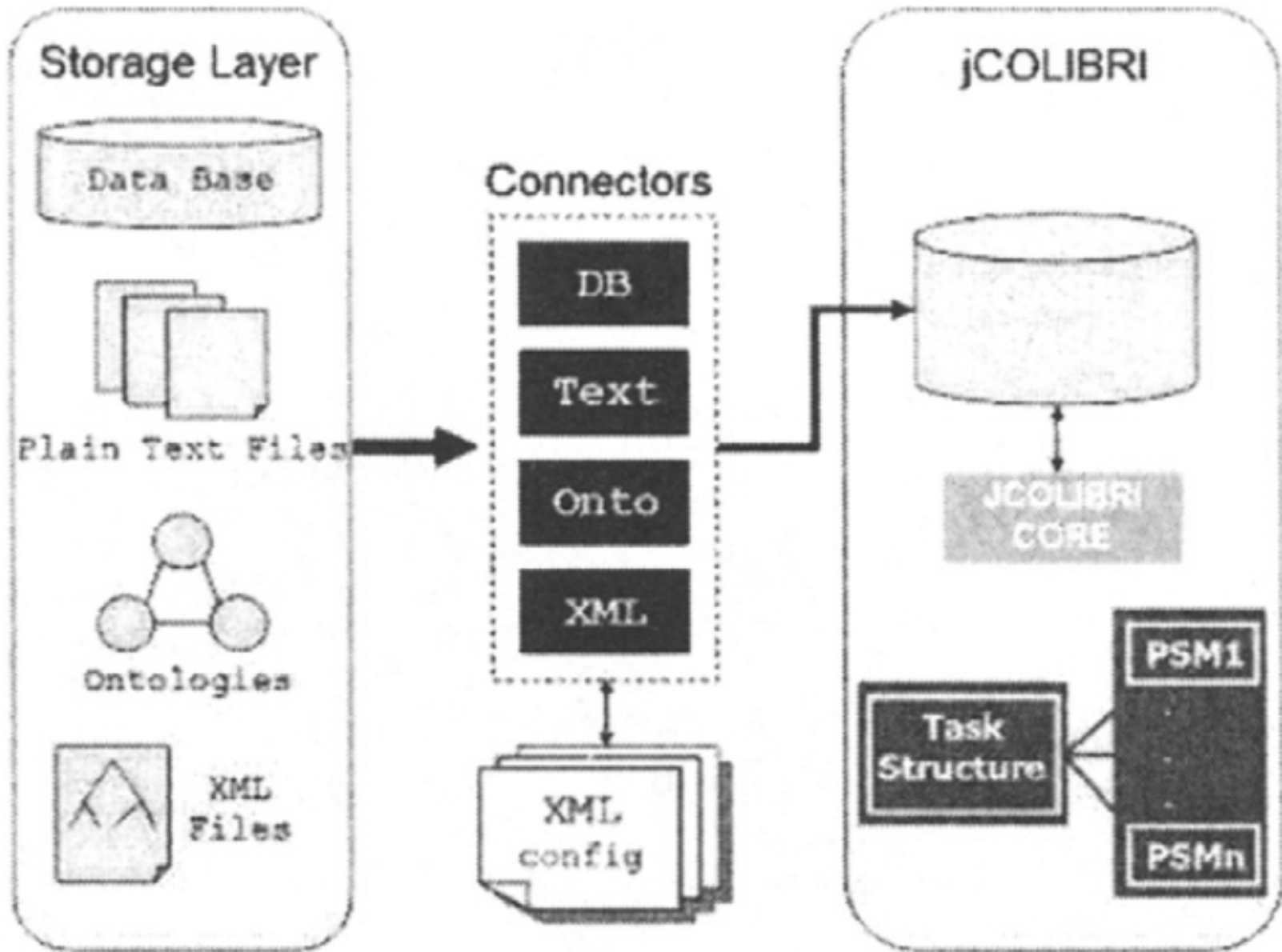
Introduction

1. What is jCOLIBRI?
2. Ontology based reasoning methods
3. Incorporation of ontology based reasoning methods in jCOLIBRI
4. Example
5. Conclusions

jCOLIBRI

- Object-oriented framework in Java for building Case Based Reasoning (CBR) systems
- jCOLIBRI is a core module which provides the basic functionality for CBR, with several optional modules to supply specific features
- 3 separate layers:
 - Interface
 - Reasoning logic
 - Persistence

jCOLIBRI



Design steps in jCOLIBRI

1. Define case structure
2. Choose similarity function
3. Case Base management
4. Task/Method decomposition

Ontology based CBR

- Ontologies are useful because they can use knowledge already require, conceptualized and implemented in a formal language
- The more knowledge, the more effective
- Usage of domain knowledge in semantic CBR processes to be more accurate

Ontology based CBR

Why is it useful in CBR?

- As a vocabulary to define the case structure
- To define the query vocabulary
- Retrieval, adaption and learning.

Ontologies as CBR **case** vocabulary

- Direct approach consisting on the use of a domain ontology in an object-oriented way
- Concepts are types or classes, individuals are values or objects and relations are the attributes describing the objects

Example: The **concept** *Destination* can be used as a type where every one of the instances are values of that type: *Lanzarote, Fuerteventura, Gran Canaria* etc.

Ontologies as CBR query vocabulary

- Two options to define the queries
 1. Use the exact same vocabulary as in the cases
 2. Using the ontology as the query vocabulary

Example: Query 1: "I want to go to Lanzarote", Query 2: "My favorite destination is Europe", Query 3: "I would like to travel to Spain".

The case whose destination is Spain, would be chosen for all of the 3 queries.

Case Retrieval using Ontologies

- Obvious that the concept hierarchy influences similarity assessment

There are different approaches:

- Classification based retrieval with two different approaches:
 1. Concept classification
 2. Instance recognition
- Computational based retrieval

Ontology based Adaption

1. Find the list of items L in the solution that needs to be adapted by following a relational path + a concept.
2. Every item in L must be substituted by a proper new item
3. Subsitute items that depended on earlier substituted items

Ontology based Adaption

- A small but powerful ontology adaption language

```
IDONTO:= /(Concept/Relation)* /Concept
IDPROPERTY:= (Relation/Concept)*
IDCASE:= "CASE."(attribute[.])*
RULE:= IDONTO,@,CONDITION,@,ADAPTATION
CONDITION:= (IDPROPERTY (=|!=) IDCASE) | (IDCASE (=|!=) String)
           | [not] (IDPROPERTY instanceOf Concept)
ADAPTATION:= SUBSTITUTION | MODIFY [FOLLOWDEPENDENCIES Relation]
SUBSTITUTION:= "SUBSTITUTE" [#CONDITION#]
MODIFY:= DIRECTMODIFICATION | ANYOTHERINSTANCEMODIFICATION
DIRECTMODIFICATION:= "DIRECT":IDPROPERTY:instance
ANYOTHERINSTANCEMODIFICATION:= "ANYOTHERINSTANCEOF":IDPROPERTY:concept
                               [#CONDITION#]
```

IDONTO identifies the path to the instance to adapt.

IDPROPERTY identifies a property of a concept.

IDCASE identifies an attribute of the case

SUBSTITUTION substitutes the instance by another one chosen randomly. Accepts can include a condition that the substitute instance must obey.

DIRECTMODIFICATION substitutes an attribute of the instance with the instance indicated by the developer.

ANYOTHERINSTANCEMODIFICATION substitutes an attribute of the instance by another instance of a concept. Accepts a condition for the substitute instance.

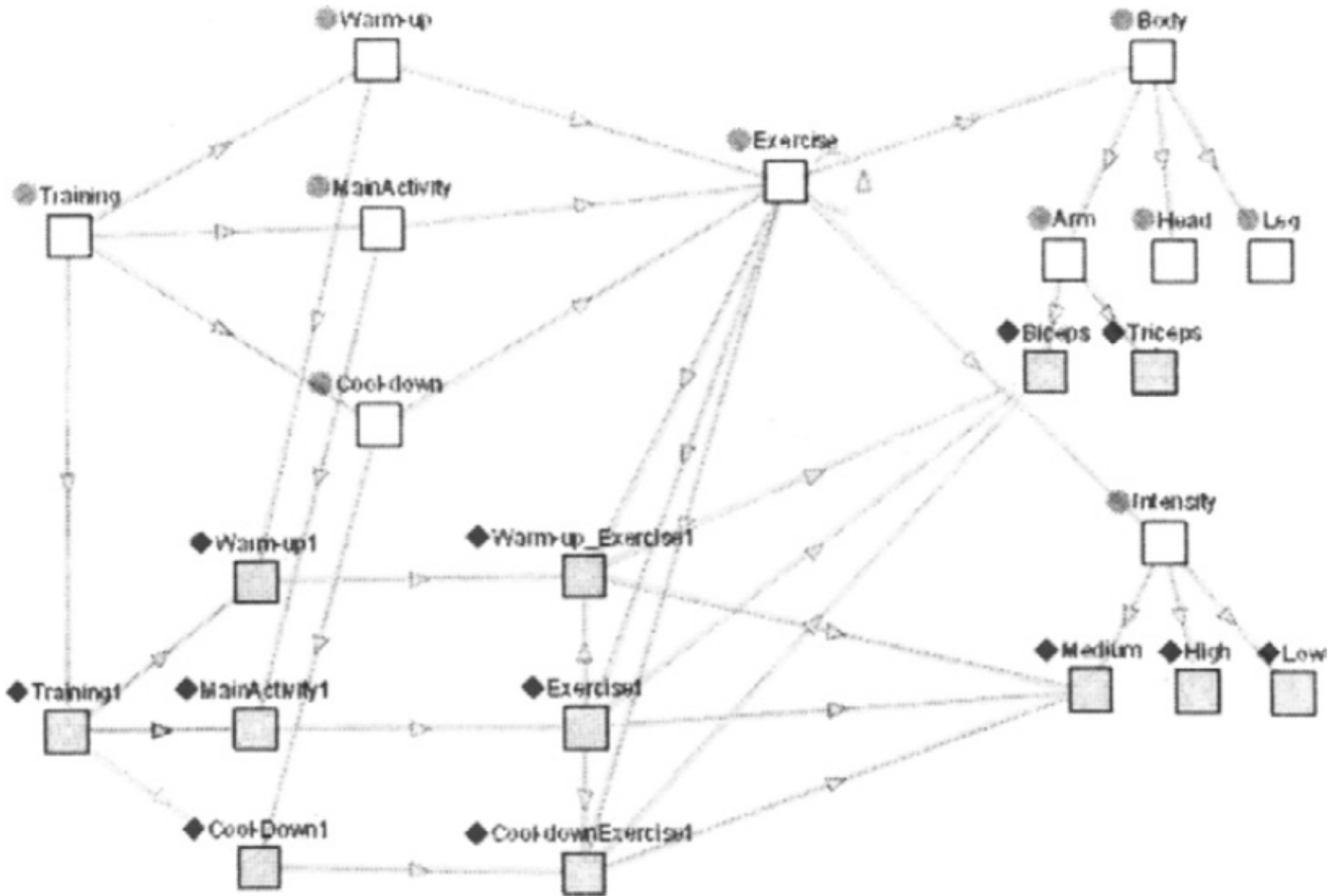
FOLLOWDEPENDENCIES applies the rule recursively to the instances related with the specified relation.

Ontology based Adaption

Rules are composed in three parts:

1. Identification of instance
2. Condition to evaluate the performing of adaption
3. Modification of the instance

Example



Example

- Query can be age, sex, state of health, lesions, etc.

Scenario: A user has a lesion in the arm and sends a query to the system. The system returns a training that contains exercises that use that body part.

```
/Training/hasMainActivity/MainActivity/hasExercise/Exercise@  
exercisedMuscles/Body == CASE.Description.lesion@  
ANYOTHERINSTANCEOF:exercisedMuscles/Body:Body  
#exercisedMuscles/Body not instanceof Arm# FOLLOWDEPENDENCIES  
relatedExercise
```

Example

Another rule changes the intensity of the exercise depending on the health of the user

/Training/hasMainActivity/MainActivity/hasExercise/Exercise@

CASE.Description.HealthState == low@

DIRECT:hasIntensity/Intensity: Low

FOLLOWDEPENDENCIES relatedExercise

Conclusions

- Describes the advantages that ontologies provide when they are used in CBR systems
- The implementation of Knowledge Intensive-CBR in jCOLIBRI through ontologies