

# Extending KDDML with a Visual Metaphor for the KDD Process

---

*Valerio Grossi  
Andrea Romei*

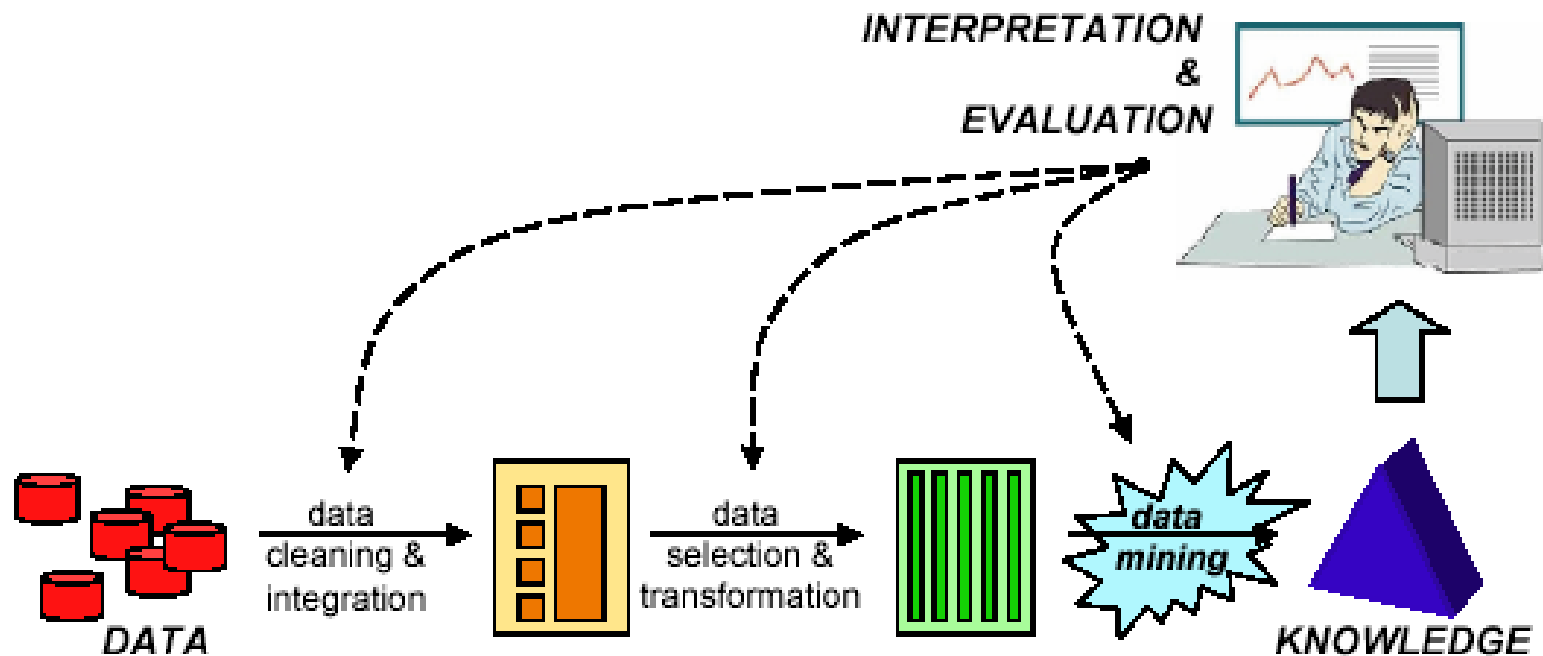
*10th International Conference on Visual Information Systems  
Salerno - Italy  
11-12 September, 2008.*

---

# What are KDD and DM?

- Many definitions, but in our view:
  - *“Knowledge Discovery in Databases (KDD) is a process, consisting of an iterative sequence of several steps, aimed at identifying valid, novel, potentially useful, and understandable patterns in data”.*
  - Data Mining (DM) refers to extracting knowledge (i.e. mining models) from large amounts of data.

# The KDD process



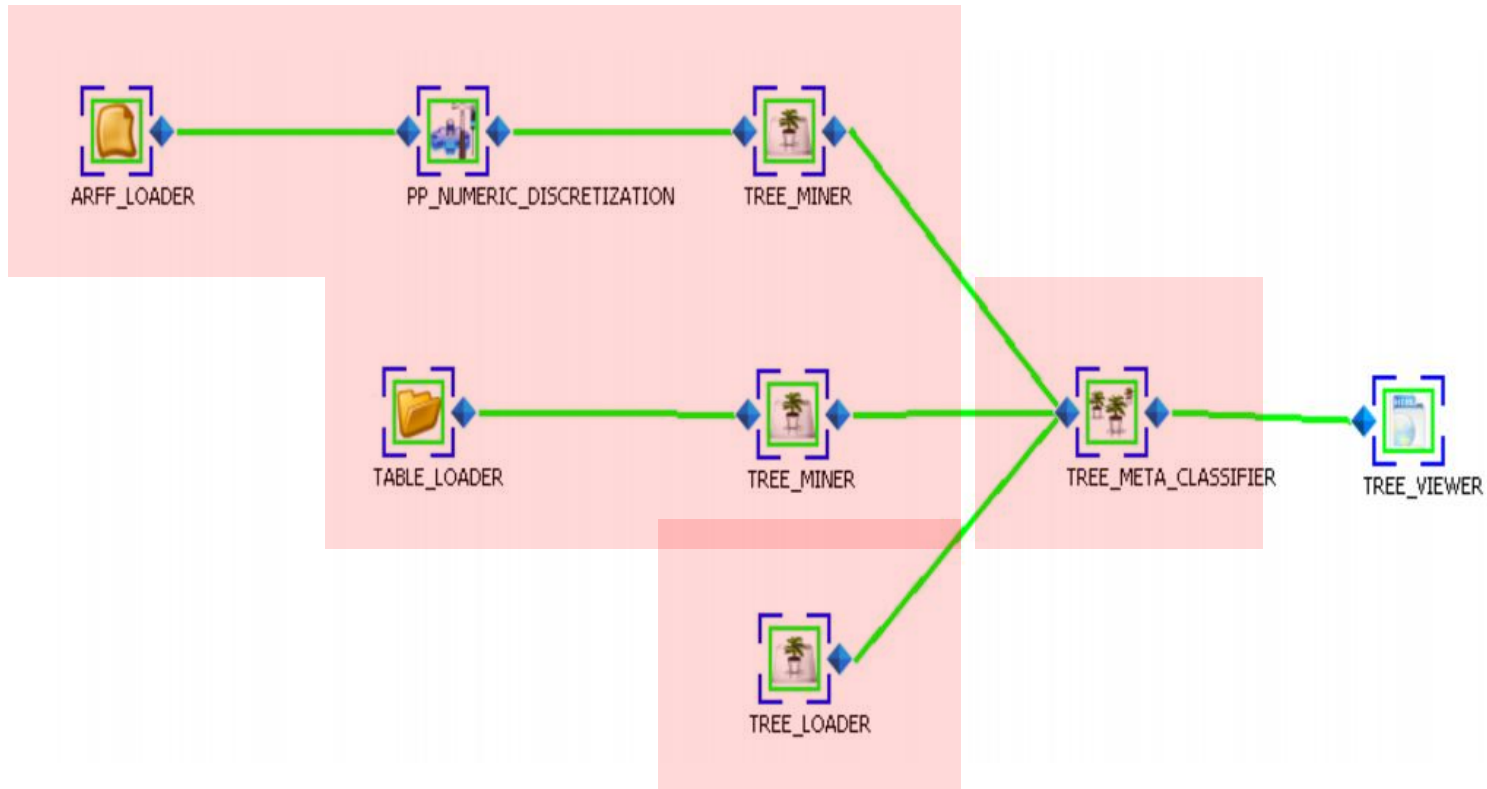
# KDDML (KDD Markup Language)

- KDDML is an XML-based middleware language (and system) in support of the KDD process.
  - Mining models and data are represented through XML documents.
  - Also KDD queries are XML documents.
    - Attributes correspond to parameters of the operator and sub-elements define arguments passed to the operator.

```
<OPERATOR_NAME att1="v1" ... attM="vM">  
  <ARG1_NAME> .... </ARG1_NAME>  
  ...  
  <ARGn_NAME> .... </ARGn_NAME>  
</OPERATOR_NAME>
```

# Example

Building a voting classifier among three classification trees.



# The KDDML query

```
<KDD_QUERY name="sample">
  <TREE_VIEWER style_sheet="tree.xsl">
    <TREE_META_CLASSIFIER xml_dest="weather_meta.xml" combination_type="committee">
      <TREE_MINER target_attribute="play">
        <PP_NUMERIC_DISCRETIZATION attribute_name="temperature">
          <ARFF_LOADER arff_file_name="weather.arff"/>
          <ALGORITHM algorithm_name="natural_binning">
            <PARAM name="number_of_intervals" value="3"/>
          </ALGORITHM>
        </PP_NUMERIC_DISCRETIZATION>
        <ALGORITHM algorithm_name="ID3"/>
      </TREE_MINER>
      <TREE_MINER target_attribute="play">
        <TABLE_LOADER xml_source="weather.xml"/>
        <ALGORITHM algorithm_name="ID3"/>
      </TREE_MINER>
    <TREE_LOADER xml_source="weather_tree.xml"/>
  </TREE_META_CLASSIFIER>
</TREE_VIEWER>
</KDD_QUERY>
```

---

# Requirements for a KDD graphical user interface

- The design and implementation of a visual language for KDD must satisfy some features, also related to the system architecture.
  - Visual aspects: graph vs. tree-like.
  - Visual programming support.
  - Extensibility.

# Visual aspect: two common abstractions

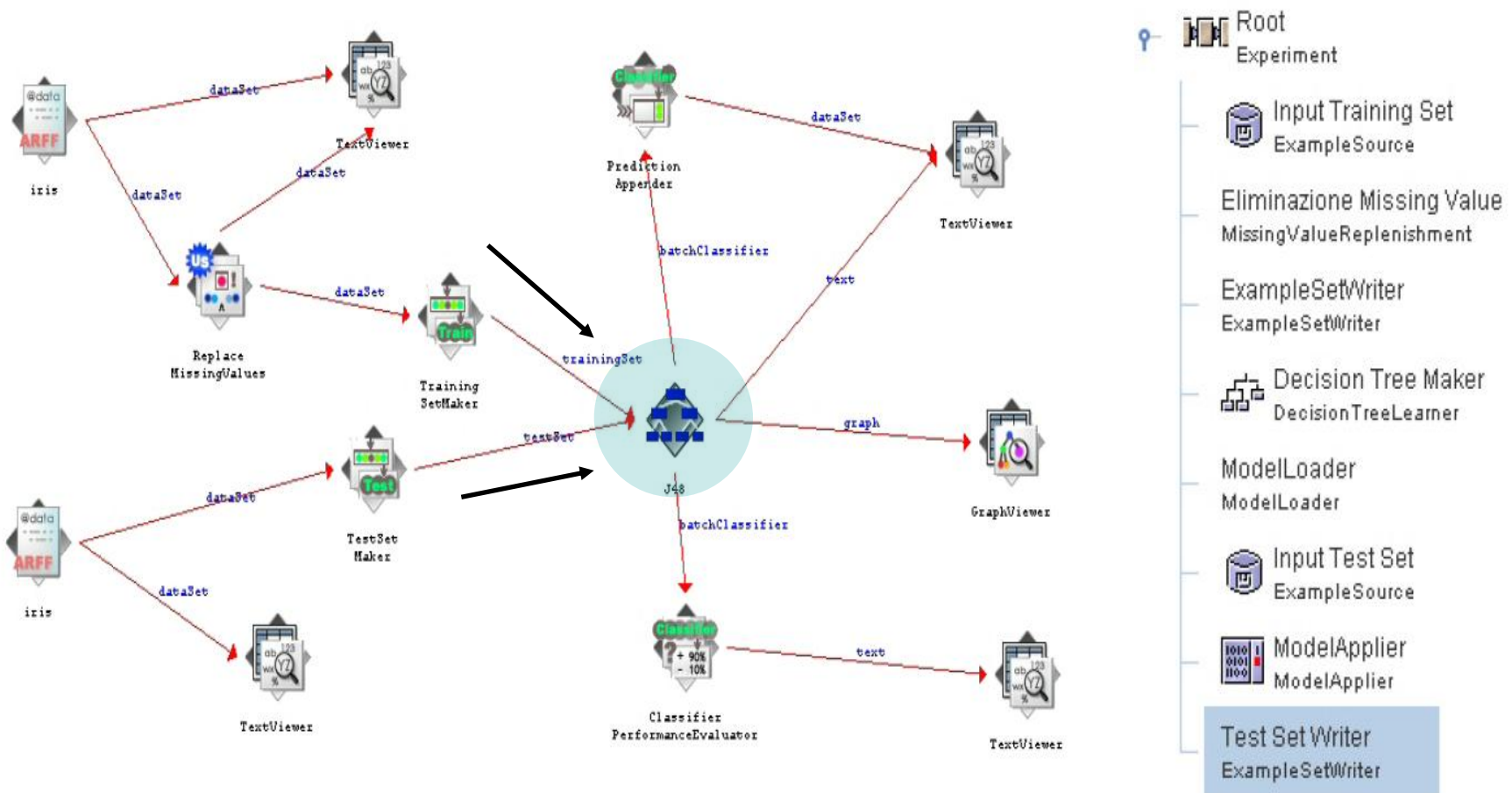
- Tree-like representation.

- All operators are arranged in a tree-like structure in which the parent node passes data to its children.
  - It forces the user to follow the steps from data input (the root of the tree) to result visualization.
  - The usage is complicated if there is an operator that needs more than one input.

- Graph representation.

- Directed acyclic graph where the nodes are operators and the edges represent the knowledge flow.
  - It enables the explicit representation of all operators receiving data from more than one node.
  - It leaves more freedom to the user and, at the same time, less guidance.

# Visual aspects: graph vs. tree



---

# Visual aspect in KDDML

- KDDML uses a graph to represent the knowledge flow.
  - Several KDDML tasks take inputs from different sources.
  - Good trade-off between user friendliness and expressiveness.
  - The XML DOM representing the query can be easily obtained by means of a right-to-left exploration of the graph.

---

# Definition of a KDD process

- The KDD process is described in terms of a hierarchical process model.
- Three levels of abstraction.
  - Phases: data preparation, modeling, evaluation, deployment and visualization.
  - Generic tasks.
  - KDDML operators.
- Our visual language is embedded into a user-friendly interface that compiles the KDD process into KDDML queries.

# Saving a KDD process

```
<KDD_QUERY name="sample">
  <TREE_VIEWER style_sheet="tree.xsl" x="490" y="250" id="1">
    <TREE_META_CLASSIFIER xml_dest="weather_meta.xml" combination_type="committee" x="390" y="250" id="2">
      <TREE_MINER target_attribute="play" x="300" y="150" id="3">
        <PP_NUMERIC_DISCRETIZATION attribute_name="temperature" x="260" y="100" id="4">
          <ARFF_LOADER arff_file_name="weather.arff" x="100" y="100" id="5"/>
          <ALGORITHM algorithm_name="natural_binning">
            <PARAM name="number_of_intervals" value="3"/>
          </ALGORITHM>
        </PP_NUMERIC_DISCRETIZATION>
        <ALGORITHM algorithm_name="ID3"/>
      </TREE_MINER>
      <TREE_MINER target_attribute="play" x="200" y="250" id="6">
        <TABLE_LOADER xml_source="weather.xml" x="150" y="350" id="7"/>
        <ALGORITHM algorithm_name="ID3"/>
      </TREE_MINER>
      <TREE_LOADER xml_source="weather_tree.xml" x="100" y="400" id="8"/>
    </TREE_META_CLASSIFIER>
  </TREE_VIEWER>
  <EDGES>
    <EDGE from="2" to="1" color="green"/>
    <EDGE from="3" to="2" color="green"/>
    <EDGE from="6" to="2" color="green"/>
    <EDGE from="8" to="2" color="green"/>
    <EDGE from="4" to="3" color="green"/>
    <EDGE from="5" to="4" color="green"/>
    <EDGE from="7" to="6" color="green"/>
  </EDGES>
</KDD_QUERY>
```

Visual  
Metaphor  
Information

---

# Visual Programming Support

- **Static analysis.**
  - To check syntactic and type correctness.
- **Run-time checking.**
  - To check errors at run-time.
- **Meta-execution.**
  - Data (and models) are divided into meta-data (meta-model) and physical data (physical model).
  - The system runs the KDD process using only the available meta-data.

# Meta-data and physical-data

```
<KDDML_TABLE data_file="weather.csv">
  <SCHEMA logical_name="weather" number_of_attributes="5" number_of_instances="20">
    <ATTRIBUTE name="outlook" number_of_missed_values="2" type="nominal">
      <NOMINAL_DESCRIPTION number_of_values="3">
        <VALUE value="rainy" cardinality="5"/>
        <VALUE value="sunny" cardinality="8"/>
        <VALUE value="overcast" cardinality="5"/>
      </NOMINAL_DESCRIPTION>
    </ATTRIBUTE>
    <ATTRIBUTE name="temperature" number_of_missed_values="0" type="numeric">
      <NUMERIC_DESCRIPTION mean="74.0" std_dev="35.36" min="64.0" max="85.0"/>
    </ATTRIBUTE>
    <ATTRIBUTE name="humidity" number_of_missed_values="4" type="numeric">
      <NUMERIC_DESCRIPTION mean="81.87" std_dev="77.18" min="65.0" max="95.0"/>
    </ATTRIBUTE>
    <ATTRIBUTE name="windy" number_of_missed_values="6" type="nominal">
      <NOMINAL_DESCRIPTION number_of_values="2">
        <VALUE value="TRUE" cardinality="4"/>
        <VALUE value="FALSE" cardinality="10"/>
      </NOMINAL_DESCRIPTION>
    </ATTRIBUTE>
    <ATTRIBUTE name="play" number_of_missed_values="0" type="nominal">
      <NOMINAL_DESCRIPTION number_of_values="2">
        <VALUE value="no" cardinality="8"/>
        <VALUE value="yes" cardinality="12"/>
      </NOMINAL_DESCRIPTION>
    </ATTRIBUTE>
  </SCHEMA>
```

Physical-data

Meta-data

# Meta-model and physical-model

```
<TreeModel algorithmName="YaDT" functionName="classification" splitCharacteristic="binarySplit">
  <MiningSchema>
    <MiningField name="outlook" usageType="active"/>
    <MiningField name="temperature" usageType="active"/>
    <MiningField name="humidity" usageType="active"/>
    <MiningField name="windy" usageType="active"/>
    <MiningField name="play" usageType="predicted"/>
  </MiningSchema>
  <Node recordCount="12.0" score="">
    <True/>
    <ScoreDistribution recordCount="6.0" value="yes"/>
    <ScoreDistribution recordCount="6.0" value="no"/>
    <Node recordCount="3.0" score="no">
      <SimplePredicate field="outlook" operator="equal" value="overcast"/>
      <ScoreDistribution recordCount="1.0" value="yes"/>
      <ScoreDistribution recordCount="1.182" value="no"/>
    </Node>
    <Node recordCount="3.0" score="yes">
      <SimplePredicate field="outlook" operator="equal" value="rainy"/>
      <ScoreDistribution recordCount="2.0" value="yes"/>
      <ScoreDistribution recordCount="0.182" value="no"/>
    </Node>
    <Node recordCount="8.0" score="no">
      <SimplePredicate field="outlook" operator="equal" value="sunny"/>
      <ScoreDistribution recordCount="3.0" value="yes"/>
      <ScoreDistribution recordCount="4.636" value="no"/>
    </Node>
  </Node>
</TreeModel>
```

Meta-model

Physical-model

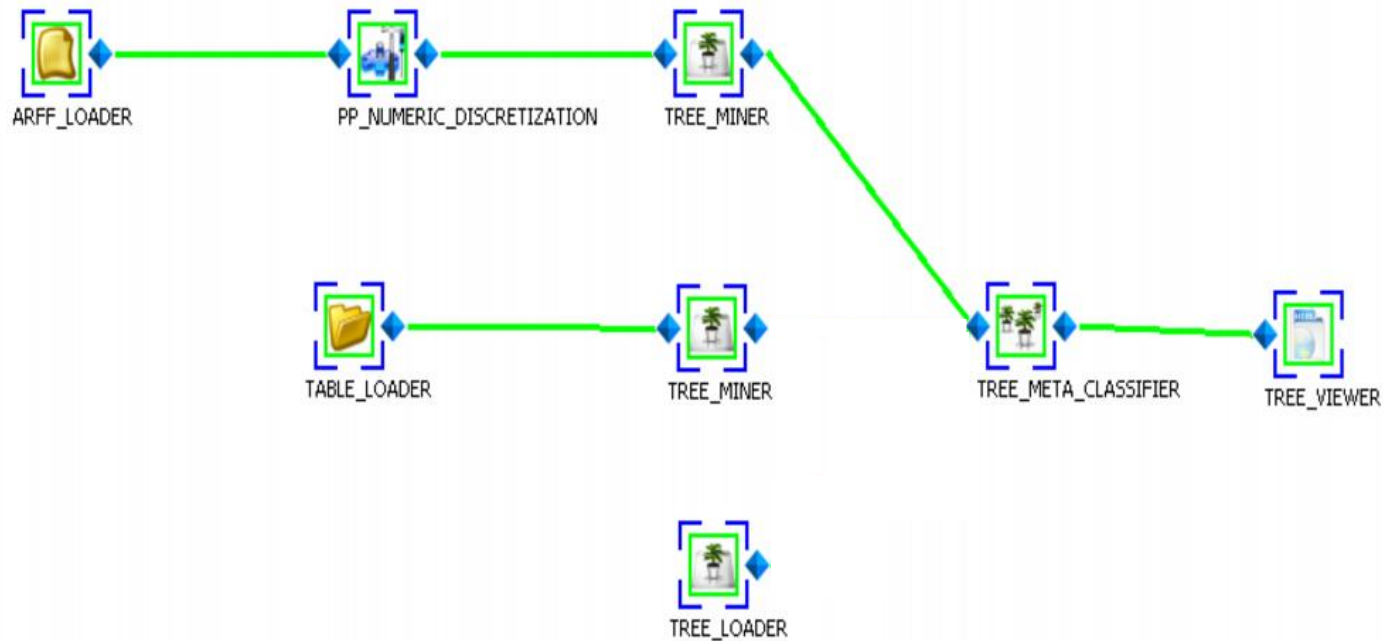
---

# Meta-execution

- Meta-execution acts implicitly whenever two nodes are connected by means of a method implemented in each KDDML operator.
  - It takes a list of meta-objects as input.
  - It returns a meta-object as output.
  - It throws an exception if an error on meta-objects occurs.
- Meta-information is directly available for the target node and this information is used to
  - Validate the rest of the query.
  - Present the partial output to the user.

# Meta-execution: example

In a classifier committee, all the trees involved must be extracted from training sets sharing the same metadata.

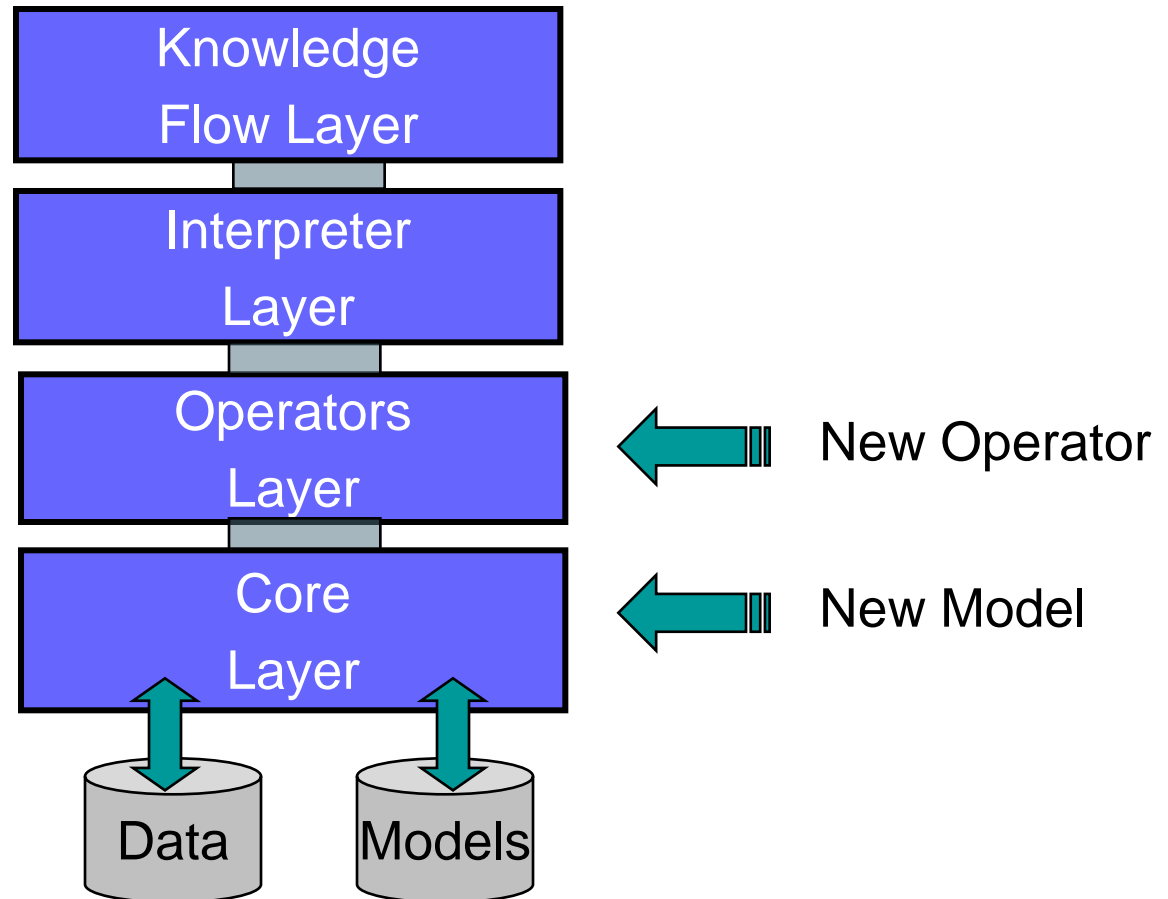


---

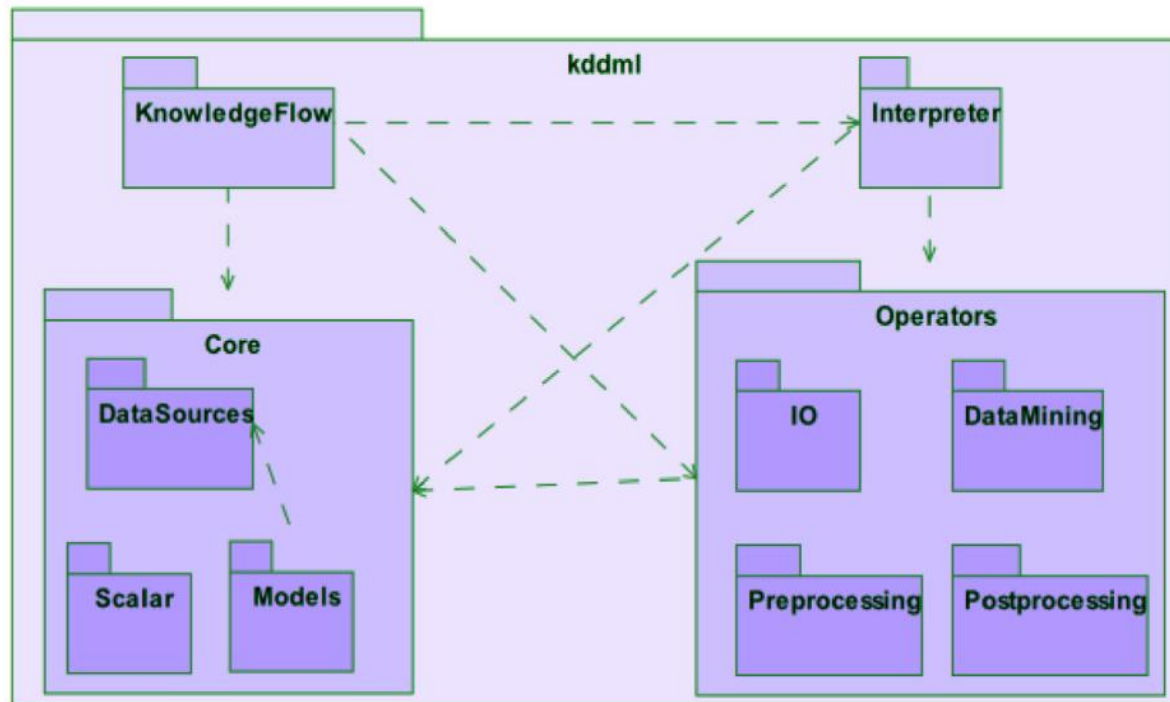
# Extensibility

- The GUI level of a KDD system has not to be modified every time a new operator is introduced.
  - The operator should be automatically available at the graphical level.
- The GUI level must be independent from the actual status of the system.
  - The system must provide a common well-defined way to extend itself, guaranteeing all the necessary information at the GUI level.

# KDDML system architecture



# The dependencies between the top-level KDDML packages



---

# Final Remarks

- The flow designer interface has been recently integrated at the top of the KDDML architecture.
  - We are currently in an open alpha-testing phase.
- Future work
  - Definition of tools enabling the visualization of PMML models.
  - A more rigorous definition of an environment capable of dealing with both logical data and knowledge.