### ADMIRE D5.6 – Tools Development Progress Report and Roadmap for Further Development

<table>
<thead>
<tr>
<th><strong>Project Title</strong></th>
<th>ADMIRE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Document Title</strong></td>
<td>ADMIRE – Tools Development Progress Report and Roadmap for Further Development</td>
</tr>
<tr>
<td><strong>Deliverable Number</strong></td>
<td>D5.6</td>
</tr>
<tr>
<td><strong>Authorship</strong></td>
<td>Adrian Mouat, Ivan Janciak, Branislav Simo, Maciej Jarka, Michal Laclavik, Carlos Buil-Aranda, Rob Baxter</td>
</tr>
<tr>
<td><strong>Document Filename</strong></td>
<td>ADMIRE-D5.6-progress-report.tex</td>
</tr>
<tr>
<td><strong>Document Version</strong></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Distribution Classification</strong></td>
<td>Public</td>
</tr>
<tr>
<td><strong>Distribution List</strong></td>
<td>ADMIRE Project Team</td>
</tr>
<tr>
<td><strong>Approval List</strong></td>
<td>Maciej Jarka, Project Manager</td>
</tr>
</tbody>
</table>

**Copyright © 2011 The ADMIRE Project**
<table>
<thead>
<tr>
<th>Personnel</th>
<th>Date</th>
<th>Comment</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>21/02/2011</td>
<td>First draft</td>
<td>0.2</td>
</tr>
<tr>
<td>AM</td>
<td>25/02/2011</td>
<td>Minor updates and Workflow Vis section</td>
<td>0.2</td>
</tr>
<tr>
<td>BS, MJ, ML, IJ</td>
<td>18/03/2011</td>
<td>Updates to Development section, Process Designer, Portals, SKSA, GPM and Workflow Visualiser</td>
<td>0.3</td>
</tr>
<tr>
<td>AM</td>
<td>21/03/2011</td>
<td>Updated ADMIRE Workbench section</td>
<td>0.4</td>
</tr>
<tr>
<td>BS</td>
<td>02/05/2011</td>
<td>Removed Orava app. portal description. Job management description added to Portal section</td>
<td>0.5</td>
</tr>
<tr>
<td>AM</td>
<td>05/05/2011</td>
<td>Wrote Roadmap, reviewed Portal section and other minor updates</td>
<td>0.6</td>
</tr>
<tr>
<td>CBA</td>
<td>12/05/2011</td>
<td>Added myExperiment section and added to Registry Client</td>
<td>0.7</td>
</tr>
<tr>
<td>AM</td>
<td>17/05/2011</td>
<td>Wrote Executive Summary</td>
<td>0.8</td>
</tr>
<tr>
<td>IJ, MJ</td>
<td>19/05/2011</td>
<td>Updated PD Section and CV section</td>
<td>0.9</td>
</tr>
<tr>
<td>AM</td>
<td>30/05/2011</td>
<td>Updated after review</td>
<td>0.10</td>
</tr>
<tr>
<td>RMB</td>
<td>31/05/2011</td>
<td>Reviewed; awaiting appendix on Process Designer UG</td>
<td>0.11</td>
</tr>
<tr>
<td>IJ, AM</td>
<td>01/06/2011</td>
<td>Added manual to appendix</td>
<td>0.12</td>
</tr>
<tr>
<td>RMB</td>
<td>02/06/2011</td>
<td>Final review and signoff</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Contents

1 Executive Summary 5

2 WP5 Development 6
   2.1 Overview of the ADMIRE Tools ........................................ 6
   2.2 Progress to PM39 ...................................................... 8

3 The ADMIRE Workbench 9
   3.1 Process Designer .................................................... 10
      3.1.1 DISPEL Editor ................................................... 11
      3.1.2 Diagram Editor .................................................. 11
      3.1.3 Registry Client ................................................ 11
      3.1.4 Registry View .................................................. 12
      3.1.5 Future Work .................................................... 13
   3.2 Gateway Process Manager ............................................. 13
      3.2.1 Future Work .................................................... 14
   3.3 DMI Models Visualization ............................................ 14
      3.3.1 Radial Coordinates Visualization ............................... 15
      3.3.2 Classification Model Visualization .............................. 17
   3.4 Table Visualiser ...................................................... 19
   3.5 Chart Visualiser ...................................................... 19
   3.6 Semantic Knowledge Sharing Assistant ................................ 20
      3.6.1 Recent Developments ........................................... 21
      3.6.2 Conclusion and Future Work ..................................... 22
   3.7 Gateway Monitor ...................................................... 23
      3.7.1 Future Work .................................................... 24
   3.8 Validation Preview .................................................... 25

4 The ADMIRE Portal Framework 26
   4.1 Knowledge Exchange through the Repository ....................... 27
   4.2 Creating process descriptions ..................................... 27
   4.3 Process execution ................................................... 28
   4.4 ADMIRE myExperiment .............................................. 30

5 Roadmap for Further Development 31

References 33

A List of Workbench Plug-ins 34

B Workbench Installation Instructions and Process Designer Quickstart Guide 36

1 Introduction 38

2 Requirements 38

3 Installation 38
# Configuration 38

4.1 Gateway Client .................................................. 39
4.2 Registry Client .................................................. 39

# Using the PROCESS DESIGNER 39

5.1 Setting the DISPEL Development Perspective .................. 40
5.2 Using the Project Wizard ....................................... 41
5.3 Working with the Registry View ................................ 41
5.4 Creating a New DISPEL File .................................... 41
5.5 Browsing the Structure of DISPEL Sentences .................. 42
5.6 Validation of a DISPEL Sentence on the ADMIRE Gateway ... 42
5.7 Generating LATEX from a DISPEL Sentence ................... 43
5.8 Initialising Diagram Editor ..................................... 43
5.9 Reducing Elements in the Diagram Editor ...................... 44

# Graphical Notation 45

# Final Walk Through 45

7.1 Hello World Example ........................................... 46
1 Executive Summary

This document is deliverable D5.6 of the ADMIRE project, which reports on the status of the Tools workpackage at the end of the project and provides a roadmap for further development.

The workpackage has successfully produced the ADMIRE Workbench, a tool based on Eclipse for helping users – normally data analysis experts – to create, validate and execute DISPEL workflows, as well as interpreting the results. The Workbench is integrated with other parts of the ADMIRE framework – Gateways for submitting workflows to and the Registry for obtaining information on available resources. Workpackage 5 has also produced several other tools, including the Gateway Monitor web application for viewing the status of running and completed workflows, the Portal framework used by Workpackage 6 to produce Portals specific to use cases and the ADMIRE myExperiment portal which helps users to collaboratively work with DISPEL. The ADMIRE Workbench is distributed as part of the ADMIRE Virtual Machine available from the ADMIRE website, or can be added to a running Eclipse installation via the update site (also linked from the website\[1\]).

While Workpackage 5 has produced a usable set of tools that are already in a state where they can be used effectively, the large and varied range of requirements and possible uses for data-intensive computing tools means that there is much more that still could be achieved. For this reason, the document concludes with the “Roadmap”, which identifies various ways in which the Workbench could be improved in order to become a truly revolutionary tool in Data Intensive computing.

\[1\]www.admire-project.eu
2 WP5 Development

2.1 Overview of the ADMIRE Tools

Figure 1 shows an overview of the ADMIRE architecture, focusing on the tools that are described in this document. This diagram has been updated from D5.5 to reflect the final set of tools available to ADMIRE developers and users.

Figure 1: Overview of the ADMIRE Tools and their interactions

The top layer shows the tools that constitute the ADMIRE Workbench. The Workbench brings together a set of compatible tools designed to support exploration and exploitation of data sources. Workbench tools are used by domain or data analysis experts to:

- explore and analyse data (Chart Visualiser, DMI Models Visualiser and Table Visualiser, see Sections 3.5, 3.3 and 3.4);
- construct complex knowledge discovery workflows visually (Process Designer, described in Section 3.1);
- edit DISPEL documents using a text editor interface (Process Designer, described in Section 3.1);
- validate constructed workflows to ensure that they can be compiled by an ADMIRE Gateway (Process Designer, described in Section 3.1);
- inspect and use context-related semantic information (Semantic Knowledge Sharing Assistant, see Section 3.6);
• submit workflows to an ADMIRE Gateway and monitor processing (Gateway Process Manager, see Section 3.2);

• validate workflows with a Gateway prior to submission and retrieve a graph representation of the workflow, (see Section 3.8);

• access and query the ADMIRE Registry (Registry Client and View, see Section 3.1.3 and 3.1.4);

• and visualise the data mining results (DMI Models Visualiser, Chart Visualiser and Table Visualiser, see Sections 3.3, 3.5 and 3.4).

For a complete overview of the Workbench see Section 3.

All Workbench tools are built as plug-ins on the Eclipse platform\(^2\). Eclipse was chosen because it already provides many facilities required by DMI-process developers as well as a stable and easily extensible platform for the development of new functionality. Each plug-in provides a set of extension points that the other components can use to communicate and share provided functionality.

Workbench components submit workflow processes, in the form of DISPEL documents, to the Gateway Process Manager for execution. The Gateway Process Manager makes use of the common Gateway Client component to send the DISPEL document to the Gateway and monitor its execution. Once completed, the results can then be viewed in one of several visualisers including the DMI Result Visualiser and the Chart Visualiser.

The layer “below the hourglass” in Figure 1 represents the enactment components in the architecture. The Gateway is a service that processes DISPEL documents and produces corresponding OGSA-DAI workflows which are executed on sites and resources that have been discovered by the Gateway. When results are available they are delivered directly from the data source to the client rather than via the Gateway. The Gateway is described under Workpackage 4.

At the lowest layer, OGSA-DAI requests are executed by the OGSA-DAI services that are accessible through a particular Gateway.

The Workflow Monitor – now called the Gateway Monitor – is also shown in Figure 1. This tool is a stand-alone web application and hence shown to the side of the Eclipse framework. The Gateway Monitor shows real-time information on the status of processes running on ADMIRE Gateways. Users can drill down into individual processes and view their workflows including information on the flow of data that can be used to identify performance issues. As well as being a powerful diagnostic tool, the Gateway Monitor provides a visual demonstration of how the distributed ADMIRE architecture works. The Gateway Monitor is described in full in Section 3.7.

In addition to these tools, Workpackage 5 is also responsible for the ADMIRE Portal framework. The portals were a major focus of the last period of the ADMIRE project, and provide high-level, simple interfaces to the use-cases which are designed to be used by non-expert end users. Section 4 describes the framework used to create the portals. Descriptions of specific portals are covered as part of Workpackage 6.

\(^2\)See www.eclipse.org
2.2 Progress to PM39

The following tasks in the updated DoW [2] were completed in the final period:

WP5.9: Extend the ADMIRE Workbench to make use of revised management extensions provided by USMT/the ADMIRE Gateway. These extensions were used by the Gateway Monitor tool [3.7] and the validation functionality added to the Process Designer [3.1].

WP5.17: Revise Service Description GUI for the new version of the Workbench. This became the updated Registry View [3.1.4].

WP5.29: Provide support for ADMIRE Portal revisions. Details of the portal framework can be found in [4]. Details on specific portals are covered in Workpackage 6.

The goal for WP5 in the final period was to produce a stable, productive platform for developing ADMIRE workflows and to create good documentation and tutorials. The following specific tasks were achieved:

- The Gateway Monitor tool was created;
- The Registry View was updated and improved following feedback from users;
- The Portal framework was extended and improved following feedback from use case developers;
- The Process Designer was kept up-to-date with the evolving DISPEL specification;
- The SKSA tool was finalised and integrated with the Repository;
- The Gateway Process Manager interface was extended and improved following feedback from users;
- A validation plug-in was created that allows users to check their DISPEL workflows are valid on a given gateway;
- A Table Visualiser plug-in was created for viewing tabular data;
- The interface for Chart Visualiser was changed to be more intuitive and user friendly;
- A comprehensive set of documentation was created for users of the Workbench, including a manual, tutorials and a demonstration video.

The full Workbench with all components was initially produced in a Release Candidate version in PM35. This version was used in the creation of DISPEL for the astronomy use case (see D6.6 [4]), which identified several problems which were then addressed in the final release of the Workbench.

The Workbench and documentation can be downloaded from the ADMIRE website[3]. Installation instructions and a quickstart guide can be found in Appendix B.

---

3 The ADMIRE Workbench

The ADMIRE Workbench is a set of compatible tools supporting various aspects of data analysis work typically carried out by data analysis experts. Users of the Workbench produce DISPEL workflows which can sent for enactment on ADMIRE Gateways which can return results for analysis within the Workbench.

The Workbench interacts with three services:

Registry: Contains the “knowledge discovery palette” available to Workbench users; that is the list of all Processing Elements, functions and types available for use. The Registry can also contain abstract metadata on other items, such as those stored in the Repository. More information on the Registry can be found in D1.9 [1].

Repository: A remote storage area. Can be used to store arbitrary data, including DISPEL documents, whether they are work in progress complete or work that a user wishes to share more widely or to store more reliably. Primarily integrated in the Workbench through the SKSA. See D4.6 [8] for more information.

Gateway: A platform to which the Workbench submits DISPEL requests for enactment.

To avoid repetitively coding functions needed by several tools and to insulate the tool development from the ADMIRE Gateway and Registry development, a set of interfaces were created for the developers of ADMIRE plug-ins and tools:

- an interface to the Gateways that packages requests, handles the Gateway protocols and unpacks responses, so that tool developers can be protected from some of the changes as the Gateway definition and implementation evolves. It includes a library of utility functions that help with generating requests and interpreting the responses. This functionality is implemented in the Gateway Client tool and Gateway Process Manager;

- an interface to the Registry, to handle the Registry protocol and to provide utility functions for generating Registry queries and for interpreting responses. This functionality is supported by the Registry Client;

- standard functions that are commonly required by the tools, e.g. DISPEL manipulation.

A full list of the Workbench plug-ins, including the names of developers can be found in Appendix A.

The ADMIRE Eclipse Update Site[4] can be used to automatically install the above ADMIRE Workbench plug-ins into a running Eclipse installation. It is recommended that users deploy the plug-ins into a fresh installation of Eclipse 3.6 with Modelling Tool[5]. The Eclipse Update Manager is shown in Figure 2.

The following sections describe the various components of the workbench in more detail. More information on the installing and using the Workbench can be found in Appendix B and on the ADMIRE website.

The following sections describe the various components of the Workbench in more detail. As this is the final report for workpackage 5, we have included details on all components, including those which have seen few or no changes in the last period.

---

3.1 Process Designer

The Process Designer is a central tool of the ADMIRE Workbench, allowing data analysis experts to create and visualise knowledge discovery processes. The modelling environment of the tool supports direct editing of the DISPEL concrete textual syntax as well as graphical construction of the knowledge discovery processes specified in DISPEL. The tool consists of several cooperating Eclipse plug-ins that form two basic editors, the DISPEL Editor (Section 3.1.1) and the Diagram Editor (Section 3.1.2). The main development objectives of the Process Designer were to provide:

- a graphical tool that assists the users working in the DMI domain with tasks related to the design of knowledge discovery processes specified in DISPEL;
- an extensible tool and architecture that will allow third parties to integrate and extend the functionality of the tool for their applications.

The implementation of the Process Designer followed the model driven engineering approach, including a modelling phase during which an internal DMI model was developed, followed by a code generation and code customization phase. To support the design and implementation of text editing facilities, different models of concrete syntax mapping tools were considered. We chose EMFText\(^6\) Eclipse’s integrated tool for agile textual syntax development. This tool uses a concrete syntax specification language to generate a DISPEL syntax parser. Moreover, EMFText allows developers to define a plain text syntax for the Ecore-based models and to generate components for loading, editing and storing model instances.

\(^6\)See [www.emftext.org](http://www.emftext.org)
The visual construction of graphs is supported by the Eclipse Graphical Modelling Framework, which follows the model-driven approach to generating graphical editors based on a domain model. The Process Designer supplies functionality for creating DISPEL sentences; querying the ADMIRE Registry; visualisation of DISPEL sentences; and validation of DISPEL sentences.

### 3.1.1 DISPEL Editor

The DISPEL Editor supports data analysis experts in coding DISPEL sentences in concrete DISPEL syntax. The editor provides both DISPEL syntax highlighting and syntax checking. It allows validation of the knowledge discovery processes for consistency and checking of their completeness. Errors are highlighted directly in the document during editing. Data analysis experts can view the internal structure of the DISPEL sentence in the “Outline View” plug-in of Eclipse. The editor opens and saves files with the “.dispel” extension. The Process Designer allows data analysis experts to design knowledge discovery processes using existing Processing Element definitions, which are loaded from the ADMIRE Registry dynamically and in real-time. The list of available Processing Elements together with their inputs and output can be viewed in the “Registry View” plug-in, which is a required component of Process Designer for retrieval of the Processing Element definitions from the Registry.

### 3.1.2 Diagram Editor

The Diagram Editor is a tool for visual composition and manipulation of the knowledge discovery process graphs. The graphical user interface of the editor enables designing of knowledge discovery processes using a visual programming approach. This allows users to modify the graphical structure of the designed knowledge discovery process instance without breaking consistency with the DISPEL concrete syntax. This means that the Diagram Editor maintains the DISPEL syntax edited in the text editor and modifies it only when there is a update to the internal model instance. It also works the other way round – changes made in the text editor are directly reflected in the graphical representation of the designed process.

In this way, both editors can be used together to edit a single DISPEL document. Information about the graphical properties of the diagrams is stored in a separate file using “.dispel diagram” extensions. Figure 3 shows the DISPEL Development Perspective provided by the Process Designer.

### 3.1.3 Registry Client

The Process Designer uses the Registry Client interface to query the Registry and verify whether referenced Processing Elements and Type definitions exist. This is done internally by the Process Designer and its DISPEL post-processing mechanism, which ensure completeness and validity of the internal DISPEL model. There are so called DISPEL resolvers, which fix broken references and use the client to retrieve appropriate definitions from the ADMIRE Registry. The API is used also by the Registry View component to obtain list of available PEs and types as presented in Section 3.1.4. The Registry Client contacts the Registry at a URL specified by the user. For development purposes or if there is no network connectivity, it is also possible to use a local file in place of the remote Registry.

In the final project period, the Registry Client was extended to include methods for retrieving details of structural and domain types. Error handling was also improved.
Internally, the Registry Client uses the Registry Client Toolkit, a Java library which is not specific to the Workbench and allows any client to query the Registry. This library includes methods for querying, registering and deleting Structural Types, Domain Types, Processing Elements and functions. The interface and underlying framework have evolved in line with the latest updates to DISPEL and the ADMIRE framework.

3.1.4 Registry View

The Registry View now allows users to retrieve lists of available PEs and types from the ADMIRE Registry, together with their detailed descriptions. It provides an extension point which is used by the Process Designer to obtain information regarding a selected PE or type in order to be able to insert it correctly into the knowledge discovery workflow being designed. Additionally, users can sort items and search for names of registered PEs and types.
In the final period of the project, the view was redesigned in order to show clearly the inputs and outputs of PEs as well as adding support for showing structural and domain types. Figure 4 shows the new Eclipse view displaying the list of PEs, their namespaces, inputs and outputs. The view was also refactored to use the Eclipse progress monitor and job scheduling functionality to load data in the background rather than cause the UI to pause.

3.1.5 Future Work

The Registry View displays nearly all of the data available to it through the ADMIRE Registry interface. However, there are a few features that could be added, including displaying annotations and listing registered functions.

Additionally, a better way of displaying domain types should be investigated; the current implementation wastes a lot of screen estate through displaying URLs which impart little information.

3.2 Gateway Process Manager

The Gateway Process Manager (GPM) provides an interface for keeping track of running Gateway processes and forwarding the results from completed processes to other plug-ins such as visualisers.

The Gateway Process Manager was developed in order to support integration between components. By using the interfaces and extension points supplied by the Gateway Process Manager, other components can register new Gateway processes and get the output from completed processes. This simplifies the model for visualiser plug-ins, which now only need to worry about reading the output from completed processes and not about the submission...
and life-cycle of processes. Workflows are monitored within Eclipse and output is read by
sending it to the appropriate visualiser or saving to a local file, as shown in Figure 5.

Additionally, the plug-in includes a simple text visualiser for viewing the results of work-
flows. The text visualiser is launched by right-clicking on the result of a workflow and selecting
“Open in Text View”. This menu is defined as an “Extension Point”, meaning other plug-ins
can register to be called from this menu.

Writing new visualisers that use the Gateway Process Manager interface is very easy: they
only need to implement an interface containing a single method. This allows developers to
create visualisers quickly for use cases specific to their domain and data structures.

In the final project period, several user-interface improvements were made:

- the configuration menu was linked from the view;
- the error reporting view was given an overhaul to be much more usable and complete;
- a button to quickly clear all processes was added;
- the process ID of processes was added, allowing users to locate the process in other tools
  such as the Gateway Monitor described in 3.7.

These improvements were made in direct response to issues reported by actual users of
the Workbench.

3.2.1 Future Work

As results are streamed directly from the server hosting the Results PE, the data can only
be read once from the GPM – the results do not persist automatically. It would be good to
have some sort of buffering service and/or to allow results saved to file to be re-opened in the
GPM.

By analysing the SType and DType information from the connection to the Results PE,
it should be possible for the GPM to suggest the correct viewer for the result.

3.3 DMI Models Visualization

Graphical representation of knowledge discovery results in ADMIRE is performed by the spe-
cialized “DMI Models Visualization” tool which is integrated with the ADMIRE Workbench.
The visualization tool is primarily designed for data analysis experts who are able to interpret
results independently from an application domain. The current version of the tool supports
visualization of Multidimensional Clusters, Decision Tree and Neural Network models that
are described by PMML version 4.0. These models are produced as results of processing
elements deployed to the ADMIRE Platform. The visualization tool itself consists of several
Eclipse plug-ins as follows:

- org.dmg.pmml this plug-in utilizes EMF to create a data model for PMML and associ-
  ated Java code. The plug-in is based on a metamodel (Ecore) generated from XML-
  Schema for the PMML version 4.0. The Java packages contain class implementations,
  interfaces and appropriate factory code for creating instances of the PMML elements.

7 Predictive Model Markup Language, dmg.org
org.dmg.pmml.edit is a plug-in which is generated by the EMF-containing provider adapters for the PMML elements.

eu.admire.visual.pmml provides a bridge between the EMF and GEF. It provides a model-view-controller framework for graphical editors and supports integration with the Eclipse platform. This plug-in is based on the definition of the description of diagrammatic graphics structures and on the mapping model, which merges the metamodel (Ecore) with the graphical elements and the tooling definitions.

eu.admire.visual.pmml.diagram is a plug-in which includes all fundamental generated Java code of diagram with additional own parts of code for correct output representation. The plug-in visualizes the root of PMML documents and their metadata, including <DataDictionary> and <Taxonomy> elements.

eu.admire.visual.pmml.diagram.layout handles the correct layout of data mining model diagrams. Each model has its own graphical presentation of elements, which is controlled by this plug-in.

eu.admire.visual.pmml.neuralnetwork.diagram is a plug-in visualising the neural network model as defined by the PMML standard version 4.0. The plug-in implements a sub-diagram related to the <NeuralNetwork> element.

eu.admire.visual.pmml.treemodel.diagram is a plug-in visualising the tree model as defined by PMML. The plug-in implements a sub-diagram related to the <TreeModel> element.

eu.admire.visual.pmml.radviz.diagram is a plug-in visualising multidimensional clusters. The plug-in implements a sub-diagram related to the <DataDictionary> element extended by the point coordinates as shown in Section 3.3.1.

The PMML format may be used to describe several different data mining models such as tree models, neural networks, clustering models, regression models, general regression models, naïve Bayes models, association models and sequence models. A PMML document may contain more than one model and additionally it can carry the initial metadata before the model is computed. The XML-Schema of PMML provides a simple extension mechanism. Using the <Extension> element, the content of any PMML document can be extended to meet the specific requirements of a data mining application. This element supports name-value pairs that allow for the insertion of application-specific information. This extension mechanism is also used by the RadVizPMML processing element, which stores coordinates of attributes and classes of the target attribute. In the following sections we describe the data mining tasks implemented in the ADMIRE Project and are supported by the visualization tool.

3.3.1 Radial Coordinates Visualization

A typical data mining approach for data exploration is identification of characteristics clusters in which the significant part of data appears or which exhibit some properties that are different from those observed in the remaining part of the considered dataset. Such data can be easily identified when it is possible to see position of the considered data in a space. This is simple
if data is described by 2 or 3 properties. RadViz method has been designed to map data described by more attributes (multidimensional data) to a planar picture. In ADMIRE, the RadViz method is used to visualize clusters of data in the dataset produced as result of an integration process. The definition of the RadViz method as a processing element in DISPEL is done as follows.

```
Type RadVizPMML is PE (  
  <Connection data, classIndex,  
  columnIndices, nominalValues> =>  
  <Connection pmml>  
);  
```

The above DISPEL code defines the RadVizPMML processing element with four input parameters (data, classIndex, columnIndices and nominalValues) and one output parameter (pmml). As a result the PE produces a PMML document that can be read on the output parameter named pmml. The PMML specification does not support description of individual data point coordinates for selected attributes. Therefore, these coordinates are defined in the outputting PMML document as extensions of the `<DataDictionary>` and `<DataField>` elements. An example of this extension element follows.

```
<DataDictionary>
```
3.3.2 Classification Model Visualization

Classification is a common data mining task used for prediction or direct classification of data records. In one of the ADMIRE application scenarios, classification is used to predict river water levels for a given water station. The classifier is produced by the Processing Element BuildClassifierLinearRegression and is then used by the ClassifierPMML processing element, which produces PMML. Its definition in DISPEL is as follows.
This Processing Element has one input named classifier and has one output named pmml. The classifier is an object produced by the preceding processing element (BuildClassifier-LinearRegression) and the output is a PMML document containing one <TreeModel> element conforming to the PMML standard. Additionally, a classifier can be expressed in the terms of a neural network, which has emerged as an important method for classification. A processing element producing PMML document containing the <NeuralNetwork> element is deployed to the ADMIRE Platform. Its DISPEL definition, including inputs and outputs is:

```
Type ClassifierPMML is PE {
    <Connection classifier> =>
    <Connection pmml>);
```

Figures 7 and 8 show screenshots of the DMI Models Visualization plug-ins presenting examples of the decision tree model and neural network model respectively.

```
3.4 Table Visualiser

The Table Visualiser tool is a simple way of displaying tuples of data. The visualiser displays results which are organised in the WebRowSet XML format. As with several other tools, the Table Visualiser interacts with the Gateway Process Manager. If installed, the “Show in Table Visualiser” option is displayed when the user right clicks on the results of completed processes in the Gateway Process Manager. The visualiser is normally used to investigate data prior to further processing.

In the future, some advanced features could be added to the tool. For now, the main functionality of the visualiser is to sort displayed data using column values. It would be also possible to provide filtering capabilities.

![Figure 9: Table Visualiser](image)

3.5 Chart Visualiser

The Chart Visualiser, or CV, is an Eclipse plug-in that can be used to visualise row sets as charts. The CV can produce various kinds of charts, including pie charts, scatter plots and histograms. During the last development period, the main update to the tool has been to add support for visualisation of data streams in real time (i.e., before processing is finished). This means that now the data analyst is able to keep track of the data flowing between two PEs in the workflow process: the chart is updated periodically, every few milliseconds. To exploit the CV in such a way, the process has to be extended with another Results PE in an appropriate place, as shown in Figure 10. In this example, the analyst wants to see a chart representing values of the age attribute flowing between PEs SQLQuery and PE1. With this improvement, results are displayed immediately, with no need to wait for the process to finish.

An additional update was to add a dialog to allow the user to select a column (or columns) and chart type dynamically. With this improvement, the workflow extension given in Figure
can be simplified through removal of the projection PE, because the user can now select the data to visualise directly in the tool. The dialog also helps the user to select an appropriate chart depending on column type. A single, numeric column can be displayed as a pie chart or histogram with a specified number of groups – values appropriate for the group range are counted. For a single nominal value (i.e., string), a bar or pie chart can be created. In this case, there is no grouping, but exact column values are used as group names. The example given in Figure 11 shows this, where a single string column containing product names is being visualised. The third option is to display scatter plot to show the correlation between two numeric values.

Figure 10: Extension of the process for real time stream visualisation in CV

Figure 11: Chart Visualiser

3.6 Semantic Knowledge Sharing Assistant

The SKSA allows users to add and share human-understandable knowledge in the context of the ADMIRE ontologies. Based on the ontological description of the active platform and application components the SKSA can detect the context in which the user is working, and attach new knowledge in human-readable form to that context. Also, any previously recorded knowledge pertinent to the user’s current context is available to the user, thus making sharing
and re-use of knowledge and information easy for non-IT experts. The tool is integrated into the ADMIRE Workbench (through the mechanism of the Eclipse plug-in) and the ADMIRE Portal. Figure 12 shows a screenshot of the Eclipse plugin version of the tool integrated with the Workbench and Figure 13 shows the web-based SKSA version integrated with the portal. The SKSA internals and approach has been described in detail in previous deliverables D5.2 [5], D5.3 [6] and D5.4 [7].

3.6.1 Recent Developments

In the reporting period covered by this document we achieved the following improvements:

- The web-based version of the SKSA GUI was developed (see Figure 13);
- The SKSA was integrated with the ADMIRE Repository and ADMIRE Registry ontologies;
- Addition of full-text search functionality for user context search;
- The SKSA was integrated with the ADMIRE portal.

The SKSA is composed of the SKSA GUI and the SKSA Core. It connects to the ADMIRE Registry and Repository as seen in Figure 14. Both the SKSA Eclipse plug-in and the Web based SKSA GUI can now connect to the ADMIRE Repository and ADMIRE-SKSA Registry. Thus any DISPEL files shared by the SKSA can be accessed by any other SKSA instance from the Workbench or portal. The ADMIRE-SKSA Registry extends the ADMIRE Registry with SKSA related ontologies and domain specific ontologies. This way shared DISPEL files (or other Repository files) can be described by ADMIRE ontologies (e.g., related PEs) as well as domain dependent ontologies (e.g., locations in flood-related use cases). The ADMIRE-SKSA Registry is part of the SKSA Core, as can be seen in Figure 14 which shows the architecture of
the SKSA. The internals of the SKSA were extended in this period to support full-text search in order to better determine user context from text queries. We have indexed the underlying Registry ontologies and ontology instances using the Apache Lucene\footnote{http://lucene.apache.org/} indexing library. Now users can enter text queries and the 10 most relevant ontology items are returned as user context. For this context the SKSA returns the most appropriate shared DISPEL files. In previous periods we have integrated the SKSA Eclipse plug-in with the ADMIRE Workbench; in the final reporting period we have developed and integrated the web-based SKSA GUI with the ADMIRE Portal Framework. In both the Workbench and Portal, users can search for DISPEL documents, open them, edit them and submit them to the Gateway.

### 3.6.2 Conclusion and Future Work

The SKSA is a support tool which can be used to share and contextually describe Repository items. We have focused on sharing DISPEL files, but the SKSA can easily be extended to share other resources such as trained models, results or other resources. The SKSA allows for the reuse of DISPEL documents in a given user context. Application users or even DISPEL developers can simply search for the most appropriate DISPEL available in the Repository to reuse it. The SKSA is integrated with (and part of) the ADMIRE Workbench and ADMIRE Portal, but in the future it can also be reused in any other application where context based sharing functionality of resources is needed. Its architecture is modular (see Figure 14) containing a reusable GUI and core components which are customizable through application
ontologies and various integration points.

3.7 Gateway Monitor

The Gateway Monitor uses data published by the ADMIRE Gateways (including the performance optimisation data) to provide an overview of the current processes executing on Gateways and the ability to drill down into a graph view of given processes.

The Gateway Monitor was not created as a Workbench plug-in as first envisaged, but instead as a stand-alone Web Application which can be used directly by data-intensive engineers or developers without the need to have access to a running Workbench.

Figure 15 shows the processes view. We can see the processes executing on the Gateways in real-time (completed processes are shown in green, running ones in yellow and processes which have failed are shown in red). We can drill down and get more information on a given process, including any error messages, the original DISPEL request and the start time. The processes view is updated continuously, making it possible to watch how a submitted DISPEL workflow is processed across Gateways.

From the processes view, we can also get to the server-side workflow(s) for the process. Opening a workflow leads to visualisations such as shown in Figure 16.

Here we can see a visualisation of the workflow graph, annotated with information on data flowing through pipes. By using the time-slider, we can see the status of the workflow at any stage in time. The play button allows the user to automatically advance the workflow in time and see how the data flows through the workflow and where and when it blocks. Currently running processes can also be viewed in near real-time, allowing users to watch the progress of their workflows. This provides data-intensive engineers with a powerful tool for identifying bottlenecks and suboptimal code in workflows. Workflows which return an error or an unexpected result can also be analysed to watch the data-flow leading up to the issue.
This form of visualisation can provide greater understanding of workflows and algorithms as well as being a useful demonstration tool.

The workflow graph visualisation uses data from the performance database that is described in D2.9 [3]. These data are exposed via a REST interface and provides information on the amount of data passed on each pipeline for the requested time period.

### 3.7.1 Future Work

There are several user interface features that could be improved:

- Using the time controls on the workflow could be more intuitive;
- The application could be more visually appealing, in particular the workflow view;
- The zoom function could allow selection of the region to be magnified;
- The text for reads/writes on a pipe sometimes overlaps.

Monitoring of the ADMIRE Registry could also be added to the processes view.

The longer-term future for this sort of tool is potentially very bright. By adding more diagnostic information and allowing more ways to analyse and visualise it, engineers will have a very powerful tool for optimising and fixing workflows.
3.8 Validation Preview

The Validation Preview allows users to ensure that their workflows are valid with regard to their chosen Gateway. The Gateway responds with a graph visualisation for valid workflows, as shown in Figure 17.

This allows users quickly to identify errors and provides them with an intuitive graph visualisation of their workflow, which can be easier to follow and reason about than raw DISPEL code.
4 The ADMIRE Portal Framework

The ADMIRE Portal represents one of the main focuses for the last period of Workpackage 5. Individual application portals provide high-level, domain specific interfaces to the ADMIRE infrastructure. The interfaces are designed to be used by a domain expert end-user, for example a call-centre manager, rather than a data analysis expert (for whom we have developed the ADMIRE Workbench). As such the individual portals abstract away many of the advanced features of the Workbench, presenting users with a few simple options that meet their expectations and understanding.

In the previous reporting period, we reassessed our choice of portal technology and made the decision to use the Google Web Toolkit (GWT). GWT-based portals provide better, more dynamic user experiences thanks to the employment of Javascript, which provides a more responsive user experience. From the portal developer’s point of view, GWT allows for faster development compared to older portal technologies through its use of the Java language for both server-side and client-side (i.e. browser) programming, the existence of a lot of predefined components, and tight integration with leading Java development tools. GWT shields developers from many (but not all) cross-browser compatibility issues, again reducing development time. Portals are domain specific applications, meaning that there is no generic Portal application for the overall project. Portals have been developed directly for the use cases to fulfill specific requirements and are described in detail in D6.6.

This section contains specification of all requirements and functionality common to all existing and future Portals. We also describe the ADMIRE myExperiment portal, which was not designed for a specific use case and builds on top of the myExperiment virtual research environment.
4.1 Knowledge Exchange through the Repository

One of the primary purposes of a Portal is to provide a simple and user-friendly method of accessing the results of data-mining (as well as running new data mining tasks). This requires the storage and sharing of data, a function supported by the ADMIRE Repository. Results can be added to the Repository either via Processing Elements executed in a DISPEL workflow or through the Portal using the Repository’s REST Web interface. The details of the User Interface to the Repository are dependent on the Portal in question, but all Portals use the same underlying framework to access the Repository.

Typical data-mining results include classification models, association rules and regression models. Often these results will be created by a data analysis expert using the Workbench and retrieved by the end-user via the Portal. A single result is likely to be made up of several files, for example a binary representation alongside a PMML representation or image files holding diagrammatic representations.

In some cases, workflows will produce successive versions of classifiers as the workflow progresses. This means that users can access preliminary results of long-running processes before the overall workflow is finished. If the preliminary results show that the classification is not going to be useful, the process can be cancelled early.

A simplified schema of showing the interaction of domain experts and data analysis experts with the ADMIRE framework and portals is depicted in Figure 18.

![Figure 18: Interaction of Experts with ADMIRE Framework and Portals](image)

4.2 Creating process descriptions

Portals are not just passive results browser. Domain experts can execute customized knowledge discovery processes from the Portal application. In most cases, these would use existing
trained models\(^\text{10}\) (classification or regression, for instance) to predict values on test data loaded from external files or resources. A domain expert does not build the entire workflow from scratch, but merely changes some parameters via web forms. The Portal then loads a core DISPEL file, replaces parameter values accordingly and produces a process description that is ready to use, all behind the scenes of the Portal. For this purpose, exploiting DISPEL functions became crucial: major workflow changes can be brought about by changing only a few function parameters, and no DISPEL-to-Java mapping is necessary. As an example, consider a scenario where a domain expert wishes to select multiple classification models to classify test data. The domain expert selects one or more classifiers from the Repository and assigns them weights. The function that receives these parameters, then passes the correct binary classifiers to the appropriate PE instances is shown below.

```java
PE(<Connection testData> => <Connection classifiedData>) classifyTestData(
        String[] modelNames, Connection priorities, String repoHost) {

    MultiClassify classify = new MultiClassify;

    for (Integer i = 0; i < modelNames.length; i++) {
        Deserialiser dsr = new Deserialiser;
        ObtainFromRepository obtain = new ObtainFromRepository;
        obtain.filename = modelNames[i];
        obtain.data => dsr.data;
        repoHost => obtain.host;
        dsr.result => classify.classifier[i];
    }

    priorities => classify.priorities;
    return PE(<Connection testData=classify.data> =>
               <Connection classifiedData=classify.result>);  
}
```

To get a concrete process description, the Portal simply has to replace the values of the `modelNames`, `priorities` and `repoHost` variables in the function invocation. In previous versions of the Portal, we used EMF\(^\text{11}\) to build processes, which was much more complicated. This version leverages the power of DISPEL functions to good effect.

### 4.3 Process execution

All ADMIRE application portals use a common job execution and management infrastructure to execute any operations. Every operation is expressed as a sequence of DISPEL statements, which are submitted to the ADMIRE Gateway as a single job. The details of the infrastructure are hidden behind the Portal so the user does not need to worry about them. Jobs (also called Gateway processes) are listed on a dedicated page of the application portal (see Figure 19 for an example). Each job has an internal ID, a name, status, execution time and a link pointing to results or error description in case the job execution fails. Status codes are the most important information and are therefore distinguished by colour.

---

\(^{10}\)The Portal architecture does not restrict the purpose of the application. It is also possible, to wrap training processes in the Portal interface

\(^{11}\)Eclipse Modelling Framework
Once a job has finished execution – indicated by the “COMPLETED” status, a user can browse the results in an application-specific way. If the job fails, an error description link points to a text box containing error message, which falls into one of the four categories: compile error, registration error, runtime error or system error. The level of detail of individual error messages depends on the sub-system that produced it. Error messages may contain details exposing the back-end ADMIRE infrastructure to the user in order to give them a chance to better identify the reason and source of the error.

In order to support more advanced users, the portal offers the possibility of submitting any DISPEL sentence (example shown in figure 20). Once submitted, such a job is added to the list of jobs on the Jobs portal page.
4.4 ADMIRE myExperiment

The ADMIRE myExperiment is a Web portal based on the myExperiment portal. With this Portal we provide to the users a way to share Processing Elements through the ADMIRE Registry and collaborate in the development of DISPEL workflows. The portal reuses the code from myExperiment and adapts it to access the ADMIRE Registry, being able to query the existing Processing Elements along with the types and functions information. The portal provides visualisation methods for all the elements in the Registry and a way for annotating those elements as well.

The ADMIRE myExperiment works as a social platform for users to interact with the existing elements in the Registry. It is possible for users to add comments to the Processing Elements or types in the registry, create groups of users, access different versions of each PE, review them, etc. In essence it offers all of the functionality provided by myExperiment but for ADMIRE and the development of DISPEL workflows. In Figure 21 we show the main visualisation screen for a list of PEs.

---

12 [www.myexperiment.org/](http://www.myexperiment.org/)
5 Roadmap for Further Development

In section 3 of this document, several Workbench component descriptions contained “Future Work” sections which described short to medium term changes that could be made to improve the component. This section focuses on the bigger picture; what features are missing that would turn the Workbench into a truly revolutionary tool for data intensive computing in general?

The following list represents a range of ideas, from blue-sky thinking that would take a major effort to realise (as well as requiring significant changes to other parts of the architecture), to smaller ideas that would be implementable in a reasonable time-frame but would still provide major benefits to users. That some of these ideas didn’t make it into the Workbench release is a shame, but without creating what we did, we may not have even been able to see their possibility.

Support for creation of new components. The Workbench should provide help for users to create new PEs and other components. For example, a wizard plug-in for creating PEs could ask the user for the types of inputs and outputs then create a Java class with the appropriate stub code. Wizards could also help with automatic wrapping of existing libraries. The Workbench should also support the creation of detailed documentation describing inputs, outputs and usage etc. This support is important not only for creating PEs, but also new types and functions.

Make data sources (and data sets) first class entities. Data sources and sets should be listed in the Registry view from where users can drag-and-drop them into the Process Designer. Various plug-ins should be available for investigating and integrating data sources e.g. displaying the available tables and columns in a database and automatically generating SQLQuery PEs for retrieving selected columns. Supported data sources should cover a wide range of possibilities, including CSV files, relational databases and RDF graphs.

Suggest possible connections for PEs and functions. By proper use of the type system, it should be easy to see which objects in the current context can be connected to a component or to list which components could be created that would match the connection. This could be taken even further for some components, such as suggesting possible column names for PEs handling WebRowSet data.

Improve testing and debugging. Better and more complete error messages should be returned to the user, with potential fixes suggested. More tools should be available for analysing workflows. Possible areas of locking and buffering should be highlighted by the Workbench.

Better visualisation of data and results. The Workbench should support investigating the raw data coming from data sources or intermediate PEs as well as the final result returned to the user. It should be easy to create graphs from data in much the same way as Microsoft Excel. The correct visualiser for data results should be selected automatically.

More “in-execution” tools. The current Workflow Visualiser described in section 3.7 gives an indication of how useful it is to see the real-time flow of data through a workflow.
This could be extended even further to allow pausing of data flows, injection of new data or redirection of flows. More tools could be added to visualise the actual data flowing through pipes rather than just the amount of data.

**More methods for discovering resources.** There are several sorts of resources important to ADMIRE, including types, functions, PEs and data sources. These should be easily discoverable by search, index or automatic suggestion. For example, appropriate resources could be suggested based on the context of the DISPEL document (e.g., types and data sources relevant to genomics when working on a genomics workflow). The SKSA has done a lot of work in this direction, but it can still be taken a lot further.

**Complete and detailed documentation of all resources.** This should be available easily to the developer, in a similar way to accessing Javadoc documentation for Java classes in Eclipse. It should also be possible to find code examples.

**Refactoring support and code suggestions.** The Workbench should identify incorrect or non-idiomatic use of DISPEL and suggest improvements. There should be a library of common patterns that developers can refer to, encapsulating solutions for problems such as developing n-fold cross validation workflows.

**More feedback from the Gateway.** At the moment, the Gateway is free to reorder and optimise workflows and to automatically insert required converters, and it is possible to envisage Gateways making even greater changes in the quest for efficiency, robustness and usability. However, at the minute these changes aren’t reported to the user, who may become confused about what exactly is happening to his/her workflow. These changes should be reported back to the Workbench in a human-readable manner.

**Support for versioning components.** There is no support for having multiple versions of a component currently, whether this means new and old implementations or implementations with different characteristics. This needs to be addressed in a way that empowers users to choose the most relevant implementation for their needs (or to allow the Gateway to chose). For example a user may want to choose a known stable version of a component, or to use the latest version with speed improvements. Alternatively they may want to use (or allow the Gateway to choose) a version certain characteristics such as low memory overhead or guaranteed response time.

**Bridging of communities.** ADMIRE needs to allow users to integrate tools from their domain e.g., LAPACK and R for mathematicians and statisticians. This overlaps with the first point describing the need to support automatically wrapping libraries, but also goes further than this; ADMIRE should not expect users to abandon their existing tools and ways of working but rather find ways to integrate and improve them with ADMIRE. Frameworks such as ADMIRE will live or die based on the availability of familiar and high-quality libraries for their users.
References


A List of Workbench Plug-ins

Table 1 lists the Workbench plug-ins by component name, project name and developer initials. The developer initials correspond to the following people:

IJ: Ivan Janciak, The University of Vienna
AM: Adrian Mouat, EPCC, The University of Edinburgh
RO: Radek Ostrowski, EPCC, The University of Edinburgh
AK: Amy Krause, EPCC, The University of Edinburgh
CB: Carlos Buil Aranda, Technical University of Madrid
MJ: Maciej Jarka, Comarch
BS: Branislav Simo, Slovak Academy of Sciences
ML: Michal Laclavik, Slovak Academy of Sciences

In addition, the Gateway Monitor described in Section 3.2 was developed by Adrian Mouat, EPCC, The University of Edinburgh.
<table>
<thead>
<tr>
<th>Name</th>
<th>Eclipse Project</th>
<th>Devel.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core DISPEL Framework</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPEL Model</td>
<td>eu.admire.dispel</td>
<td>IJ</td>
</tr>
<tr>
<td>DISPEL Edit</td>
<td>eu.admire.dispel.edit</td>
<td>IJ</td>
</tr>
<tr>
<td>DISPEL Editor</td>
<td>eu.admire.dispel.editor</td>
<td>IJ</td>
</tr>
<tr>
<td>DISPEL Eclipse Resource</td>
<td>eu.admire.dispel.resource.dispel</td>
<td>IJ</td>
</tr>
<tr>
<td>DISPEL Eclipse Resource UI</td>
<td>eu.admire.dispel.resource.dispel.ui</td>
<td>IJ</td>
</tr>
<tr>
<td>DISPEL Eclipse Resource LATEX</td>
<td>eu.admire.dispel.resource.dispel.tex</td>
<td>IJ</td>
</tr>
<tr>
<td>DISPEL Parser</td>
<td>org.emftext.commons.antlr3_3_0</td>
<td>IJ</td>
</tr>
<tr>
<td>Workbench Perspective</td>
<td>eu.admire.workbench.perspective</td>
<td>AM</td>
</tr>
<tr>
<td>Workbench Wizards</td>
<td>eu.admire.workbench.wizards</td>
<td>IJ</td>
</tr>
<tr>
<td><strong>Process Optimization Framework</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMI Model</td>
<td>eu.admire.model.dmi</td>
<td>IJ</td>
</tr>
<tr>
<td>DMI Model Transformations</td>
<td>eu.admire.dispel.m2m</td>
<td>IJ</td>
</tr>
<tr>
<td>DMI Process Optimization Engine</td>
<td>eu.admire.poe</td>
<td>IJ</td>
</tr>
<tr>
<td><strong>Gateway Client</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gateway Client Toolkit</td>
<td>GatewayClientToolkit</td>
<td>AM, RO, AK</td>
</tr>
<tr>
<td><strong>Registry Client</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registry Client Toolkit</td>
<td>AdmireRegistryClientToolkit</td>
<td>CB, AK</td>
</tr>
<tr>
<td>Registry Client Utils</td>
<td>eu.admire.workbench.registry.client</td>
<td>IJ, AM</td>
</tr>
<tr>
<td><strong>Process Designer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Designer</td>
<td>eu.admire.workbench.pd</td>
<td>IJ</td>
</tr>
<tr>
<td>Process Designer Packages</td>
<td>eu.admire.workbench.pd.packages</td>
<td>IJ</td>
</tr>
<tr>
<td>Process Designer Methods</td>
<td>eu.admire.workbench.pd.methods</td>
<td>IJ</td>
</tr>
<tr>
<td>Process Designer Utils</td>
<td>eu.admire.workbench.utils</td>
<td>IJ</td>
</tr>
<tr>
<td><strong>Registry Viewer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registry Client UI</td>
<td>eu.admire.workbench.registry.view</td>
<td>IJ, AM</td>
</tr>
<tr>
<td><strong>Gateway Process Manager</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gateway Process Manager</td>
<td>eu.admire.workbench.gateway.gatewayprocessmanager</td>
<td>AM</td>
</tr>
<tr>
<td><strong>Result Visualisers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table Visualiser</td>
<td>eu.admire.workbench.tablevisualiser</td>
<td>MJ</td>
</tr>
<tr>
<td>Dot Visualiser</td>
<td>eu.admire.workbench.dotvisualiser</td>
<td>MJ</td>
</tr>
<tr>
<td>Chart Visualiser</td>
<td>eu.admire.workbench.cv</td>
<td>MJ</td>
</tr>
<tr>
<td>Text Viewer</td>
<td>eu.admire.workbench.resulttextviewer</td>
<td>AM</td>
</tr>
<tr>
<td>DOT Tree Visualisation</td>
<td>eu.admire.workbench.dotvisualiser</td>
<td>MJ</td>
</tr>
<tr>
<td>PMML Visualisation Models</td>
<td>eu.admire.visual.pmml</td>
<td>IJ, ML</td>
</tr>
<tr>
<td>PMML Basic Visualisation Diagrams</td>
<td>eu.admire.visual.pmml.diagram</td>
<td>IJ, ML</td>
</tr>
<tr>
<td>PMML Visualisation Layouts</td>
<td>eu.admire.visual.pmml.layout</td>
<td>IJ, ML</td>
</tr>
<tr>
<td>PMML NN Visualisation Diagram</td>
<td>eu.admire.visual.pmml.neuralnetwork.diagram</td>
<td>IJ, ML</td>
</tr>
<tr>
<td>PMML Tree Visualisation Diagram</td>
<td>eu.admire.visual.pmml.treemodel.diagram</td>
<td>IJ, ML</td>
</tr>
<tr>
<td>PMML Radial Visualization Diagram</td>
<td>eu.admire.visual.pmml.radviz.diagram</td>
<td>IJ, ML</td>
</tr>
<tr>
<td>PMML 4.0 Model</td>
<td>org.dmg.pmml</td>
<td>IJ</td>
</tr>
<tr>
<td>PMML 4.0 Edit</td>
<td>org.dmg.pmml.edit</td>
<td>IJ</td>
</tr>
<tr>
<td><strong>SKSA</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic Assistant</td>
<td>eu.admire.workbench.sksa</td>
<td>BS, ML</td>
</tr>
<tr>
<td><strong>Workflow Visualiser</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workflow Preview</td>
<td>eu.admire.workbench.preview</td>
<td>IJ, AM</td>
</tr>
</tbody>
</table>

Table 1: List of ADMIRE Workbench Plug-ins
B Workbench Installation Instructions and Process Designer Quickstart Guide

The following section contains installation instructions for the workbench and a quickstart guide for the Process Designer.
ADMIRE Workbench Installation
and Process Designer Quickstart Guide

Updated 03/05/11

The ADMIRE Project

www.admire-project.eu

Funded by the European Commission
(Framework 7 ICT 215024)
1 Introduction

In this document we introduce the ADMIRE Workbench as a tool for data analysis experts. In more detail we describe: a) the installation procedure; b) the main features and of the Process Designer how to use them and; c) a concrete example showing how to create and visualise a simple process in DISPEL.

2 Requirements

The ADMIRE Workbench is fully implemented in Java and is based on several third-party tools and libraries. The minimal requirements are:

- Java Runtime Environment 1.6+
- Eclipse 3.6 (Helios) including:

3 Installation


Download the Eclipse Modelling Tools (includes Incubating components) package for your operating system and unzip it to a local folder.

The easiest way to install the latest release of the Workbench designer is to use the Eclipse Update Manager functionality inside Eclipse. The manager downloads and installs all of the required plug-ins and resolves dependencies during the installation process automatically. To use the manager start Eclipse and follow these instructions:

From Eclipse’s menu select ‘Help’→‘Install new Software...’→‘Add repository’, and point it to the ADMIRE Update site at: [http://www.admire-project.eu/ADMIREUpdateSite/](http://www.admire-project.eu/ADMIREUpdateSite/)

Select all the available plug-ins and accept the license conditions. Finish the installation procedure with Eclipse restart.

4 Configuration

After restart of Eclipse open ‘DISPEL Development’ perspective from Eclipse’s menu under ‘Window’→‘Open perspective’→‘Other’ and configure the ADMIRE service clients as described in the next sections.
4.1 Gateway Client

From the ‘Window’→‘Preferences’ select ‘Gateway Client’ and point the ‘Gateway URL’ to the address of your local Gateway, as shown in Figure 23.

4.2 Registry Client

From the ‘Window’→‘Preferences’ select ‘Registry Client’ and point the ‘Registry URL’ to: to the address of your project’s Registry as shown in Figure 24.

Note that a file can be used instead of a remote Registry.

5 Using the Process Designer

The Process Designer is implemented as an Eclipse plug-in so for details on using the Eclipse Platform, refer to the “Workbench User Guide” at http://www.eclipse.org/documentation/. In the following sections we present the basic features of the Process Designer for creating DISPEL workflows.
5.1 Setting the DISPEL Development Perspective

In the Eclipse Platform a ‘Perspective’ determines the visible actions and views within a window. To open the appropriate perspective from Eclipse’s menu select ‘Window’→‘Open Perspective’→‘Other’ and select ‘DISPEL Development’. This action will open all the required plug-ins within the Eclipse window.
5.2 Using the Project Wizard

The ADMIRE Project Wizard helps users to create new ADMIRE projects. The wizard creates an initial structure of the project containing a set of examples and additional files for DISPEL development. Create a new ADMIRE project by selecting ‘File’ → ‘New’ → ‘Other’ → ‘ADMIRE’ → ‘ADMIRE Project’. Give it a name and provide a location for the new project.

![ADMIRE Project Wizard](image)

Figure 26: ADMIRE Project Wizard

5.3 Working with the Registry View

The Registry View plug-in lists all the definitions from the related ADMIRE Registry. Sorted-enabled listing allows its users to search, select and present definitions of: 1) Processing Elements, 2) Domain Types and 3) Structural Types. The description of processing elements contains information about input and output connections and their types. A selected processing element can be inserted to the Diagram Editor using drag and drop feature – this is not enabled in the DISPEL Editor. Note that only the processing elements available in the registry can be imported in the process being designed, otherwise the DISPEL Editor will set an error flag. By default, the Registry View view is available in the DISPEL Development perspective but it can also be opened from Eclipse menu ‘Window’ → ‘Show View’ → ‘Other’ → ‘ADMIRE’ → ‘Registry View’.

5.4 Creating a New DISPEL File

Each DISPEL sentence to be sent to the ADMIRE Gateway for enactment must be in a file with “.dispel” extension. To create such a file, as a part of ADMIRE Project, right click in the ‘Project Explorer’ view and select ‘ADMIRE dispel’ wizard from ‘New’ → ‘File’ → ‘ADMIRE’. Give a name to the file and make sure that its extension is “.dispel”. This wizard will create the new file in the selected project with the given name and with initial content containing a package name (i.e. eu.admire) for the new DISPEL sentence.
5.5 Browsing the Structure of DISPEL Sentences

The Process Designer also exposes the internal structure of a DISPEL sentence using the ‘Outline View’. The structure of the sentence corresponds to the concrete syntax tree, which enables data analysis experts to visually understand the content of the DISPEL sentence and navigate through its structure. Figure 29 shows the structure of a DISPEL sentence in the Eclipse view.

5.6 Validation of a DISPEL Sentence on the ADMIRE Gateway

Any DISPEL sentence opened in the DISPEL Editor can be submitted to the ADMIRE Gateway for validation. If there are no syntactic errors in the sentence a graph showing an executable workflow can be visualised using the ‘Preview View’ as shown in Figure 31. Note that the visualised graph might be different from the one created in the Diagram Editor because the
Figure 29: Outline of DISPEL sentence structure validated one is optimised for execution on a specific ADMIRE platform.

Figure 30: Initialisation of the Diagram Editor

Figure 31: Workflow Preview
5.7 Generating \LaTeX\ from a DISPEL Sentence

A tool for generating \LaTeX\ code from a DISPEL is available by clicking \(\text{button}\) on the DISPEL Editor toolbar. This is helpful for inserting nicely formatted DISPEL code in publications.

![Figure 32: \LaTeX\ generated from DISPEL](image)

5.8 Initialising Diagram Editor

Visualisation of an existing DISPEL sentence is done in the Process Designer by invoking diagram initialization action by right-click on “.dispel” file in the ‘Project Explorer View’ and selecting ‘Initialise dispel_diagram file’ as shown in Figure 30. This action creates a new file with the same name as the original file but with a “.dispel_diagram” extension. This will activate the Diagram Editor which will present a graph corresponding to the process specified in DISPEL.

5.9 Reducing Elements in the Diagram Editor

A typical DISPEL graph contains many processing elements, connections, variables and literal values. In order to simplify the graph, the variable and literal elements can be hidden by clicking \(\text{button}\) on the DISPEL Diagram toolbar. To renew the graph click on \(\text{button}\).
6 Graphical Notation

DISPEL is not designed to encode data mining algorithms but rather to compose graphs consisting of various processing elements, which implement those algorithms. To visualise these graphs, we specified a graphical notation for the processes described in DISPEL. The graphical notation is used by the Process Designer to represent the DMI processes in a visually readable form that supports comprehension of the processes and facilitates the design phase of the preparation of DISPEL sentences before their enactment. The graphical objects representing the main elements of the DISPEL graphical notation are presented in Table 2.

<table>
<thead>
<tr>
<th>Graphical Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Processing Element" /></td>
<td><strong>Processing Element</strong>: Multiple instances of a PE may be used in one process. Each instance represents a node in the graph depicted using this graphical object labelled with the name of the PE and having containers for the lists of input and output connectors. The graphical element shows also name of a variable referring to the instance.</td>
</tr>
<tr>
<td><img src="image2" alt="Variable" /></td>
<td><strong>Variable</strong>: This graphical object represents a variable defined in the process and labelled with its name and type.</td>
</tr>
<tr>
<td><img src="image3" alt="Literal" /></td>
<td><strong>Literal</strong>: Denotes a primitive value used as input to a PE. Optionally each literal value can be labelled with its type name.</td>
</tr>
<tr>
<td><img src="image4" alt="Function" /></td>
<td><strong>Function</strong>: Encapsulates a data flow within a DISPEL function. Double-clicking on this graphical element opens the graph contained in the function.</td>
</tr>
</tbody>
</table>

7 Final Walk Through

In the following section we take you step-by-step through the process of building and visualising a DISPEL sentence. Before we start make sure that you have successfully installed and configured the Workbench as described in Section 1.
7.1 Hello World Example

This is a very simple example showing the basic features of Process Designer described above. Follow these steps:

1. Create a new DISPEL file as described in Section 5.4.

2. Copy and paste the following DISPEL code to the editor as shown in Figure 33.

```java
package eu.admire {
    use uk.org.ogsadai.Echo;
    use eu.admire.Results;
    PE(<> => <Connection out>) HelloWorldFunction(String world)
    {
        Echo echo = new Echo;
        String hello = "Hello " + world;
        hello => echo.input ;
        return PE(<> => <Connection out=echo.output>);
    }
    PE(<> => <Connection out>) HelloWorld = HelloWorldFunction("World");
    HelloWorld helloWorld = new HelloWorld;
    Results result = new Results;
    "output" => result.name;
    helloWorld.out => result.input;
    submit result;
}
```

4. Initialise a diagram file as described in Section 5.8. The Diagram Editor opens the visualisation as shown in 34.

5. Double click on the package element and then the Diagram Editor opens the visualisation of the corresponding graph as shown in 35.
Figure 33: Hello World Example in the DISPEL Editor
Figure 34: Visualisation of Packages in the Diagram Editor

Figure 35: Visualisation of the Hello World Example in the Diagram Editor