

CADDSuite and Generic Knime Nodes

Marc Röttig

Applied Bioinformatics Group

Center for Bioinformatics Tübingen (ZBIT)

10/04/2011

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

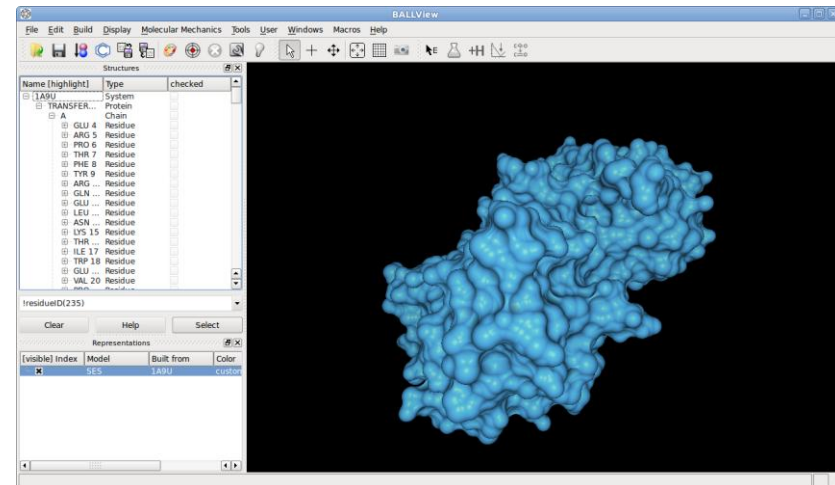




- Applied Bioinformatics group (ABI)
 - Founded in 2003 by Prof. Oliver Kohlbacher
 - Currently we have 15 researchers
 - Expertise in :
 - Proteomics/Metabolomics (OpenMS project)
 - Drug design (CADD Suite project)
 - Molecular modelling (BALL project)
 - Sequence analysis (SeqAn, joint work with FU Berlin)
 - Systems biology (UniPAX project)
 - Immunoinformatics



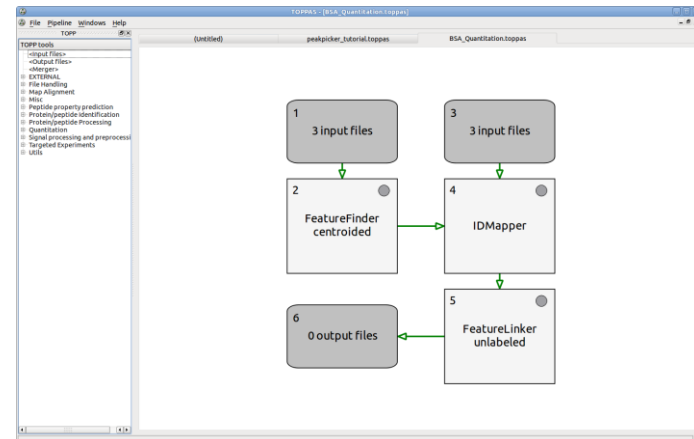
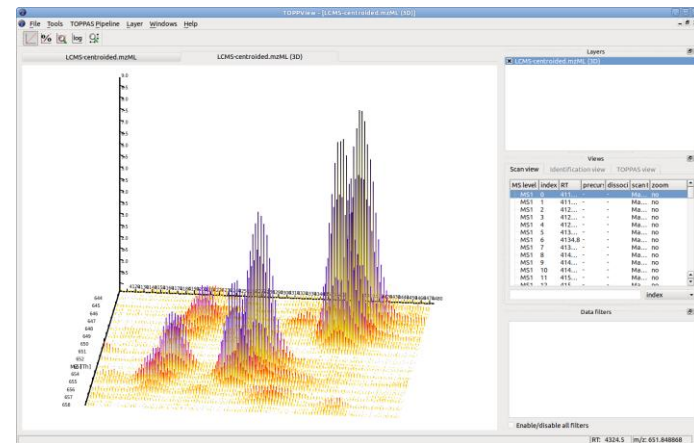
- Biochemical algorithms library
- C++ based framework for Molecular Modelling
- Offers:
 - Extensive set of data structures
 - Molecular Mechanics
 - Molecular Dynamics
 - Advanced solvation
 - Supports many file formats
 - BALLView visualization





- Computer aided drug design suite
- C++ based library for common tasks in CADD
- Offers:
 - File format conversion tools
 - Molecule preparation tools (ligand&receptors)
 - Molecular Docking algorithms (IMGDock,..)
 - QSAR and analysis tools (QuEasy)
 - Database functionality (storage&retrieval of compounds and associated results)

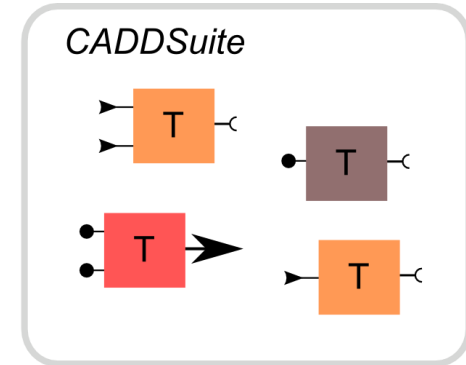
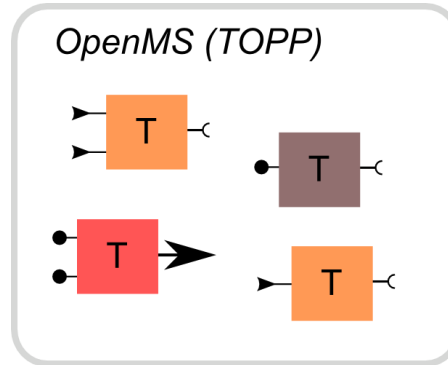
- Open mass spectrometry library
- C++ based library, developed at ZBIT and FU Berlin
- TOPPView
 - Visualizer of MS data
 - QT based
- TOPPAS
 - OpenMS Proteomics Pipeline
 - workflow modelling engine



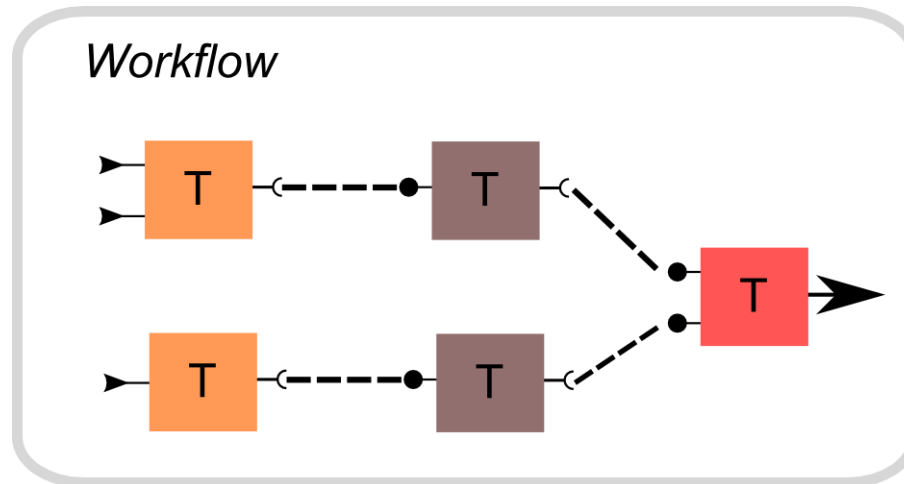


- C++ library of efficient algorithms and data structures for the analysis of sequences with the focus on biological data
- Offers:
 - Sequence/Graph handling infrastructure
 - Pairwise / multiple Alignments
 - ReadMapping
 - Indexing schemes of sequence data bases
- SeqAn used by Illumina's Casava pipeline

- We have several libraries, mostly C++ based, each with a bunch of tools



- Each tool should be usable as “node” within workflows





- Plethora of workflow modelling software out there (even our own, TOPPAS)
- We need a WF modelling platform for desktop prototyping of workflows
- **Solution:** Do not reinvent the wheel but integrate libraries and tool kits into a well known WF system (KNIME)



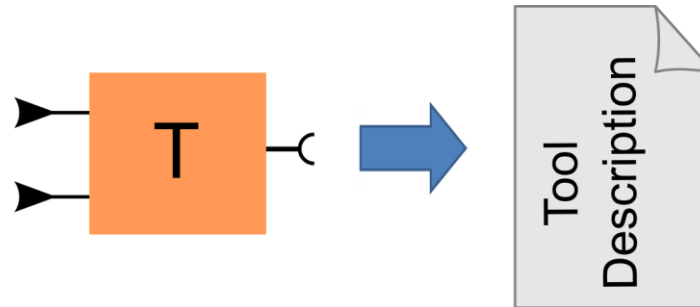
- **Aim 1:** integrate tools from CADDSuite, OpenMS and SEQAN into KNIME
- **Aim 2:** make use of existing infrastructure and proven concepts from OpenMS workflow system, do not reinvent the wheel for each project
- **Aim 3:** Automatic integration through a common interface (also for future tools)



- Modelling and execution of workflows happens at three levels
 - **Desktop**: local prototyping of WFs (**KNIME**)
 - **Cluster**: execution of WFs on cluster, web-based modelling and submission (**Galaxy**)
 - **Grid**: execution of WFs on distributed set of clusters (grid) (**WSPgrade**, **Unicore**)
- Overall goal is to be able to transfer WFs from each level to next level (i.e. KNIME -> Galaxy)



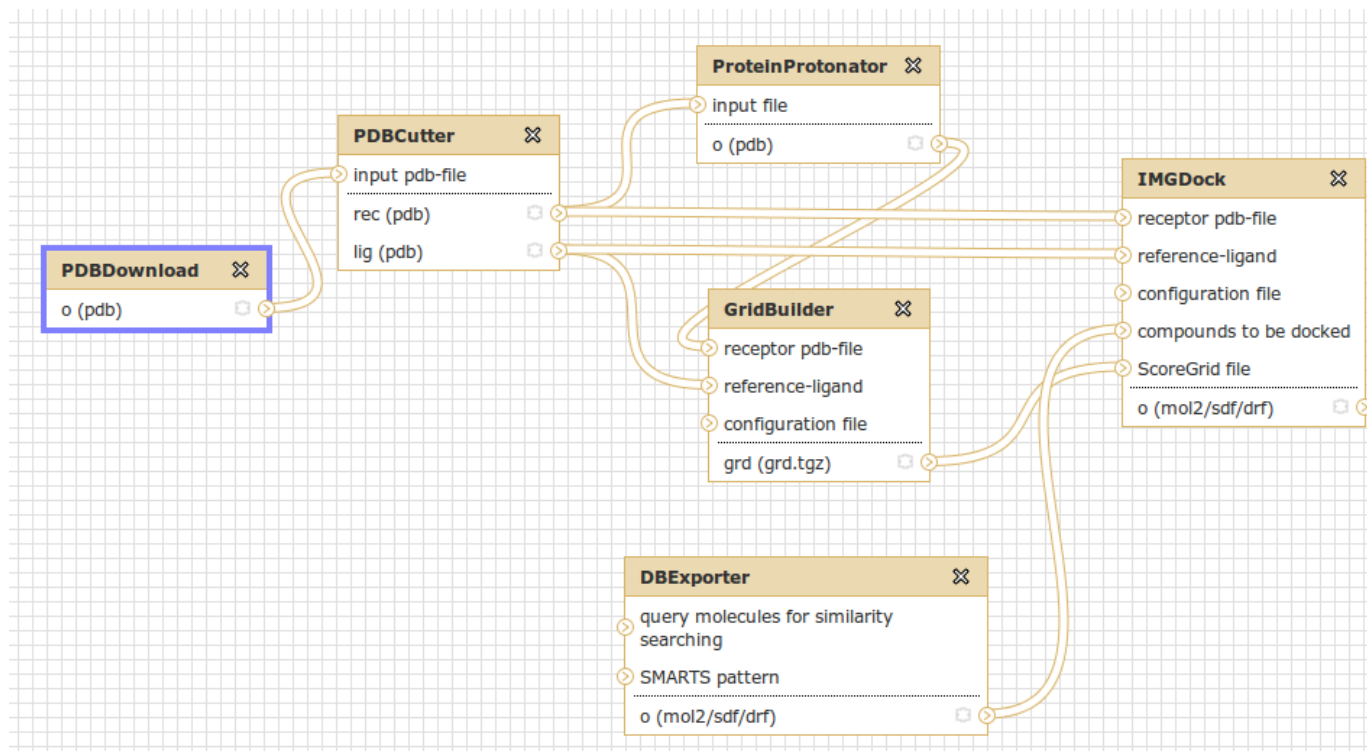
- We already have a tool (node) description format (CTD) from OpenMS



- We also have some experience with integration into Galaxy Workflow System and liked their MIME type system



- CTD helps us to define info needed by a **node**
- Each I/O file will go along one **edge** of a workflow
- edges/node ports are MIME typed

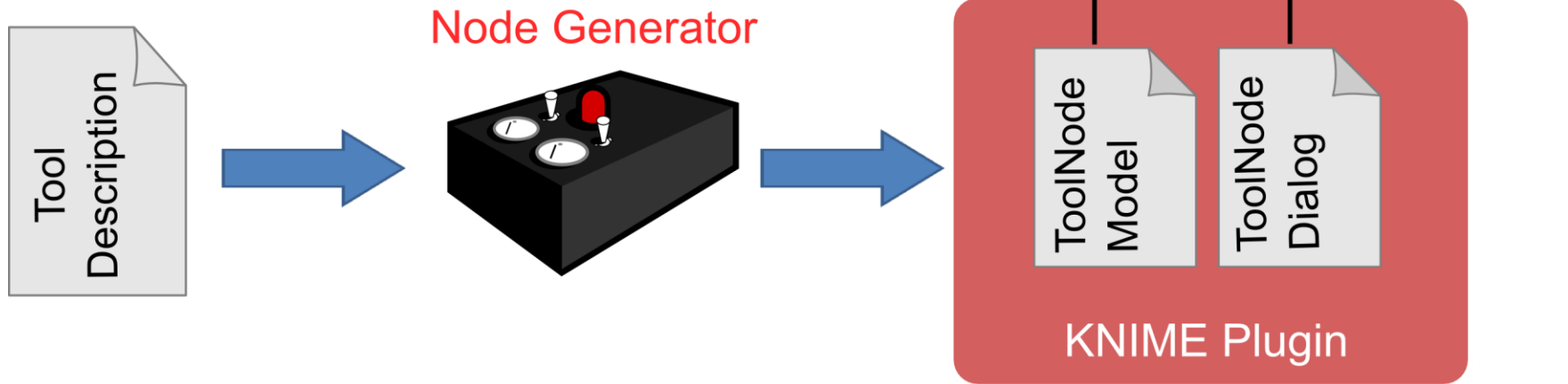




- Known from Email Mime types (RFC 2046) and Unix OSes
- Used for example by Galaxy to type input ports
- GKN has a central MIME type registry
- New plugins can add MIME types to this registry
- I/O files always have a MIME type and are transported along edges in tables
- MIME types are identified by well known file extensions (pdb,sdf,mzML,...)



Generic Knime Nodes





- Each TOPP tool can generate description of its interface (Common Tool Description, CTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<tool status="internal">
<name>NoiseFilterSGolay</name>
<version>1.9.0</version>
<description><![CDATA[Removes noise from profile spectra by using a Savitzky Golay filter.]]></description>
<manual><![CDATA[Removes noise from profile spectra by using a Savitzky Golay filter.]]></manual>
<docurl>http://www-bs2.informatik.uni-tuebingen.de/services/OpenMS/OpenMS-release/html/TOPP_NoiseFilterSGolay.html</docurl>
<category>Signal processing and preprocessing</category>
<PARAMETERS version="1.3" xsi:noNamespaceSchemaLocation="http://open-ms.sourceforge.net/schemas/Param_1_3.xsd" xmlns:xsi="http://www.w3.org/200
  <NODE name="NoiseFilterSGolay" description="Removes noise from profile spectra by using a Savitzky Golay filter.">
    <ITEM name="version" value="1.9.0" type="string" description="Version of the tool that generated this parameters file." tags="advanced" />
    <NODE name="1" description="Instance &apos;l&apos; section for &apos;NoiseFilterSGolay&apos;">
      <ITEM name="in" value="" type="string" description="input raw data file " tags="input file,required" restrictions="*.mzML" />
      <ITEM name="out" value="" type="string" description="output raw data file " tags="output file,required" restrictions="*.mzML" />
      <NODE name="algorithm" description="Algorithm parameters section">
        <ITEM name="frame_length" value="11" type="int" description="The number of subsequent data points used for smoothing." />
        <ITEM name="polynomial_order" value="4" type="int" description="Order or the polynomial that is fitted." />
      </NODE>
    </NODE>
  </NODE>
</PARAMETERS>
</tool>
```



- General tool description in header

```
<tool>  
<name>NoiseFilterSGolay</name>  
<version>1.9.0</version>  
<description>Removes noise using a Savitzky Golay filter.</description>  
<manual>Some manual text here.</manual>  
<docurl>http://openms.de/1.9/doc/NoiseFilterSGolay.html</docurl>  
<category>Signal processing and preprocessing</category>  
...  
...  
</tool>
```



- Parameter and I/O channel description

```
<tool>
```

```
...
```

```
<ITEM name="in" value="" type="string" description="input raw data file"  
      tags="input file,required" restrictions="*.mzML" />
```

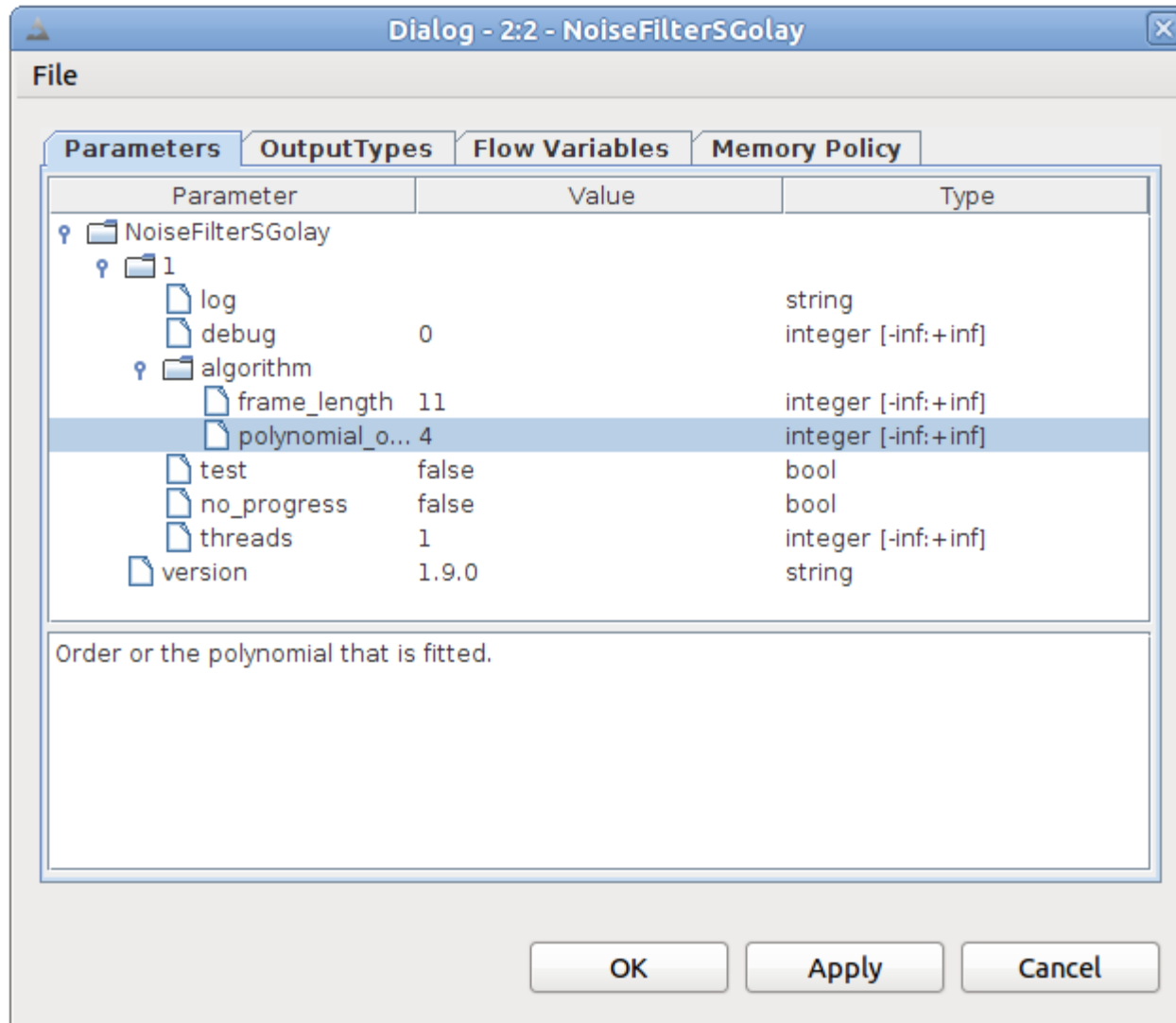
```
<ITEM name="out" value="" type="string" description="output raw data file"  
      tags="output file,required" restrictions="*.mzML" />
```

```
<NODE name="algorithm" description="Algorithm parameters section">  
  <ITEM name="frame_length" value="11" type="int"  
        description="number data points used for smoothing." />  
  <ITEM name="polynomial_order" value="4" type="int"  
        description="Order of the polynomial that is fitted." />
```

```
</NODE>
```

```
...
```

```
</tool>
```





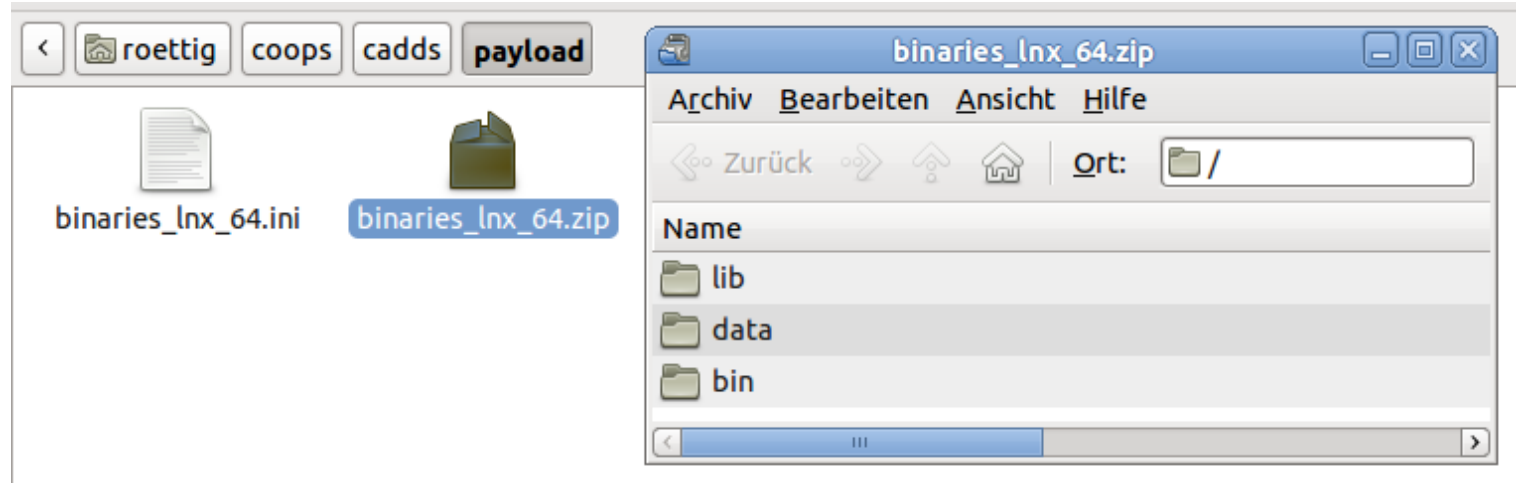
- To create a GKN plugin we need:
 - Set of CTDs describing the tools
 - One mimetypes.xml file describing the MIME types

```
<mimetypes>
```

```
  <mimetype name="FASTA" ext="FASTA" descr="..." />
```

```
</mimetypes>
```

- Payloads for each platform (Win, Mac, *nix)





- For each defined MIME type *X* (i.e. FASTA)
 - a class *X*FileCell and
 - an interface *X*FileValue extending DataValue
- is created, allowing storage of MIMEFile “handles” within KNIME tables

Row ID	? MIMEFILE
Row 0	PDBMimeFileCell

- For each Tool an NodeConfiguration object is created from the CTD
- Execution of Nodes is completely handled by base GKN NodeModel's execute



tmp mimetypes

Name	Größe	Typ	Änderungsdatum
BinaryGridFileCell.class	1,6 KB	Java-Klasse	Mo 03 Okt 2011 15:32:32 CEST
BinaryGridFileValue.class	229 Bytes	Java-Klasse	Mo 03 Okt 2011 15:32:32 CEST
PDBFileCell.class	1,5 KB	Java-Klasse	Mo 03 Okt 2011 15:32:32 CEST
PDBFileValue.class	215 Bytes	Java-Klasse	Mo 03 Okt 2011 15:32:32 CEST

GridBuilder



tmp GridBuilder config

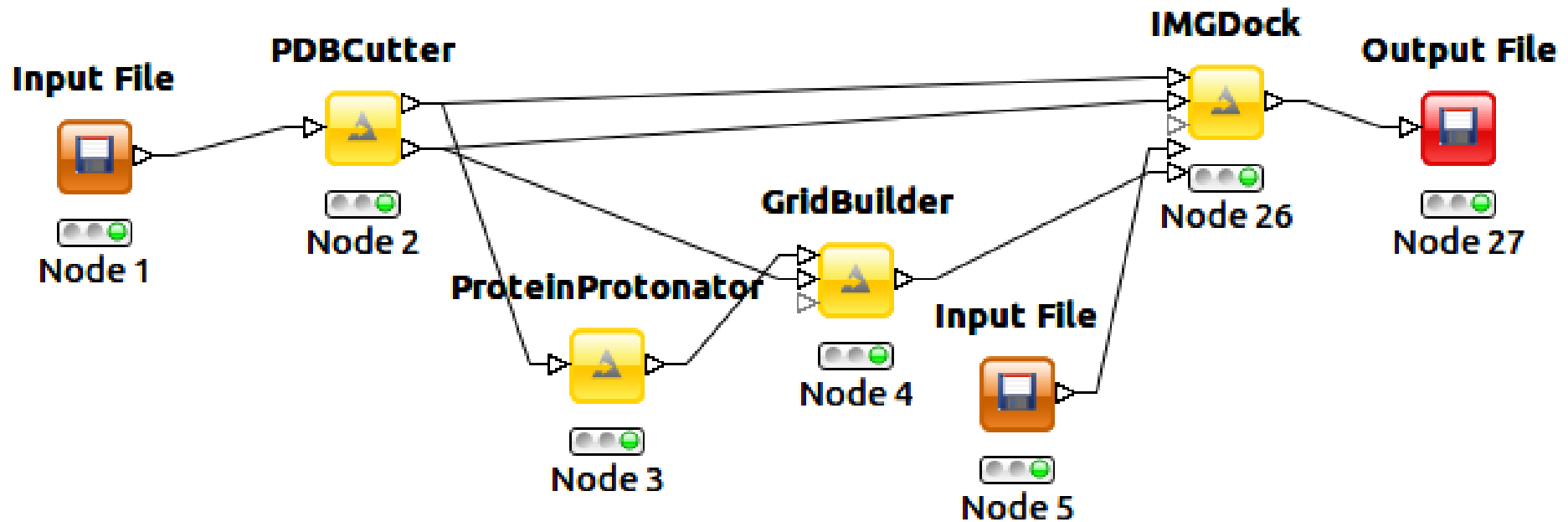
Name	Größe	Typ	Änderungsdatum
config	1 Objekt	Ordner	Mo 03 Okt 2011 15:32:32 CEST
config.xml	2,1 KB	XML-Dokument	Mo 03 Okt 2011 15:32:32 CEST
GridBuilderNodeView.class	558 Bytes	Java-Klasse	Mo 03 Okt 2011 15:32:30 CEST
GridBuilderNodeModel.class	4,7 KB	Java-Klasse	Mo 03 Okt 2011 15:32:30 CEST
GridBuilderNodeFactory.xml	1,7 KB	XML-Dokument	Mo 03 Okt 2011 15:32:32 CEST
GridBuilderNodeFactory.class	2,0 KB	Java-Klasse	Mo 03 Okt 2011 15:32:30 CEST
GridBuilderNodeDialog.class	559 Bytes	Java-Klasse	Mo 03 Okt 2011 15:32:30 CEST



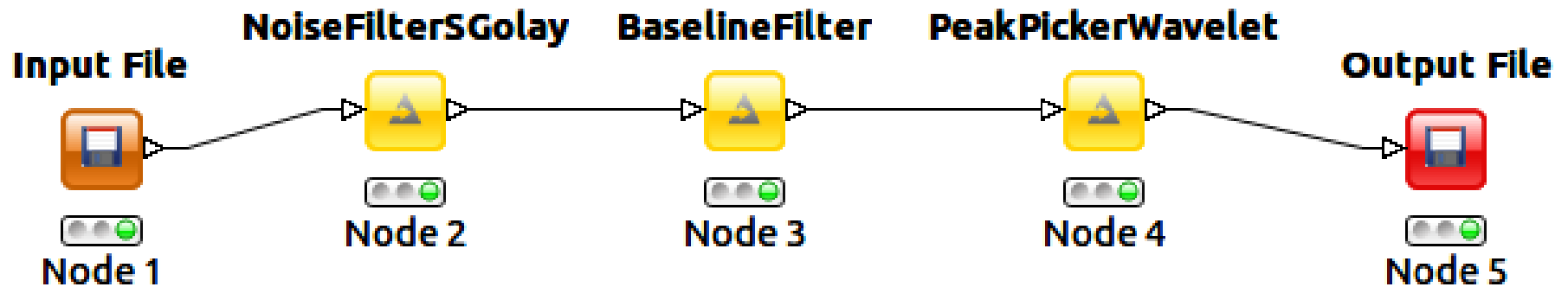
- GenericKnimeNodes
 - IO
 - Input File
 - Input Files
 - Output File
 - Output Files
 - KNIMEConversion
 - BeanShell
 - ColumnMerger
 - ColumnToList
 - FileToTable
 - ListZipLoopStart
 - TableToFile
 - View
 - File Viewer

- Community Nodes
 - CADDSuite
 - Analysis
 - CADDSuite
 - Checks and evaluations
 - Convert, combine and store
 - Docking
 - ConstraintsFinder
 - GridBuilder**
 - IMGDock
 - InteractionConstraintDefiner
 - PocketDetector
 - SpatialConstraintDefiner
 - WaterFinder
 - Get Data
 - Preparation
 - QuEasy (QSAR)
 - AutoModel
 - InputPartitioner
 - InputReader
 - ModelCreator
 - MolPredictor

GKN workflow (CADD Suite)



small docking workflow

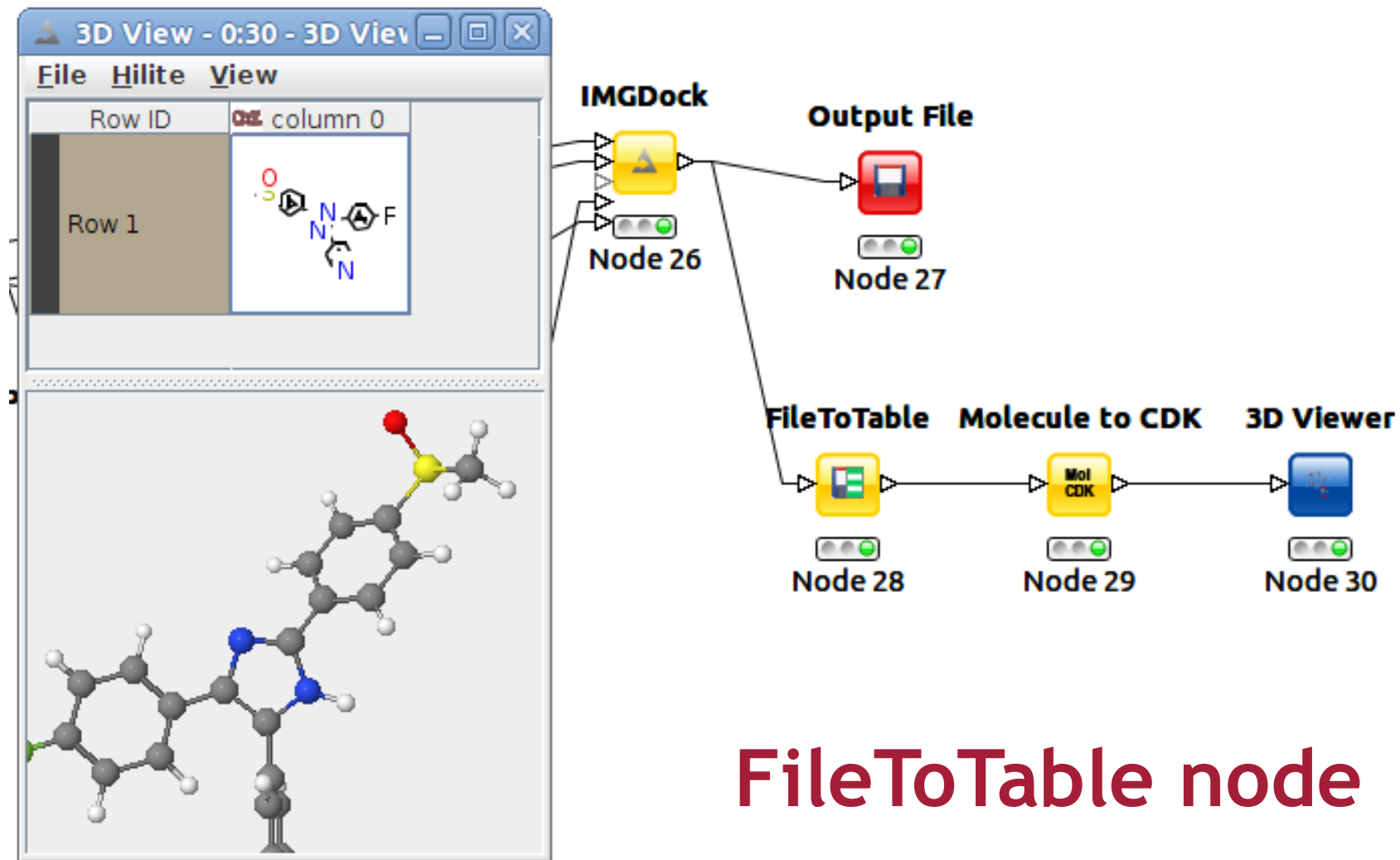


small peak picking workflow



- Use of explicit I/O ports for Nodes is quite different from classical KNIME usage
- Usually data was only transported within tables
- With GKN data is now „hidden“ in MIMEFiles
- This is quite handy for really big files that need to be piped through the workflow
- But in the end (maybe after some filtering) we'd like to do classical KNIME analysis (statistics, WEKA, ...) with our data

From GKN back to KNIME



FileToTable node



- GKN base plugin defines an extension point
- Demanglers can be added through this point
- A Demangler interconverts between the two worlds (MIMEFile → column of basic entries)

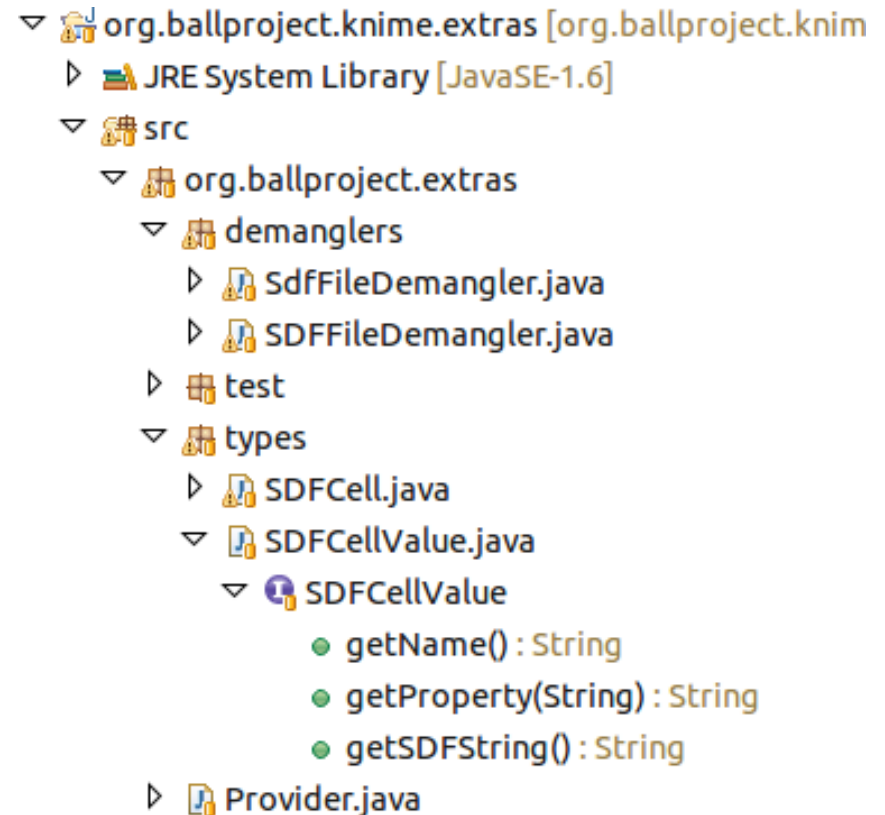
```
<extension-point id="DemanglerProvider" name="DemanglerProvider" schema="DemanglerProvider.exsd"/>
```

```
public interface Demangler extends Serializable
{
    DataType getSourceType();
    DataType getTargetType();
    Iterator<DataCell> demangle(MIMEFileCell cell);
    MIMEFileCell mangle(Iterator<DataCell> iter);
}
```

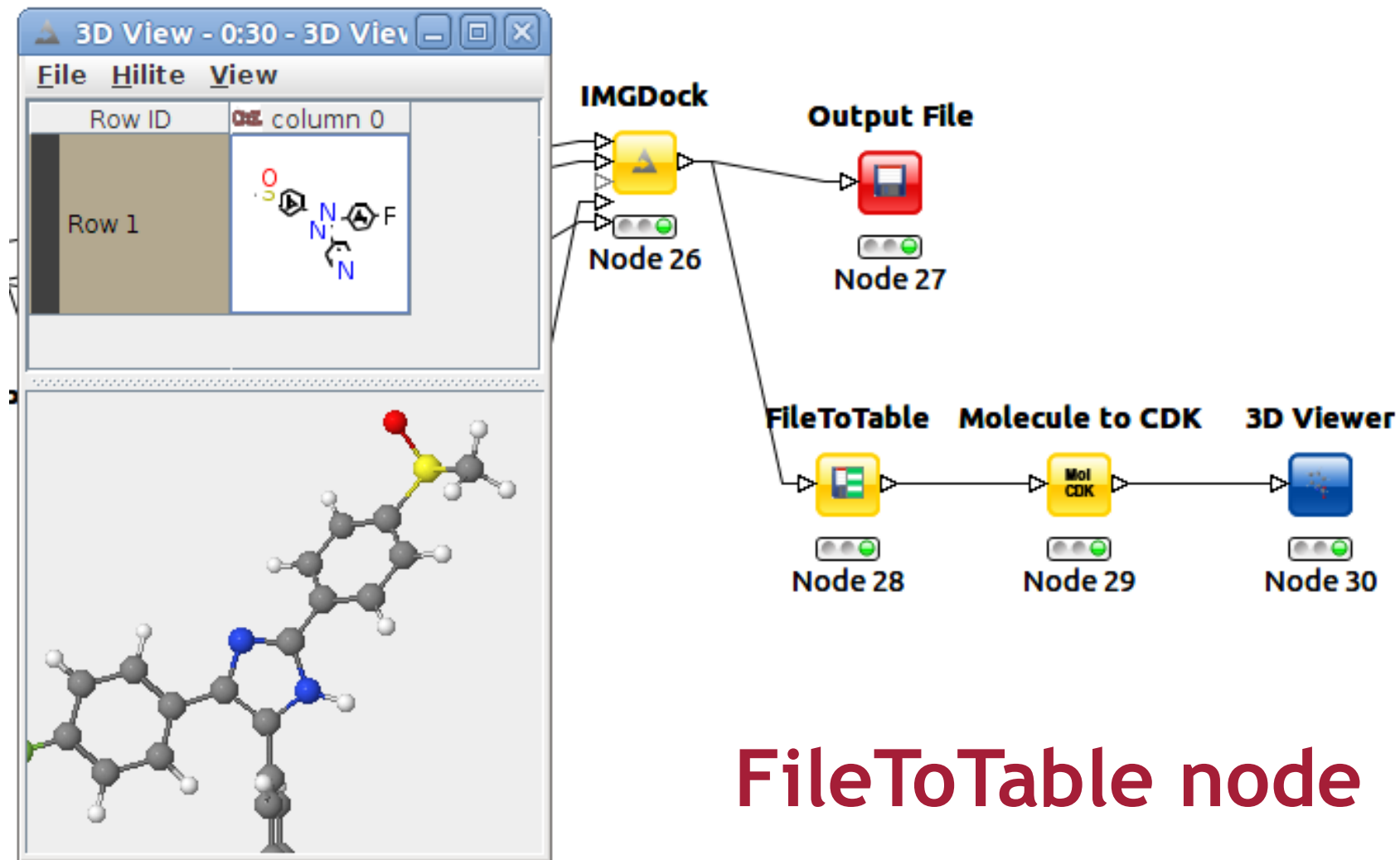
```
public interface DemanglerProvider
{
    List<Demangler> getDemanglers();
}
```



- org.ballproject.knime.extras fragment delivers a SDFDemangler
- It demangles SDFFileCells to KNIME SdfCells and a also new SDFCell type

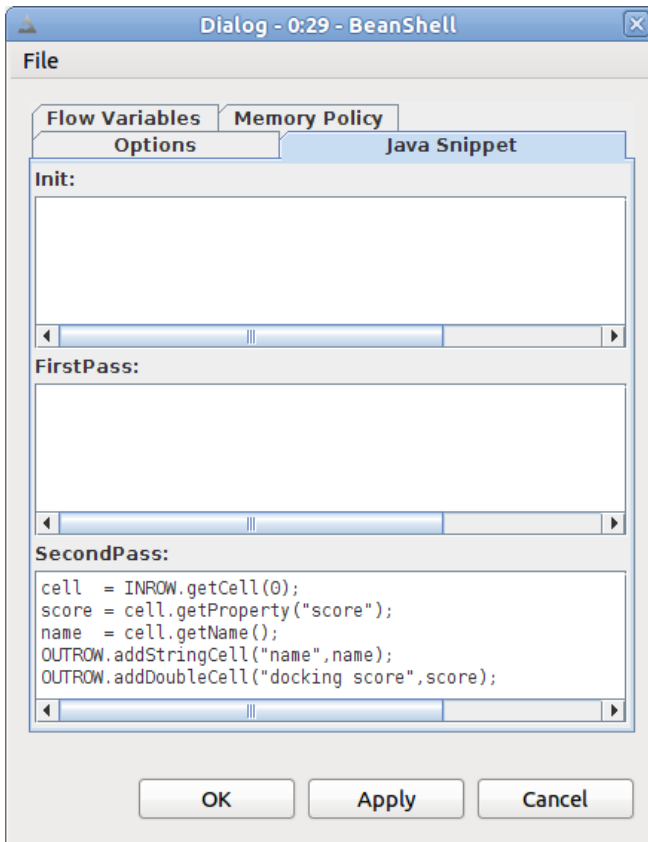


From GKN back to KNIME

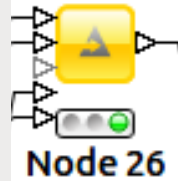


FileToTable node

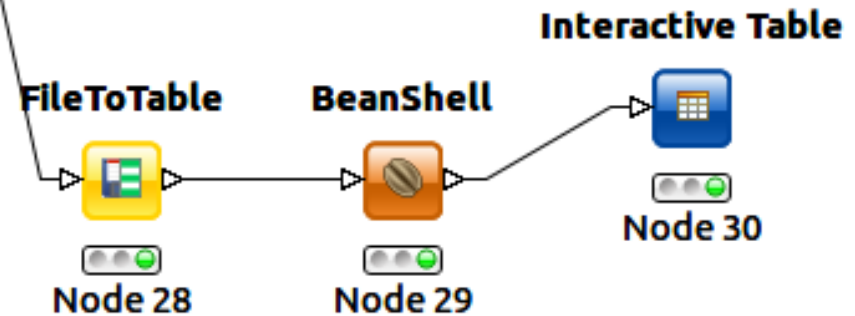
Demangler + Bean Shell



IMGDock



Row ID	name	docking score
Row 1	9543416	-15.342



```
cell = INROW.getCell(0);  
score = cell.getProperty("score");  
name = cell.getName();  
OUTROW.addStringCell("name",name);  
OUTROW.addDoubleCell("docking score",score);
```

Wrapping non-CTD tools



```
<?xml version="1.0" encoding="UTF-8"?>
<tool status="external">
<name>BLAST</name>
<version>2.2.1</version>
<description><![CDATA[BLAST.]]></description>
<manual><![CDATA[BLAST.]]></manual>
<docurl>http://www.google.de</docurl>
<category>Sequence Tools</category>
<type/>
<mapping><![CDATA[
<mapparam CLIswitch="-i" name="blastall.i"/>
<mapparam CLIswitch="-d" name="blastall.d"/>
<mapparam CLIswitch="-o" name="blastall.o"/>
<mapparam CLIswitch="-p" name="blastall.p"/>
<mapparam CLIswitch="-e" name="blastall.e"/>
<mapparam CLIswitch="-M" name="blastall.M"/>
<!-- we want XML as output-->
<setparam name="-m" value="7"/>
]]>
</mapping>
<path>/usr/bin/blastall</path>
<PARAMETERS version="1.3" xsi:noNamespaceSchemaLocation="http://open-ms.sourceforge.net/schemas/Param_1_3.xsd" xmlns:xsi="">
  <NODE name="blastall" description="">
    <ITEM name="i" value="" type="string" description="input file" tags="input file,required" restrictions="*.FASTA" />
    <ITEM name="o" value="" type="string" description="output file" tags="output file,required" restrictions="*.BLASTXML" />
    <ITEM name="p" value="blastp" type="string" description="mode" restrictions="blastp,blastn"/>
    <ITEM name="d" value="" type="string" tags="required" description="database path" restrictions=""/>
    <ITEM name="e" value="0.001" type="float" description="cutoff evalue" restrictions=""/>
    <ITEM name="G" value="-1" type="float" description="gap opening" restrictions=""/>
    <ITEM name="E" value="-1" type="float" description="gap extension" restrictions=""/>
    <ITEM name="M" value="BLOSUM62" type="string" description="scoring matrix" restrictions="BLOSUM62,BLOSUM45"/>
  </NODE>
</PARAMETERS>
</tool>
```

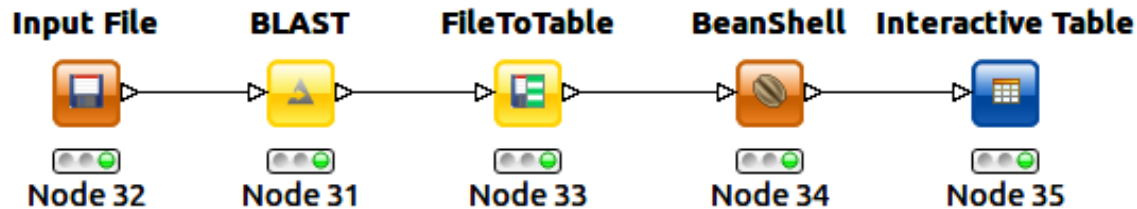
Wrapping non-CTD tools



Table View - 0:35 - Interactive Table

File Hilite Navigation View Output

Row ID	S id	D pid	S hitseq
Row 1	gnl BL_ORD_ID 1	1	MKAAVWNEFKKALEIKEVERPKLEEGEVLVKIEACGVCHTDLHAAHGDWPIK
Row 2	gnl BL_ORD_ID 5	0.503	MRAAVTKDHK-VSIEDKKLRALKPGEALVQTEYCGVCHTDLHVKNADFGDY
Row 3	gnl BL_ORD_ID 4	0.503	MRAAVTKDHK-VSIEDKKLRALKPGEALVQTEYCGVCHTDLHVKNADFGDY
Row 4	gnl BL_ORD_ID 19	0.5	MRAAVTKDHK-VSIEDKKLRALKPGEALVQTEYCGVCHTDLHVKNADFGDY
Row 5	gnl BL_ORD_ID 14	0.5	MRAAVTKDHK-VSIEDKKLRALKPGEALVQTEYCGVCHTDLHVKNADFGDY
Row 6	gnl BL_ORD_ID 8	0.5	MRAAVTKDHK-VSIEDKKLRALKPGEALVQTEYCGVCHTDLHVKNADFGDY
Row 7	gnl BL_ORD_ID 7	0.5	MRAAVTKDHK-VSIEDKKLRALKPGEALVQTEYCGVCHTDLHVKNADFGDY

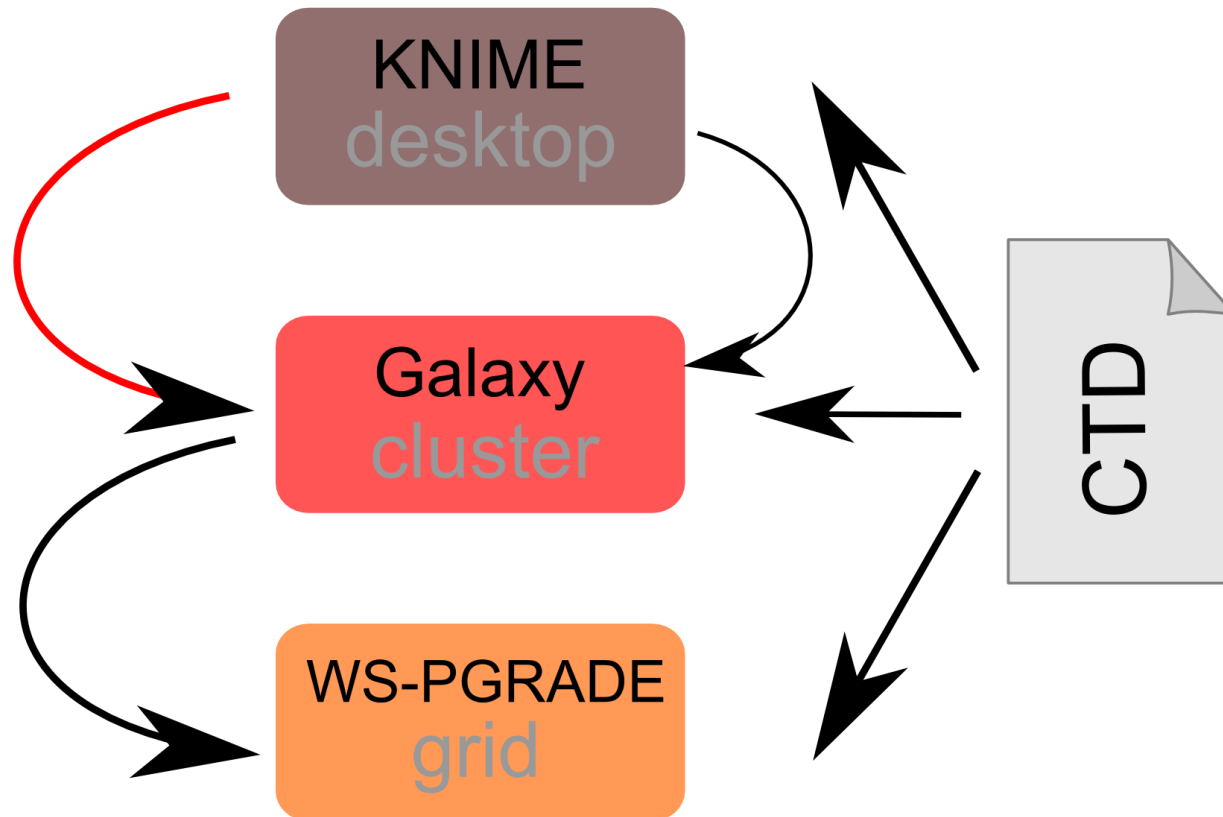


```
cell = INROW.getCell(0);  
pid = cell.getIdentity();  
id = cell.getId();  
hitseq = cell.getHitSequence();  
OUTROW.addStringCell("id",id);  
OUTROW.addDoubleCell("pid",pid);  
OUTROW.addStringCell("hitseq",hitseq);
```



*workflow
conversion*

*node
generation*



- Thanks go out to:
 - Thorsten Meinl for initial help on KNIME
 - Hannes Junker for help with OpenMS integration
 - Marcel Schumann for help with CADDSuite integration
 - the audience for your attention

Any questions or suggestions ?

<https://github.com/roettig/GenericKnimeNodes>