Workflow Mining:
Identification of frequent patterns in a large collection of KNIME workflows
Overview

• Motivation
  • History & current status
  • Why “Workflow Mining”?  

• “Static Workflow Mining”
  • Extraction of workflow graphs
  • Frequent sub-graph mining
  • Results

• “Dynamic Workflow Mining”
  • KNIME Profiler Plugin
  • Analysis of profiling logs

• Conclusions and Outlook
Overview

• **Motivation**
  - History & current status
  - Why “Workflow Mining”?  

• “Static Workflow Mining”  
  - Extraction of workflow graphs  
  - Frequent sub-graph mining  
  - Results

• “Dynamic Workflow Mining”  
  - KNIME Profiler Plugin  
  - Analysis of profiling logs

• Conclusions and Outlook
History

• 2009
  • First look at KNIME, decision to perform in-depth evaluation
  • Identification of critical functional gaps
  • Inhouse developer training
  • Start of collaboration with KNIME.com
• 2010
  • Implementation of missing nodes, integration of various external chemistry tools
  • Licensing of additional node collections from commercial vendors
  • Migration of deployed workflows (e.g. QSAR-models for ADME parameters)
• 2011
  • Migration of end-users from computational chemistry
  • Inhouse end-user trainings
  • Extension of user base to other groups (e.g. computational biology, IS, HTS)
Development of KNIME usage
Current status

- A significant group of end-users is applying KNIME on a regular / daily basis
  - Consistently more than 100 new, unique workflows are created per month
  - Significant exchange and sharing of workflows among users

- A lot of feedback from end-users
  - Bug reports
  - Feature requests
  - How do I ...
  - ...

- Collecting feedback is simple, but ...
  - ... how to prioritize feature requests ?
  - ... what are best practices / common pitfalls / bottlenecks ?
Use data mining to support decisions!

- Difficult to assess significance of requests in a growing and diverse user community
  - Many different application scenarios
  - How are our users actually using KNIME?
  - Different people solve the same problem in a different way!
  - Which nodes are really important?

- Identify key users to cover diversity in user community
  - Still, a lot of gut feeling involved

- Analyze existing workflow collection to get reliable usage statistics
Overview

• Motivation
  • History & current status
  • Why “Workflow Mining”?

• “Static Workflow Mining”
  • Extraction of workflow graphs
  • Frequent sub-graph mining
  • Results

• “Dynamic Workflow Mining”
  • KNIME Profiler Plugin
  • Analysis of profiling logs

• Conclusions and Outlook
Extraction of useful descriptors from KNIME workflows

KNIME workflows are stored in a generic XML format.

A simple parser extracts the workflow graph:
- Nodes
- Connections

The analysis discards:
- Node configuration / labels
- Connection type / direction
- Workflow layout
- Workflow annotations
- Personal information
- ...
Frequent sub-graph mining

• What is frequent sub-graph mining?
  • Problem: Given a database of (labeled) graphs $G = \{G_1, ..., G_n\}$ return all (connected) sub-graphs $S = \{S_1, ..., S_k\}$ occurring at least $x$ times in $G$.

• Related to frequent item set mining & association rule mining

• Many applications in the life-sciences
  • Chemical structures
  • Biological networks
  • …

• Different algorithms and implementations available
  • MoSS node in KNIME for chemical data
  • …
• Using all workflows from central installation (~4,000 as of 2012-01-10)
• 3,899 workflows were parseable and non-empty
• 64,357 nodes
• 62,841 connections
Results – Top 10 nodes

Java Snippet
- Node 8: 9907

GroupBy
- Node 4: 5275

Joiner
- Node 5: 4534

Rename
- Node 3: 4290

Row Filter
- Node 11: 4144

Column Filter
- Node 10: 3326

File Reader
- Node 1: 1720

DB Open v1
- Node 1: 1707

Sorter
- Node 3: 1628

Java Edit Variable
- Node 2: 1570
Results – Top 11-20 nodes

- **Table Reader**: 1567
- **Concatenate**: 1267
- **Table Writer**: 1212
- **Metanode**: 1136
- **DB PreparedQuery v1**: 1051
- **Reference Row Filter**: 893
- **Double To Int**: 882
- **DB MethodsView v1**: 790
- **JPython Script 1:1**: 683
- **DB Query v1**: 660
Results – Workflow patterns with 10% support (selection)

- Allow addition of multiple columns in Java Snippet?
- Collaboration with KNIME.com

- Allow for more flexibility in column naming / typing?

- Make users aware of “Replace columns” option

- Allow joining multiple input tables? (e.g. “Joiner (optional in)”)
Results – Workflow patterns with 7.5% support (selection)

- Issues with sorting stability until v2.3.4 (Bug 2682)

- Develop a concept for simple and efficient data transfer among related workflows?

- Improve column typing in inhouse database integration
Overview

• Motivation
  • History & current status
  • Why “Workflow Mining”?

• “Static Workflow Mining”
  • Extraction of workflow graphs
  • Frequent sub-graph mining
  • Results

• “Dynamic Workflow Mining”
  • KNIME Profiler Plugin
  • Analysis of profiling logs

• Conclusions and Outlook
Static vs. dynamic workflow analysis

• Collection consists mainly of ad hoc-workflows
  • On average, only few executions per workflow
  • Non-interactive batch processes responsible for most workflow executions (20,000,000+ property predictions over the last 12 months)
• Static analysis of all stored workflows might yield biased results

• No insights into bottlenecks and performance issues
  • Influence of different implementations for the same node
  • Improvements of the KNIME framework

• Limited ability to capture influence of “control flow” constructs
  • Include or exclude inactive branches in analysis?
  • How to handle loops?
<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Workflow Path</th>
<th>Node Name</th>
<th>Node Index</th>
<th>Custom Name</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-11-12;06:45:54.662</td>
<td></td>
<td>&lt;Metanode&gt;</td>
<td>0 Node 0</td>
<td>EXECUTED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:54.670</td>
<td></td>
<td>&lt;Metanode&gt;</td>
<td>0 Node 0</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.237</td>
<td>PROP4</td>
<td>DB Open v1</td>
<td>39 Node 39</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.240</td>
<td>PROP4</td>
<td>DB Open v1</td>
<td>39 Node 39</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.262</td>
<td>PROP4</td>
<td>SDF Reader (deprecated)</td>
<td>46 Node 46</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.317</td>
<td>PROP4</td>
<td>SDF Reader (deprecated)</td>
<td>46 Node 46</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.372</td>
<td>PROP4</td>
<td>DB Query v1</td>
<td>2 Node 2</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.706</td>
<td>PROP4</td>
<td>DB Query v1</td>
<td>2 Node 2</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.751</td>
<td>PROP4</td>
<td>Rename</td>
<td>47 Node 47</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.757</td>
<td>PROP4</td>
<td>Rename</td>
<td>47 Node 47</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.761</td>
<td>PROP4</td>
<td>Double To Int</td>
<td>13 Node 13</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.764</td>
<td>PROP4</td>
<td>Double To Int</td>
<td>13 Node 13</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.769</td>
<td>PROP4</td>
<td>Concatenate</td>
<td>48 Node 48</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.774</td>
<td>PROP4</td>
<td>Concatenate</td>
<td>48 Node 48</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.778</td>
<td>PROP4</td>
<td>Column Filter</td>
<td>28 Node 28</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.779</td>
<td>PROP4</td>
<td>Column Filter</td>
<td>28 Node 28</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.790</td>
<td>PROP4</td>
<td>Molecule Type Cast</td>
<td>5 Node 5</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.793</td>
<td>PROP4</td>
<td>Molecule Type Cast</td>
<td>5 Node 5</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.797</td>
<td>PROP4</td>
<td>Column Filter</td>
<td>10 Node 10</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.798</td>
<td>PROP4</td>
<td>Column Filter</td>
<td>10 Node 10</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.807</td>
<td>PROP4</td>
<td>Row Filter</td>
<td>11 Node 11</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.807</td>
<td>PROP4</td>
<td>Row Filter</td>
<td>11 Node 11</td>
<td>CONFIGURED</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.868</td>
<td>PROP4</td>
<td>Corina v1</td>
<td>6 Node 6</td>
<td>IDLE</td>
<td></td>
</tr>
<tr>
<td>2011-11-12;06:45:55.876</td>
<td>PROP4</td>
<td>Corina v1</td>
<td>6 Node 6</td>
<td>CONFIGURED</td>
<td></td>
</tr>
</tbody>
</table>

- Collected execution profiles for 10,000+ non-interactive batch runs
- Not used in interactive sessions, no monitoring of end-user behavior
Results - Top 10 nodes (by number of executions)

Java Snippet

- Node 8: 161.824

Column Filter

- Node 10: 65.282

Row Filter

- Node 11: 37.590

GroupBy

- Node 4: 25.774

Rename

- Node 3: 41.138

Reference Row Filter

- Node 1: 24.773

Joiner

- Node 5: 40.672

Double To Int

- Node 8: 21.351

jPython Script 1:1

- Node 3: 38.530

Concatenate

- Node 9: 19.952
Results – Top 10 nodes (by consolidated runtime)

- **DBBinarySimilaritySearch v1**
  - Node 1
  - 522.013 sec

- **BI Generic Model Integration v1**
  - Node 2
  - 419.561 sec

- **Moe Descriptors v1**
  - Node 4
  - 291.633 sec

- **Volsurf Descriptors**
  - Node 5
  - 212.338 sec

- **Corina v1**
  - Node 6
  - 138.287 sec

- **Blabber**
  - Node 7
  - 75.838 sec

- **Java Snippet**
  - Node 8
  - 66.679 sec

- **Daylight Descriptors v1**
  - Node 9
  - 58.606 sec

- **Table Reader**
  - Node 10
  - 55.274 sec

- **Joiner**
  - Node 11
  - 53.019 sec
Overview

• Motivation
  • History & current status
  • Why “Workflow Mining”?

• “Static Workflow Mining”
  • Extraction of workflow graphs
  • Frequent sub-graph mining
  • Results

• “Dynamic Workflow Mining”
  • KNIME Profiler Plugin
  • Analysis of profiling logs

• Conclusions and Outlook
Conclusions and Outlook

• "Static Workflow Mining" serves as a tool to identify typical patterns in KNIME workflows
  • Supports prioritization of bug reports / feature requests
  • Identification of best practices and common pitfalls
  • Plans to provide useful Metanodes based on frequent patterns

• "Dynamic Workflow Mining" monitors workflow behavior under real-world conditions
  • Identification of bottlenecks and critical steps in workflows
  • Performance optimizations for batch-processes planned
  • Additional option to perform real-time monitoring of system health

• Large workflow collection an asset, but also a burden
  • Have to ensure compatibility of new node versions
Acknowledgements

Johannes Koppe (IS BP R&DM)
Andreas Teckentrup (IS BP R&DM)
Bernd Wiswedel (KNIME.com)

All our KNIME users for their feedback, their suggestions und their patience!