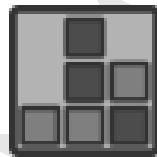


# ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-meta



Peter Reutemann

December 22, 2016

©2009-2014



THE UNIVERSITY OF  
**WAIKATO**  
*Te Whare Wānanga o Waikato*



Except where otherwise noted, this work is licensed under  
<http://creativecommons.org/licenses/by-sa/4.0/>

# Contents

|          |                                  |           |
|----------|----------------------------------|-----------|
| <b>1</b> | <b>Dynamic use of templates</b>  | <b>7</b>  |
| <b>2</b> | <b>Copying callable actors</b>   | <b>9</b>  |
| <b>3</b> | <b>Including external actors</b> | <b>11</b> |
|          | <b>Bibliography</b>              | <b>13</b> |



# List of Figures



# Chapter 1

## Dynamic use of templates

The templating mechanism described in the “core-module” manual, shows how to speed up the inception of new flows. But the templates can also be used in a dynamic way at runtime using the following actors:

- *TemplateStandalone* – for templates that generate standalones
- *TemplateSource* – for templates that generate sources
- *TemplateTransformer* – for templates that generate transforming sub-flows
- *TemplateSink* – for templates that generate sinks

The sub-flow generation is done in a lazy way, i.e., only when the aforementioned template actor is executed, the template is generated. The sub-flow is used till either the end of the flow execution or if a variable changes that is attached to the template itself. In the latter case, the sub-flow gets re-generated the next time the template actor gets executed. This dynamic sub-flow generation in conjunction with variable use, allows to adapt and change the flow at runtime. The example *adams-core-template.flow* demonstrates this.





## Chapter 2

# Copying callable actors

Callable actors can not only be used as synchronization points in the flow. It is also possible to *copy* them, using them as templates. If you don't want to use external flows, but still need to use the same sub-flow multiple times and avoid the bottle neck of synchronous execution, then you can use one of the following actors to create a copy of the callable at the very same location:

- *CopyCallableStandalone* – copies a callable standalone
- *CopyCallableSource* – copies a callable source
- *CopyCallableTransformer* – copies a callable transformer
- *CopyCallableSink* – copies a callable sink



## Chapter 3

# Including external actors

Similar to the external actors of the *adams-core* module, the following actors allow the use of external flow snippets. However, these flows simply replace themselves with the content of the external flow and cannot be changed dynamically with variables. Flexibility has been traded here for performance.

- *IncludeExternalStandalone* – includes an external standalone
- *IncludeExternalSource* – includes an external source
- *IncludeExternalTransformer* – includes an external transformer
- *IncludeExternalSink* – includes an external sink



# Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System  
<http://adams.cms.waikato.ac.nz/>