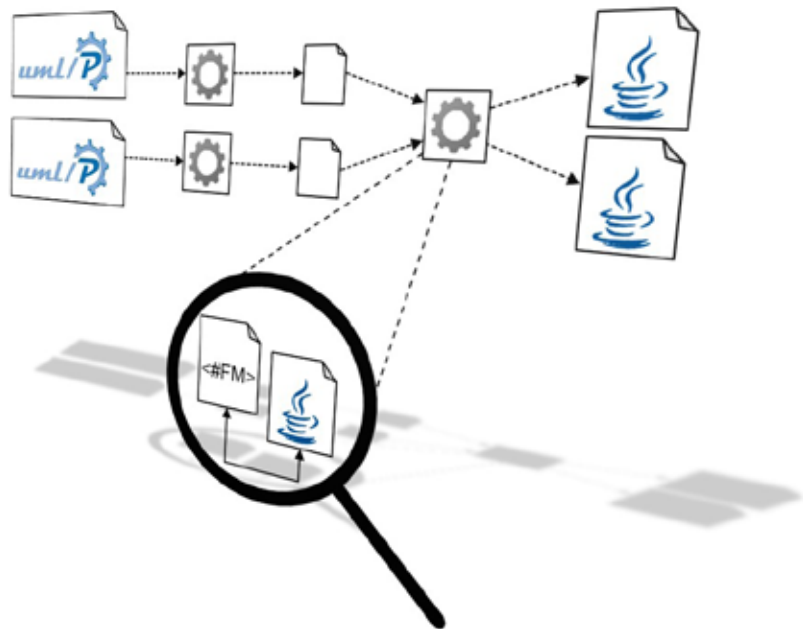


Timo Greifenberg  
Steffen Hillemacher  
Bernhard Rumpe

# Towards a Sustainable Artifact Model

Artifacts in Generator-Based  
Model-Driven Projects



Aachener Informatik-Berichte,  
Software Engineering  
Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe

Band 30

# **Aachener Informatik-Berichte, Software Engineering**

herausgegeben von  
Prof. Dr. rer. nat. Bernhard Rumpe  
Software Engineering  
RWTH Aachen University

Band 30

**Timo Greifenberg**  
**Steffen Hillemacher**  
**Bernhard Rumpe**  
RWTH Aachen University

## **Towards a Sustainable Artifact Model**

Artifacts in Generator-Based Model-Driven Projects

Shaker Verlag  
Aachen 2017

**Bibliographic information published by the Deutsche Nationalbibliothek**

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Copyright Shaker Verlag 2017

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publishers.

Printed in Germany.

ISBN 978-3-8440-5678-5

ISSN 1869-9170

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen

Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9

Internet: [www.shaker.de](http://www.shaker.de) • e-mail: [info@shaker.de](mailto:info@shaker.de)

# Abstract

Model-driven development (MDD) is an enabler for the automatic generation of programming language files for products or for tests from explicitly defined models. MDD projects manage a large magnitude of artifacts (files, etc.) with various relationships.

A large class of artifact relations comes from artifacts *using* others, e.g., via importing types and signatures. This form of usage strongly differs from *generation* dependencies, where one artifact is generated, compiled, and transformed from or to other artifacts.

An MDD project usually entails a number of potentially dependent process steps, where a chain of artifact generations, compilations, and packagings arises. During these steps a multitude of artifacts are created, read or even executed. Those artifacts are thus related to each other in various ways.

The number and complexity of occurring dependencies and other relationships between development artifacts can lead to several problems, such as poor maintainability and long development times of both, MDD tools and the product, in an MDD process. To tackle these problems, it is important to understand which artifacts are involved and how these artifacts are related to each other in MDD projects.

In this report, we (1) develop an abstract and rather general artifact model and (2) apply the artifact model by examining in detail the kinds of artifacts and related concepts relevant for a form of wide-spread projects, namely Java projects. We also dive into the core of generative projects, by looking at the generator as a set of artifacts executed at design time. Artifacts are regarded as storable and explicitly named elements of MDD projects, such as model files, directories, libraries, and source code files. Thus, artifacts are the physical manifestation of all information in an MDD project.

For a precise definition of all relevant concepts, we introduce the *Artifact Model* (AM), which allows the precise, model-based specification of involved kinds of artifacts, corresponding concepts, and their relations. The AM can also be considered as a specific form of *meta-model* for models representing the concrete elements and relations between these in MDD projects.



# Contents

<b>1. Introduction</b>	<b>1</b>
1.1. How to Read the CD4A and OCL Specification . . . . .	4
1.2. Contents of the Report . . . . .	5
1.3. Acknowledgements . . . . .	5
<b>2. Essence of Artifact Models</b>	<b>7</b>
2.1. Artifacts and Artifact Containers . . . . .	7
2.1.1. Artifacts . . . . .	8
2.1.2. Relations between Artifacts . . . . .	9
2.1.3. Artifact Containers . . . . .	10
2.1.4. Directories . . . . .	12
2.1.5. Archives . . . . .	12
2.2. Systems and Tools . . . . .	13
2.2.1. Systems . . . . .	13
2.2.2. Modules . . . . .	14
2.3. How to Use the Artifact Model . . . . .	15
<b>3. Extending the Artifact Model to Java</b>	<b>17</b>
3.1. Java Source and Class Files . . . . .	17
3.1.1. Java Artifacts . . . . .	18
3.1.2. Java Source Files . . . . .	18
3.1.3. Java Class Files . . . . .	19
3.1.4. Java Archives . . . . .	20
3.2. Relation Between Java Artifacts and Types . . . . .	20
3.2.1. Packages . . . . .	21
3.2.2. Java Types . . . . .	21
3.3. Detailed Examination of Java Artifacts and Types . . . . .	25
3.3.1. Case 1: Java Compiles All Files It Relies On . . . . .	26
3.3.2. Case 2: Where Java Looks for Types . . . . .	27
3.3.3. Case 3: Where Java Looks for Inner Types . . . . .	29
3.3.4. Case 4: How Java Handles Archives . . . . .	30
3.3.5. Summary . . . . .	31
<b>4. Modelling Languages and Their Definitions</b>	<b>33</b>
4.1. Languages . . . . .	34

4.2. Grammar-Based Definitions . . . . .	34
4.3. ModelFiles Conform to Languages . . . . .	35
<b>5. Class Diagrams in the Artifact Model</b>	<b>37</b>
<b>6. Generators and their Artifacts in MontiCore</b>	<b>41</b>
6.1. Static Artifact Structures . . . . .	41
6.1.1. Templates . . . . .	42
6.1.2. Generators . . . . .	44
6.2. Dynamic Monitoring of Tool Executions . . . . .	45
6.2.1. Representing Actions and Events . . . . .	45
6.2.2. Actions in a Generation Process . . . . .	47
6.2.3. Tools Read and Create Artifacts . . . . .	49
6.2.4. Template and Java Files Contribute to Artifacts . . . . .	50
<b>7. Artifacts in Maven-managed Java Projects</b>	<b>53</b>
7.1. Maven Modules . . . . .	54
7.2. Relations between Maven Modules . . . . .	55
7.3. Target Directories and Target Artifacts . . . . .	56
7.4. POM and VCSRootDir . . . . .	57
7.5. Maven Phases . . . . .	59
7.6. Executing Maven . . . . .	60
<b>8. Applications of the Artifact Model</b>	<b>65</b>
8.1. Analyses based on Tool Monitoring . . . . .	66
8.2. Understanding the Module/Artifact Architecture . . . . .	66
8.3. Generated Systems . . . . .	69
8.4. Template Relations Induced by Generated Artifacts . . . . .	70
8.5. Incremental Toolchain Execution . . . . .	71
8.6. Unused Imports . . . . .	73
<b>9. Conclusion</b>	<b>75</b>
9.1. Considerations on the Artifact Model . . . . .	75
9.2. Multi-level considerations on CD4A and OCL . . . . .	76
<b>Bibliography</b>	<b>79</b>
<b>A. Merged Artifact Model</b>	<b>91</b>
<b>B. Entire Application Model</b>	<b>109</b>
<b>Index</b>	<b>113</b>