Heiko Klare
IPD Reussner
Am Fasanengarten 5, 76131 Karlsruhe
heiko.klare@kit.edu, 0721/608-45995

Professorinnen und Professoren

der KIT-Fakultät für Informatik

Karlsruhe, 19.10.2020

**Vorstellung meines Promotionsthemas im Rahmen des Promotionsgesprächs (ehemals Professorenrunde)**

Sehr geehrte Professorinnen, sehr geehrte Professoren,

ich beabsichtige, meine Promotionsprüfung an der KIT-Fakultät für Informatik des Karlsruher Instituts für Technologie abzulegen. Ich werde bei der Erstellung meiner Dissertation federführend von

Prof. Dr. Ralf Reussner

betreut. Meine Arbeit trägt derzeit den Arbeitstitel

Building Transformation Networks for Consistent Evolution of Multiple Models .

Ich möchte mich gerne bei Ihnen vorstellen und Ihnen das Thema meiner Arbeit erläutern. Eine kurze Zusammenfassung meiner Arbeit habe ich diesem Schreiben beigelegt.

Den wesentlichen Beitrag meiner Arbeit zur Erweiterung des Stands der Forschung in der Informatik fasse ich hiermit ganz kurz zusammen:

Die Entwicklung eines Softwaresystems mittels mehrerer Werkzeuge erfordert die Konsistenzerhaltung der einzelnen Artefakte. Existierende Ansätze auf Basis von Transformationen sind auf zwei Artefakte beschränkt oder projektspezifisch. In meiner Dissertation untersuche ich systematisch, wie mehrere Transformationen zwischen jeweils zwei Artefakten unabhängig entwickelt und modular gekoppelt werden können, um alle Artefakte konsistent zu halten. Ich stelle eine Formalisierung des Problems, sowie konstruktive und analytische Lösungsmethoden vor, deren Korrektheit ich beweise und deren Anwendbarkeit ich empirisch validiere.

Wenn Sie Interesse und Zeit für ein Gespräch mit mir haben, nennen Sie mir bitte mit beiliegendem Antwortschreiben einen möglichen Termin. Sollte ein Gespräch in nächster Zeit nicht möglich sein oder sollte Ihnen die schriftliche Zusammenfassung der Arbeit zunächst genügen, würde ich mich sehr freuen, wenn Sie mir dies mitteilen würden.

Mit freundlichem Gruß

Heiko Klare

Absenderdaten des Lehrstuhls

Heiko Klare
IPD Reussner, Geb. 50.34, Raum 328
Am Fasanengarten 5
76131 Karlsruhe

Ort, Datum

**Rückantwort – Gespräch über Promotionsthema**

☐ Ich möchte gerne mit Ihnen über Ihr Promotionsthema sprechen

Terminvorschlag: hier einen Terminvorschlag eintragen

☐ Von meiner Seite aus besteht derzeit kein Gesprächsbedarf.

Sonstige Mitteilung:

Hier kann der Lehrstuhl eine Nachricht an den/die Doktoranden/in eintragen

Mit freundlichem Gruß

Absendername

Karlsruher Institut für Technologie

Dissertationsvorhaben Heiko Klare

Erstgutachter: Prof. Dr. Ralf Reussner

Zweitgutachter: Prof. Dr. Colin Atkinson (Universität Mannheim)

# Building Transformation Networks for Consistent Evolution of Multiple Models

*In meiner Promotion habe ich die Konsistenzerhaltung verschiedener Artefakte zur Beschreibung eines Softwaresystems untersucht, indem ich die Kopplung von Transformationen zwischen den Spezifikationssprachen dieser Artefakte formalisiert, analysiert und mit Methoden unterstützt habe.*

Bei der Entwicklung eines Softwaresystems nutzen die verschiedenen Entwickler/-innen und weiteren Interessenvertreter/-innen diverse Sprachen zur Beschreibung unterschiedlicher Belange. Meist stellt der Programmcode das zentrale Artefakt dar, wird jedoch, implizit oder explizit, durch Spezifikationen der Architektur, des Deployments oder der Anforderungen ergänzt. Zur Spezifikation dieser Artefakte werden neben der Programmiersprache verschiedene andere Sprachen verwendet, beispielsweise UML für Architekturmodelle, OpenAPI für Schnittstellen-Spezifikationen oder Docker für Deployment-Spezifikationen von Microservices. Zur Erstellung eines funktionsfähigen Softwaresystems müssen diese Artefakte eine einheitliche, widerspruchsfreie Spezifikation des gesamten Systems darstellen. Beispielsweise müssen Service-Schnittstellen in all diesen Artefakt einheitlich repräsentiert sein. Wir sagen, die Artefakte müssen konsistent sein.

In der modellgetriebenen Entwicklung werden solch verschiedene Artefakte, dort allgemein *Modelle* genannt, bereits als wesentliche zentrale Entwicklungsbestandteile genutzt, um auch Teile des Programmcodes aus ihnen abzuleiten. Dies betrifft beispielsweise die Softwareentwicklung für Fahrzeuge [8, 21] oder allgemein die Entwicklung cyber-physikalischer Systeme. Zur Konsistenzerhaltung der Modelle werden hier oftmals Transformationen eingesetzt, die nach Änderungen eines Modells die anderen Modelle anpassen. Die bisherige Forschung beschränkt sich jedoch auf die Konsistenzerhaltung zweier Modelle [3] und die aufeinander abgestimmte, projektspezifische Konsistenzerhaltung mehrerer Modelle [1, 2]. Ein systematischer Entwicklungsprozess, in dem einzelne Transformationen unabhängig und modular entwickelt und in verschiedenen Kontexten wiederverwendet werden können, wird hierdurch jedoch nicht unterstützt.

In meiner Dissertation befasse ich mich daher mit der Frage, wie Entwickler mehrere Transformationen zu einem Netzwerk kombinieren können, welches die Transformationen in einer geeigneten Reihenfolge ausführen kann, sodass abschließend alle Modelle konsistent zueinander sind [13, 10]. Dies geschieht unter der Annahme, dass einzelne Transformationen zwischen zwei Sprachen unabhängig voneinander entwickelt und daher nicht aufeinander abgestimmt werden. Meine Beiträge unterteilen sich dabei in die Untersuchung der Korrektheit einer solchen Kombination und die Auswirkung auf und Optimierung von Qualitätseigenschaften eines solchen Netzwerkes.

Ich leite zunächst einen adäquaten Korrektheitsbegriff her und definiere ihn mittels eines angemessenen Formalismus. Dieser Korrektheitsbegriff impliziert die Notwendigkeit dreier weiterer Betrachtungen. Erstens fordert er von Transformationen eine *Synchronisations*-Eigenschaft, zweitens setzt er eine Form von *Kompatibilität* zwischen den Transformationen voraus, und drittens erfordert er das Finden einer korrekten Ausführungsreihenfolge der Transformationen, einer *Orchestrierung*. Ich leite her, wie mit bestehenden Sprachen Transformationen entwickelt werden können, die die Synchronisations-Eigenschaft erfüllen [17], und schlage auf Basis einer formal bewiesenen Eigenschaft ein entsprechendes Konstruktionsverfahren vor, dessen Anwendbarkeit ich in Fallstudien im

Karlsruher Institut für Technologie

Kontext der komponentenbasierten Softwareentwicklung evaluiere. Weiterhin definiere ich formal die Eigenschaft der Kompatibilität von Transformationen [15], für welche ich ein formales Analyseverfahren vorschlage und dessen Korrektheit ich beweise. Hieraus entwickele ich eine praktische Realisierung, deren Anwendbarkeit ich ebenfalls in Fallstudien nachweise. Außerdem untersuche ich, wie für Transformationen eine valide Orchestrierung, also eine Ausführungsreihenfolge nach der alle Modelle konsistent sind, gefunden werden kann [7]. Dabei beweise ich, dass das allgemeine zugrundeliegende Problem unentscheidbar ist. Ich stelle Indikatoren dafür vor, dass auch mögliche Einschränkungen des Problems, um dessen Entscheidbarkeit zu erreichen, die Anwendbarkeit unpraktikabel einschränken. Ich schlage daher einen Algorithmus vor, der die Lösung konservativ approximiert, und für den ich Korrektheit und eine wohldefinierte Eigenschaft zur Beschreibung seiner Nützlichkeit beweise. Des Weiteren kategorisiere ich Fehler, die auftreten können, wenn die identifizierten Korrektheitseigenschaften nicht erfüllt werden, und analysiere deren Relevanz in oben genannten Fallstudien [17]. Hieraus leite ich zum einen ab, inwieweit das fehlerhafte Verhalten des Netzwerkes Rückschlüsse auf die Art des ursächlichen Fehler zulässt, und zeige zum anderen, dass sich die meisten potentiellen Fehler mit den vorgeschlagenen Verfahren per Konstruktion vermeiden lassen.

Zur Untersuchung von Qualitätseigenschaften eines Netzwerkes von Transformationen identifiziere ich zunächst relevante Eigenschaften und untersuche, wie sich verschiedene Typen von Netzwerktopologien auf diese auswirken [10]. Hierbei zeigt sich, dass insbesondere Korrektheit und Modularität im Widerspruch stehen und die Wahl der Topologie ein Abwägen bei der Optimierung dieser Eigenschaften erfordert. Ich leite hieraus ein Konstruktionsverfahren für Transformationsnetzwerke ab [12], welches die Notwendigkeit einer Abwägung zwischen den Qualitätseigenschaften abmildert und, unter gewissen Voraussetzungen, Korrektheit per Konstruktion gewährleistet. Für dieses Verfahren stelle ich außerdem eine Spezifikationssprache vor [12], die den Entwicklungsprozess strukturiert. Ich evaluiere die Anwendbarkeit des Verfahrens und der Sprache durch die Implementierung der bereits für die Korrektheitseigenschaften verwendeten Fallstudien aus der komponentenbasierten Softwareentwicklung. Hierzu validiere ich insbesondere die Erfüllbarkeit der Voraussetzungen zur Anwendung des Verfahrens in realistischen Szenarien, sowie die Kompaktheit der vorgestellten Sprache.

Die Beiträge meiner Arbeit unterstützen sowohl Forscher/-innen, als auch Transformationsentwickler/-innen und Transformationsanwender/-innen bei der Analyse und Konstruktion von Netzwerken von Transformationen [22]. Sie stellen für Forscher/-innen und Transformationsentwickler/-innen systematisches Wissen über die Korrektheit und weitere Qualitätseigenschaften solcher Netzwerke bereit, und zeigen insbesondere, welche Teile dieser Eigenschaften per Konstruktion oder per Analyse erreicht bzw. nachgewiesen werden können, sowie mit welchen Fehlern bei der Ausführung gerechnet werden muss. Zusätzlich stelle ich mit den Beiträgen konkrete, praktisch nutzbare Verfahren bereit, mit denen korrekte, modulare Netzwerke konstruiert, analysiert und ausgeführt werden können, wovon sowohl Transformationsentwickler/-innen als auch Transformationsanwender/-innen profitieren.

Karlsruher Institut für Technologie

## Referenzen


[1] A. Cleve, E. Kindler, P. Stevens und V. Zaytsev. „Multidirectional Transformations and Synchronisations (Dagstuhl Seminar 18491)“. In: *Dagstuhl Reports* 8.12 (2019), S. 1–48.

[2] Z. Diskin, H. König und M. Lawford. „Multiple Model Synchronization with Multiary Delta Lenses“. In: *Fundamental Approaches to Software Engineering*. Springer International Publishing, 2018, S. 21–37.

[3] P. Stevens. „Maintaining consistency in networks of models: bidirectional transformations in the large“. In: *Software and Systems Modeling* 19.1 (2020), S. 39–65.


## Eigene Veröffentlichungen


[4] S. Ananieva, T. Berger, A. Burger, T. Kehrer, H. Klare, A. Koziolek, H. Lönn, R. Sethu, G. Taentzer und B. Westfechtel. „Conceptual Modeling Group“. In: *Software Evolution in Time and Space: Unifying Version and Variability Management (Dagstuhl Seminar 19191)*. Bd. 9. Dagstuhl Reports 5. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, S. 21–24.

[5] S. Ananieva, T. Kehrer, H. Klare, A. Koziolek, H. Lönn, R. Sethu, A. Burger, G. Taentzer und B. Westfechtel. „Towards a Conceptual Model for Unifying Variability in Space and Time“. In: *2nd International Workshop on Variability and Evolution of Software-Intensive Systems*. VariVolution ’19. ACM, 2019.

[6] S. Ananieva, H. Klare, E. Burger und R. Reussner. „Variants and Versions Management for Models with Integrated Consistency Preservation“. In: *12th International Workshop on Variability Modelling of Software-Intensive Systems*. VAMOS 2018. ACM, 2018, S. 3–10.

[7] J. Gleitze, H. Klare und E. Burger. *Finding a Universal Execution Strategy for Model Transformation Networks*. submitted to Fundamental Approaches to Software Engineering (FASE) 2021.

[8] H. Guissouma, H. Klare, E. Sax und E. Burger. „An Empirical Study on the Current and Future Challenges of Automotive Software Release and Configuration Management“. In: *44th Euromicro Conference on Software Engineering and Advanced Applications*. SEAA 2018. IEEE, 2018, S. 298–305.

[9] R. Heinrich, D. Werle, H. Klare, R. Reussner, M. Kramer, S. Becker, J. Happe, H. Koziolek und K. Krogmann. „The Palladio-Bench for Modeling and Simulating Software Architectures“. In: *40th International Conference on Software Engineering: Companion Proceedings*. ICSE ’18. ACM, 2018, S. 37–40.

[10] H. Klare. „Multi-model Consistency Preservation“. In: *21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. MODELS 2018. ACM, 2018, S. 156–161.

[11] H. Klare, E. Burger, M. E. Kramer, M. Langhammer, T. Saglam und R. Reussner. „Ecoreification: Making Arbitrary Java Code Accessible to Metamodel-Based Tools“. In: *20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. MODELS 2017. IEEE Computer Society, 2017.

[12] H. Klare und J. Gleitze. „Commonalities for Preserving Consistency of Multiple Models“. In: *22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019, S. 371–378.

Karlsruher Institut für Technologie


[13] H. Klare, M. E. Kramer, M. Langhammer, D. Werle, E. Burger und R. Reussner. „Enabling consistency in view-based system development – The Vitruvius approach“. In: *Journal of Software and Systems* 171 (2021).

[14] H. Klare, M. Langhammer und M. E. Kramer. „Projecting UML Class Diagrams from Java Code Models“. In: *4th Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling)*. VAO '16. Karlsruhe Institute of Technology. 2016, S. 11–18.

[15] H. Klare, A. Pepin, E. Burger und R. Reussner. *A Formal Approach to Prove Compatibility in Transformation Networks*. Techn. Ber. 3. Karlsruher Institut für Technologie (KIT), 2020. 40 S.

[16] H. Klare, T. Saglam, E. Burger und R. Reussner. „Applying Metamodel-based Tooling to Object-oriented Code“. In: *7th International Conference on Model-Driven Engineering and Software Development*. MODELSWARD 2019. INSTICC. SCiTePress, 2019.

[17] H. Klare, T. Syma, E. Burger und R. Reussner. „A Categorization of Interoperability Issues in Networks of Transformations“. In: *Journal of Object Technology* 18.3 (2019). The 12th International Conference on Model Transformations, 4:1–20.

[18] M. E. Kramer, G. Hinkel, H. Klare, M. Langhammer und E. Burger. „A Controlled Experiment Template for Evaluating the Understandability of Model Transformation Languages“. In: *Second International Workshop on Human Factors in Modeling co-located with 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. Bd. 1805. CEUR Workshop Proceedings. CEUR-WS.org, 2016, S. 11–18.

[19] J. Meier, H. Klare, C. Tunjic, C. Atkinson, E. Burger, R. Reussner und A. Winter. „Single Underlying Models for Projectional, Multi-View Environments“. In: *7th International Conference on Model-Driven Engineering and Software Development*. MODELSWARD 2019. INSTICC. SCiTePress, 2019, S. 119–130.

[20] J. Meier, C. Werner, H. Klare, C. Tunjic, U. Aßmann, C. Atkinson, E. Burger, R. Reussner und A. Winter. „Classifying Approaches for Constructing Single Underlying Models“. In: *Model-Driven Engineering and Software Development*. Springer International Publishing, 2020, S. 350–375.

[21] E. Sax, R. Reussner, H. Guissouma und H. Klare. *A Survey on the State and Future of Automotive Software Release and Configuration Management*. Techn. Ber. 11. Karlsruher Institut für Technologie (KIT), 2017. 19 S.

[22] M. Tichy und H. Klare. „Human Factors: Interests of Transformation Developers and Users“. In: *Multidirectional Transformations and Synchronisations (Dagstuhl Seminar 18491)*. Bd. 8. Dagstuhl Reports 12. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, S. 16–20.

[23] K. Yurchenko, M. Behr, H. Klare, M. Kramer und R. Reussner. „Architecture-Driven Reduction of Specification Overhead for Verifying Confidentiality in Component-Based Software Systems“. In: *MODELS 2017 Satellite Event (MoDeVVa Workshop), co-located with 20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. Bd. 2019. MODELS 2017. CEUR-WS, 2017, S. 321–323.