University of Stuttgart
Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW)

Language Composition Operators

A Literature Review

Jérôme Pfeiffer,
David Schmalzing,
Andreas Wortmann

Photo by Matt Howard on Unsplash

Motivation

Uncovering the current state of language composition

- Efficiently engineering software languages demands their reuse through composition
- Composition operators emerged acting on different
  - constituents of languages,
  - technological spaces,
  - and purposes.
- Ten years ago, Erdweg et. al. classified language composition into 5 categories.
- Innovations in software language engineering question the validity of these categories

→ Uncovering the current state of language composition and drawing a detailed map of language composition operators to guide SLE researchers and practitioners

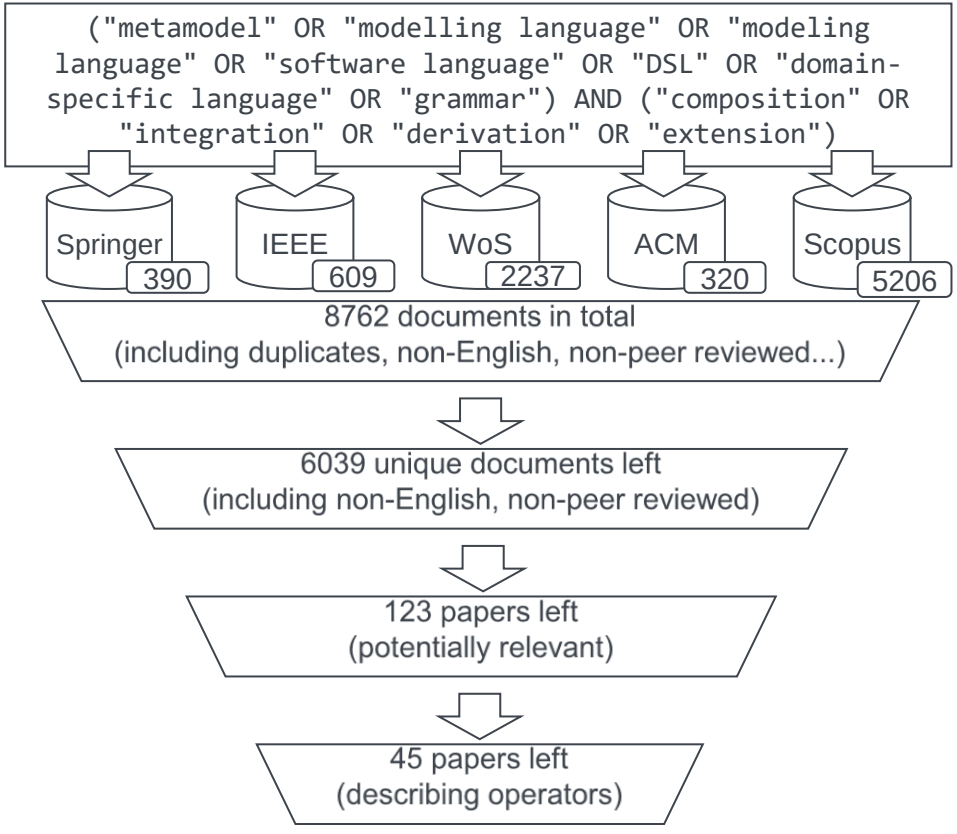
Language Composition Untangled [1]

Classifying composition operators for software languages

- Language Extension:
  - Extend a base language definition with rules.
  - Example: Adding rules to an inherited grammar
- Language Restriction:
  - Language extension restricts the language.
  - Example: Use context condition to prevent usage of certain modeling elements.
- Language Unification:
  - Composition on equal terms, i.e., without direction, using glue code.
  - Example: Names on transitions of statecharts refer to names of properties in a classdiagram
- Self-extension:
  - Embedding of languages into a host language by providing a host language model that encapsulates the embedded language's concept
  - Example: A class library in a GPL
- Extension Composition:
  - Language extensions can work together

Research Method

Conducting a systematic literature review on software language composition



- Classified the studies along the classification of LCU
- Developed a questionnaire for detailed analyses and comparability of extracted information
- Classified the first 20 studies in parallel among authors to avoid misunderstandings

Research Question 1

Which language composition operators exist?

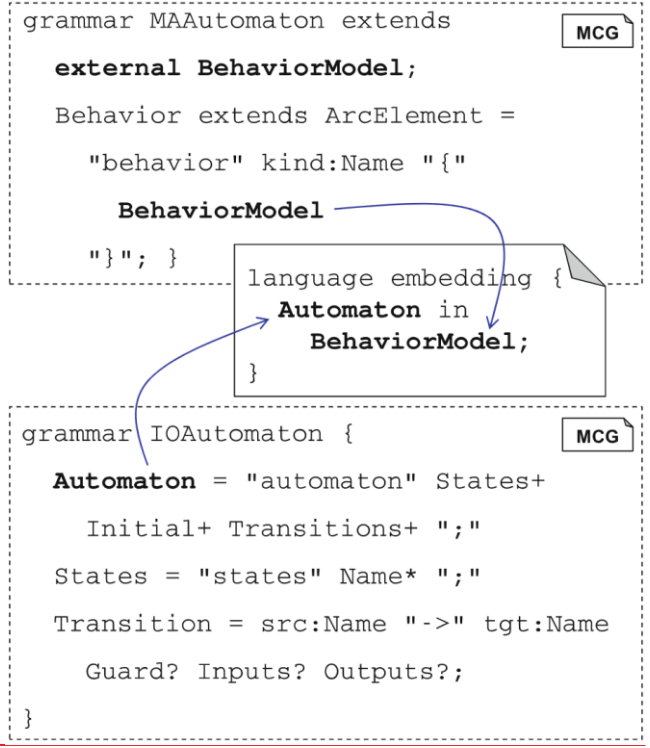
Which language composition operators exist?

Operators on Syntax

Grammar Embedding [14,22,25,32]

Language extension operator on syntax

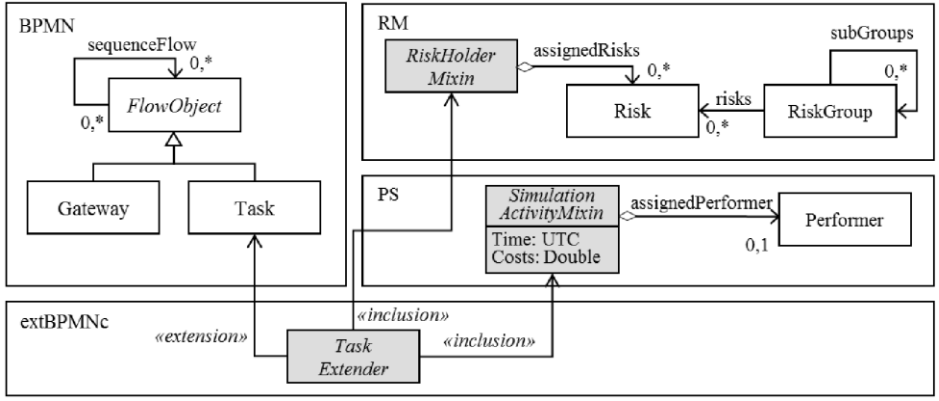
- Parameters: A production of a base grammar and a production of a client grammar
- Result: A new grammar in which the production of the base grammar is extended with an alternative containing the client grammar's production
- Effect on language instances (models): Models of the new grammar may use base and client grammar production instances in the same model.
- Additional: Explicit extensions (e.g., keyword `interface` in MontiCore)
- Technological space: MontiCore, SugarJ



Metamodel Mixins [71]

Language extension operator on syntax

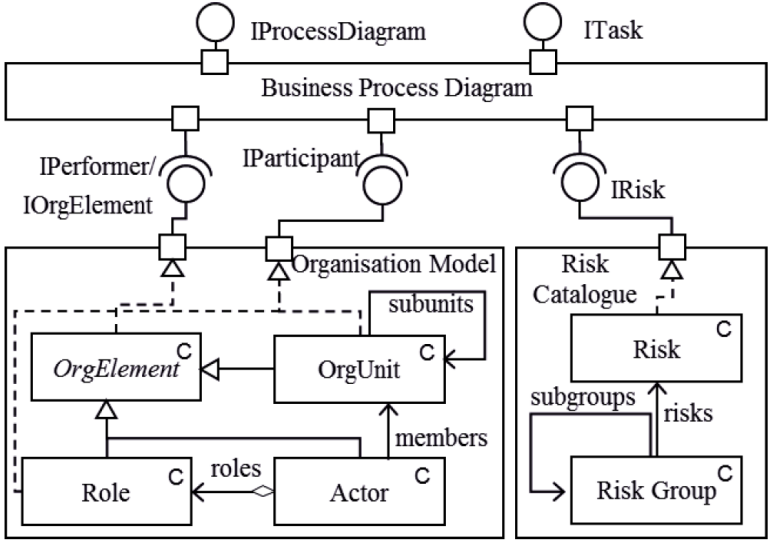
- Parameters: A parent metamodel, a mixin metamodel (abstract), concepts of the mixin metamodel that should be added to the parent metamodel
- Result: A metamodel featuring concepts from the parent metamodel including the mixin element
- Additional: The result of a mixin cannot be used as a mixin element
- Technological space: ADOxx



Metamodel Fragment Composition [70]

Language extension operator on syntax

- Metamodel Fragment is a container for a metamodel exposing provided and required interfaces
  - Provided interface exposes metaclasses
  - Required interface demands implementation by another metamodel fragment
- Parameter: Mapping of client's provided interface class to base fragment's required interface
- Result: Extended metamodel with realized required interface
- Technological space: n.a.



Annotation-Based Language Unification [53]

Language unification operator on syntax

- Parameters: Two textual syntax definition
- Result: Unified syntax by using annotations in the base language realizing concepts of the client language
- Base language must support annotations
- Technological space: n.a.

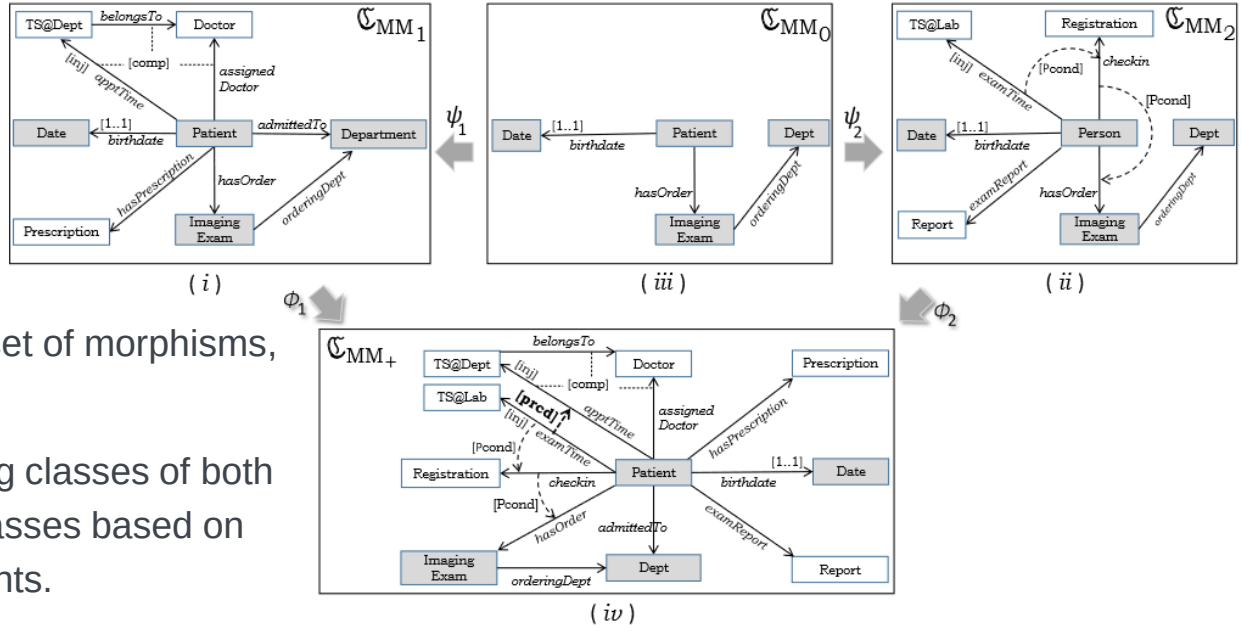
```
public class Person {
  private int id; //…
}
```

```
<entity class="model.Person" name="Person">
<table name="PERSON"/>
<attributes>
 <id name="id">
  <column name="IDENTIFIER"/>
 </id>
...
</entity>
```

```
@Entity(name = "Person")
@Table(name = "PERSON")
public class Person {
@Id
@Column(name = "IDENTIFIER")
private int id;
//…
}
```

Metamodel Merging [51,56]

Language unification operator on syntax



- Parameters: Two metamodels, set of morphisms, set of constraints
- Result: One metamodel featuring classes of both metamodel including merged classes based on name similarity fulfilling constraints.
- Technological space: n.a.

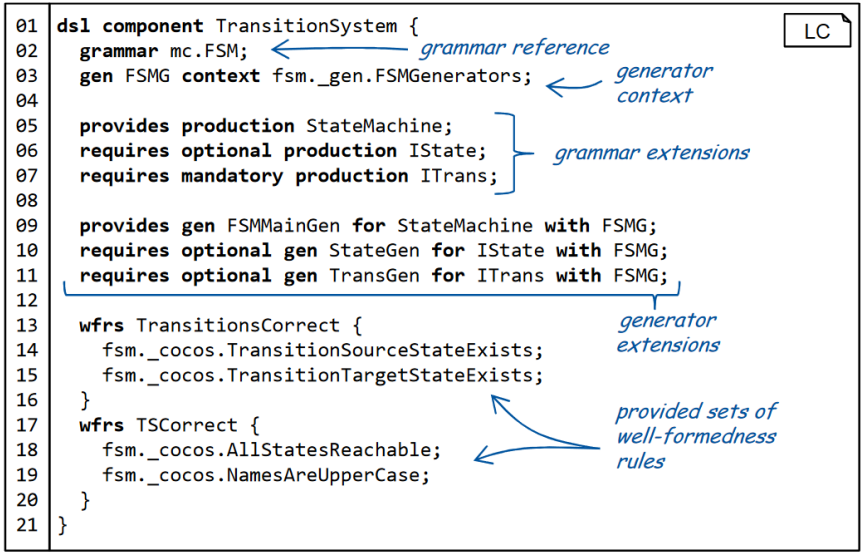
Which language composition operators exist?

Operators on Syntax and Semantics

Language Component Embedding [12, 15]

Language extension operator on syntax and semantics

- Language Component:
  - Comprises a grammar, well-formedness rules and a code generator
  - The interface consists of provided and required extensions making constituents explicit
- Parameters: Required interface point of a base language component, provided interface point of a client language component
- Result: A new language component and a new language embedding client provided extensions into base languages required extensions
- Technological space: MontiCore



Object-Oriented Language Unification [50]

Language unification operator on syntax´and semantics

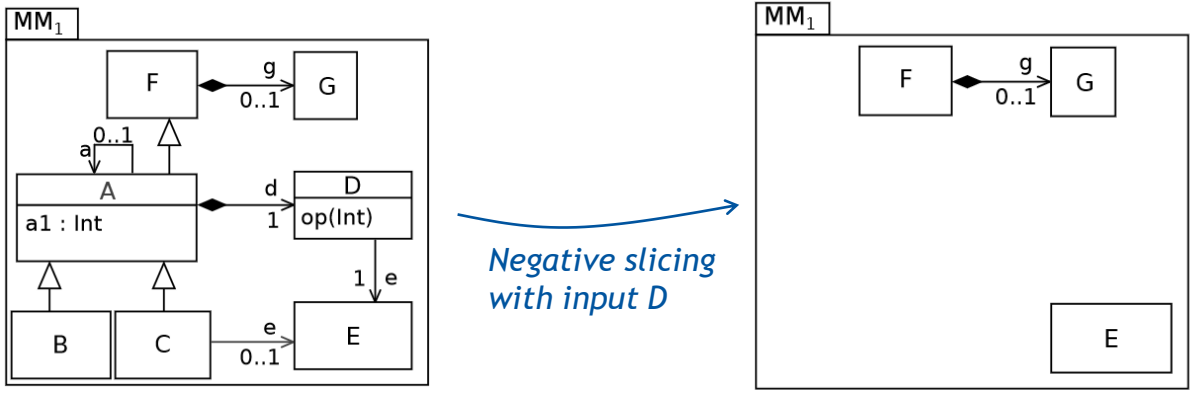
- Language module description
  - comprises definitions of CS, AS and computation rules (e.g. for semantics)
  - Computation rules are unambiguously related to a single AS element
- Parameters: Two language modules, a set of glue rules overriding rules of the input languages
- Result: A new language module extending from both modules and comprising overriding rules.
- Technological space: LISA

```
1   language RobotUnificationExprAdd extends Robot, ExprAdd {
2     rule extends start {
3       compute {}
4     }
5     rule overrides command {
6       COMMAND ::= left EXPR compute {
7         COMMAND.outx = COMMAND.inx – EXPR.val;
8         COMMAND.outy = COMMAND.iny; };
9       COMMAND ::= right EXPR compute {
10        COMMAND.outx = COMMAND.inx + EXPR.val;
11        COMMAND.outy = COMMAND.iny; };
12      COMMAND ::= up EXPR compute {
13        COMMAND.outx = COMMAND.inx;
14        COMMAND.outy = COMMAND.iny + EXPR.val; };
15      COMMAND ::= down EXPR compute {
16        COMMAND.outx = COMMAND.inx;
17        COMMAND.outy = COMMAND.iny – EXPR.val; };
18    }
19  }
```

Language Slicing [21]

Language restriction operator on syntax and semantics

- Parameters: Metamodel, elements to slice
- Result: New metamodel without the sliced elements
- Technological space: GEMOC / Melange

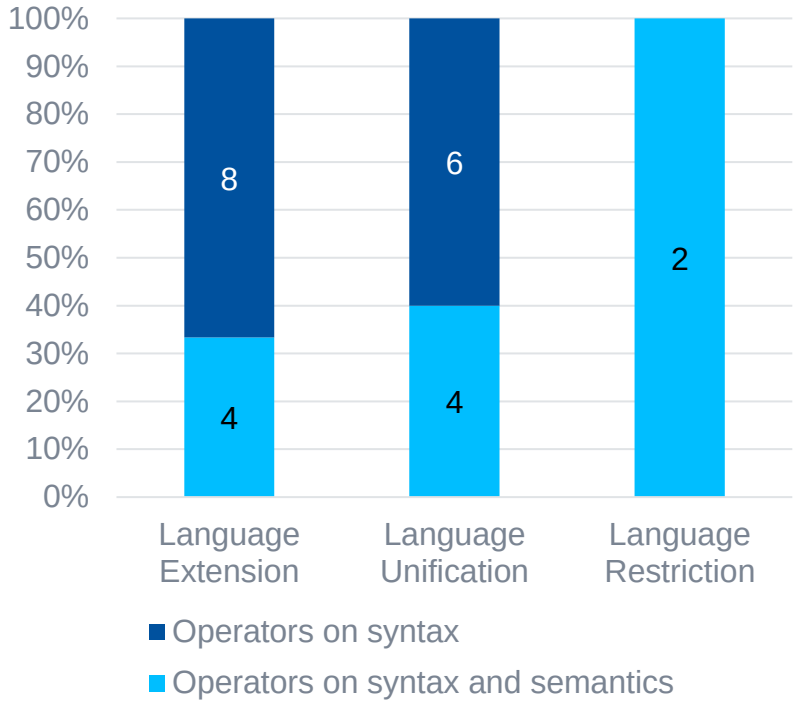


Research Question 2

Which language definition dimensions are supported?

Language Constituents Supported by Composition Operators

- 10/24 (41.6%) operators support composition of syntax **and** semantics
- Syntax mainly defined in grammars and metamodels
- Semantics realized via code generators, internal and external interpreters, and aspects
- No operators composing
  - Grammars and metamodels
  - Semantics in different realizations



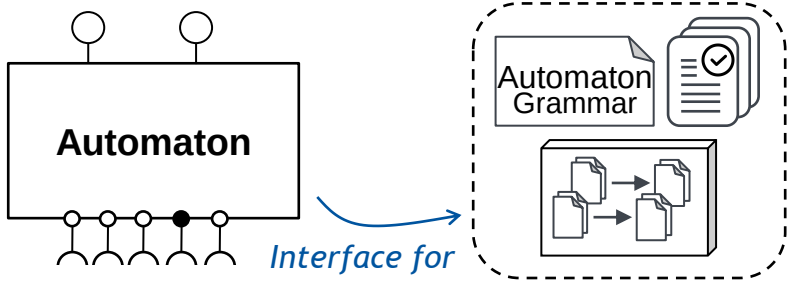
Research Question 3

Which properties do language composition operators have?

Blackbox Language Composition Operators

How much knowledge about language internals is necessary for composition?

- Language Component Embedding [12,15] and Unification [55]
  - Language components representing language fragments including syntax and semantics
  - Provided and required extensions expose the language's concepts
  - Composition via mapping between interfaces enables black-box language composition
- Metamodel Fragment Composition [70]
  - Fragments encapsulate metamodels
  - Expose metamodel elements via provided and required interfaces
  - Composition by binding interfaces



Traceability and Modularity of Composition

The original languages stay independent origins after composition

Modular

- Grammar Embedding [14, 22, 25, 32]:
  - Creates a new grammar where the base production is extended with the client production as an alternative
  - The original grammars are referenced in the resulting grammar
  - No information is lost

Non-modular

- Metamodel Merging [51, 56]:
  - Merges two metamodels into one
  - After the composition the concept's origins are not visible in the result anymore
- Metamodel Mixins [71]:
  - Takes a metamodel and a mixin model as input
  - Creates a new metamodel comprising metamodel classes and mixin classes
  - It is not clear which concepts origin from mixin

Closed Under Composition

Can the result of composition again be used as input for the operator?

- Metamodel Mixins [71]
  - Parameters: metamodel, mixin element
  - The resulting metamodel is not usable as mixin
- Language Union [18]
  - Merge new rules into an existing language
  - Rules do not need to be part of another language
  - Resulting language definition cannot be merged into an existing language
- Language Module Restriction [18, 63]
  - Extend a language without inheriting concepts
  - E.g., selecting concepts not to be reused

Challenges

Our observations lead to challenges for future investigations

1 Heterogeneous composition

- Composing languages across different technological spaces
- E.g., embedding a Neverlang language into a Xtext language

2 Black-box composition

- Hiding implementation details of languages
- Only three operators supported currently

3 Automated composition

- Minimizing the manual effort and white-box knowledge after the composition
- Relevant for black-box approaches

4 Alignment of operators

- Do we need this many composition operators?
- How similar are the operators?
- Which are the ones most frequently used?

Summary

- **Motivation:** Uncovering the current state of language composition ten years after the classification of “Language Composition Untangled”

- **Research Questions:**
  1. Which composition operators exist?
  2. Which language dimension are supported?
  3. Which properties do the operators have?

→ 8762 papers in initial search → 45 relevant

- **Results:**
  - We found 24 operators
    - Extension of syntax (8) and semantics (4)
    - Unification of syntax (6) and semantics (4)
    - Restriction on syntax and semantics (2)
  - 2/3 of the operators are technology-specific

a) Black-box

b) Traceability and modularity

c) Closed under composition

University of Stuttgart
Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW)

Jérôme Pfeiffer

Research Assistant

email jerome.pfeiffer@isw.uni-stuttgart.de

phone +49 (0) 711 685-94500

University of Stuttgart
Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW)
Seidenstrasse 36 • 70174 Stuttgart • Germany

Literature

[1] Sebastian Erdweg, Paolo G. Giarrusso, and Tillmann Rendel. 2012. Language Composition Untangled. In Proceedings of the Twelfth Workshop on Language Descriptions, Tools, and Applications (Tallinn, Estonia) (LDTA '12). Association for Computing Machinery, New York, NY, USA, Article 7, 8 pages.

[14] Arvid Butting, Katrin Hölldobler, Bernhard Rumpe, and AndreasWortmann. 2021. Compositional Modelling Languages with Analytics and Construction Infrastructures Based on Object-Oriented Techniques - The MontiCore Approach. In Composing Model-Based Analysis Tools. Springer, 217–234.

[22] Lukas Diekmann and Laurence Tratt. 2013. Parsing composed grammars with language boxes. In Workshop on Scalable Language Specifications.

[25] Sebastian Erdweg and Felix Rieger. 2013. A framework for extensible languages. In Proceedings of the 12th international conference on Generative programming: concepts & experiences. 3–12.

[32] Arne Haber, Markus Look, Pedram Mir Seyed Nazari, Antonio Navarro Perez, Bernhard Rumpe, Steven Völkel, and Andreas Wortmann. 2015. Composition of heterogeneous modeling languages. In International Conference on Model-Driven Engineering and Software Development. Springer, 45–66.

[58] Christoph Rieger, Martin Westerkamp, and Herbert Kuchen. 2018. Challenges and Opportunities of Modularizing Textual Domain-Specific Languages. MODELSWARD (2018), 387–395.

[70] Srđan Živković and Dimitris Karagiannis. 2015. Towards metamodelling-in-the-large: Interface-based composition for modular metamodel development. In Enterprise, Business-Process and Information Systems Modeling. Springer, 413–428.

[53] Milan Nosál, Matúš Sulír, and Ján Juhár. 2016. Language composition using source code annotations. Computer Science and Information Systems 13, 3 (2016), 707–729.

[9] Benjamin Braatz and Christoph Brandt. 2014. A framework for families of domain-specific modelling languages. Software & Systems Modeling 13, 1 (2014), 109–132.

[51] Bart Meyers, Antonio Cicchetti, Esther Guerra, and Juan De Lara. 2012. Composing textual modelling languages in practice. In Proceedings of the 6th International Workshop on Multi-Paradigm Modeling. 31–36.

Literature

[56] Fazle Rabbi, Yngve Lamo, and Lars Michael Kristensen. 2017. A Model Driven Engineering Approach for Heterogeneous Model Composition. In International Conference on Model-Driven Engineering and Software Development. Springer, 198–221.

[12] Arvid Butting, Robert Eikermann, Oliver Kautz, Bernhard Rumpe, and Andreas Wortmann. 2018. Modeling language variability with reusable language components. In Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1. 65–75.

[15] Arvid Butting, Jerome Pfeiffer, Bernhard Rumpe, and Andreas Wortmann. 2020. A compositional framework for systematic modeling language reuse. In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. 35–46.

[50] Marjan Mernik. 2013. An object-oriented approach to language compositions for software language engineering. Journal of Systems and Software 86, 9 (2013), 2451–2464.

[21] Thomas Degueule, Benoit Combemale, Arnaud Blouin, Olivier Barais, and Jean-Marc Jézéquel. 2015. Melange: A meta-language for modular and reusable development of dsls. In Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering. 25–36.

[55] Jérôme Pfeiffer and Andreas Wortmann. 2021. Towards the Black-Box Aggregation of Language Components. In 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). IEEE, 576–585.

[71] Srźan źivkoviź and Dimitris Karagiannis. 2016. Mixins and Extenders for Modular Metamodel Customisation. In Proceedings of the 18th International Conference on Enterprise Information Systems. 259–270.

[18] Matteo Cimini. 2020. On the effectiveness of higher-order logic programming in language-oriented programming. In International Symposium on Functional and Logic Programming. Springer, 106–123.

[63] Edoardo Vacchi and Walter Cazzola. 2015. Neverlang: A framework for feature-oriented language development. Computer Languages, Systems & Structures 43 (2015), 1–40.