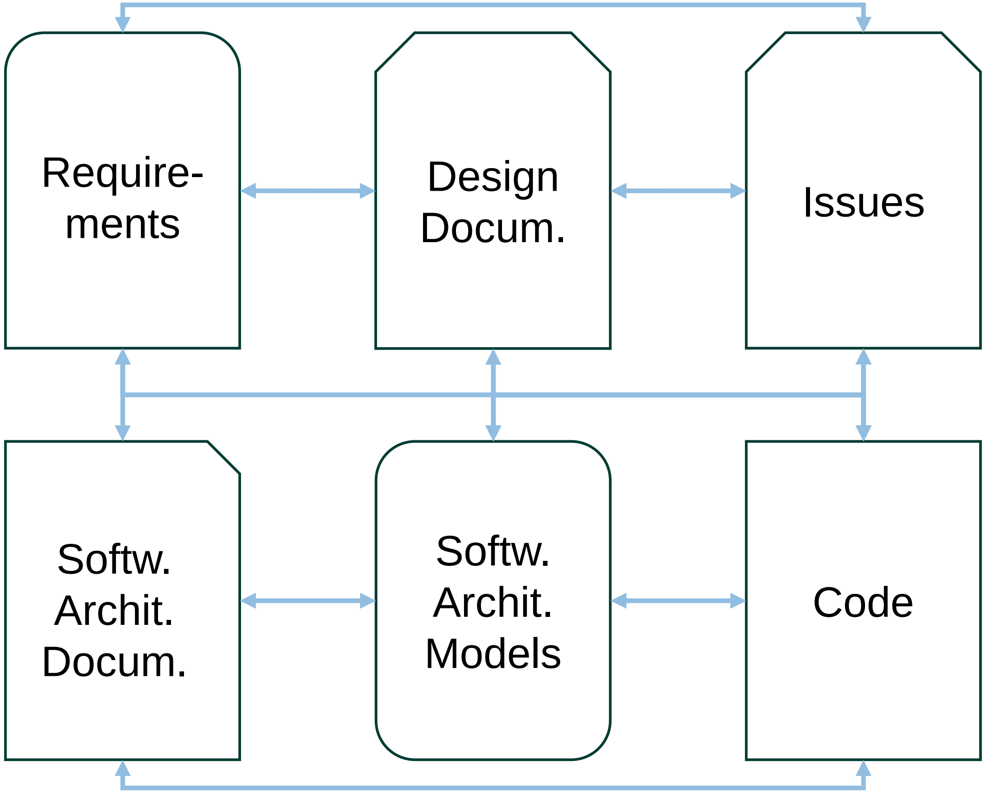
Karlsruhe Institute of Technology

Recovering Trace Links Between Software Documentation And Code

Jan Keim, Sophie Corallo, Dominik Fuchß, Tobias Hey, Tobias Telge, Anne Koziolek

ICSE 2024, Lisbon, Portugal

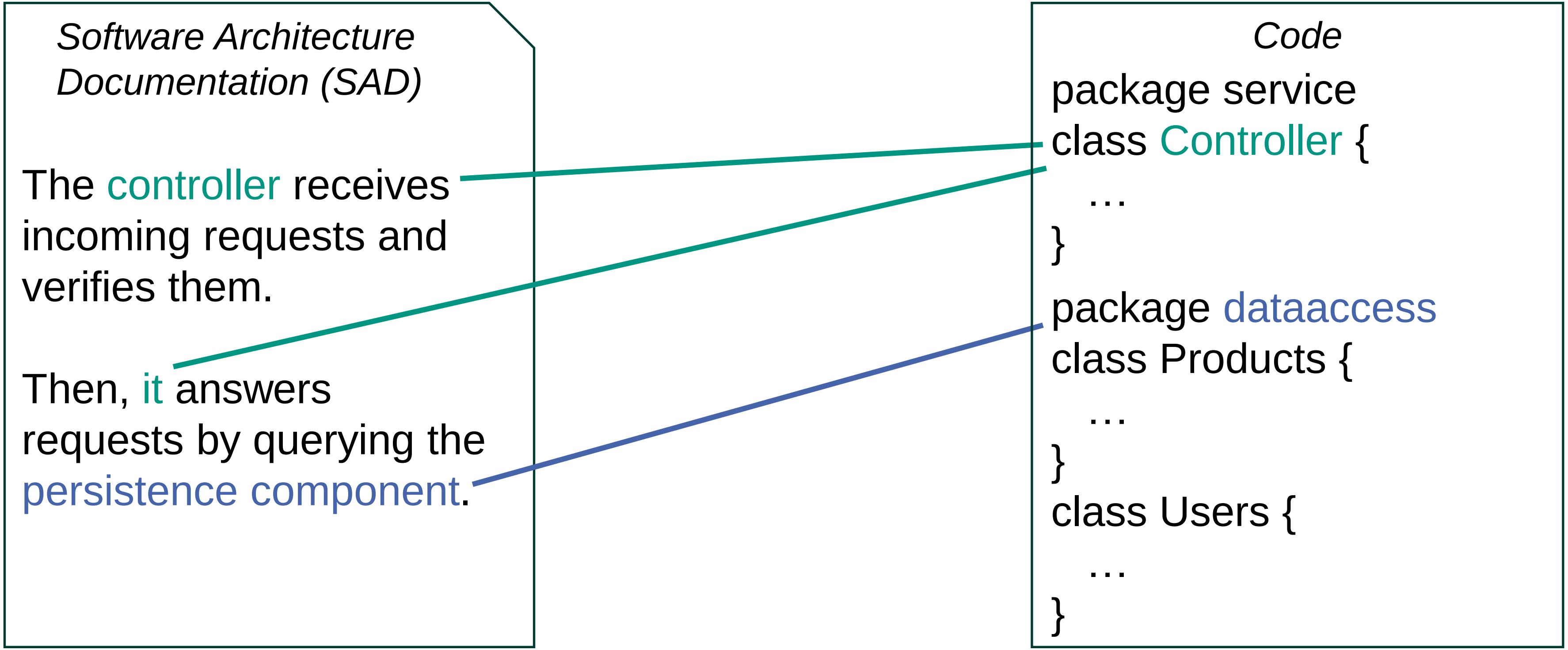
Why should we care about trace links?



Trace links are evidently useful for

- Software Maintenance
- Bug Localization
- Change Impact An.
- System Security
- …

Traceability Link Recovery is difficult



Software Architecture Documentation (SAD)

The controller receives incoming requests and verifies them.

Then, it answers requests by querying the persistence component.

Code

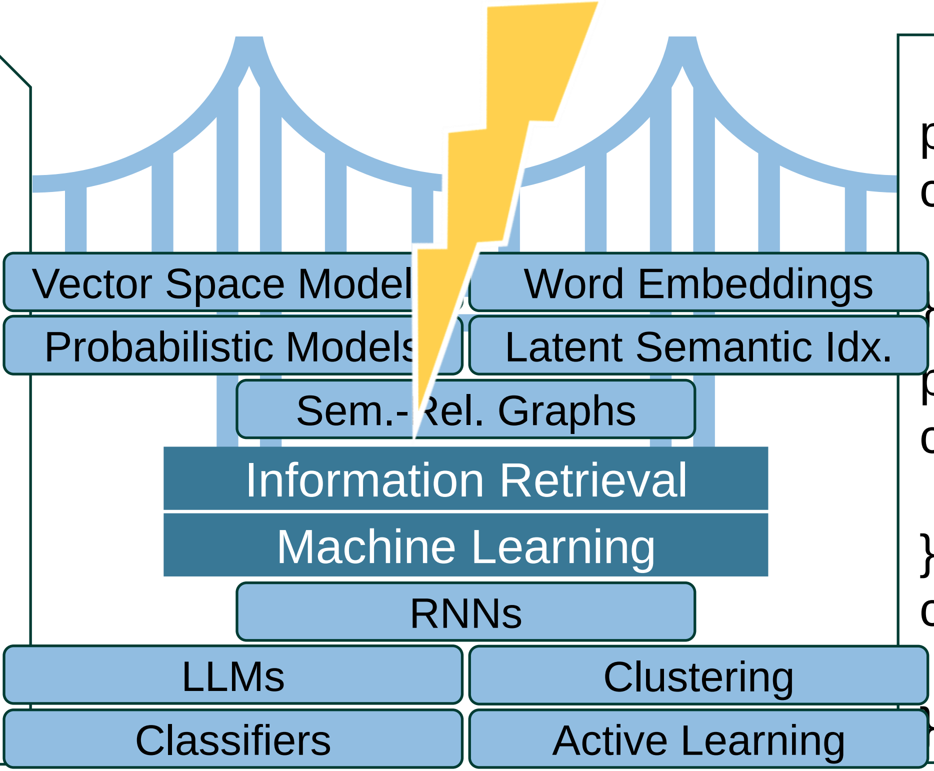
```
package service
class Controller {
   …
}
package dataaccess
class Products {
   …
}
class Users {
   …
}
```

Approaches need to bridge the semantic gap

Software Architecture Documentation (SAD)

The controller receives incoming requests and verifies them.

Then, it answers requests by querying the persistence component.



Code

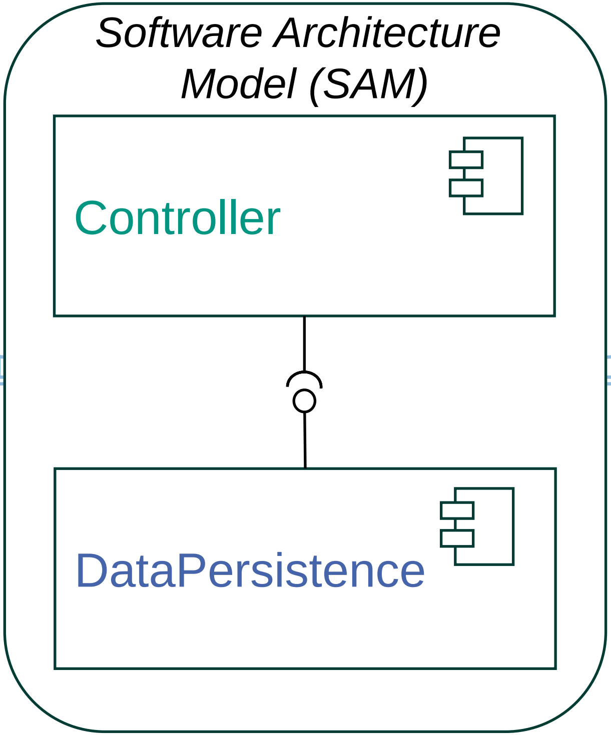
```
package service
class Controller {
  …
}

package dataaccess
class Products {
  …
}
class Users {
  …
}
```

Intermediate Artifacts to reduce the gap

Software Architecture Documentation (SAD)

The controller receives incoming requests and verifies them.

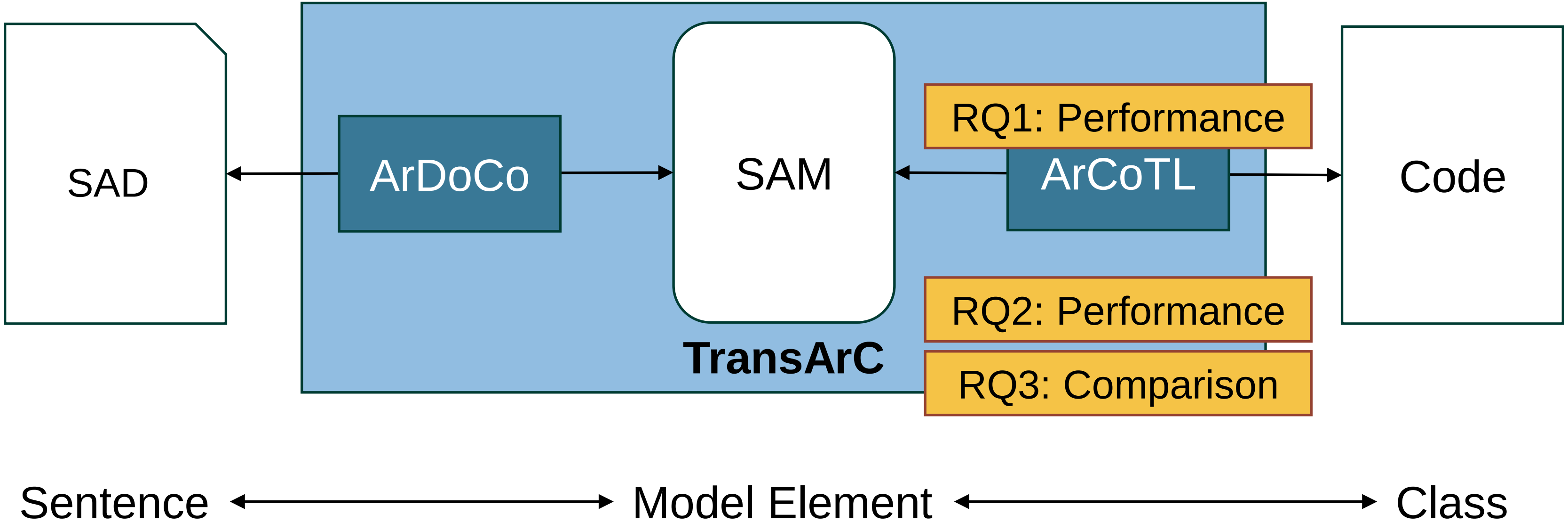
Then, it answers requests by querying the persistence component.



Code

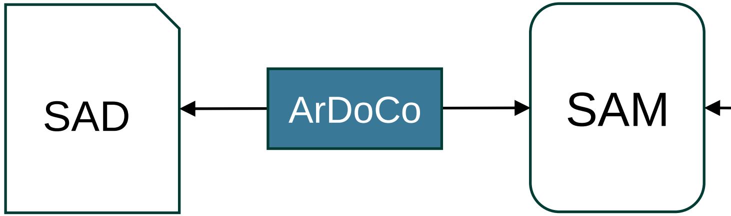
```
package service
class Controller {
   …
}
package dataaccess
class Products {
   …
}
class Users {
   …
}
```

Transitively combine TLR approaches



ArDoCo – TLR between SAD and SAM

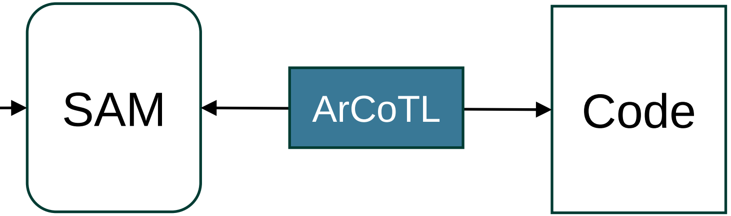
- Existing work [Keim et al. 2021], [Keim et al. 2023]
- Entity identification in text
- Heuristic-based
- Tracing entities to model based on similarity

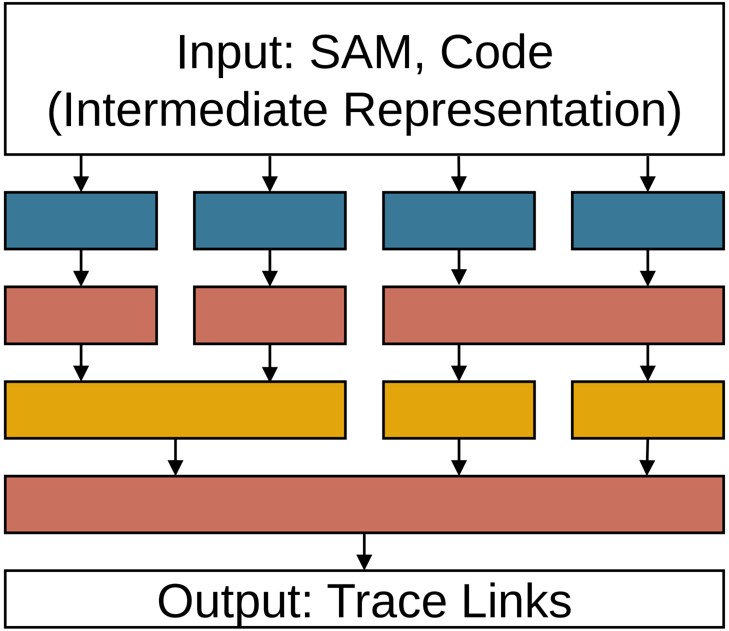


[Keim et al. 2021] „Trace Link Recovery for Software Architecture Documentation“, ECSA 2021
[Keim et al. 2023] „Detecting Inconsistencies in Software Architecture Documentation Using Traceability Link Recovery”, ICSA 2023

ArCoTL – TLR between SAM and Code

- Transform input into intermediate representations
  - Simplified architecture model
  - Code model
- Computational graph
- Heuristics-based
  - Standalone heuristics
  - Dependent heuristics
- Aggregators and filters



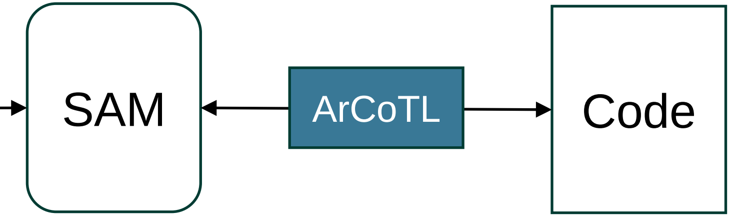


Evaluation Setup

Artifact Type		MediaStore	TeaStore	TEAMMATES	BigBlueButton (BBB)	JabRef
SAD	#Sentences	37	43	198	85	13
SAM	#Model Elements	23	19	16	24	6
Code	#Files	97	205	832	547	1,979
SAM-Code	#Trace Links	60	164	1,616	730	1,956
SAD-Code	#Trace Links	50	707	7,610	1,295	8,240

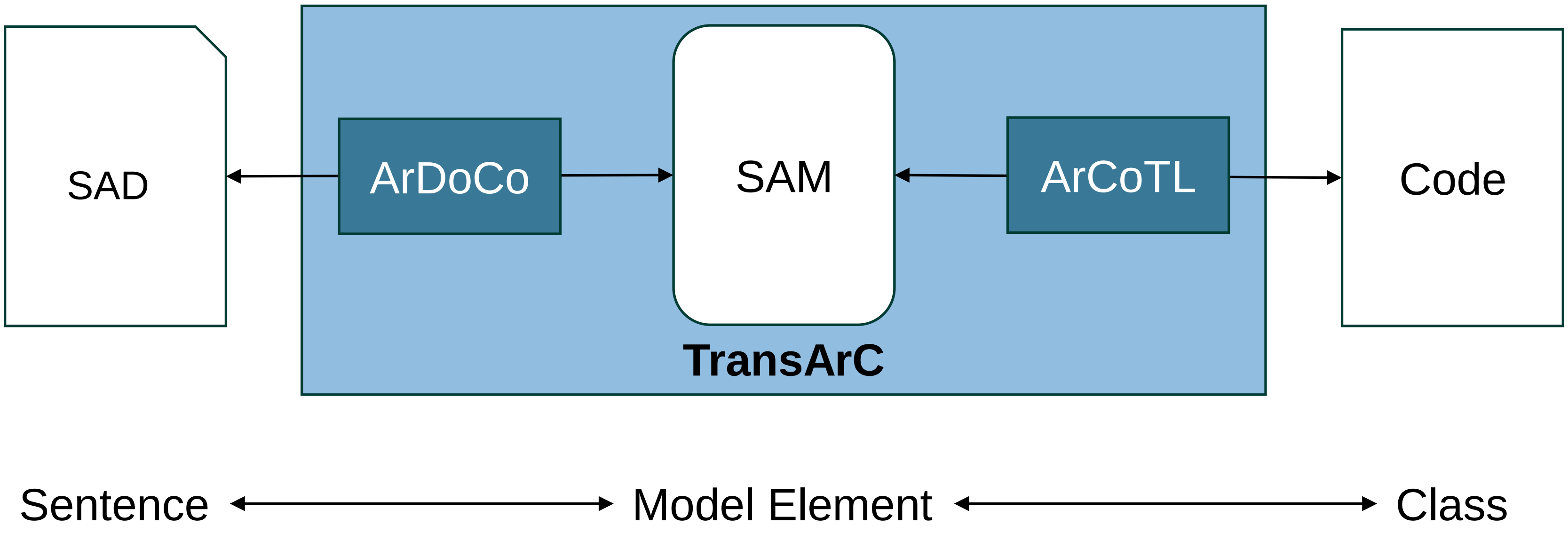
ArCoTL: Evaluation Results

Project	Precision	Recall	F1-Score
MediaStore	0.98	1.00	0.99
TeaStore	0.98	0.98	0.98
TEAMMATES	1.00	1.00	1.00
BigBlueButton	0.94	0.96	0.95
JabRef	1.00	1.00	1.00
Average	**0.98**	**0.99**	**0.98**



RQ1 (Performance)
ArCoTL achieves excellent results

TransArC – Transitive Trace Links



TransArC: Evaluation

- Same evaluation projects
- Four Baseline Approaches for comparison
  - TAROT: IR-based for Requirements-to-Code-TLR [Gao et al. 2022]
  - FTLR: IR-based for Requirements-to-Code-TLR [Hey et al. 2021]
  - CodeBERT: LLM trained to map method documentation and code [Feng et al. 2020]
  - ArDoCode: ArDoCo that treats code as model

[Gao et al. 2022] „Using Consensual Biterms from Text Structures of Requirements and Code to Improve IR-Based Traceability Recovery“, ASE 2022
[Hey et al. 2021] „Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations”, ICSME 2021
[Feng et al. 2020] „ CodeBERT: A Pre-Trained Model for Programming and Natural Languages”, EMNLP 2020

TransArC: Evaluation Results (F1-Scores)

Approach	MediaStore	TeaStore	Teammates	BBB	JabRef	Avg.	W. Avg.
TAROT	0.13	0.27	0.11	0.10	0.49	0.22	0.29
FTLR	0.19	0.21	0.10	0.07	0.48	0.21	0.28
CodeBERT	0.17	0.36	0.12	0.12	0.61	0.28	0.36
ArDoCode	0.09	0.31	0.53	0.13	0.80	0.37	0.62
TransArc	**0.68**	**0.83**	**0.80**	**0.84**	**0.94**	**0.82**	**0.87**

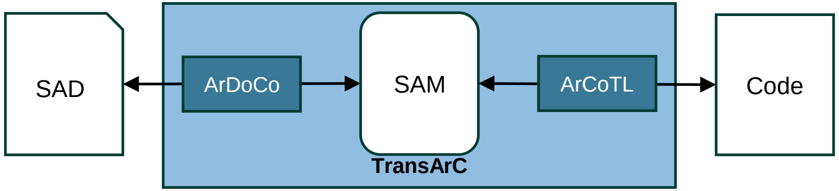
+122%

+40%

<u>RQ2 (Performance)</u>
TransArC achieves excellent results

<u>RQ3 (Comparison)</u>
TransArC significantly outperforms the baseline approaches

Conclusion

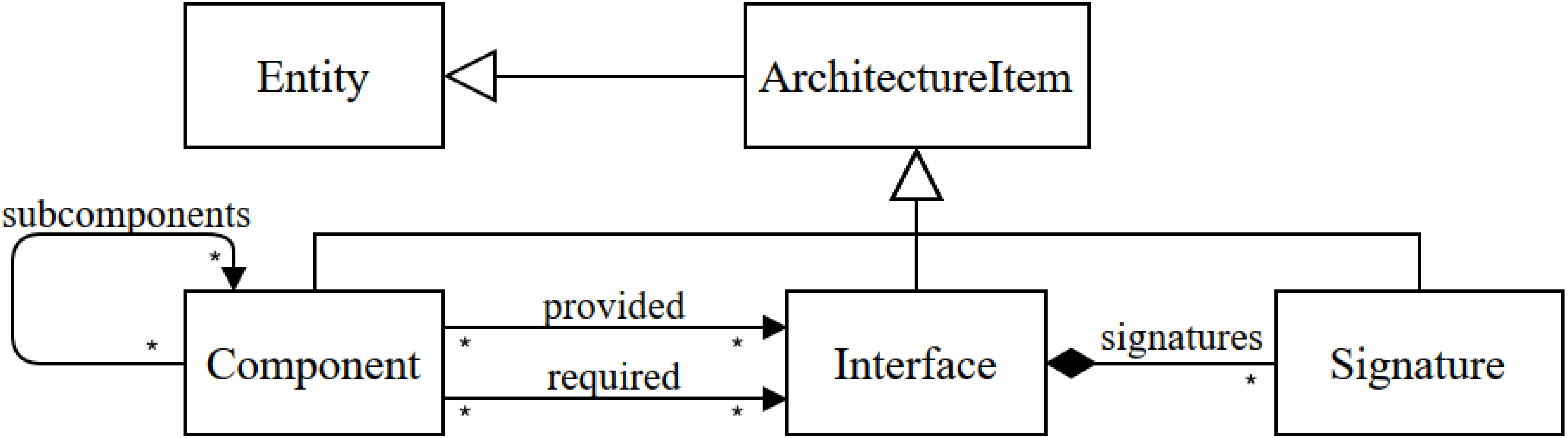


- TransArC transitively combines ArDoCo and the new approach ArCoTL to try to better bridge the semantic gap
- In the evaluation,
  - ArCoTL performs excellently (avg. F1: 0.98)
  - TransArC significantly outperforms the baseline approaches (avg. F1: 0.82 → +122%)
- Outlook
  - Evaluate on more (different) projects
  - Experiment with other kinds of artifacts
  - Explore further intermediate artifacts
  - Combine our approach(es) with others

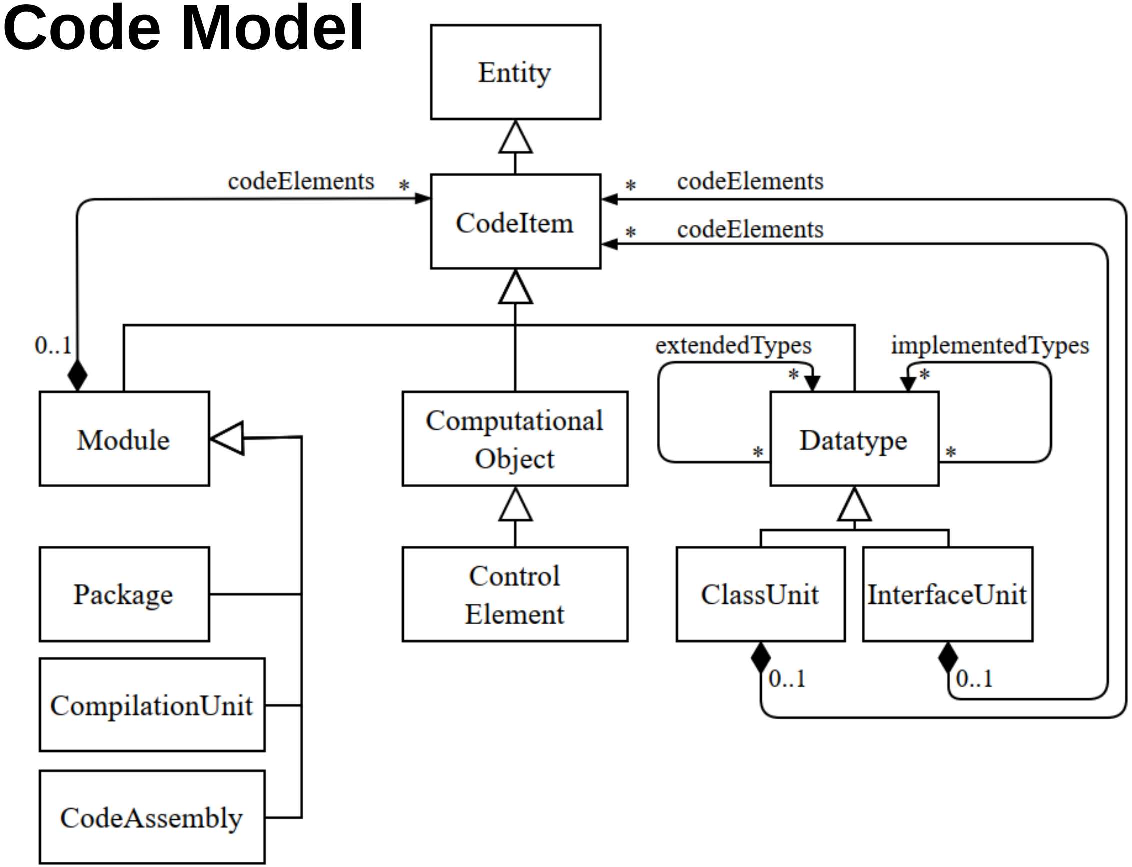
ardoco.de/c/icse24

Appendix

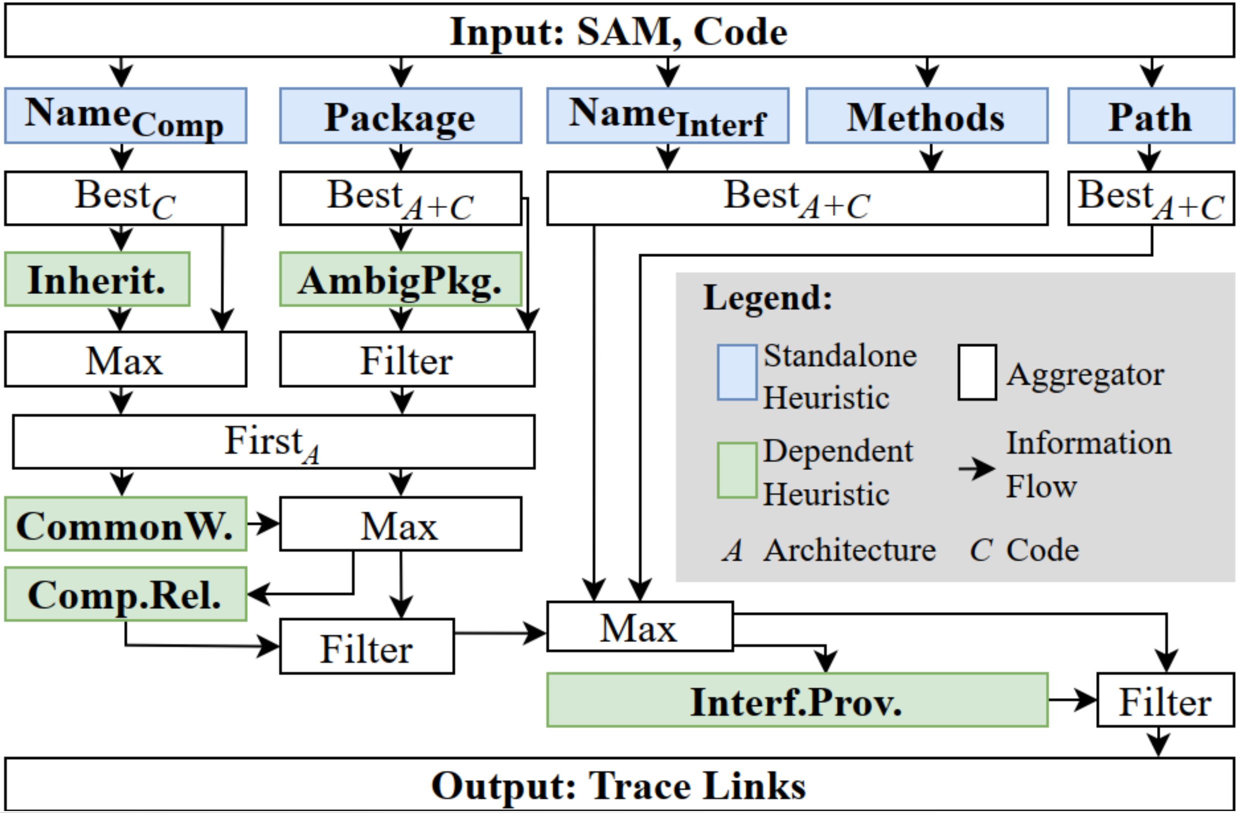
Appendix: Architecture Model



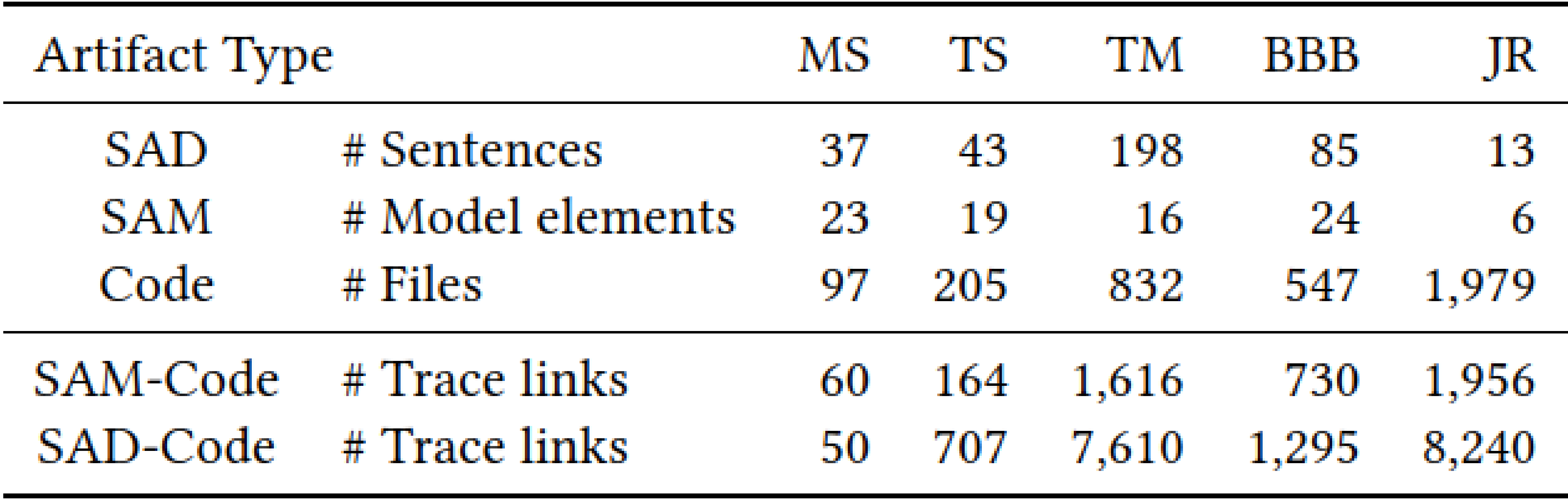
Appendix: Code Model



Appendix: Computational Graph

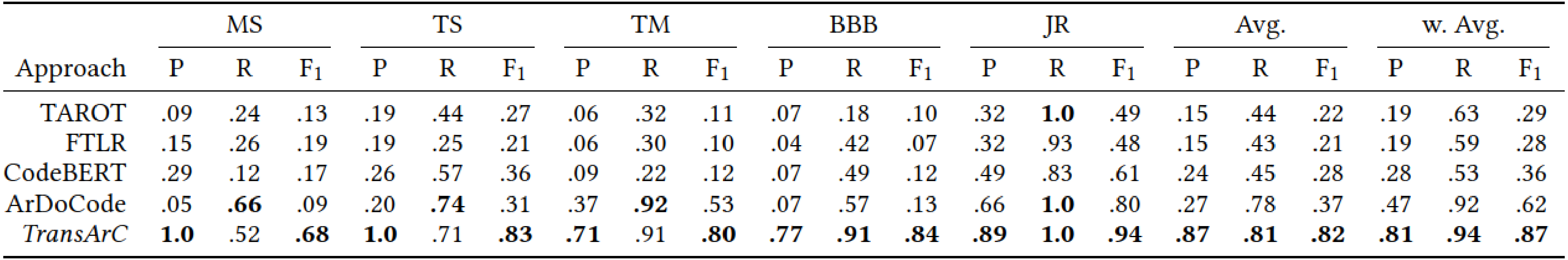


Appendix: Evaluation Data



Artifact Type		MS	TS	TM	BBB	JR
SAD	# Sentences	37	43	198	85	13
SAM	# Model elements	23	19	16	24	6
Code	# Files	97	205	832	547	1,979
SAM-Code	# Trace links	60	164	1,616	730	1,956
SAD-Code	# Trace links	50	707	7,610	1,295	8,240

Appendix: Detailed Results TransArC



	MS			TS			TM			BBB			JR			Avg.			w. Avg.		
Approach	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
TAROT	.09	.24	.13	.19	.44	.27	.06	.32	.11	.07	.18	.10	.32	**1.0**	.49	.15	.44	.22	.19	.63	.29
FTLR	.15	.26	.19	.19	.25	.21	.06	.30	.10	.04	.42	.07	.32	.93	.48	.15	.43	.21	.19	.59	.28
CodeBERT	.29	.12	.17	.26	.57	.36	.09	.22	.12	.07	.49	.12	.49	.83	.61	.24	.45	.28	.28	.53	.36
ArDoCode	.05	**.66**	.09	.20	**.74**	.31	.37	**.92**	.53	.07	.57	.13	.66	**1.0**	.80	.27	.78	.37	.47	.92	.62
TransArC	**1.0**	.52	**.68**	**1.0**	.71	**.83**	**.71**	.91	**.80**	**.77**	**.91**	**.84**	**.89**	**1.0**	**.94**	**.87**	**.81**	**.82**	**.81**	**.94**	**.87**