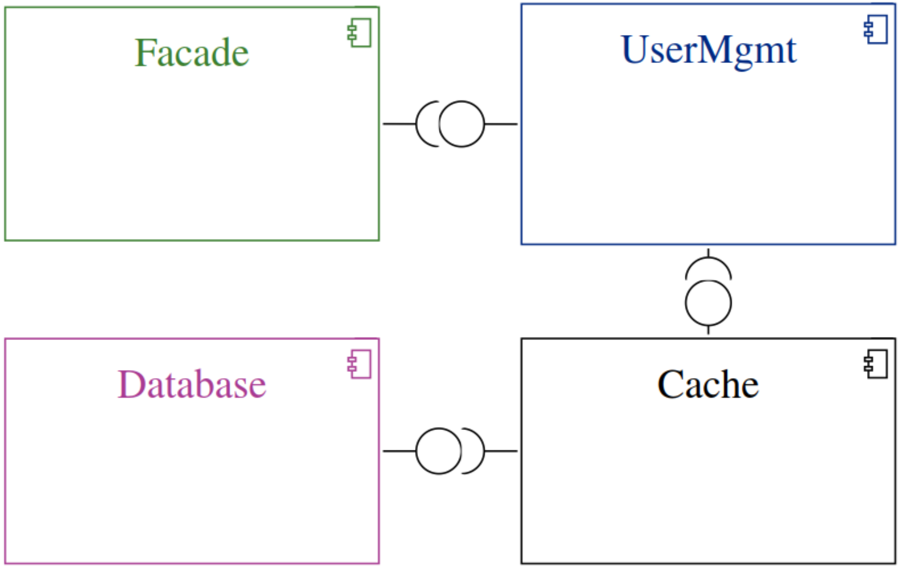
# Detecting Inconsistencies in Software Architecture Documentation Using Traceability Link Recovery

Jan Keim, Sophie Corallo, Dominik Fuchß, Anne Koziolek

ICSA23 – L'Aquila

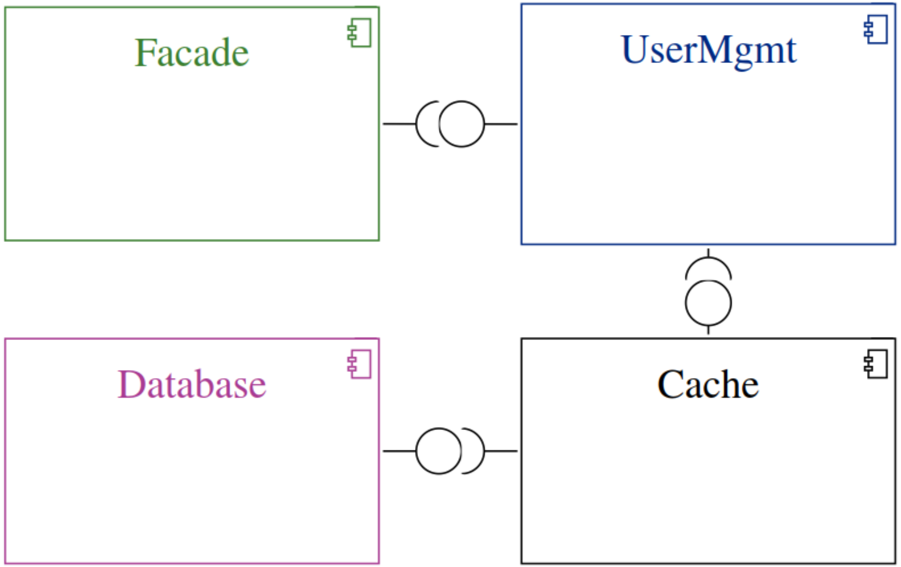
# Software Architecture Documentation (SAD)




1) The system adheres to layered architecture.
2) The Facade is the entry point to the service.
3) It passes calls to the user management.
4) The user management then accesses the DB.
5) The Common component contains utility functionality.

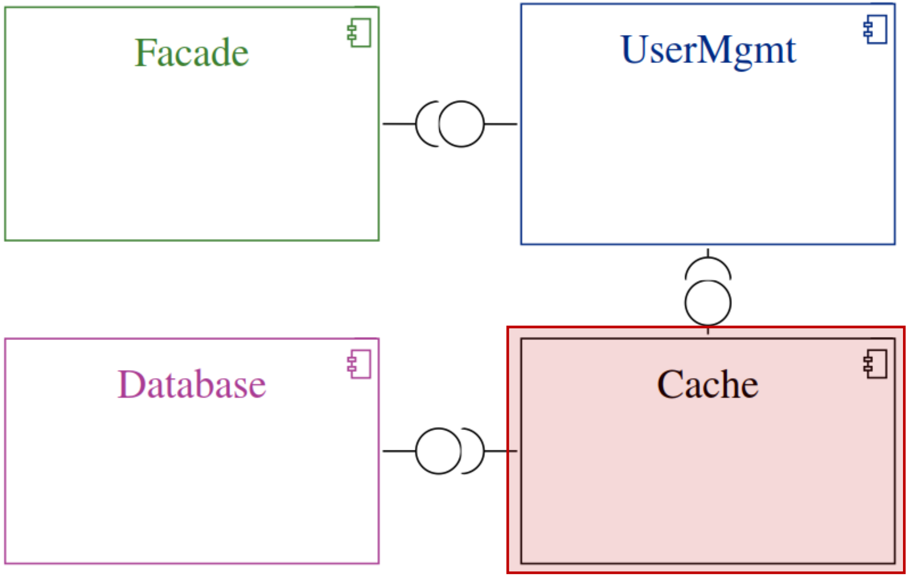
# Connecting SADs with Traceability Link Recovery

using the TLR approach SWATTR [Keim2021]



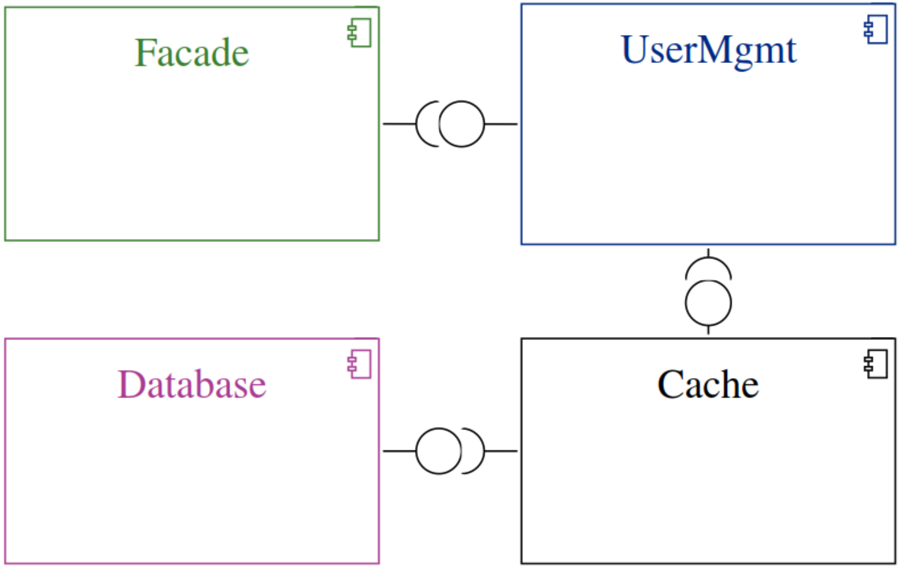

1) The system adheres to layered architecture.
2) The Facade is the entry point to the service.
3) It passes calls to the user management.
4) The user management then accesses the DB.
5) The Common component contains utility functionality.

# Unmentioned Model Elements




1) The system adheres to layered architecture.
2) The Facade is the entry point to the service.
3) It passes calls to the user management.
4) The user management then accesses the DB.
5) The Common component contains utility functionality.

# Missing Model Elements




1) The system adheres to layered architecture.
2) The Facade is the entry point to the service.
3) It passes calls to the user management.
4) The user management then accesses the DB.
5) The Common component contains utility functionality.

# Research Questions & Contributions

1) To what extent do changes to the previous approach SWATTR improve the performance for Traceability Link Recovery?
2) How does the approach perform for detecting unmentioned model elements?
3) How well does the approach detect missing model elements?

Contributions

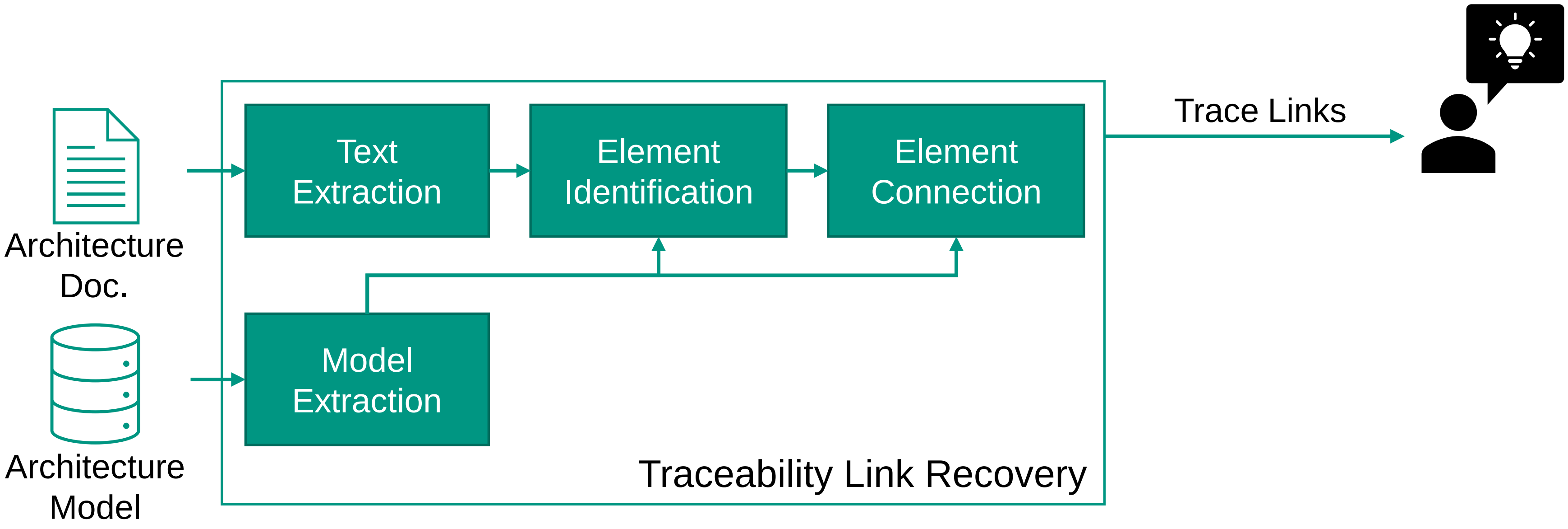
1. Extending TLR and add capabilities to identify inconsistencies
2. Novel approach (ArDoCo) to identify inconsistencies
3. Replication package

# Related Work

- Inconsistency Detection between API/Code documentation and Code,
  e.g., Kim & Kim 2016
- Inconsistency Detection for requirements,
  e.g., Fantechi & Spinicci 2005, Kamalrudin et al. 2010
- Inconsistency Detection for Software Architecture,
  e.g., Lytra & Zdun 2014

→ No work looking at inconsistencies between natural language software architecture documentations and software architecture models

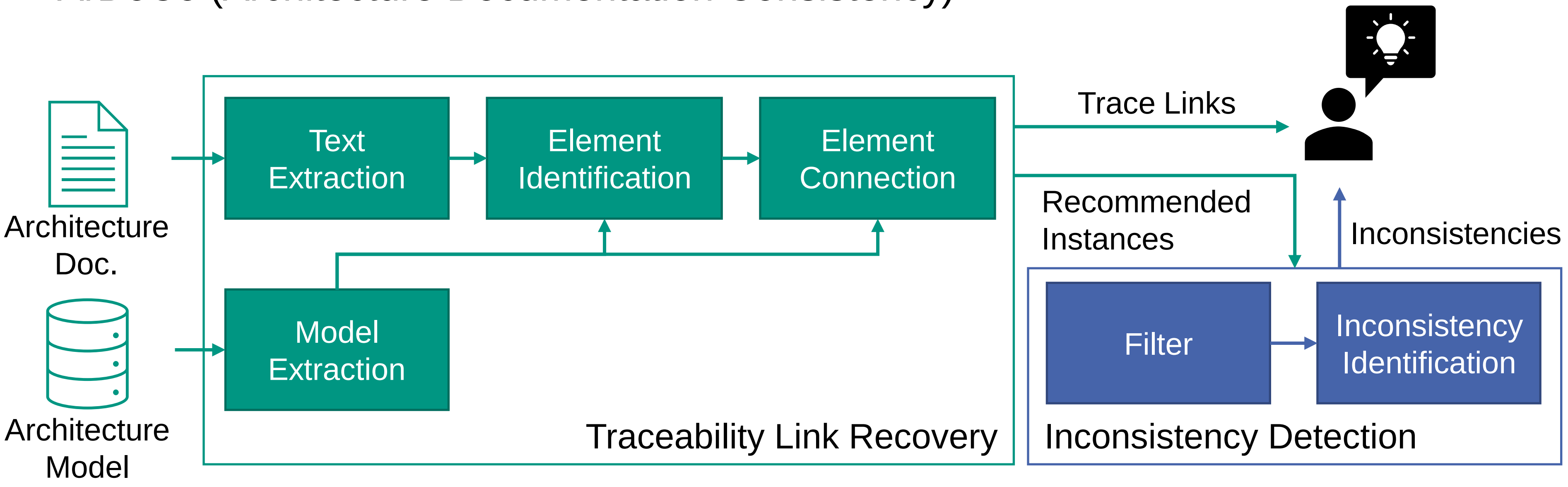
# Background: SWATTR




[Keim2021]

# Our Approach

ArDoCo (Architecture Documentation Consistency)



# Adaptations to Traceability Link Recovery

- Handling of compound nouns
  → Making use of (noun) phrases
  → Adding and adapting heuristics for phrases
- Handling of project’s name

- Slightly updated use of word similarity metrics:
  Combination of Jaro-Winkler and Levenshtein distance

# Detecting Unmentioned Model Elements

Karlsruhe Institute of Technology

- Look for absent trace links for model elements (e.g., components)
- Each model element needs to have at least *one* trace link

- Configuration options to adjust to needs
  - Minimum number of needed trace links
  - Types of model elements that are checked (e.g., components, interfaces)
  - Regex-based whitelist

# Detecting Missing Model Elements

- Make use of Recommended Instances (RIs) of SWATTR

→ RIs without a trace link are (potential) inconsistencies

- Problem: SWATTR detects many RIs to increase recall for TLR
- Therefore, filtering RIs based on
  - (dynamic) threshold regarding overall confidence
  - confidence for name and type of the RI
  - Number of occurrences
  - Unwanted words: general and project/domain-specific blacklists

# Evaluation Projects

| Project | Language (kLOC) | | Forks | Contributors |
|---|---|---|---|---|
| **MediaStore (MS)** | Java | 4 | - | - |
| **TeaStore (TS)** | Java | 12 | 0.1k | ~ 15 |
| **TEAMMATES (TM)** | Java | 91 | 2.6k | ~ 500 |
| | TypeScript | 54 | | |
| **BigBlueButton (BBB)** | JavaScript | 69 | 5.8k | ~ 180 |
| | JSX | 47 | | |
| | Scala | 22 | | |
| | Java | 21 | | |
| **JabRef (JR)** | Java | 157 | 2.0k | ~ 490 |

# Evaluation: Traceability Link Recovery

Karlsruhe Institute of Technology

RQ1: To what extent do changes to the previous approach SWATTR improve the performance for TLR?

- Goals
  - To measure how well we can link sentences that mention a certain model element to the model elements
  - To compare the results
- Process
  - Comparison with gold standard

- Metrics
  - Precision, Recall, F1 – Score
  - Accuracy, Specificity
  - Φ – Coefficient
  - Average, Weighted Average

# Evaluation: Traceability Link Recovery

| Project | Precision | | Recall | | F1-Score | | Accuracy | |
|---|---|---|---|---|---|---|---|---|
| MS | | 1.0 | | .62 | | .77 | | .98 |
| TS | 1.0 | 1.0 | .93 | .74 | .97 | .85 | 1.0 | .99 |
| TM | .52 | .56 | .70 | .90 | .60 | .69 | .97 | .97 |
| BBB | .81 | .88 | .62 | .83 | .70 | .85 | .98 | .99 |
| JR | .82 | .90 | 1.0 | 1.0 | .90 | .95 | .97 | .97 |
| w. Avg. | .80 | .83 | .79 | .82 | .79 | .80 | .98 | .98 |
| | .81 | | .81 | | .80 | | .98 | |

Historic

Current

# Evaluation: Comparing TLR results

- Baseline Approach
  - Assumption: Elements that should be linked have equal or really similar naming
  - Extracts n-grams for sentences and model elements (n = {1,2,3})
  - Compares n-grams from text and models using normalized Levenshtein distance
  - Create TLs if comparison shows (high) similarity

| Approach | Precision* | Recall* | F1-Score* | Accuracy* |
|---|---|---|---|---|
| Baseline | .80 | .37 | .50 | .89 |
| SWATTR | .49 | .63 | .52 | .94 |
| ArDoCo | **.81** | **.81** | **.80** | **.98** |

* weighted Average

# Evaluation: Inconsistency Detection - UMEs

RQ2: How does the approach perform for detecting unmentioned model elements?

- Goal
  - To measure how well we can detect unmentioned model elements
- Process
  - Comparison with gold standard

- Metrics
  - Precision, Recall, F1 – Score
  - Accuracy, Specificity
  - Φ – Coefficient
  - Average, Weighted Average

# Evaluation: Inconsistency Detection - UMEs

| Project | # Elements | | Precision | | Recall | | F1-Score | | Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|
| MS | | 4 | | .67 | | 1.0 | | .80 | | .88 |
| TS | 6 | 5 | 1.0 | 1.0 | .83 | 1.0 | .91 | 1.0 | .91 | 1.0 |
| TM | 1 | | 1.0 | | 1.0 | | 1.0 | | 1.0 | |
| BBB | 4 | 1 | .50 | 1.0 | .75 | 1.0 | .60 | 1.0 | .73 | 1.0 |
| JR | 3 | 1 | 1.0 | 1.0 | .67 | 1.0 | .80 | 1.0 | .83 | 1.0 |
| w. Avg. | | | .86 | .88 | .79 | 1.0 | .80 | .93 | .85 | .95 |
| | | | .87 | | .88 | | .86 | | .90 | |

Historic
Current

# Evaluation: Inconsistency Detection - MMEs

RQ3: How well does the approach detect missing model elements?

- Goals
  - To measure how well the approach detects missing model elements
  - To compare with a simple baseline
  - To measure the influence of filter lists
- Process
  - Remove model elements to create (artificial) inconsistencies
- Metrics
  - Precision, Recall, F1 – Score
  - Accuracy, Specificity
  - Φ – Coefficient
  - Average, Weighted Average

# Evaluation: Inconsistency Detection - MMEs

| Project | Precision | | Recall | | F1-Score | | Accuracy | |
|---|---|---|---|---|---|---|---|---|
| MS | | .21 | | .79 | | .33 | | .70 |
| TS | .16 | .96 | .98 | .70 | .28 | .79 | .38 | .96 |
| TM | .17 | .18 | .63 | .76 | .26 | .28 | .86 | .85 |
| BBB | .09 | .89 | .18 | .46 | .11 | .43 | .81 | .96 |
| JR | .22 | 1.0 | .11 | .44 | .15 | .44 | .57 | .85 |
| w. Avg. | .14 | .60 | .47 | .63 | .19 | .43 | .71 | .87 |
| | .39 | | .64 | | .34 | | .77 | |

Historic
Current

# Discussion

- Good results, some outliers
- Outliers when text and model diverge too much → Low precision

- Threats to Validity
  - Few (open source) cases, unclear how well this generalizes
  - Artificial inconsistencies introduced when evaluating MME-detection
  - Benchmark dataset with potentially biased gold standards

# Conclusion

Karlsruhe Institute of Technology

- We looked into automatic detection of inconsistencies in software architecture documentation using trace links
- We improved our approach for TLR and propose an approach to identify missing model elements and unmentioned model elements
- We evaluated using five projects
  - TLR: F1-Score 0.81, Accuracy 0.98
  - ID – UMEs: F1-Score 0.89, Accuracy 0.93
  - ID – MMEs: F1-Score 0.39, Accuracy 0.77
  - Outperforming baselines
- Needed Improvements & Future Work
  - Make use of relations and check their consistency
  - Experiment with deep learning/language models

Replication package

# References


- [Keim2021] J. Keim, S. Schulz, D. Fuchß, C. Kocher, J. Speit, and A. Koziolek, “Tracelink recovery for software architecture documentation,” in Software Architecture, S. Biffl, E. Navarro, W. Löwe, M. Sirjani, R. Mirandola, and D. Weyns, Eds. Springer International Publishing, 2021, pp. 101–116.
- [KimKim2021] S. Kim and D. Kim, “Automatic identifier inconsistency detection using code dictionary,” Emp. Softw. Engg., vol. 21, no. 2, p. 565–604, 2016.
- [FantechiSpinicci2005] A. Fantechi and E. Spinicci, “A content analysis technique for inconsistency detection in software requirements documents.” 2005, pp. 245–256.
- [Kamalrudin2010] M. Kamalrudin, J. Grundy, and J. Hosking, “Managing consistency between textual requirements, abstract interactions and essential use cases,” in IEEE Annual Computer Software and Applications Conference, 2010, pp. 327–336.
- [LytraZdun2014] I. Lytra and U. Zdun, “Inconsistency management between architectural decisions and designs using constraints and model fixes,” in 2014 23rd Australian Software Engineering Conference, 2014, pp. 230–239.