UNIVERSITÄT
LEIPZIG

SPLC 2019, Challenge Solutions Track

A Graph-Based Feature Location Approach Using Set Theory

Paris, September 11, 2019

Richard Müller and Ulrich Eisenecker

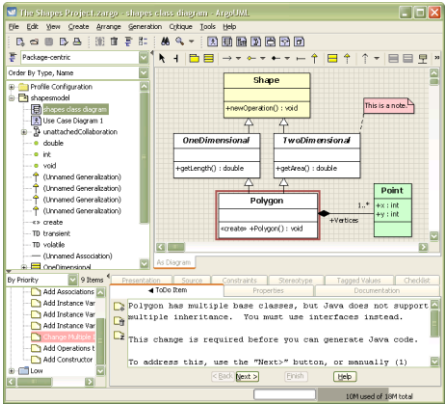
FEATURE LOCATION

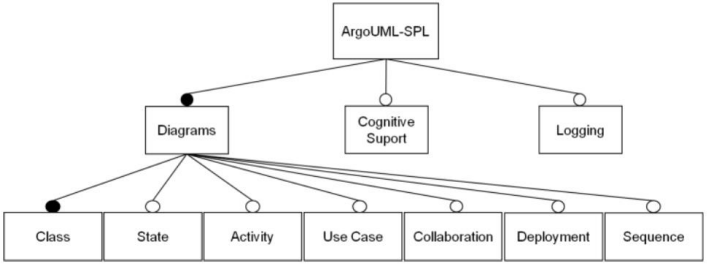
- A feature is *“a prominent or distinctive and user visible aspect, quality, or characteristic of a software system or systems“*
- Feature location techniques aim at identifying a mapping from features to source code elements
- Important for software maintenance of single systems and of Software Product Lines (SPLs)
- In SPLs single features as well as feature combinations and feature negations have to be identified

[Kang et al. 1990, Dit et al 2013, Rubin and Chechik 2013]

ARGOUML SPL BENCHMARK

- Set of scenarios each containing a set of ArgoUML variants
- List of features with names and descriptions
- Java program to automatically calculate precision, recall, and F1 score based on the feature location results and the ground-truth





[Martinez et al. 2018]

FEATURE LOCATION TAXONOMY

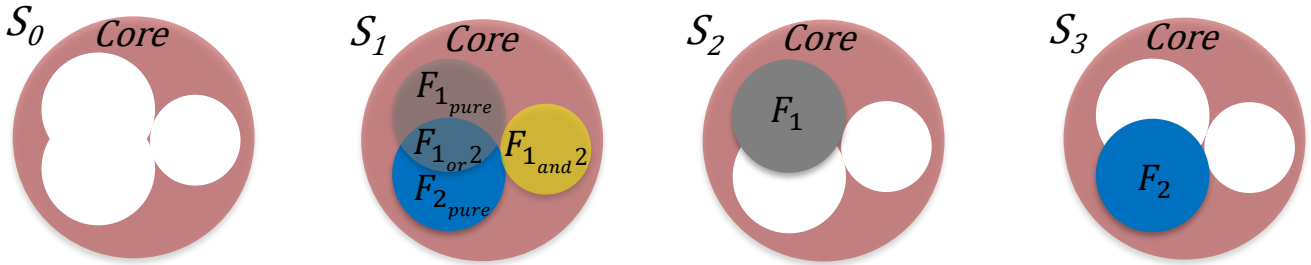
- Input
  - Java source code artifacts
- Data source
  - Software graphs created by the Java source code plugin of jQAssistant
- Method
  - Static analysis and set theory
- Output
  - Text file for each feature with class and method traces
- Evaluation
  - Traditional scenario of the ArgoUML SPL benchmark

[Dit et al. 2013]

CONCEPT WITH TWO FEATURES

- A SPL S consisting of four variants S_0 to S_3 and containing two features F_1 and F_2 can be described by the following sets

System	Features	Traces
S_0	*none*	T_0
S_1	F_1, F_2	T_1
S_2	F_1	T_2
S_3	F_2	T_3



$Core$	Traces belonging to no feature	$F_{1_{and}2}$	Traces, that are only included if F_1 and F_2 are present
$F_{1_{pure}}$	Exclusive traces required for F_1	F_1	Traces of $F_{1_{pure}}$ and $F_{1_{or}2}$
$F_{1_{or}2}$	Common traces of F_1 and F_2		

SET OPERATIONS WITH TWO FEATURES

(1) $F_1 := F_{1_{pure}} \cup F_{1_{or}2} := T_2 \setminus T_0$

(2) $F_2 := F_{2_{pure}} \cup F_{1_{or}2} := T_3 \setminus T_0$

(3) $F_{1_{pure}} \cup F_{1_{and}2} := (T_1 \setminus T_0) \setminus T_3$

(4) $F_{2_{pure}} \cup F_{1_{and}2} := (T_1 \setminus T_0) \setminus T_2$

(5) $F_{1_{and}2} := \left(F_{1_{pure}} \cup F_{1_{and}2}\right) \cap \left(F_{2_{pure}} \cup F_{1_{and}2}\right)$

System	Features	Traces
S_0	*none*	T_0
S_1	F_1, F_2	T_1
S_2	F_1	T_2
S_3	F_2	T_3

SET OPERATIONS WITH EIGHT FEATURES (TRADITIONAL SCENARIO)

(6) $$F_{1_{pure}} := ((T_1 \setminus T_0) \setminus T_2) \setminus \bigcup_{i=1}^{8} \bigcup_{j=i+1}^{8} F_{i_{and}j}$$

(7) $$F_{2_{pure}} := ((T_1 \setminus T_0) \setminus T_3) \setminus \bigcup_{i=1}^{8} \bigcup_{j=i+1}^{8} F_{i_{and}j}$$

(8) $$F_{1_{and}2} := ((T_1 \setminus T_0) \setminus T_2) \cap ((T_1 \setminus T_0) \setminus T_3)$$

System	Features	Traces
S_0	*None*	T_0
S_1	$F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8$	T_1
S_2	$F_2, F_3, F_4, F_5, F_6, F_7, F_8$	T_2
S_3	$F_1, F_3, F_4, F_5, F_6, F_7, F_8$	T_3
S_4	$F_1, F_2, F_4, F_5, F_6, F_7, F_8$	T_4
S_5	$F_1, F_2, F_3, F_5, F_6, F_7, F_8$	T_5
S_6	$F_1, F_2, F_3, F_4, F_6, F_7, F_8$	T_6
S_7	$F_1, F_2, F_3, F_4, F_5, F_7, F_8$	T_7
S_8	$F_1, F_2, F_3, F_4, F_5, F_6, F_8$	T_8
S_9	$F_1, F_2, F_3, F_4, F_5, F_6, F_7$	T_9

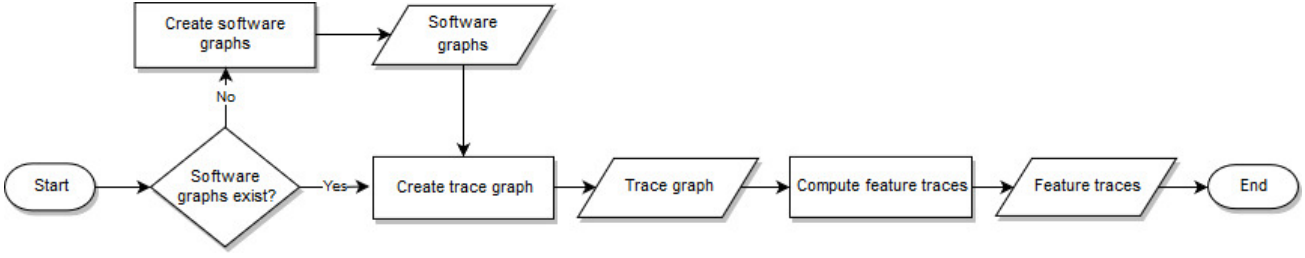
- These operations have to be repeated for features F_3 to F_8
- Due to the given configurations, it is not possible to determine the sets of common (or) feature traces in the traditional scenario

HOW MANY SYSTEMS AND HOW MANY ELEMENTARY SETS EXIST IN THE CHALLENGE?

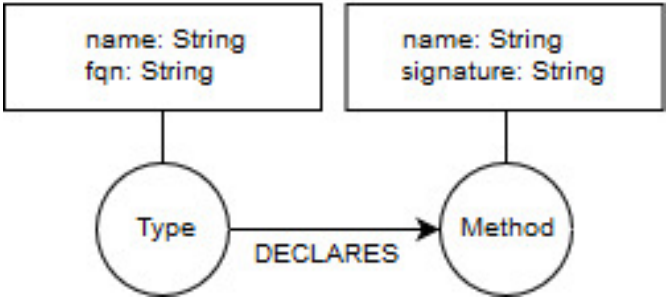
- $Systems\,(f) \coloneqq 1 + \sum_{i=1}^{f} \binom{f}{i} \coloneqq 2^f$
- $Elementary\ Sets\,(f) \coloneqq f + 2 * \sum_{i=2}^{f} \binom{f}{i} \coloneqq 2^{f+1} - f - 2$
- The ArgoUML SPL contains 8 features
  - There are 256 possible variants
    - $Systems\,(8) \coloneqq 256$
  - and 502 elementary sets
    - $Elementary\ \ Sets\,(8) \coloneqq 502$

IMPLEMENTATION

- Java source code plugin for jQAssistant (jQA)
- Graph Database Neo4j
- Query language Cypher with Awesome Procedures On Cypher (APOC)



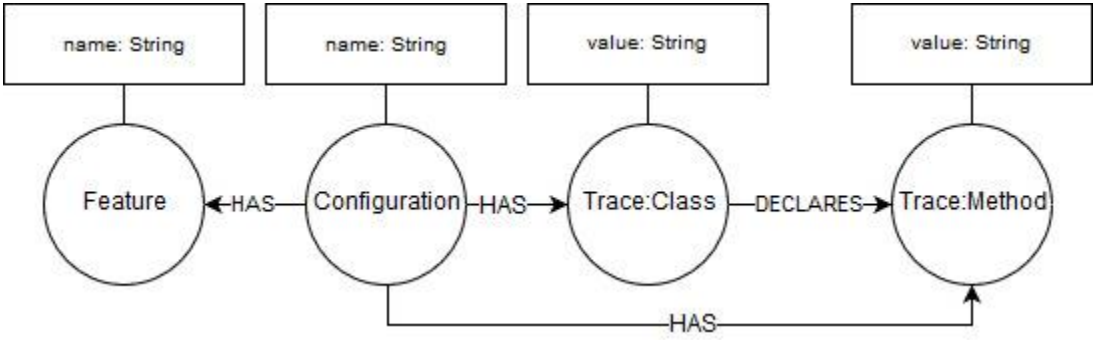
SOFTWARE GRAPH




```
MATCH (type:Type)
WHERE type.fqn STARTS WITH 'org.argouml'
AND NOT (:Type)-[:DECLARES]->(type) AND NOT type.fqn CONTAINS '$'
AND NOT type.fqn CONTAINS 'Anonymous' AND NOT size(type.name) = 1
RETURN DISTINCT type.fqn AS type

MATCH (type:Type)-[:DECLARES]->(method:Method)
// same conditions as above
RETURN DISTINCT type.fqn AS type, method.name AS method, method.
signature AS signature
```

TRACE GRAPH




```
MATCH (c:Configuration)-[:HAS]->(ct:Trace)
WHERE c.name = 'P01_AllDisabled.config' SET ct:Core
// ...
MATCH (c:Configuration)-[:HAS]->(fct:Trace)
WHERE c.name = 'P02_AllEnabled.config'
WITH cts, collect(fct) AS fcts
WITH apoc.coll.subtract(fcts, cts) AS fts
FOREACH (f IN fts | SET f:FeatureTrace )
//...
```

EVALUATION WITH PRUNED GROUND-TRUTH

Name	Precision	Recall	F1-score
ACTIVITYDIAGRAM	1.00	0.49	0.66
ACTIVITYDIAGRAM_and_STATEDIAGRAM	1.00	1.00	1.00
COGNITIVE	1.00	1.00	1.00
COGNITIVE_and_DEPLOYMENTDIAGRAM	1.00	0.93	0.96
COGNITIVE_and_SEQUENCEDIAGRAM	1.00	1.00	1.00
COLLABORATIONDIAGRAM	1.00	0.95	0.98
COLLABORATIONDIAGRAM_and_SEQUENCEDIAGRAM	1.00	1.00	1.00
DEPLOYMENTDIAGRAM	1.00	1.00	1.00
LOGGING	1.00	0.67	0.80
SEQUENCEDIAGRAM	1.00	0.96	0.98
STATEDIAGRAM	1.00	0.63	0.77
USECASEDIAGRAM	1.00	1.00	1.00
Average	1.00	0.89	0.93

STRENGTHS & WEAKNESSES

- The approach locates features, feature combinations, and feature negations with a precision of 1.0 and a recall of 1.0 provided, the required sets of software traces are available
- The implementation of the technique is very compact
- Initially, we manually defined the elementary feature subsets in a given scenario, now, an adequate algorithm is implemented
- Currently, the jQA plugin does not consider imports, fields, and method statements
- Time performance (6 min per variant)

CONCLUSION

- We mapped the given problem to set theory and developed a graph-based solution for the traditional scenario with Java, Neo4j, and Cypher
- Our solution locates exclusive (pure) traces and pair-wise feature interaction (and) traces to complete classes and methods of 12 out of 24 features and feature combinations with a precision of 1.0 and a recall of 1.0

FUTURE WORK

- Extend the jQA plugin to scan imports, fields, and method statements
- Include the the algorithm for defining all elementary feature subsets for a given scenario to automate feature location in the context of the ArgoUML SPL benchmark
- Identify performance bottlenecks and apply adequate optimizations
- Combine the approach with alternative techniques in scenarios with less or no variants

ARTIFACTS

- Challenge solution for the traditional scenario
  - https://github.com/softvis-research/argouml-spl-benchmark
- Java source code plugin for jQAssistant
  - https://github.com/softvis-research/jqa-javasrc-plugin/tree/splc-challenge
- Algorithm to calculate all elementary feature sets of a given scenario
  - https://github.com/softvis-research/featurelocation

REFERENCES

- Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature location in source code: a taxonomy and survey. J. Softw. Evol. Process 25, 1 (2013), 53–95.
- Jabier Martinez, Nicolas Ordoñez, Xhevahire Tërnava, Tewfik Ziadi, Jairo Aponte, Eduardo Figueiredo, and Marco Tulio Valente. 2018. Feature Location Benchmark with argoUML SPL. In Proc. 22Nd Int. Syst. Softw. Prod. Line Conf. Vol. 1 (SPLC'18). ACM, New York, NY, USA, 257–263.
- Julia Rubin and Marsha Chechik. 2013. A Survey of Feature Location Techniques. In Domain Eng. Prod. Lines, Lang. Concept. Model. 29–58.
- Kang K. C., Sholom G Cohen, James A Hess, William E Novak, A Spencer Peterson, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. 1990. Feasibility Study Feature-Oriented Domain Analysis (FODA). Technical Report November. Carnegie-Mellon University Software Engineering Institute.
- Richard Müller, Dirk Mahler, Michael Hunger, Jens Nerche, and Markus Harrer. 2018. Towards an Open Source Stack to Create a Unified Data Source for Software Analysis and Visualization. In Proc. 6th IEEE Work. Conf. Softw. Vis. IEEE, Madrid, Spain.

UNIVERSITÄT
LEIPZIG

THANK YOU.

Richard Müller and Ulrich Eisenecker
Information Systems Institute, Software Engineering Department

rmueller@wifa.uni-leipzig.de
@rimllr
https://github.com/softvis-research
http://softvis.wifa.uni-leipzig.de