Luciano Marchezan

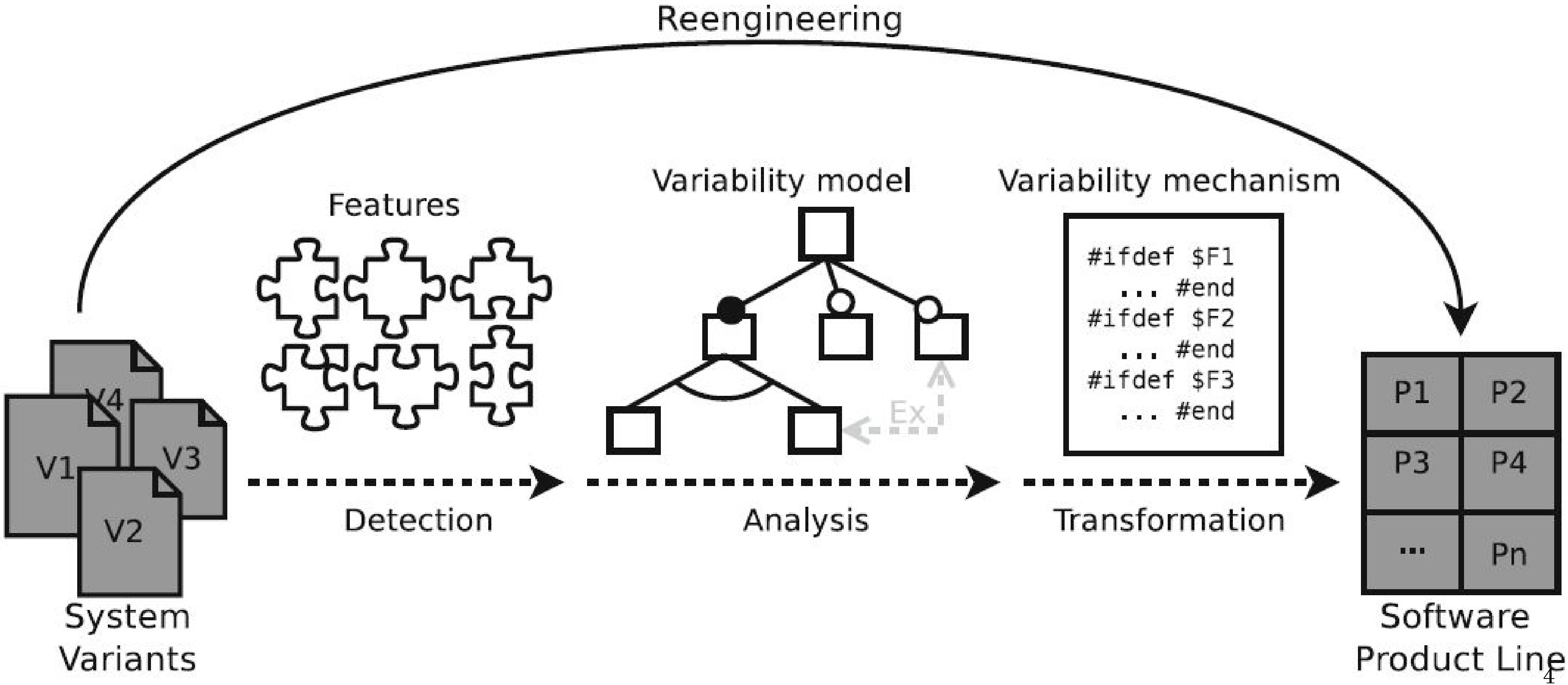
Research goals

- Create a set of guidelines for the reengineering process of legacy systems towards software product lines.
    - The first step will focus on detection and analysis phases.
- Study and analyse the strategies, techniques and input/output artifacts to identify the best choices for different scenarios.
- Build a tool to validate some of these techniques.
- Validate in a study case.

Reengineering legacy applications into software product lines: a systematic mapping

Wesley K. G. Assunção · Roberto E. Lopez-Herrejon · Lukas Linsbauer · Silvia R. Vergilio · Alexander Egyed , 2017

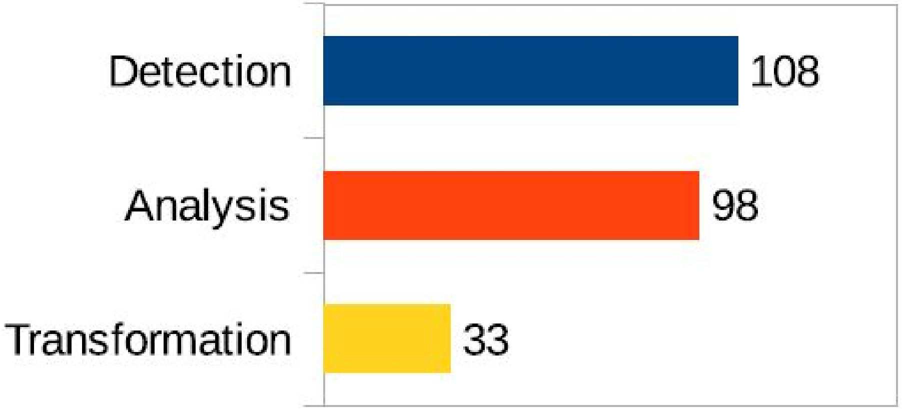
The Software Reengineering Process



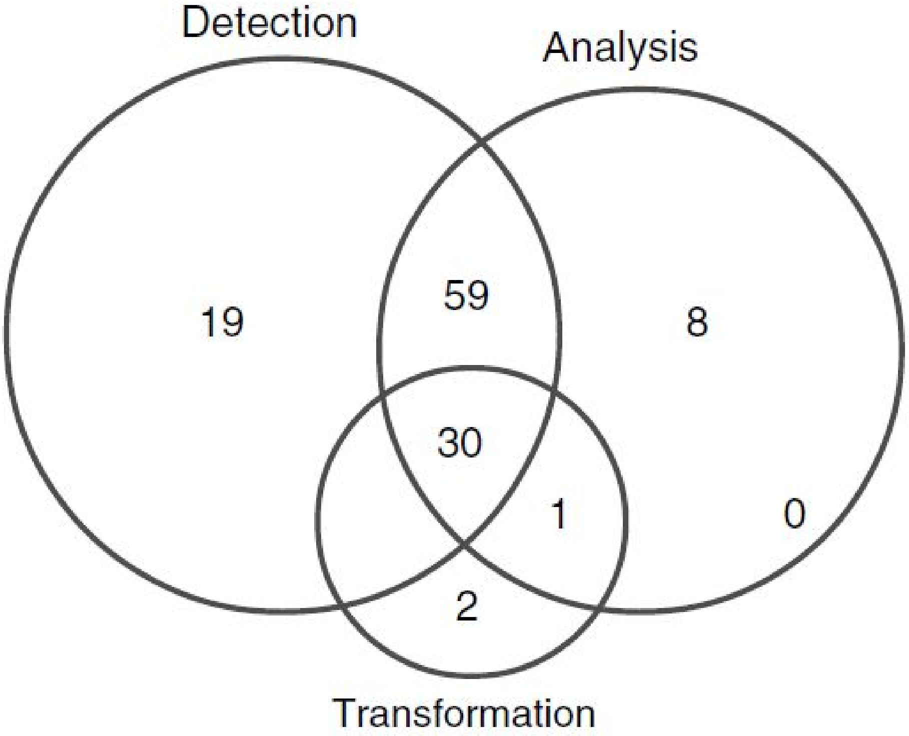
The Software Reengineering Process

- Detection: Detecting the variability and commonality among existing products (represented as features).
- Analysis: Organization of discovered variability and commonality and creation of the variability model (FM).
- Transformation: Artefacts that implement the features and the variability model are used to create the SPL, using a variability mechanism.

Distribution of studies among phases



a Primary sources per phase



b Intersection among the phases

Reengineering Strategies

Expert-driven: based on the expertise of specialists: software engineers, software architects, developers, stakeholders, etc.

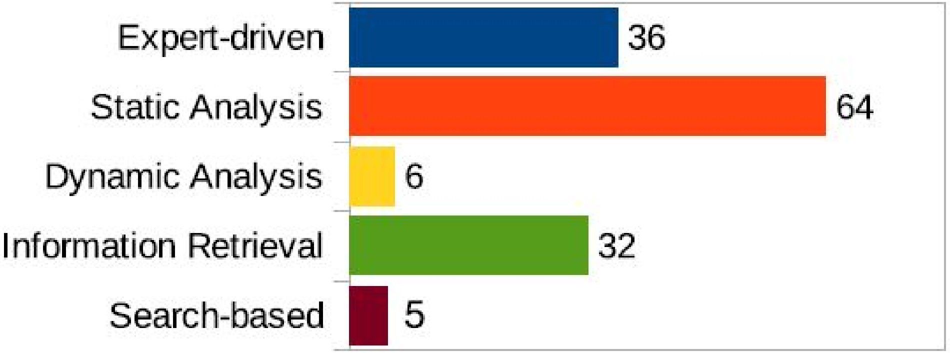
Static analysis: following or analysing structural information of static artefacts without their execution.

Dynamic analysis: the use of tools to collect and analyse information about the artifact's execution, in general considering a low-level of abstraction, such as source code.

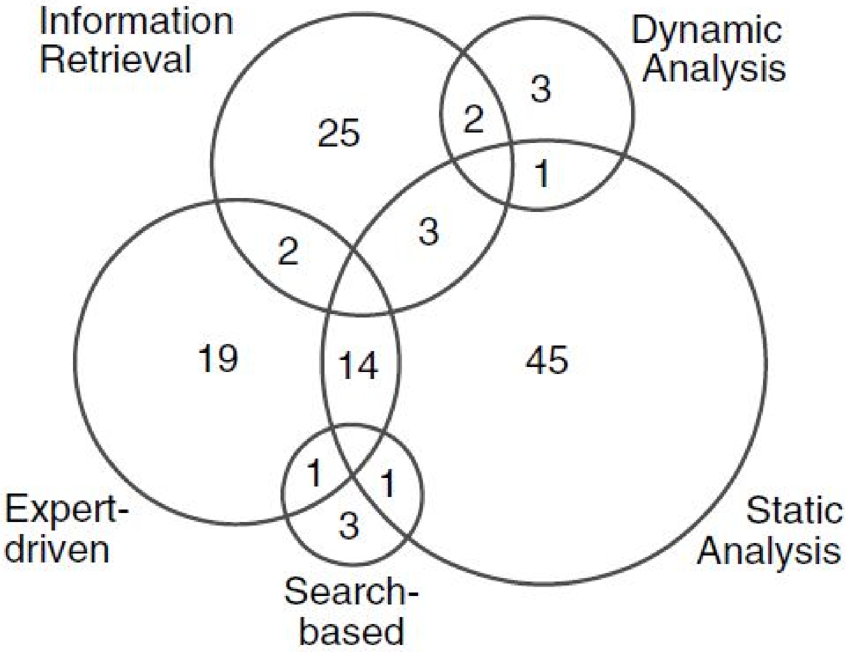
Information Retrieval: Considers the textual similarity in identifiers, comments and other artifacts.

Search-based: applies algorithms from the optimization field, such as Genetic Algorithms.

Reengineering Strategies

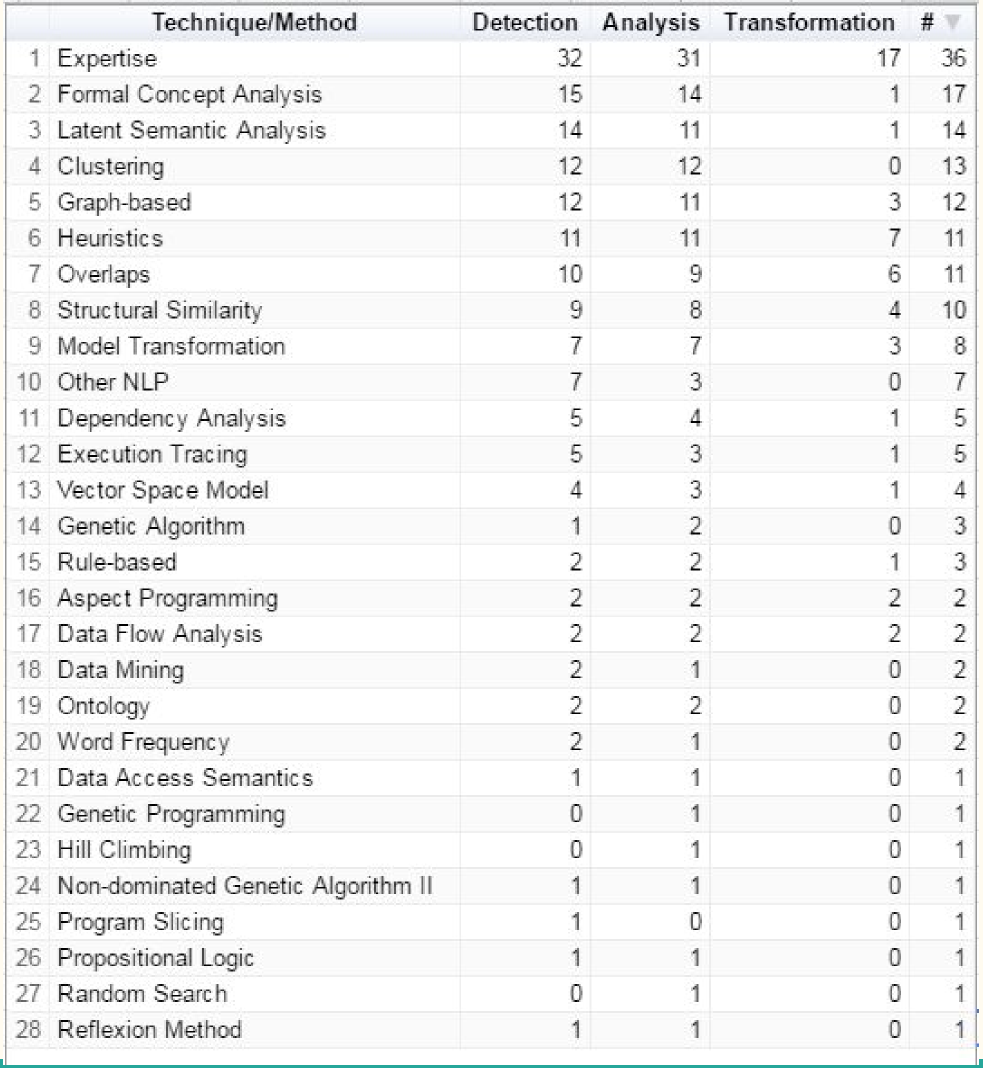


a Primary sources per strategy

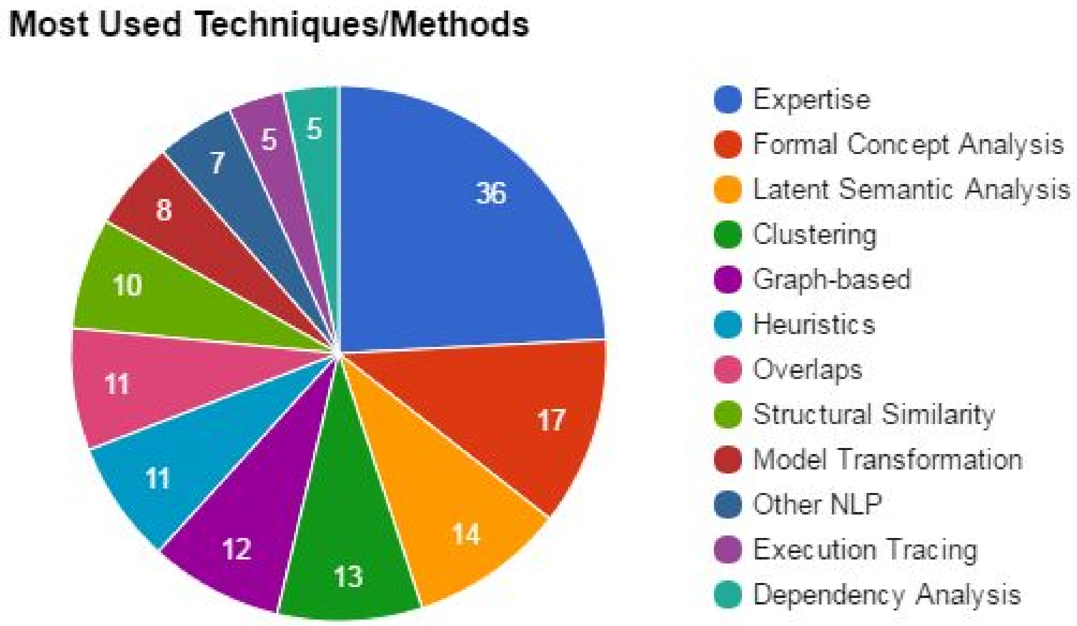


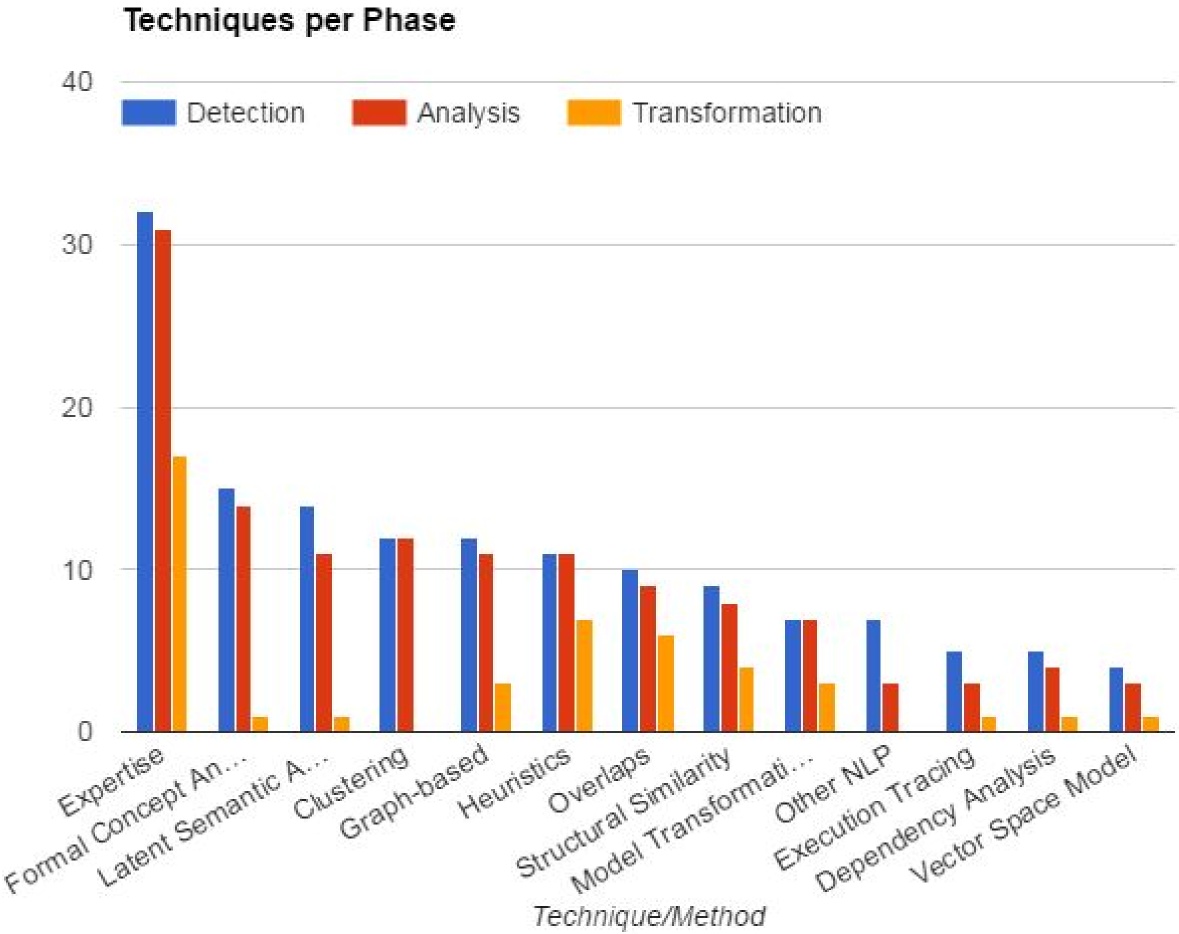
b Intersection among the strategies

Techniques/Methods



	Technique/Method	Detection	Analysis	Transformation	#
1	Expertise	32	31	17	36
2	Formal Concept Analysis	15	14	1	17
3	Latent Semantic Analysis	14	11	1	14
4	Clustering	12	12	0	13
5	Graph-based	12	11	3	12
6	Heuristics	11	11	7	11
7	Overlaps	10	9	6	11
8	Structural Similarity	9	8	4	10
9	Model Transformation	7	7	3	8
10	Other NLP	7	3	0	7
11	Dependency Analysis	5	4	1	5
12	Execution Tracing	5	3	1	5
13	Vector Space Model	4	3	1	4
14	Genetic Algorithm	1	2	0	3
15	Rule-based	2	2	1	3
16	Aspect Programming	2	2	2	2
17	Data Flow Analysis	2	2	2	2
18	Data Mining	2	1	0	2
19	Ontology	2	2	0	2
20	Word Frequency	2	1	0	2
21	Data Access Semantics	1	1	0	1
22	Genetic Programming	0	1	0	1
23	Hill Climbing	0	1	0	1
24	Non-dominated Genetic Algorithm II	1	1	0	1
25	Program Slicing	1	0	0	1
26	Propositional Logic	1	1	0	1
27	Random Search	0	1	0	1
28	Reflexion Method	1	1	0	1


Most Used Techniques/Methods
36
17
14
13
12
11
11
10
8
7
5
5
● Expertise
● Formal Concept Analysis
● Latent Semantic Analysis
● Clustering
● Graph-based
● Heuristics
● Overlaps
● Structural Similarity
● Model Transformation
● Other NLP
● Execution Tracing
● Dependency Analysis


Techniques per Phase
40
30
20
10
0
Detection
Analysis
Transformation
Expertise
Formal Concept An…
Latent Semantic A…
Clustering
Graph-based
Heuristics
Overlaps
Structural Similarity
Model Transformati…
Other NLP
Execution Tracing
Dependency Analysis
Vector Space Model
Technique/Method

Input Artifacts

- **Domain Information:** information get through domain analysis such as: high level description of systems in specific domain.
- **Requirements:** feature descriptions, customer requests, test sets generated, implementation and operation aspects, etc.
- **Design models:**class diagrams, state machines, entity-relationship database model or even feature models.
- **Source code.**

Output Artifacts

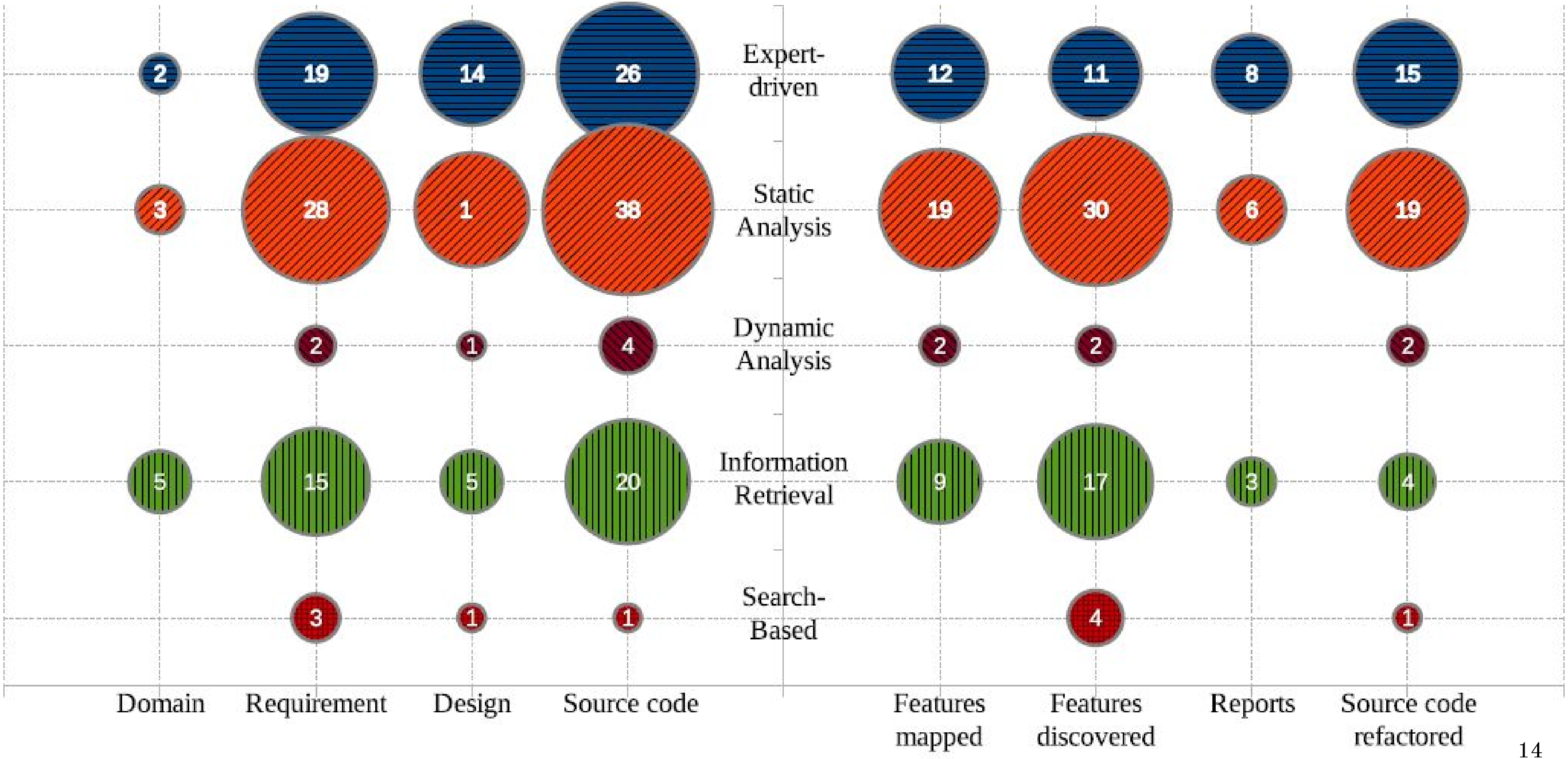
Features discovered: Features identified or mined from artefacts.

Features mapped: Traceability links between known features and artefacts related with them.

Reports: Reports with information such as the variability among the systems, impact on the reengineering to SPLs, and potential reuse in legacy system variants.

Source code refactored: Source code refactored is an output provided to allow a better organization of the features with the SPLE.

Input and Output used by strategies



Research Phases

- Select the most used techniques for detection and analysis phases;
- Read the papers which use these techniques to extract the following information:
  - Domains where the techniques/methods are used;
  - Drawbacks and benefits of each technique/method in different scenarios;
  - Techniques/methods that are mixed;
  - More details about the artifacts used;
- Propose a process/approach using the information gathered;
- Validate the approach;

References

Assunção, Wesley KG, et al. "Reengineering legacy applications into software product lines: a systematic mapping." Empirical Software Engineering (2017): 1-45.