Laleh M. Eshkevari, Ph.D Dissertation Defense

14 December 2015

Automatic Detection and Classification of Renamings

Supervisors:

Dr. Antoniol
Dr. Guéhéneuc

Department of Computer and Software Engineering
Ecole Polytechnique de Montreal, Quebec, Canada

Outline

- Context and Motivation
- Thesis Statement
- Taxonomy of Renaming
- Detection
- Classification

Java and PHP

- Conclusion and Future Works

Context and Motivation

Identifiers are added, deleted, or modified, i.e., renamed.

Why identifiers are renamed?

- Improve consistency
- Adjust naming convention
- Correct typos

Context and Motivation

- *Developer A:*

“There’s a balance to be struck: - identifiers are communication, and as the code is refactored it is critical that identifiers continue to correctly describe their purpose - changing identifiers tends to break APIs, and sometimes they’re used for unintended purposes, over-frequent change is not good.”

- *Developer B:*

“I encountered a problem when my colleague wrote Java code which uses reflection. I avoided renaming some classes/methods which will be inspected by the reflection, since doing so can introduce unpredictable bugs.”

Examples of Renaming

`e -> t`	`parameter, Exception -> Throwable`
`g -> generalization`	`local var, MGeneralization -> Object`
`length -> l`	`local var, int`
`sessState -> sessionState`	`local var, SessionState`
`jj_3R_70 -> jj_3R_69`	`method, private, boolean, final`
`verifyAXFR -> verifyStream`	`method, public, byte`

```
rebuildTypesAffectedByMissingSecondaryTypes ->
                                          rebuildTypesAffectedBySecondaryTypes

MicroContainerNotAdvisedAnnotationOverrideProxyAdvisorTestCase ->
         MicrocontainerAdvisedAnnotationOverrideProxyAdvisorTestCase
```

Developers Opinion on Renamings

Invited: 739 developers

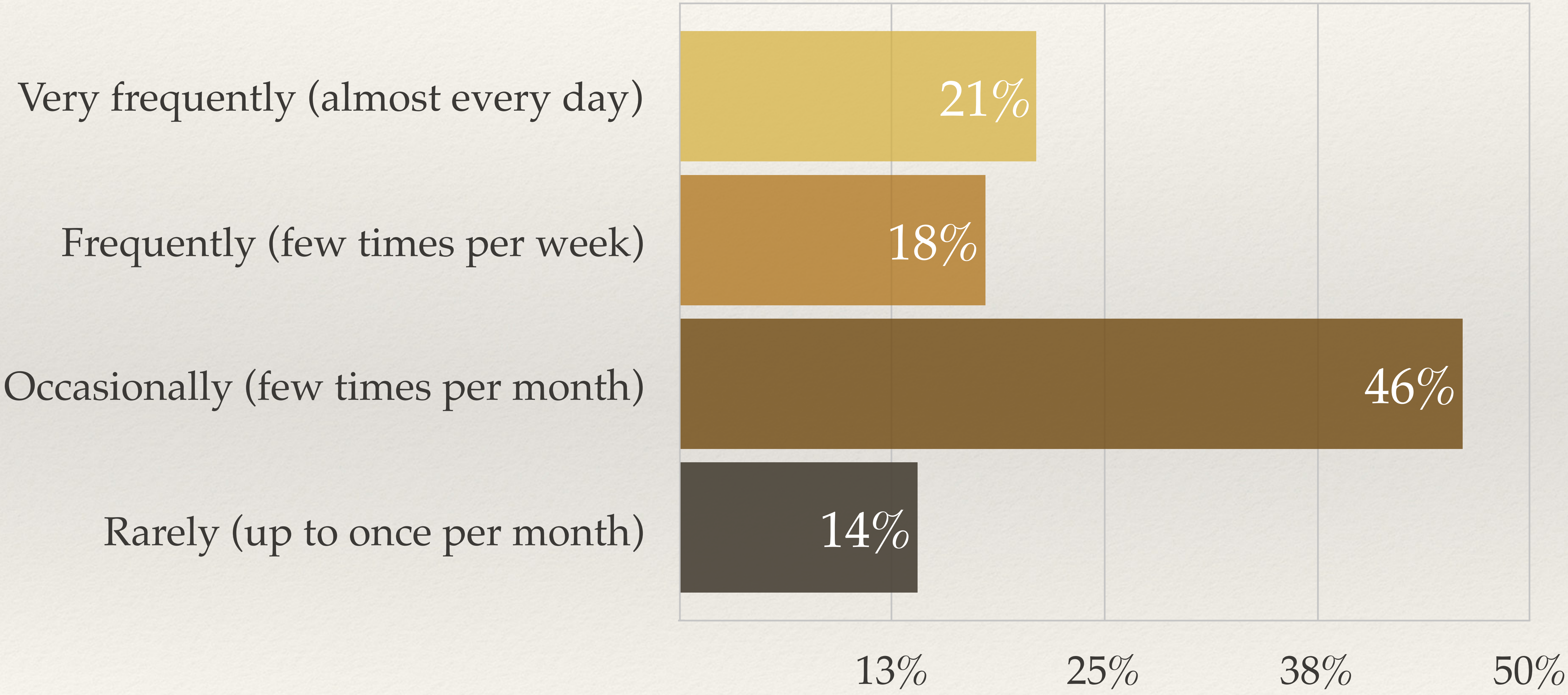
Open-source and industrial programs

Object-Oriented

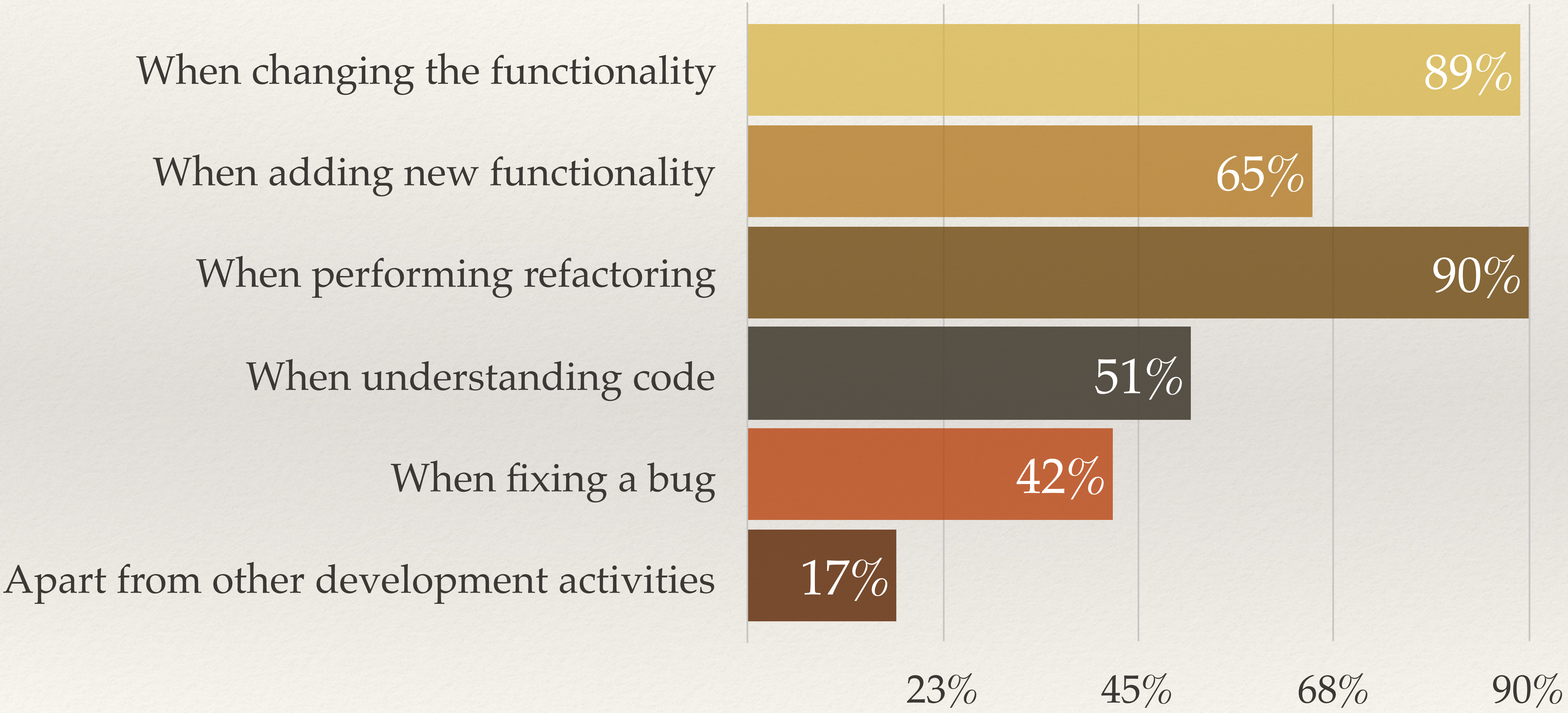
Participated: 71

- How often do developers rename?
- When do they rename?
- Is renaming straightforward?
- Already postpone a renaming?

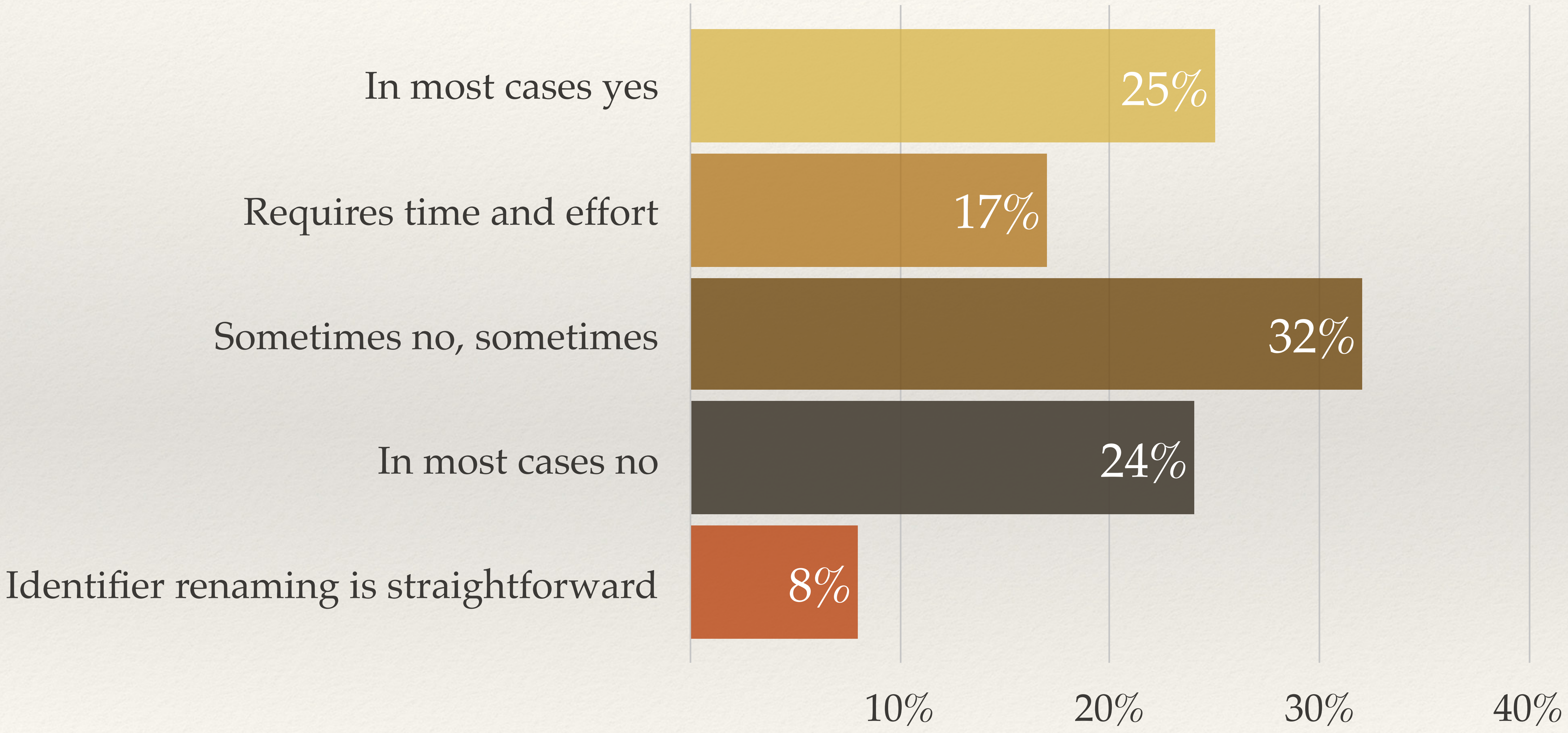
How often do developers rename?



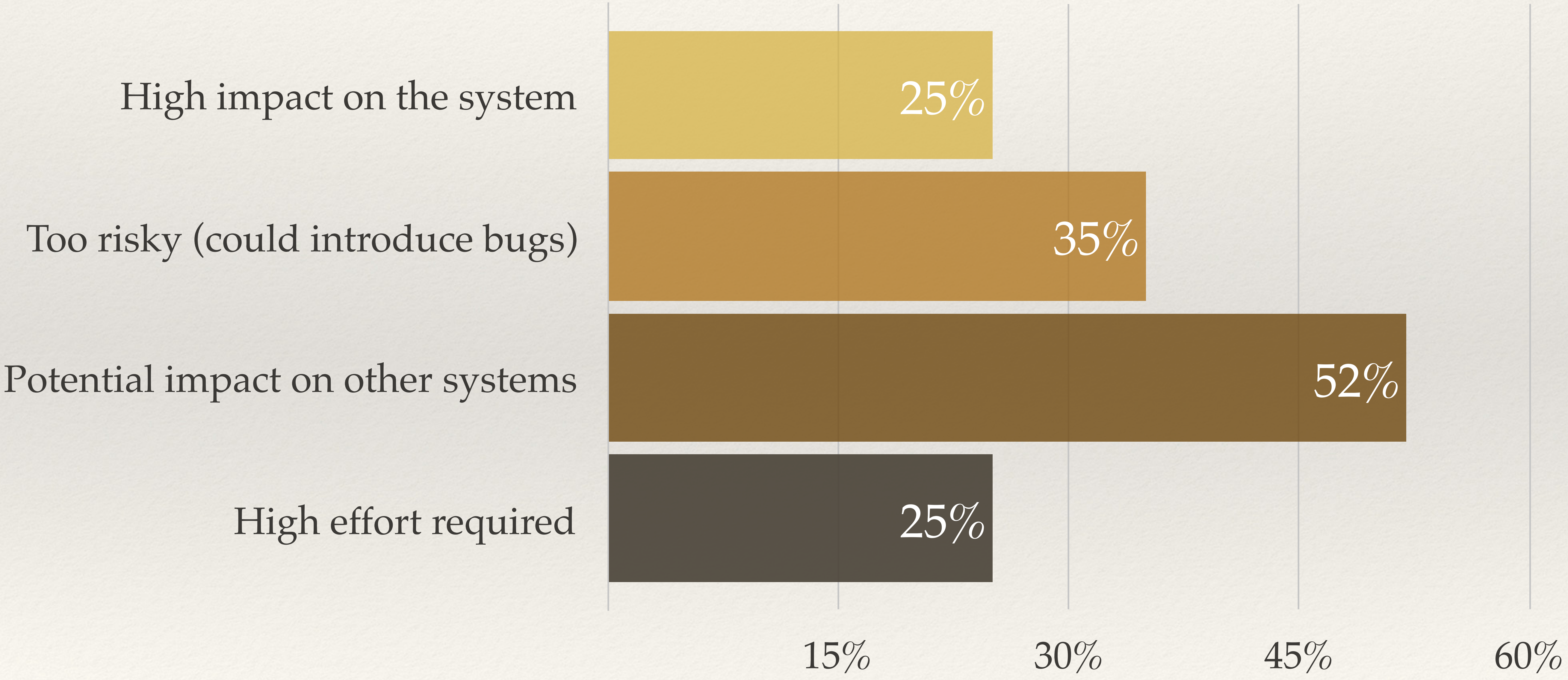
When do the developers rename?



Is renaming straightforward?



Already postponed a renaming?



Thesis Statement

Goal: To understand when, why, and how developers rename identifiers.

Detection and linguistic analysis of identifier renamings provides valuable insight on how, why, when developers rename identifiers.

Tool supports, programming language, and naming convention are factors that impact renamings frequency.

Taxonomy of Renamings

- We defined a Taxonomy of renaming based on grounded-theory approach [Strauss, 1987; Glaser, 1992].
- We manually analyzed 500 renaming to identify dimensions of renaming [Eshkevari *et al*, Arnaoudova *et al*].

Entity kind

Form of renamings

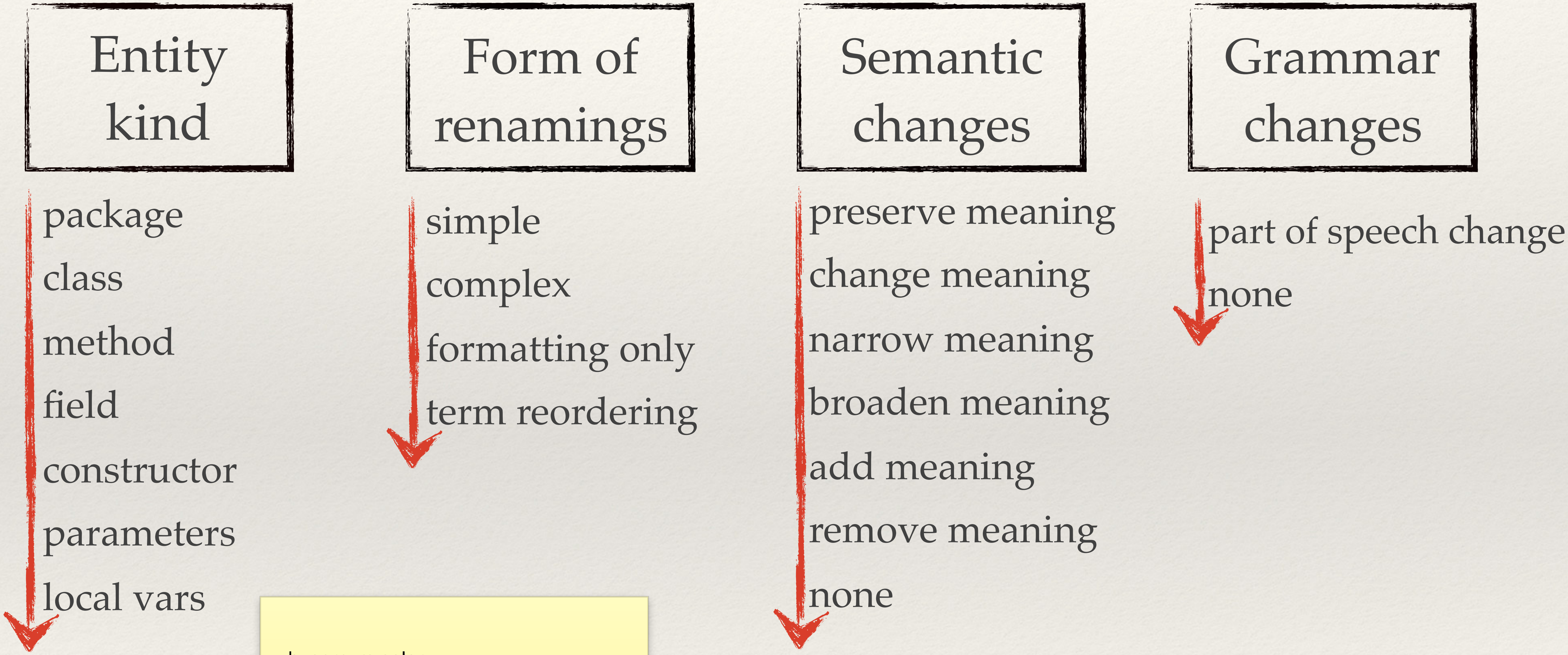
Semantic changes

Grammar changes

Eshkevari *et al*. An exploratory study of identifier renamings. **MSR 2011**

Arnaoudova *et al*. REPENT : Analyzing the Nature of Identifier Renamings. **TSE 2014**

Taxonomy of Renamings



Entity kind

- package
- class
- method
- field
- constructor
- parameters
- local vars

Form of renamings

- simple
- complex
- formatting only
- term reordering

Semantic changes

- preserve meaning
- change meaning
- narrow meaning
- broaden meaning
- add meaning
- remove meaning
- none

Grammar changes

- part of speech change
- none

hypernym color
hyponym red

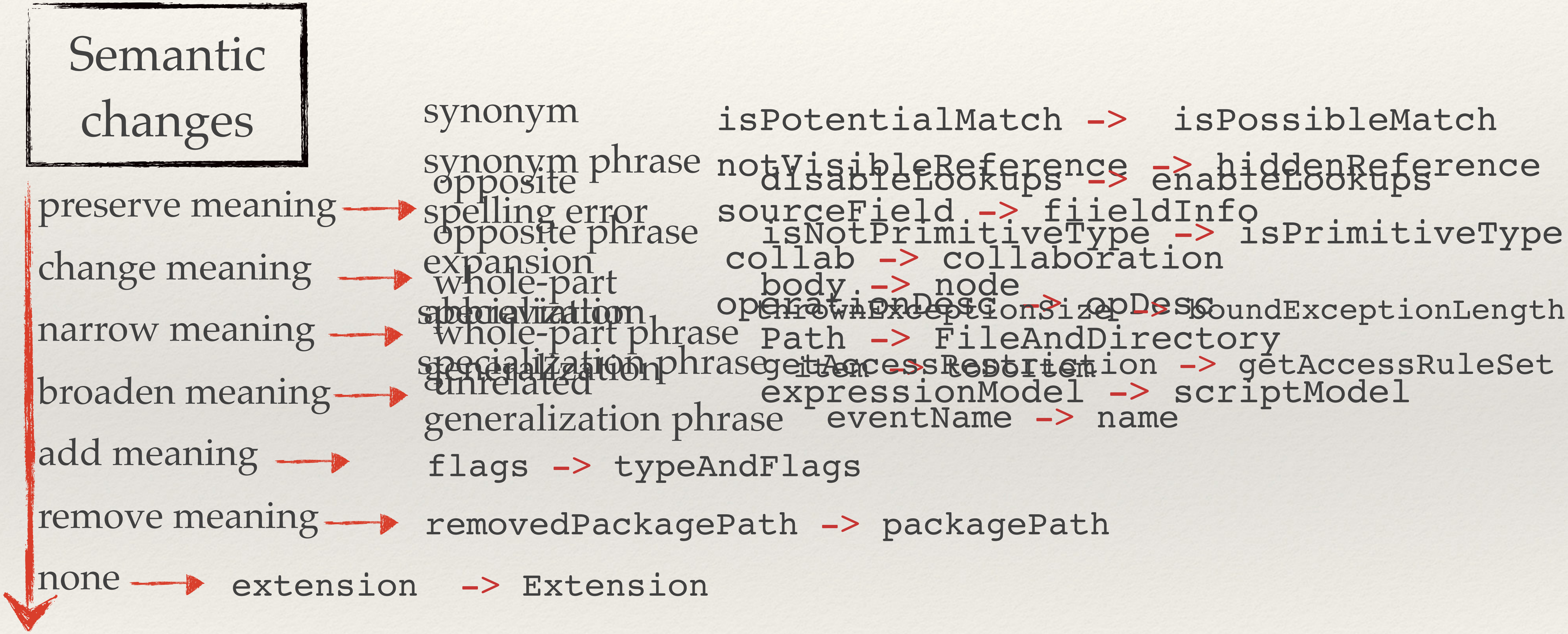
holonym tree
meronym trunk

Taxonomy of Renamings

Form of renamings

simple	`override -> overriding`
complex	`IsAssignmentWithNoEffectMASK -> AssignmentHasNoEffect`
formatting only	`JavaExtension -> JAVA_EXTENSION`
term reordering	`setDelaySocketClose -> setSocketCloseDelay`

Taxonomy of Renamings



Semantic changes

preserve meaning

change meaning

narrow meaning

broaden meaning

add meaning

remove meaning

none

synonym `isPotentialMatch -> isPossibleMatch`

synonym phrase `notVisibleReference -> hiddenReference`

opposite `disableLookups -> enableLookups`

spelling error `sourceField -> fiieldInfo`

opposite phrase `isNotPrimitiveType -> isPrimitiveType`

expansion `collab -> collaboration`

whole-part `body -> node`

abbreviation `operationDesc -> opDesc`

specialization `thrownExceptionSize -> boundExceptionLength`

whole-part phrase `Path -> FileAndDirectory`

specialization phrase `getAccessRestriction -> getAccessRuleSet`

generalization `item -> toDoItem`

unrelated `expressionModel -> scriptModel`

generalization phrase `eventName -> name`

`flags -> typeAndFlags`

`removedPackagePath -> packagePath`

`extension -> Extension`

Taxonomy of Renamings

Grammar changes

part of speech change → `getUpdatedSize -> updateFigGroupSize`

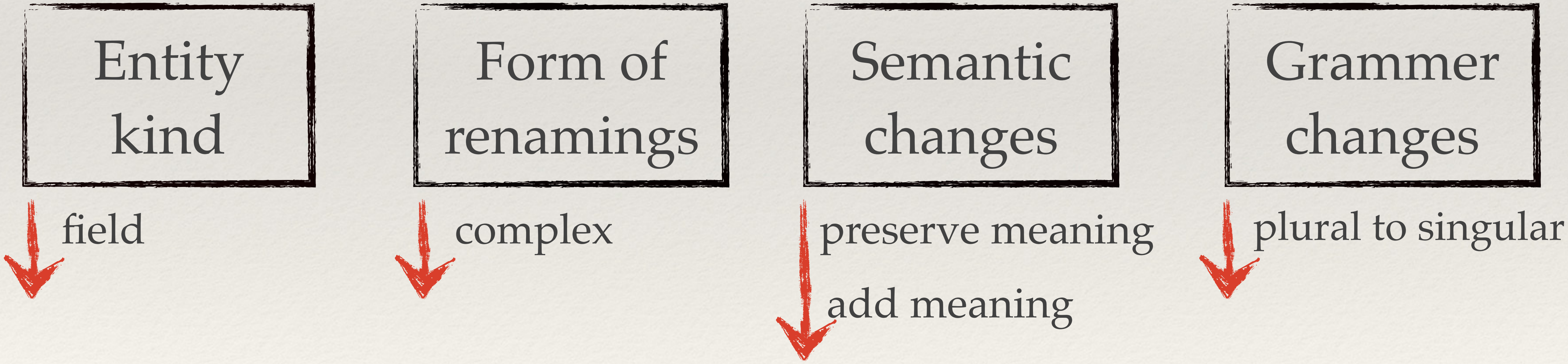
none → `isPotentialMatch -> isPossibleMatch`

Example

```
private int invParamsPtr =-1;

private int invalidParamReferencesPtr=-1;
```

`invParamsPtr -> invalidParamReferencesPtr`



Entity kind

field

Form of renamings

complex

Semantic changes

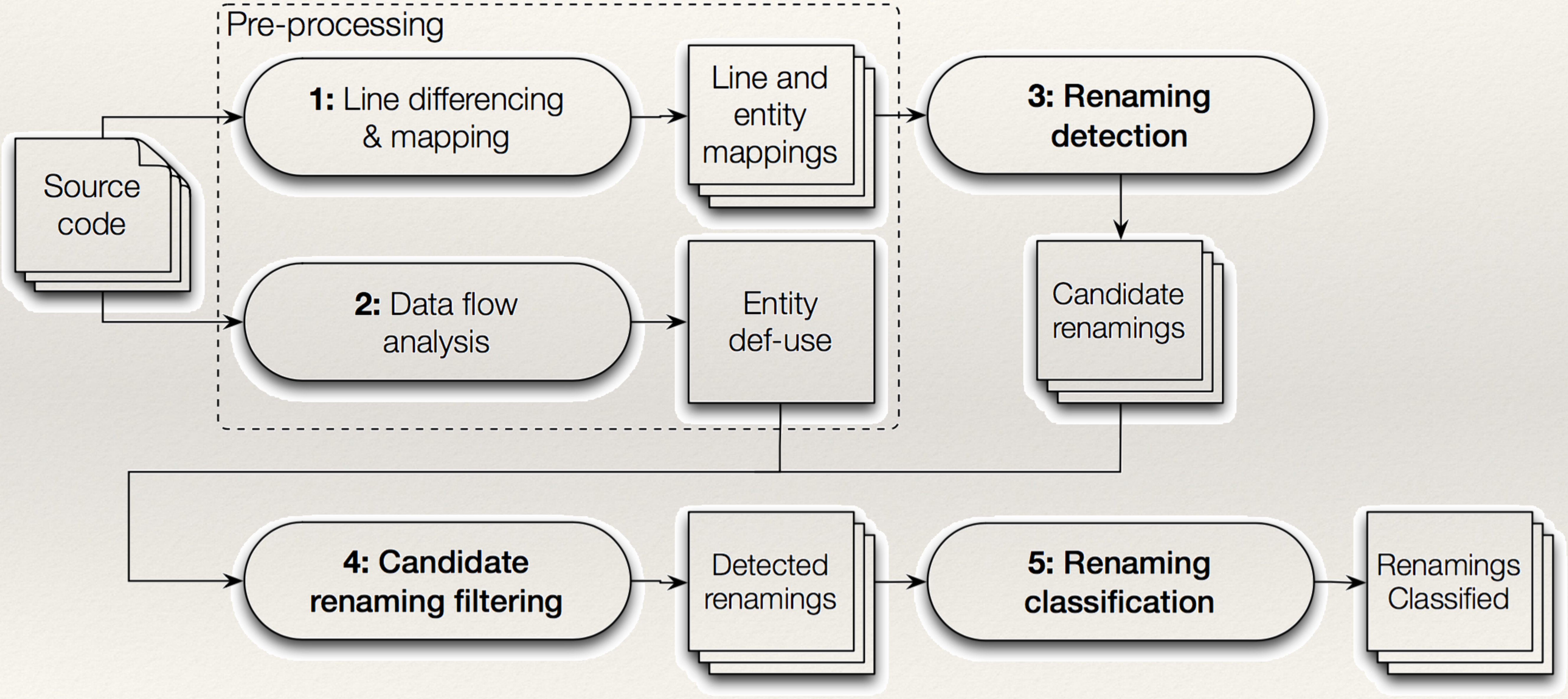
preserve meaning

add meaning

Grammer changes

plural to singular

Detection and Classification Approach

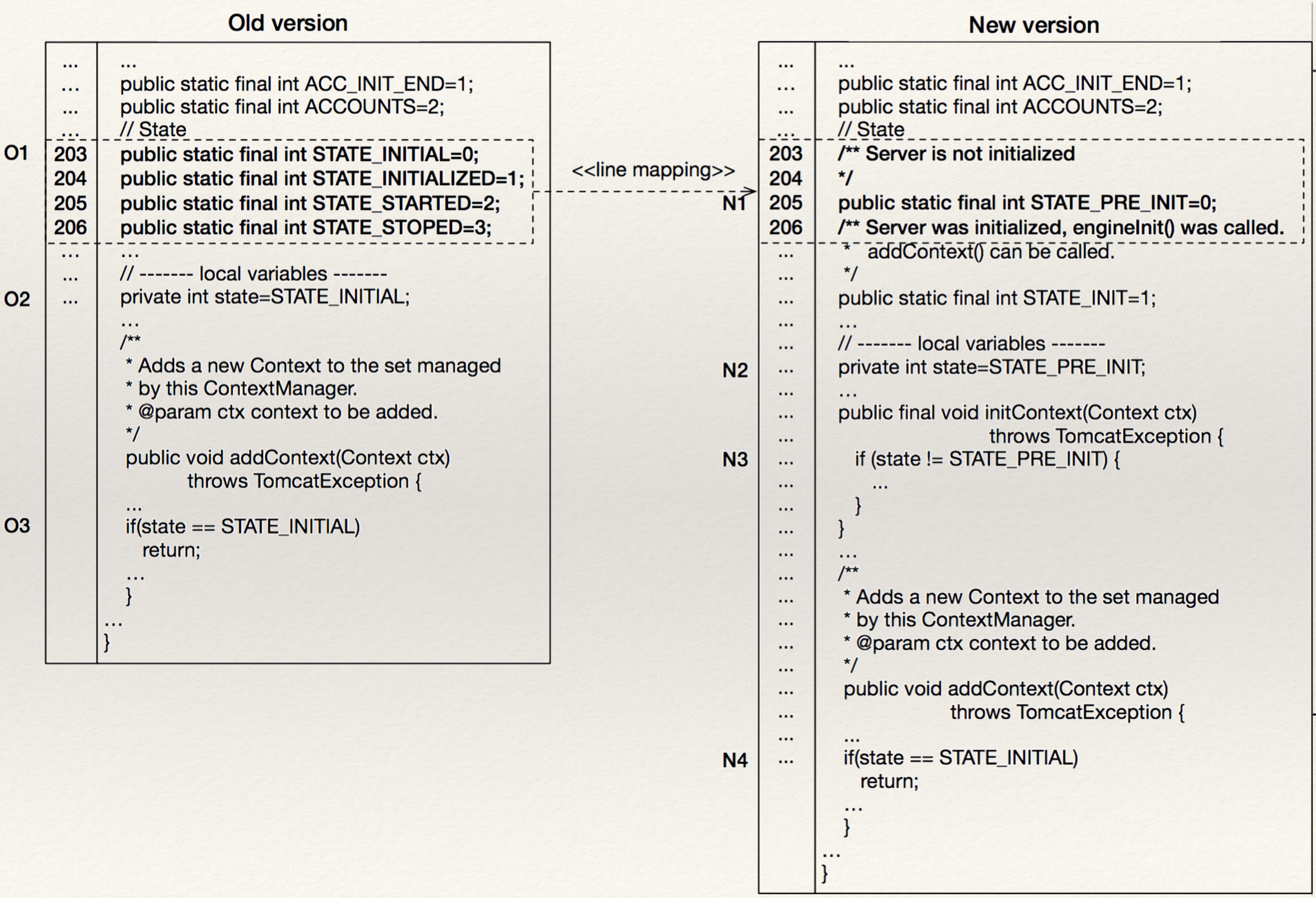


Renaming Detection

Detection

Line mapping

Entity mapping



Old version

```
...  ...
...  public static final int ACC_INIT_END=1;
...  public static final int ACCOUNTS=2;
...  // State
203  public static final int STATE_INITIAL=0;
204  public static final int STATE_INITIALIZED=1;
205  public static final int STATE_STARTED=2;
206  public static final int STATE_STOPED=3;
...  ...
...  // ------- local variables -------
...  private int state=STATE_INITIAL;
     ...
     /**
      * Adds a new Context to the set managed
      * by this ContextManager.
      * @param ctx context to be added.
      */
      public void addContext(Context ctx)
                 throws TomcatException {
      ...
      if(state == STATE_INITIAL)
         return;
      ...
      }
   ...
   }
```

New version

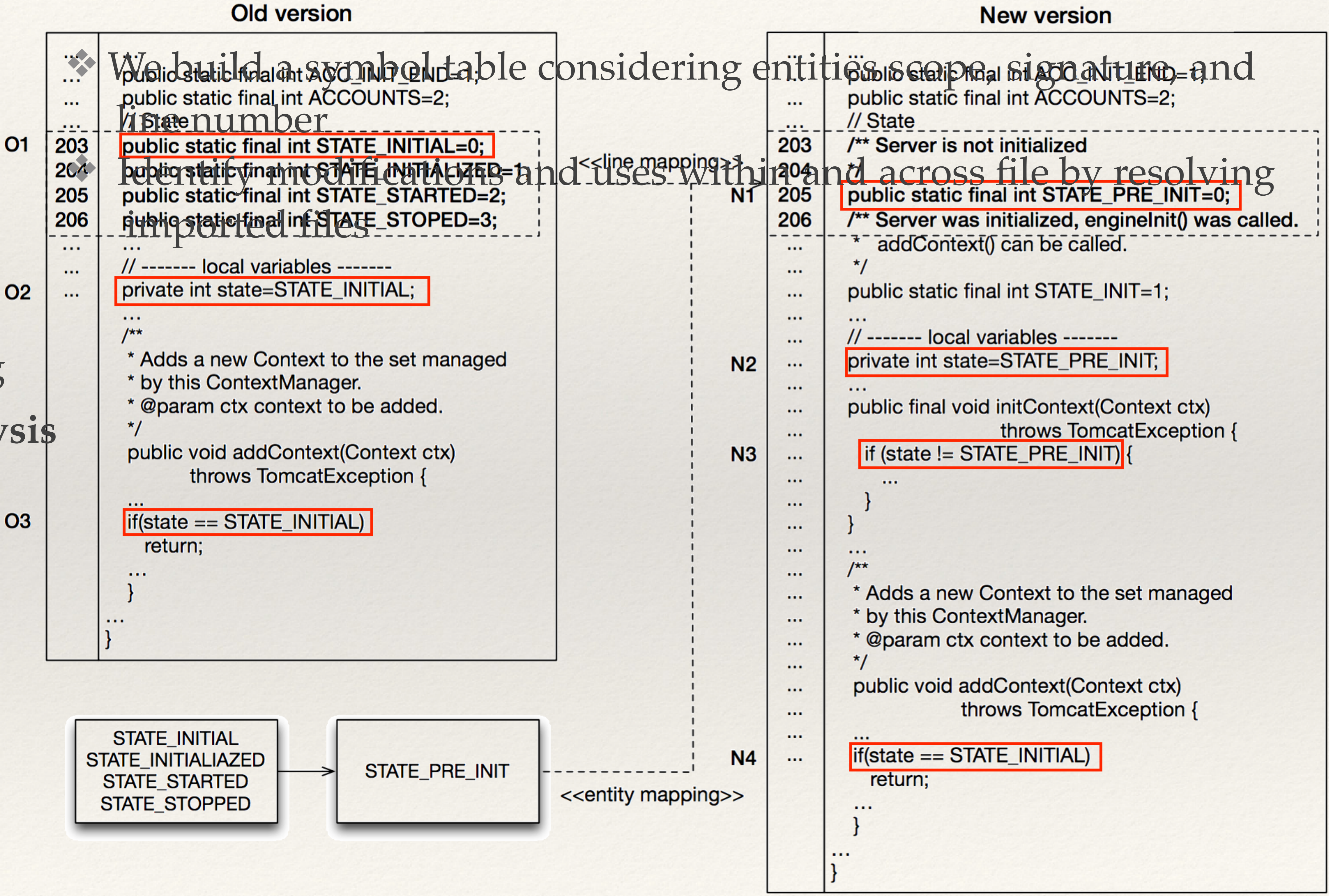
```
...  ...
...  public static final int ACC_INIT_END=1;
...  public static final int ACCOUNTS=2;
...  // State
203  /** Server is not initialized
204  */
205  public static final int STATE_PRE_INIT=0;
206  /** Server was initialized, engineInit() was called.
...   *  addContext() can be called.
...   */
...  public static final int STATE_INIT=1;
...  ...
...  // ------- local variables -------
...  private int state=STATE_PRE_INIT;
...  ...
...  public final void initContext(Context ctx)
...                   throws TomcatException {
...    if (state != STATE_PRE_INIT) {
...      ...
...    }
...  }
...  ...
...  /**
...   * Adds a new Context to the set managed
...   * by this ContextManager.
...   * @param ctx context to be added.
...   */
...   public void addContext(Context ctx)
...                 throws TomcatException {
...   ...
...   if(state == STATE_INITIAL)
        return;
      ...
      }
   ...
   }
```

Renaming Detection

Detection

- Line mapping
- Entity mapping
- **Data flow analysis**

- We build a symbol table considering entities scope, signature, and line number
- Identify modifications and uses within and across file by resolving imported files



Old version

```
...  public static final int ACC_INIT_END=1;
...  public static final int ACCOUNTS=2;
...  // State
203  public static final int STATE_INITIAL=0;
204  public static final int STATE_INITIALIZED=1;
205  public static final int STATE_STARTED=2;
206  public static final int STATE_STOPED=3;
...  ...
...  // ------- local variables -------
...  private int state=STATE_INITIAL;
     ...
     /**
      * Adds a new Context to the set managed
      * by this ContextManager.
      * @param ctx context to be added.
      */
      public void addContext(Context ctx)
                throws TomcatException {
      ...
      if(state == STATE_INITIAL)
         return;
      ...
      }
   ...
   }
```

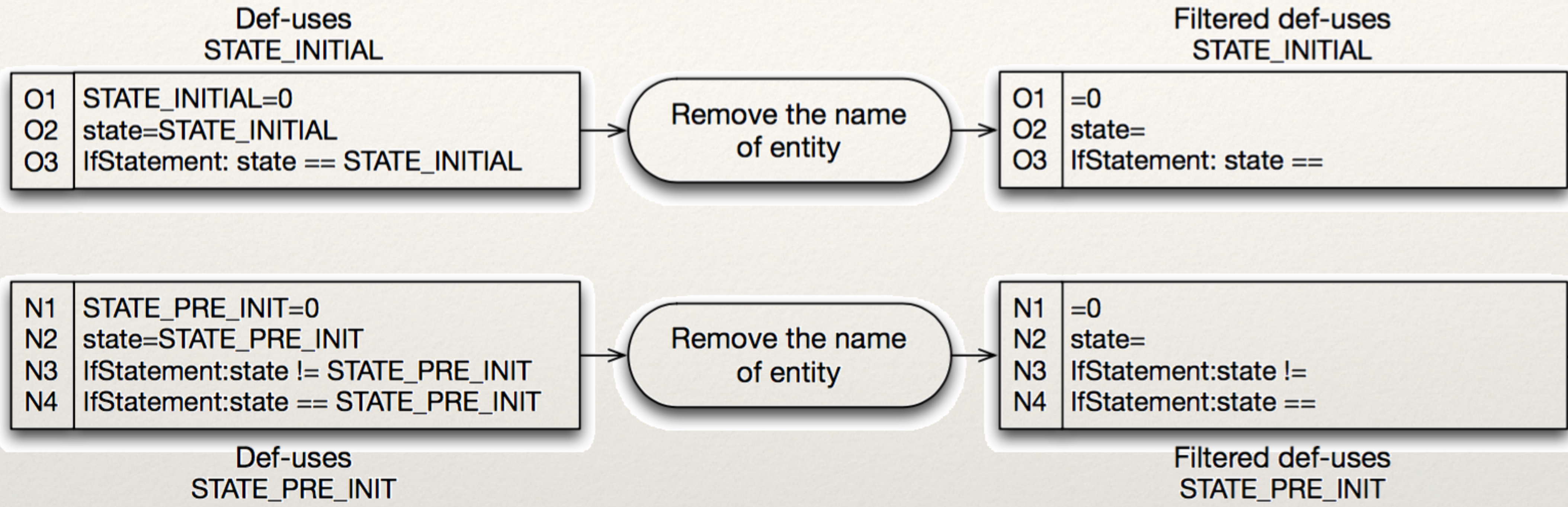
New version

```
...  public static final int ACC_INIT_END=1;
...  public static final int ACCOUNTS=2;
...  // State
203  /** Server is not initialized
204   */
205  public static final int STATE_PRE_INIT=0;
206  /** Server was initialized, engineInit() was called.
...   *  addContext() can be called.
...   */
...  public static final int STATE_INIT=1;
...  ...
...  // ------- local variables -------
...  private int state=STATE_PRE_INIT;
...  ...
...  public final void initContext(Context ctx)
...                     throws TomcatException {
...    if (state != STATE_PRE_INIT) {
...      ...
...    }
...  }
...  ...
...  /**
...   * Adds a new Context to the set managed
...   * by this ContextManager.
...   * @param ctx context to be added.
...   */
...   public void addContext(Context ctx)
...                 throws TomcatException {
...   ...
...   if(state == STATE_INITIAL)
...     return;
...   ...
...   }
   ...
   }
```

Renaming Detection

Detection

- Line mapping
- Entity mapping
- Data flow analys
- **Score of mapping**



Def-uses STATE_INITIAL

O1 STATE_INITIAL=0
O2 state=STATE_INITIAL
O3 IfStatement: state == STATE_INITIAL

Remove the name of entity

Filtered def-uses STATE_INITIAL

O1 =0
O2 state=
O3 IfStatement: state ==

N1 STATE_PRE_INIT=0
N2 state=STATE_PRE_INIT
N3 IfStatement:state != STATE_PRE_INIT
N4 IfStatement:state == STATE_PRE_INIT

Def-uses STATE_PRE_INIT

Remove the name of entity

N1 =0
N2 state=
N3 IfStatement:state !=
N4 IfStatement:state ==

Filtered def-uses STATE_PRE_INIT

Hungarian algorithm
maximize score

	N1	N2	N3	N4
O1	**1.00**	0.00	0.00	0.00
O2	0.00	**1.00**	0.00	0.00
O3	0.00	0.00	0.98	**1.00**

Statement score matrix

Statement Similarity Threshold (SST)

Declaration Similarity Threshold (DST)

- We use the Normalized Levenshtein edit Distance (NLD)

$$NLD = \frac{LD(S_i,S_j)}{Length(S_i) + Length(S_j)}$$

$$similarity\_score = 1 - NLD$$

Renaming Detection

Detection

- Line mapping
- Entity mapping
- Data flow analysis
- **Score of mapping**

```
O1 -> N1
O2 -> N2
O3 -> N4
```

	N1	N2	N3	N4
O1	**1.00**	0.00	0.00	0.00
O2	0.00	**1.00**	0.00	0.00
O3	0.00	0.00	0.98	**1.00**

Statement score matrix

```
score(STATE_INITIAL,STATE_PRE_INIT)=1+1+1=3
```

$$\mathtt{score}(E_l,E_k)=\begin{cases}\mathtt{score=sum}(S_{i,j}) \\ i,j \in \mathtt{mapped\ statements} & \mathtt{numMatched >= NST} \\ \mathtt{score=0} & \mathtt{numMatched < NST}\end{cases}$$

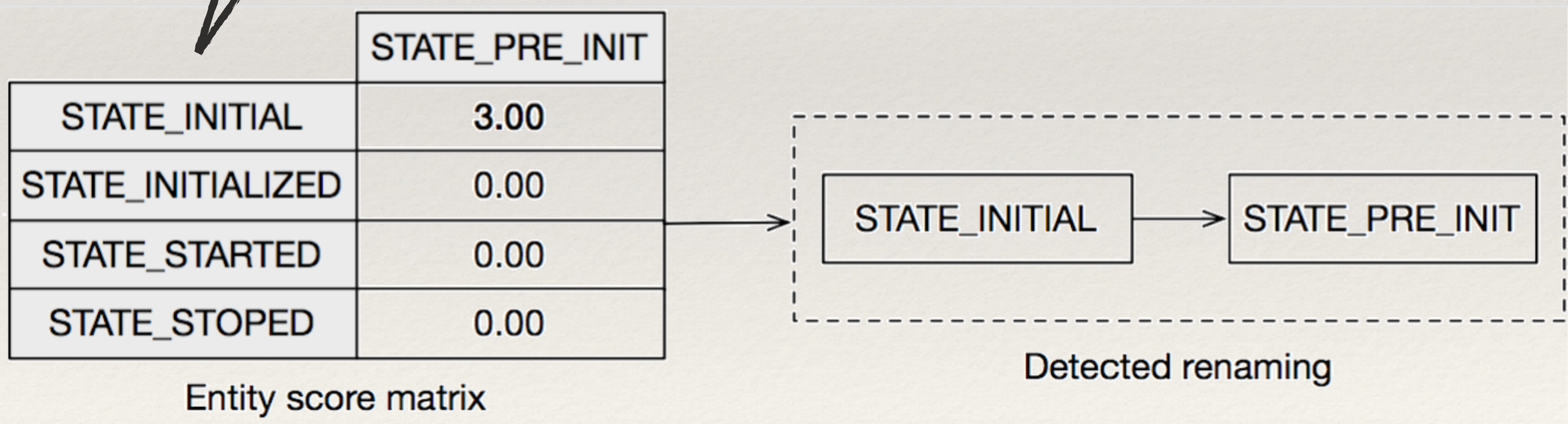
Number of matched Statement Threshold

Renaming Detection

Detection

- Line mapping
- Entity mapping
- Data flow analysis
- Score of mapping
- **Renaming detection**

Hungarian algorithm
maximize score



	STATE_PRE_INIT
STATE_INITIAL	**3.00**
STATE_INITIALIZED	0.00
STATE_STARTED	0.00
STATE_STOPED	0.00

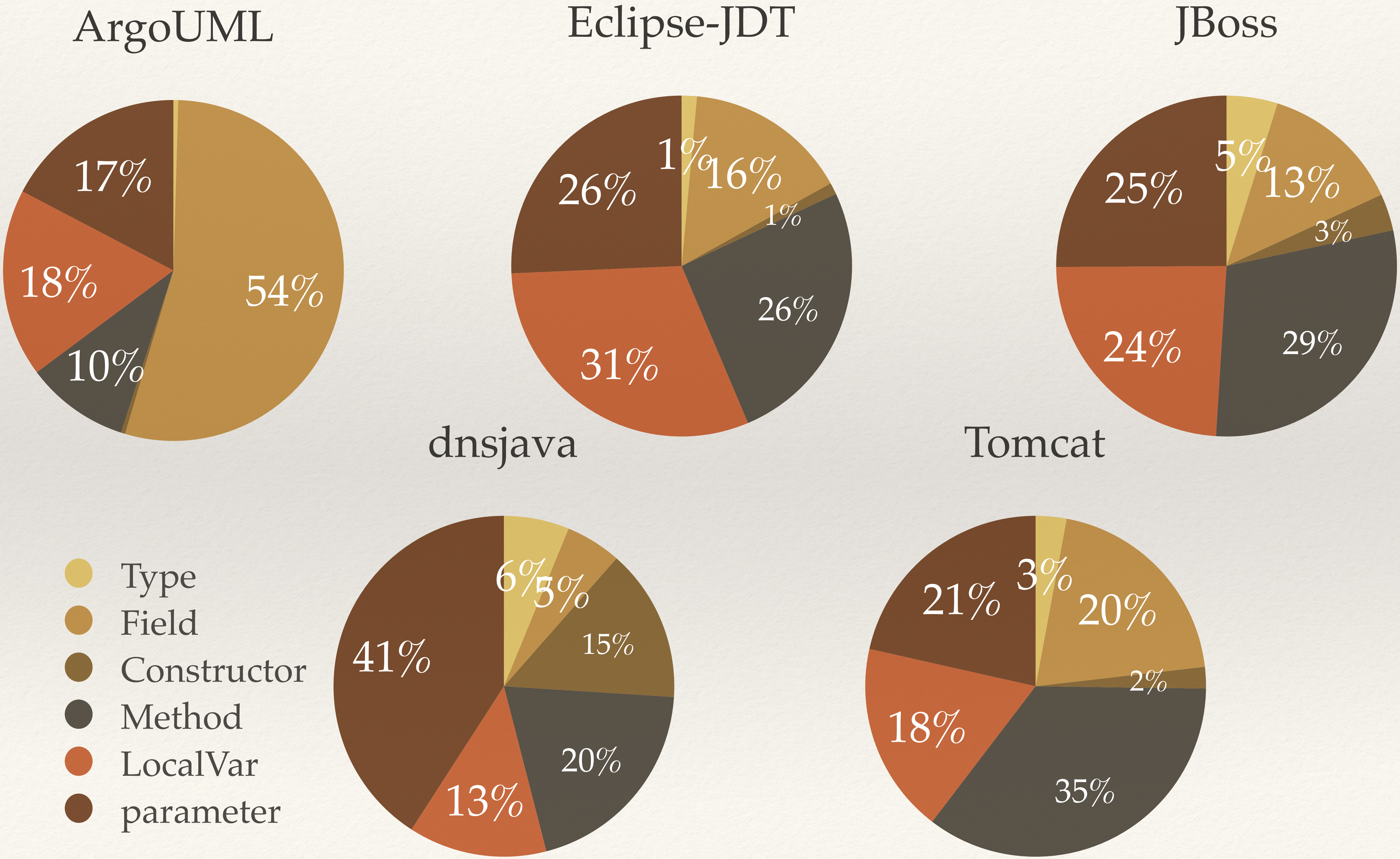
Entity score matrix

Detected renaming

Analyzed Programs

Programs	Period	Total files	Revisions
Tomcat	1999–2006	12,205	46,498
Eclipse-JDT	2001–2006	5,758	54,571
ArgoUML	1998-2012	300	68,400
JBoss	1999–2011	40,003	25,028
dnsjava	1998–2011	365	1,415

Detection Results



Detection Accuracy

How accurate is the set of renamings detected by REPENT?

- Sample size, 95%, ∓5% = 1,723
- Two evaluators, voting, conflict resolved by third evaluator

$$\text{Precision} = \frac{|\,\text{TPS}\,|}{|\,\text{TPS}\,| + |\,\text{FPS}\,|} = \mathbf{88\%}$$

Low precision in detection of parameter:
dnsjava: 54%
Tomcat : 67%

Not enough parameters when calibrating the thresholds

Programs	Precision
Tomcat	80%
Eclipse-JDT	**94%**
ArgoUml	**97%**
JBoss	**91%**
dnsjava	78%

Detection Accuracy

How complete is the set of renamings detected by REPENT?

- Commit logs “renam”, remove false positives

$$\text{Recall} = \frac{|\,\text{DCR} \cap \text{DR}\,|}{|\,\text{DCR}\,|} = \mathbf{92\%}$$

Eclipse-JDT: Failed to identify Class renamings due to missed file renamings.

ArgoUML: 3/4 documente renamings is identified. The missed case was a combination combination of renaming and refactoring

Programs	Recall
Tomcat	**100%**
Eclipse-JDT	63%
ArgoUml	75%
JBoss	**96%**
dnsjava	**98%**

Detection Summary

- We identify 33,812 renamings in five open source programs.

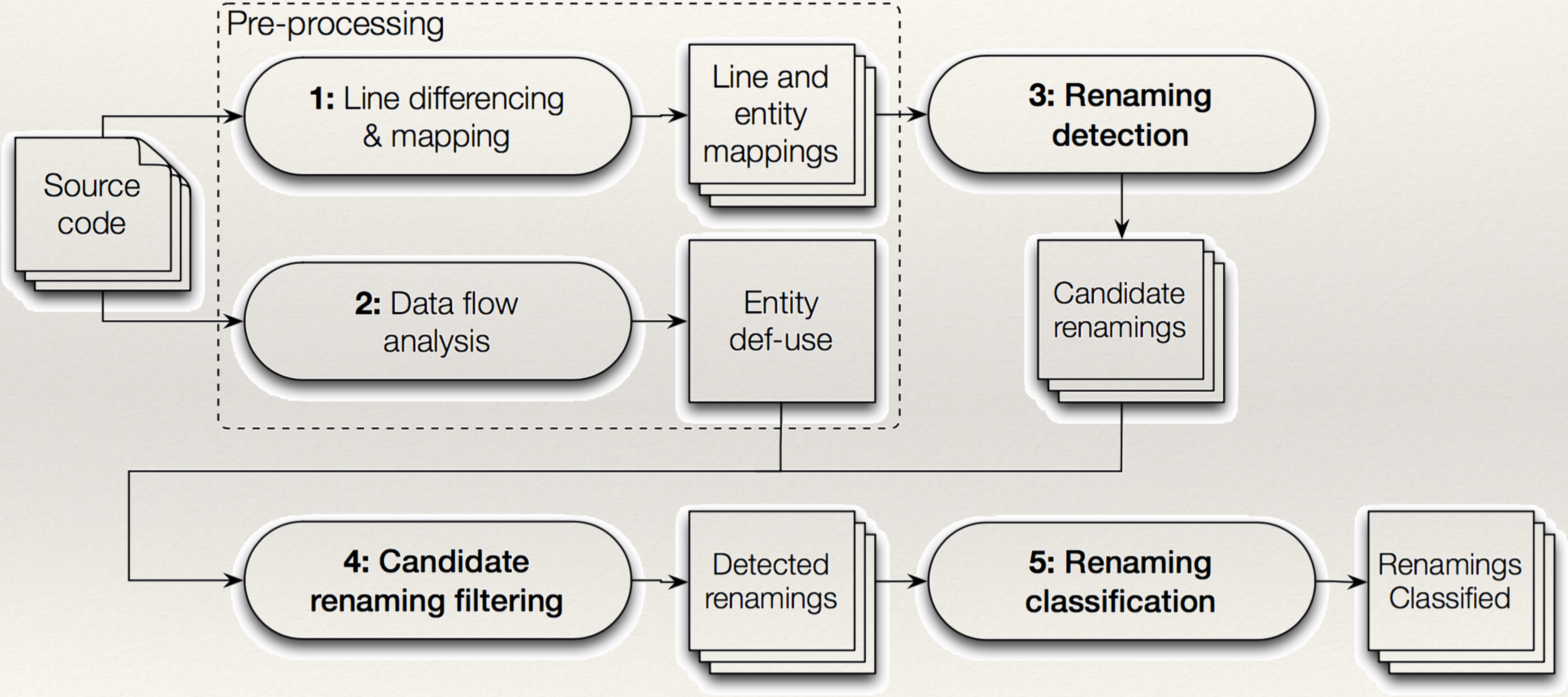
- We manually validated a sample size (95% +- 5%) of 1,723 renamings.

- The overall precision of detection is 88%.

- The overall recall of detection is 92%.

- The high precision and recall make our approach suitable for identifying renamings.

Detection and Classification Approach



Taxonomy of Renamings

Entity kind

- package
- class
- method
- field
- constructor
- parameters
- local vars

Form of renamings

- simple
- complex
- formatting only
- term reordering

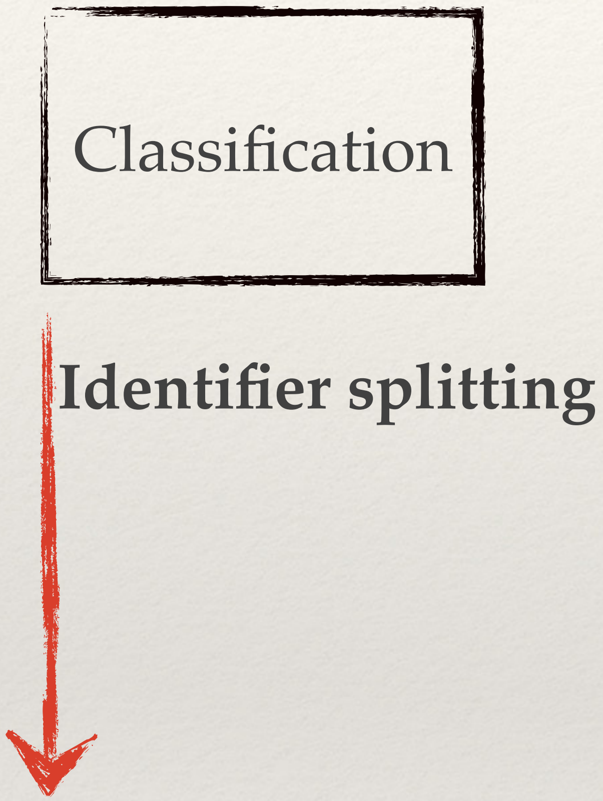
Semantic changes

- preserve meaning
- change meaning
- narrow meaning
- broaden meaning
- add meaning
- remove meaning
- none

Grammar changes

- part of speech change
- none

Renaming Classification



Classification

Identifier splitting

```
invParamsPtr -> invalidParamReferencesPtr

inv Params Ptr  invalid Param References Ptr
```

Renaming Classification

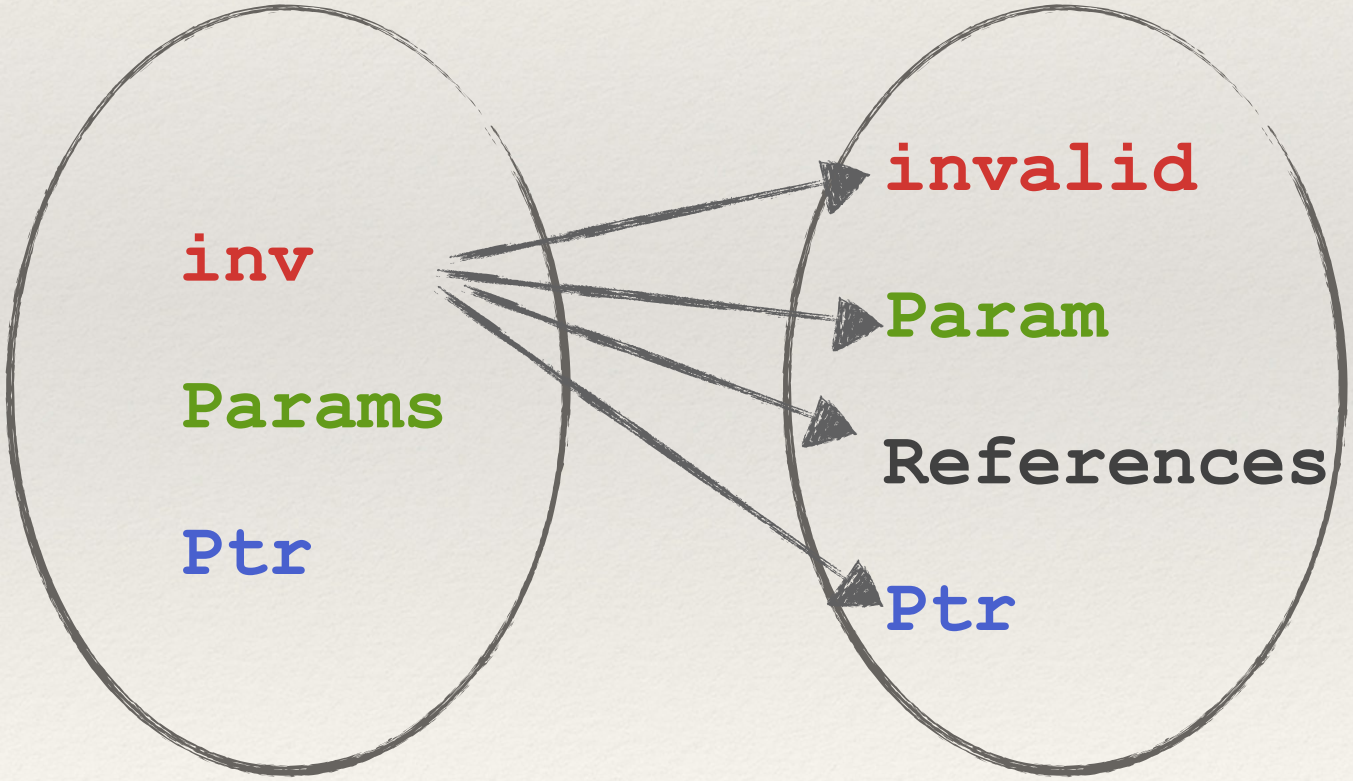
Classification

Identifier splitting

Term mapping

```
invParamsPtr -> invalidParamReferencesPtr

inv Params Ptr  invalid Param References Ptr
```



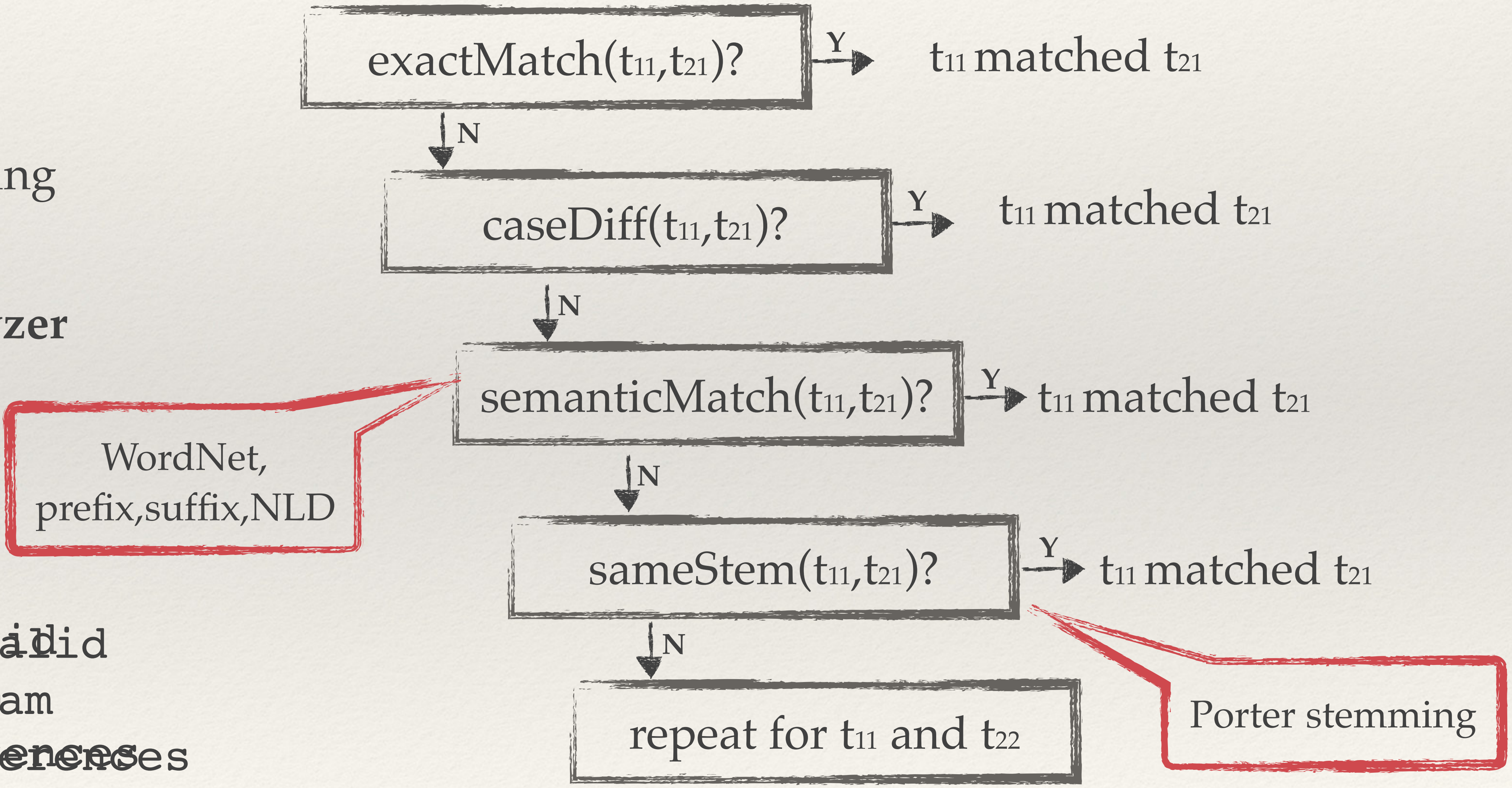
Renaming Classification

`invParamsPtr -> invalidParamReferencesPtr`

`inv Params Ptr  invalid Param References Ptr`

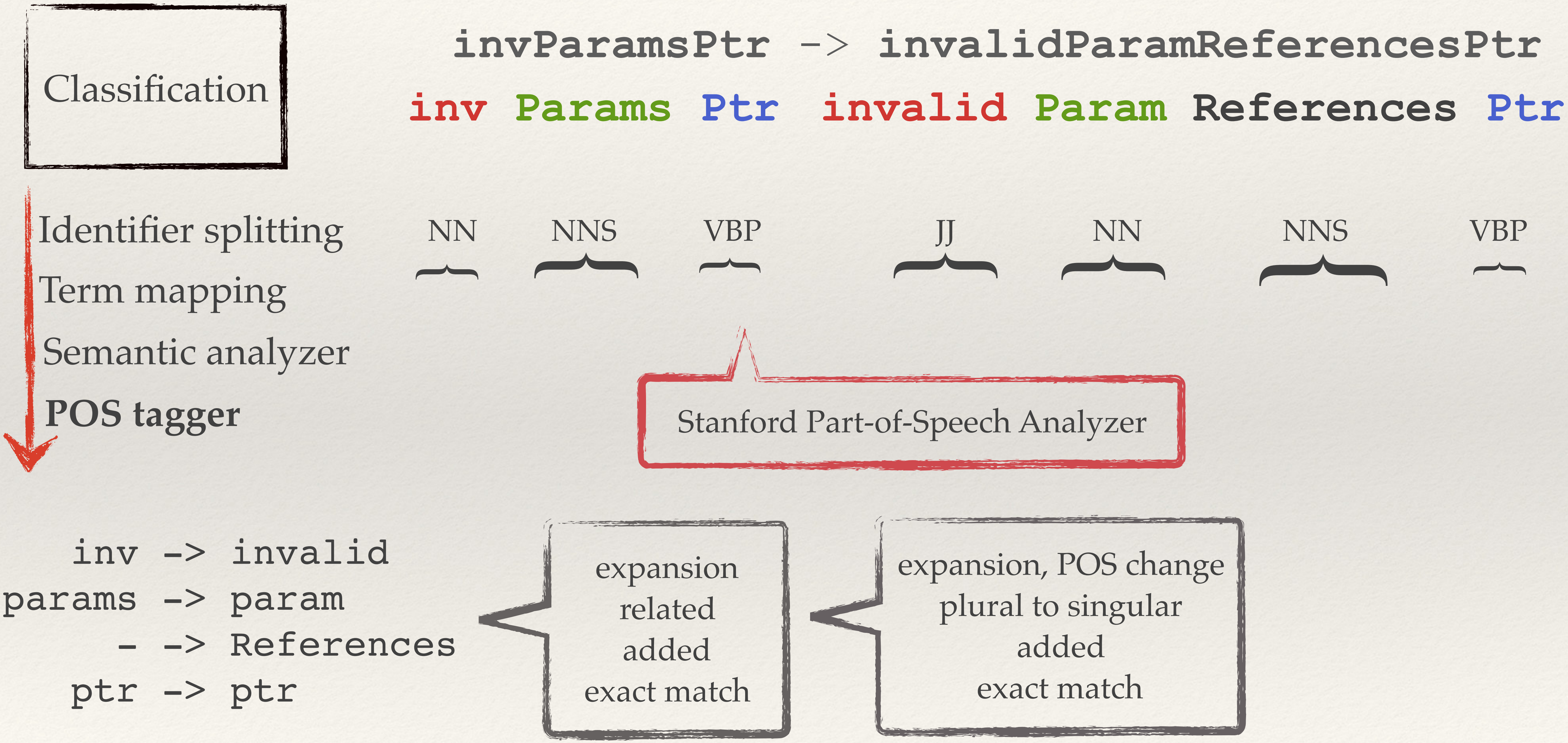
Classification

- Identifier splitting
- Term mapping
- **Semantic analyzer**



```
   inv -> invalid
params -> param
     - -> References
   ptr -> ptr
```

Renaming Classification



Classification

Identifier splitting

Term mapping

Semantic analyzer

POS tagger

```
invParamsPtr -> invalidParamReferencesPtr
```

```
inv Params Ptr  invalid Param References Ptr
```

NN NNS VBP JJ NN NNS VBP

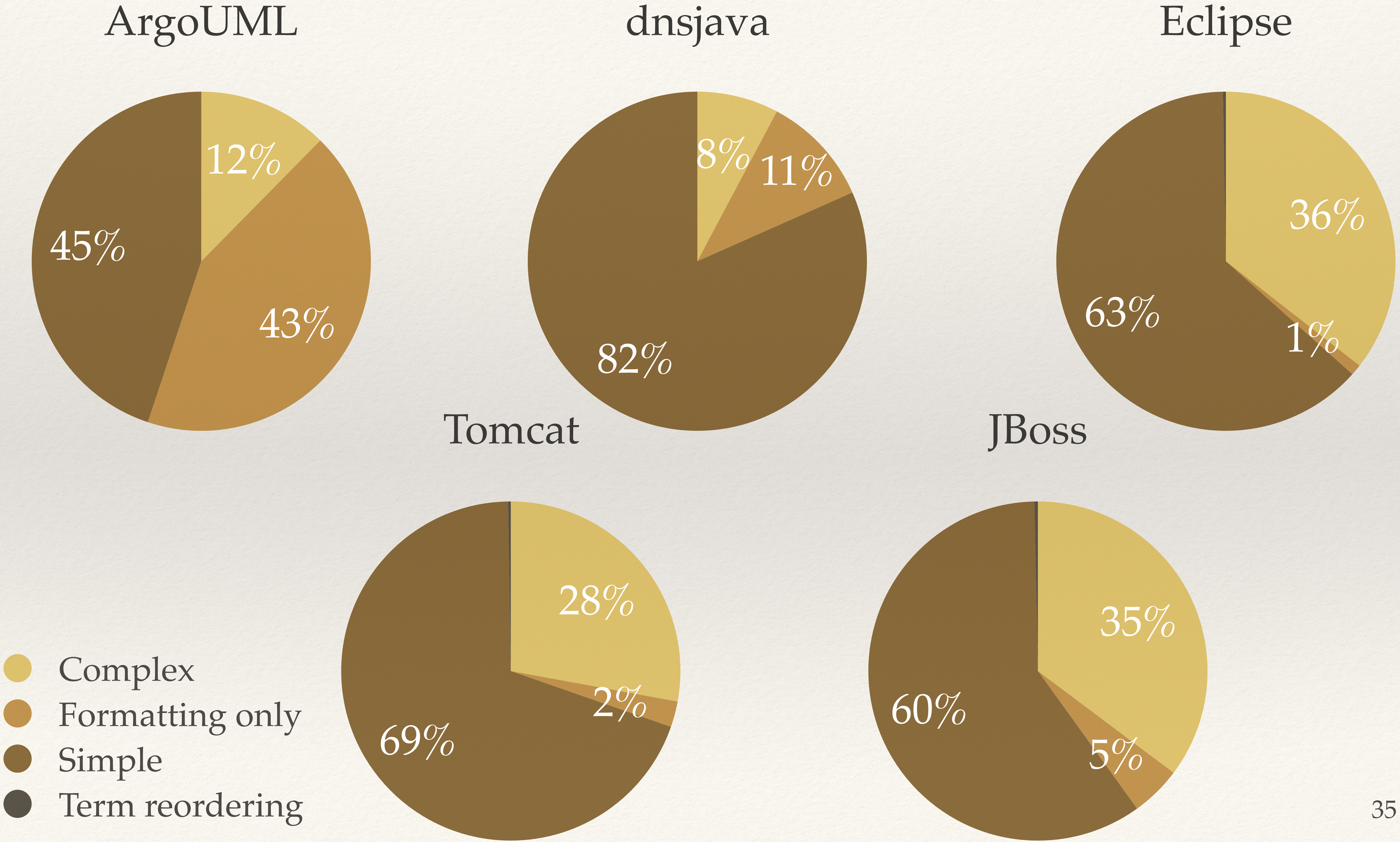
Stanford Part-of-Speech Analyzer

```
   inv -> invalid
params -> param
     - -> References
   ptr -> ptr
```

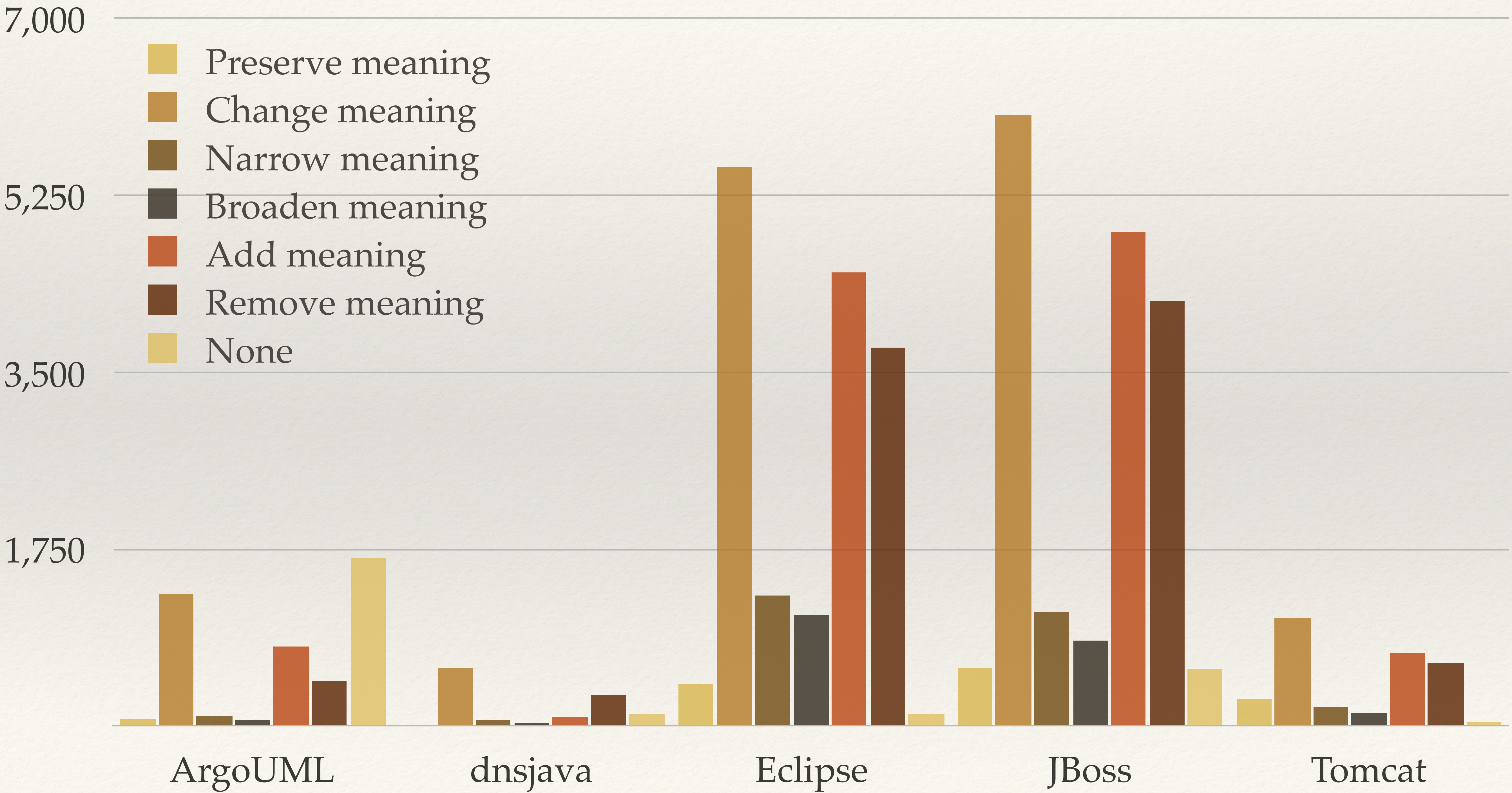
expansion
related
added
exact match

expansion, POS change
plural to singular
added
exact match

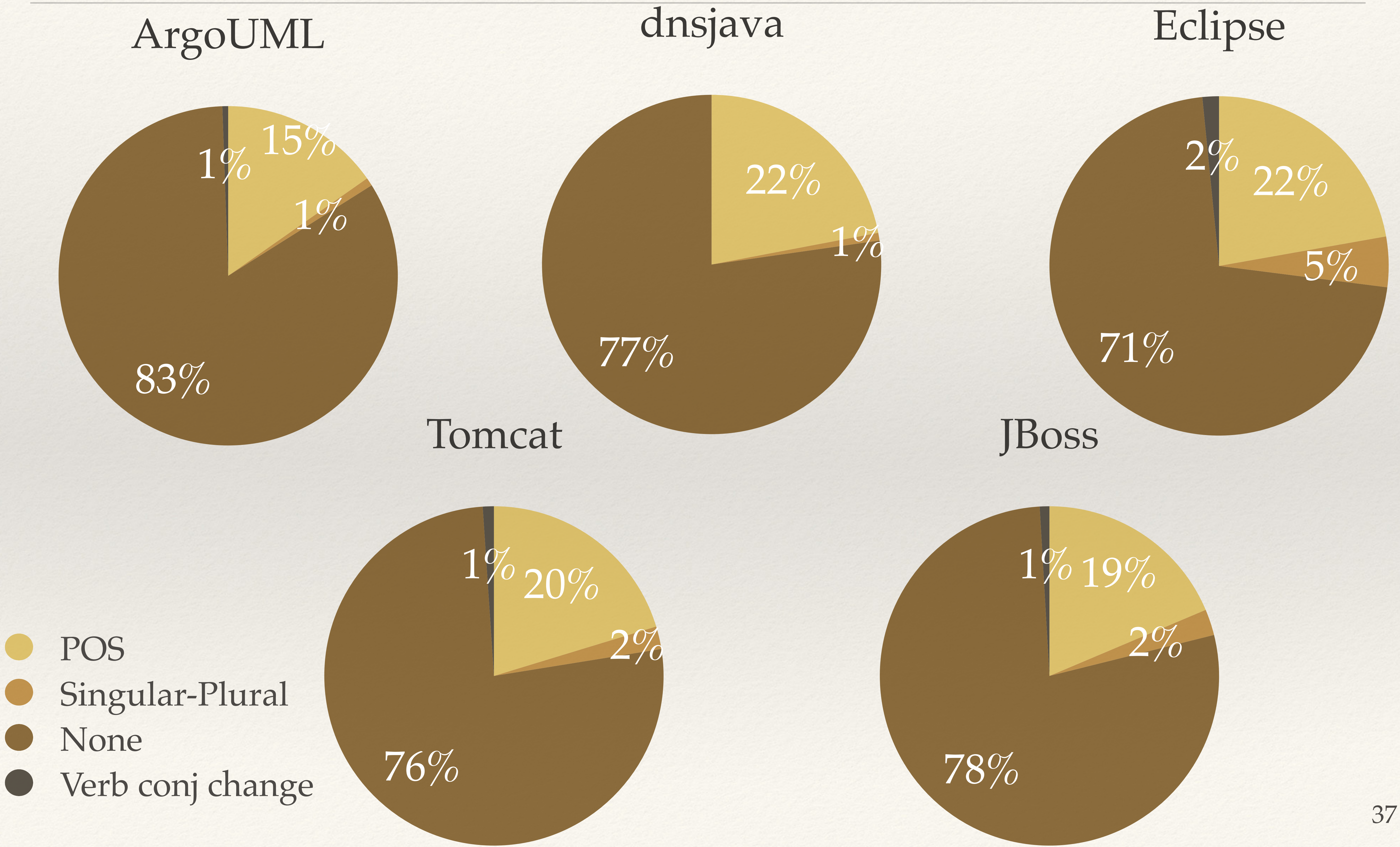
Results for Form of Renaming



Results Semantic changes



Results Grammar Change



Classification Accuracy

How accurate is the set of classified renamings?

- Sample size, 95%, ∓10% , for each level of dimension
- 330 , 1102, 355, for each dimension respectively.
- Two evaluators, voting, conflict was resolve

Programs	Form of renaming	Semantic changes	Grammar changes
Tomcat	**96%**	72%	61%
Eclipse-JDT	**96%**	**82%**	75%
ArgoUML	**100%**	**88%**	**88%**
JBoss	**98%**	79%	72%
dnsjava	**100%**	**92%**	**100%**

Classification Accuracy

Programs	Form of renaming	Semantic changes	Grammar changes
Tomcat	**96%**	72%	61%
Eclipse-JDT	**96%**	**82%**	75%
ArgoUML	**100%**	**88%**	**88%**
JBoss	**98%**	79%	72%
dnsjava	**100%**	**92%**	**100%**

wrong term mapping

narrow or broaden meaning

add or remove detailing

`is -> get` **hyponym**

`long -> short` **antonym**

- wrong splitting, all lower-cased

fairly accurate

- wrong term-mapping

- wrong relations between terms

singular / plural

verb conj change

low precision in other POS

Gupta et al. Part-of-Speech Tagging of Program Identifiers for Improved Text-based Software Engineering Tools . ICP 2013

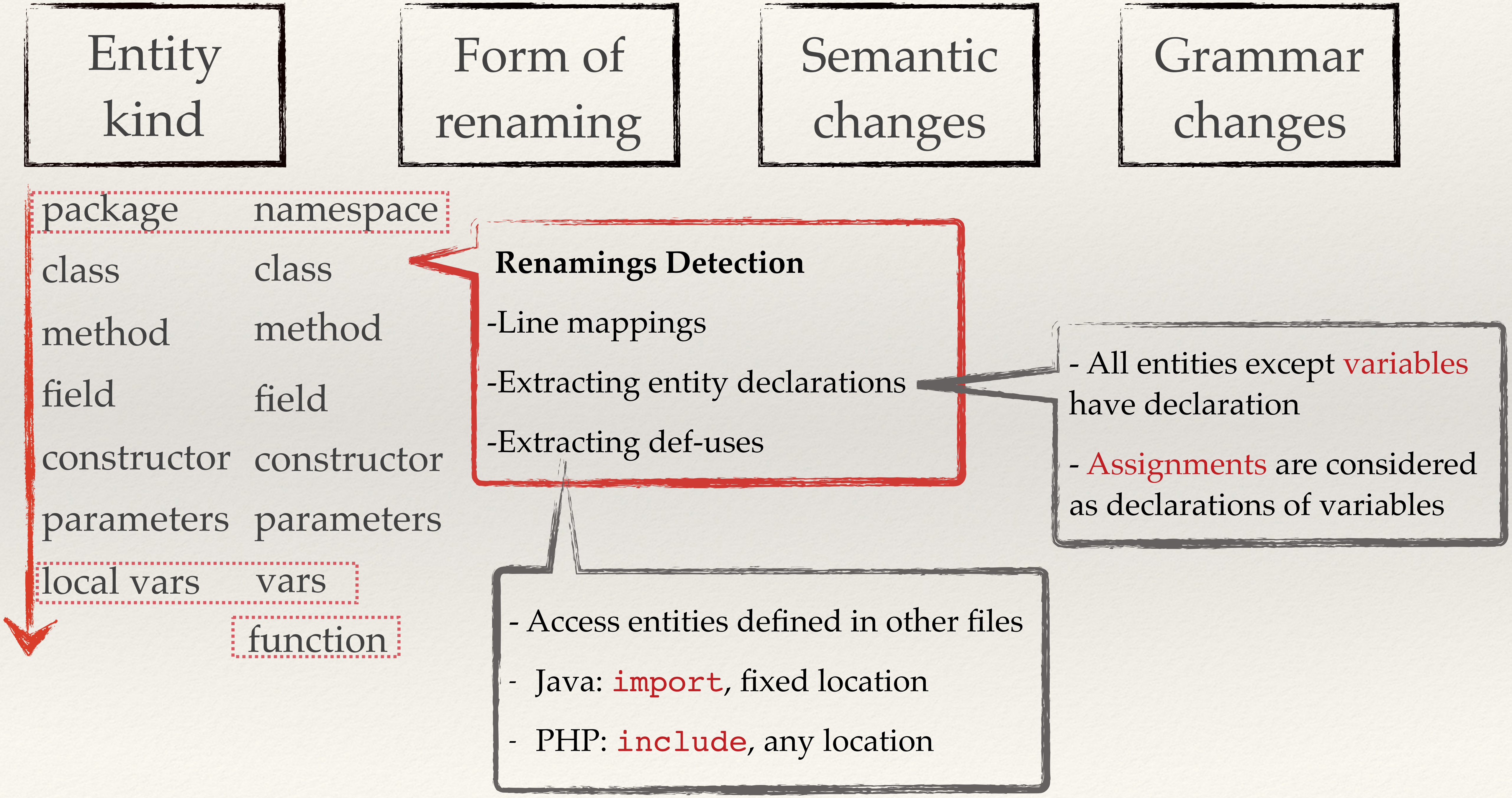
Applicability to other languages

- Java is a statically type and object-oriented language.

- We are interested to investigate the applicability of our approach to a language different from Java.

- We choose PHP as it is a popular language, it is a dynamically type language and it allows scripting, procedural and object-oriented programming.

Challenges!!



Entity kind

Form of renaming

Semantic changes

Grammar changes

package	namespace
class	class
method	method
field	field
constructor	constructor
parameters	parameters
local vars	vars
	function

Renamings Detection

-Line mappings

-Extracting entity declarations

-Extracting def-uses

- All entities except variables have declaration

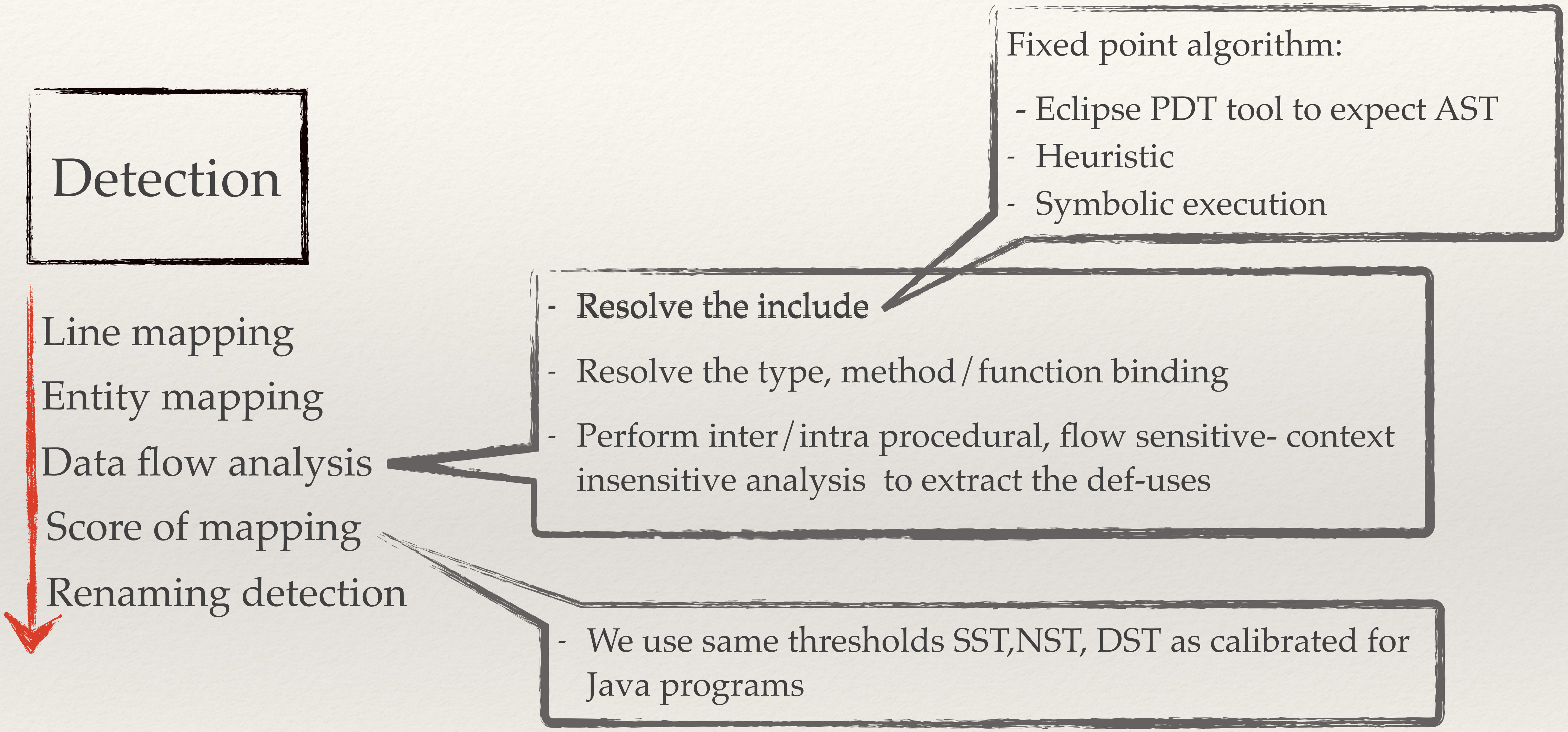
- Assignments are considered as declarations of variables

- Access entities defined in other files

- Java: `import`, fixed location

- PHP: `include`, any location

PHP Renamings Detection



Detection

Line mapping

Entity mapping

Data flow analysis

Score of mapping

Renaming detection

Fixed point algorithm:

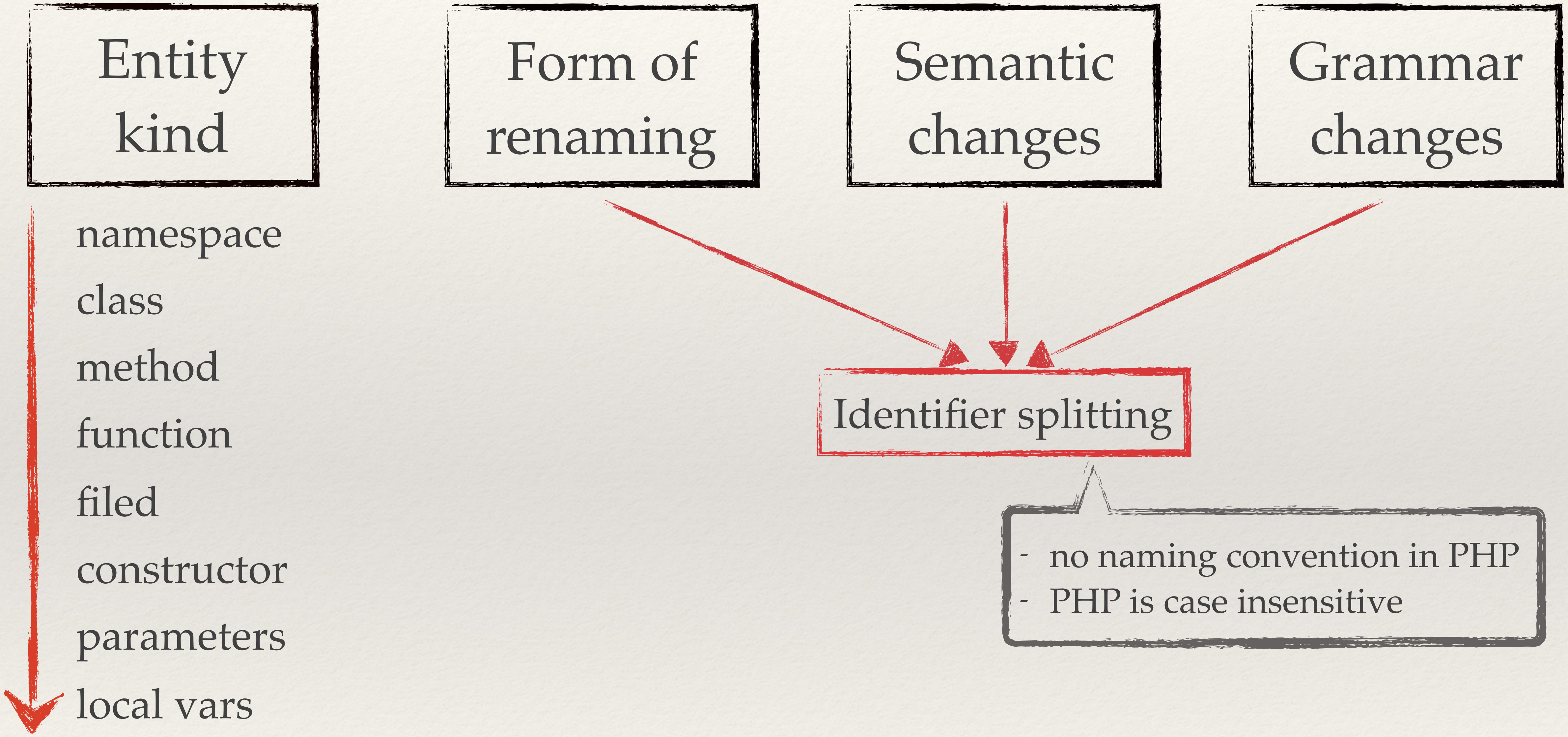
- Eclipse PDT tool to expect AST
- Heuristic
- Symbolic execution

- **Resolve the include**
- Resolve the type, method/function binding
- Perform inter/intra procedural, flow sensitive- context insensitive analysis to extract the def-uses

- We use same thresholds SST,NST, DST as calibrated for Java programs

Eshkevari *et al.* Identifying and Locating Interference Issues in PHP Applications. **ICPC 2014**

Challenges!!



Entity kind

- namespace
- class
- method
- function
- filed
- constructor
- parameters
- local vars

Form of renaming

Semantic changes

Grammar changes

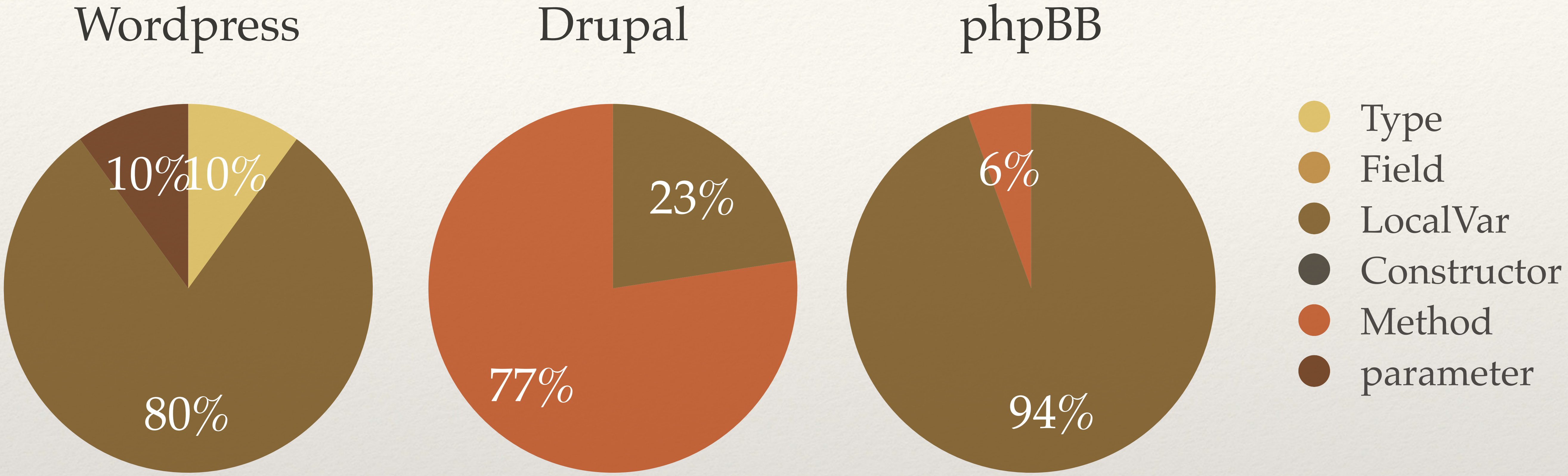
Identifier splitting

- no naming convention in PHP
- PHP is case insensitive

Analyzed Programs

Programs	Period	Total files	Revisions
Wordpress	06 March 2015 06 April 2015	547	386
Drupal	19 March 2011 30 May 2011	492	322
phpBB	18 Jun 2006 21 July 2006	143	368

Detection Results

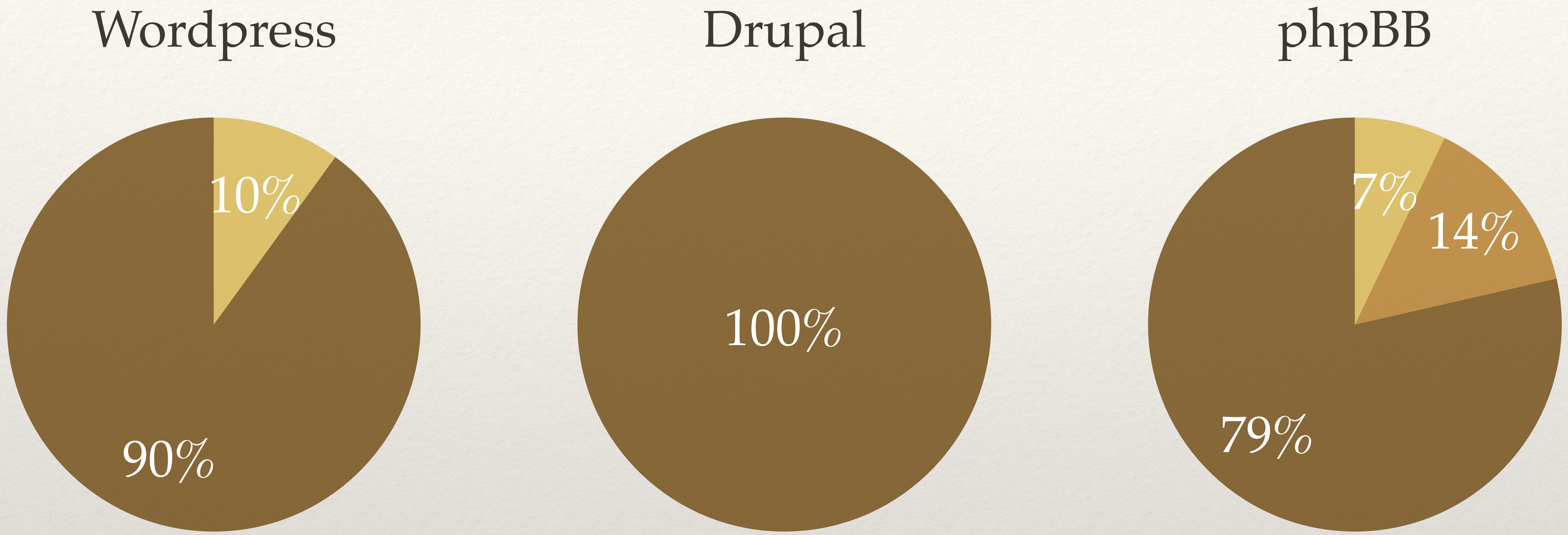


$$\text{Precision} = \frac{|\text{ TPS }|}{|\text{ TPS }| + |\text{ FPS }|} = \mathbf{85\%}$$

$$\text{Recall} = \frac{|\text{ TPS }|}{|\text{ TPS }| + |\text{ FNS }|} = \mathbf{78\%}$$

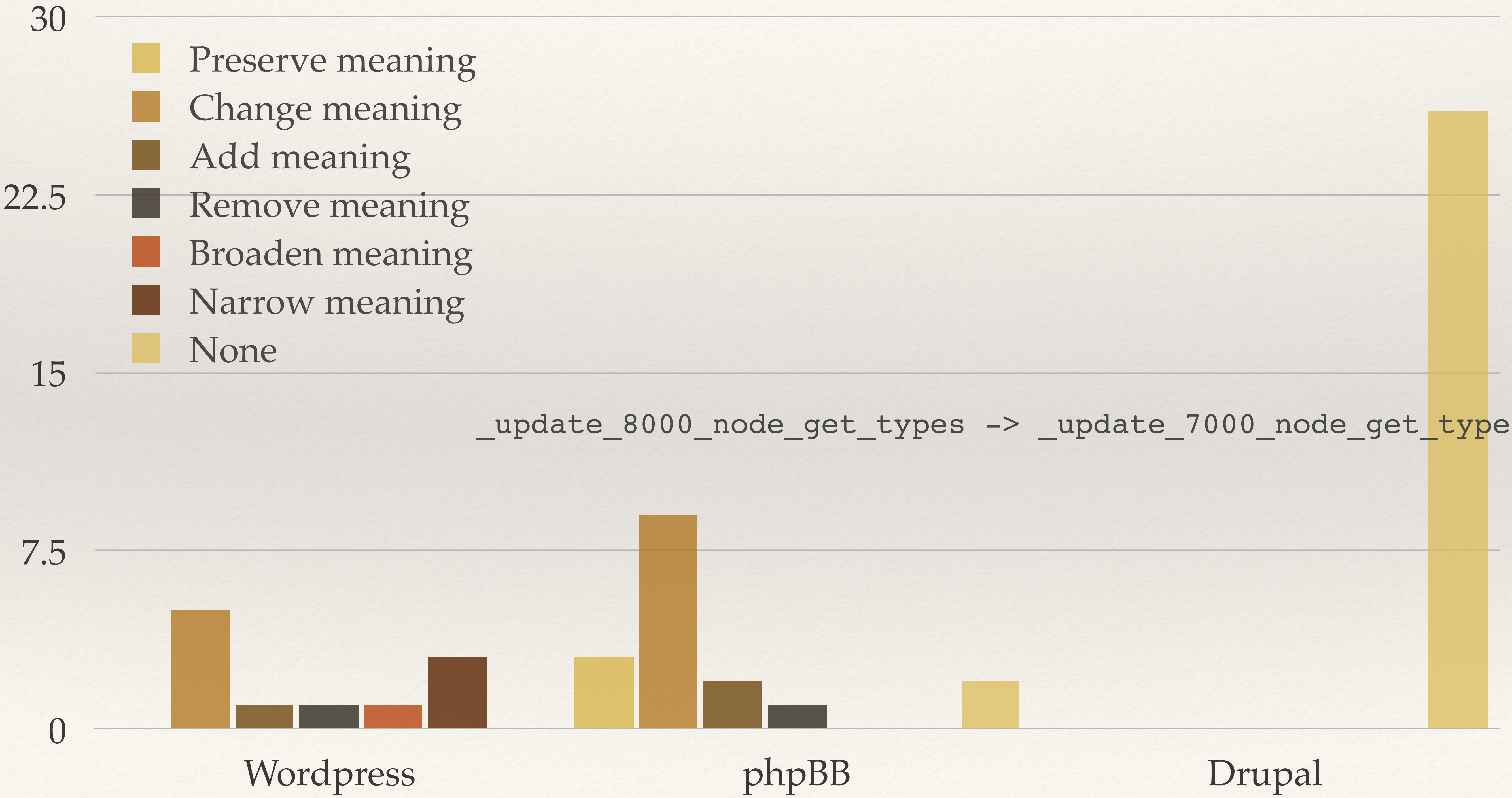
Programs	Precision	Recall
Wordpress	**100%**	62%
Drupal	78%	81%
phpBB	**84%**	**87%**

Results for Form of Renaming



Programs	Precision
Wordpress	**100%**
Drupal	**100%**
phpBB	**100%**

Results Semantic changes



`_update_8000_node_get_types -> _update_7000_node_get_types`

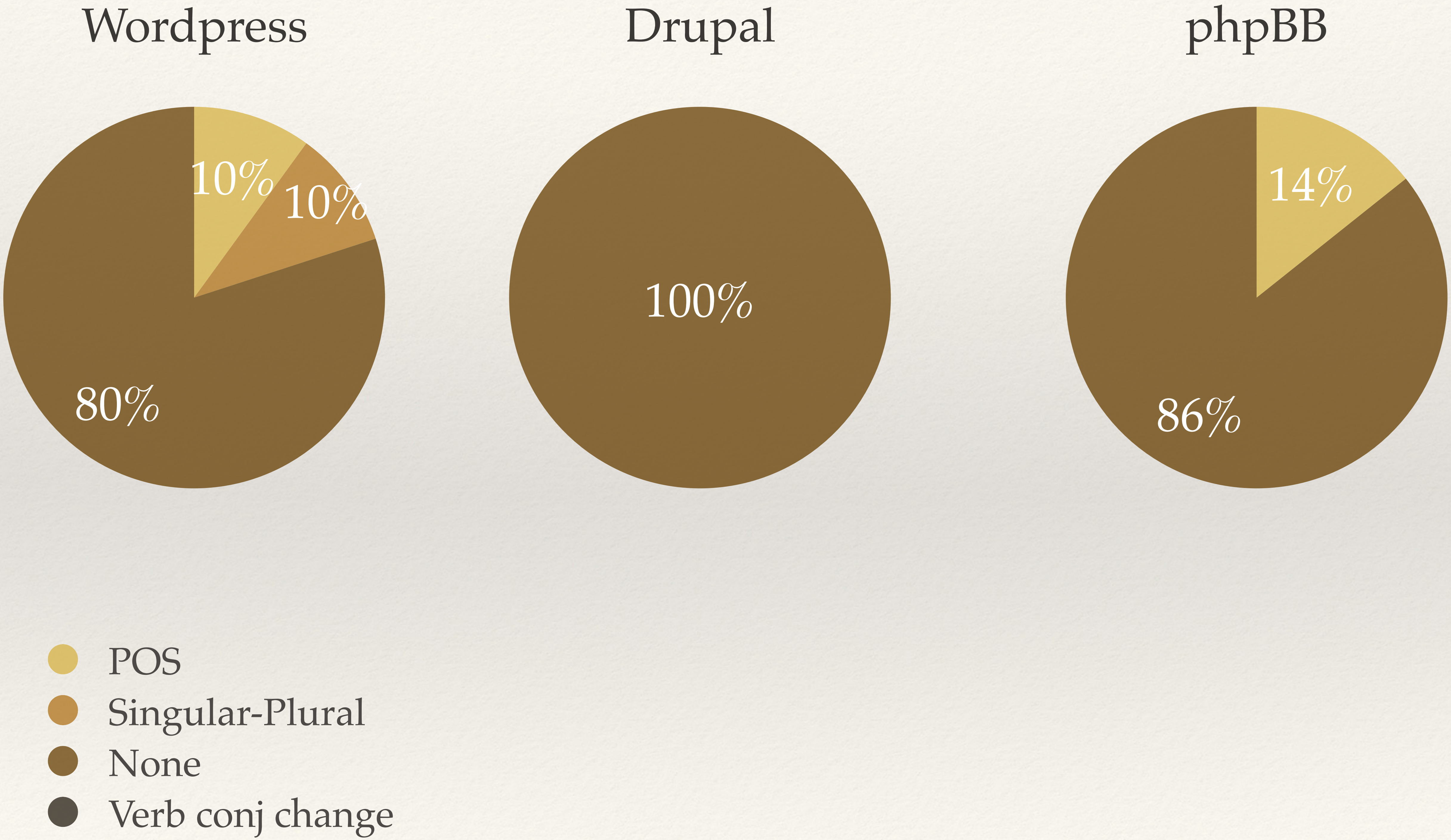
Precision of Semantic Change

Semantic change	Wordpress	Drupal	phpBB
Preserve meaning	-	-	100%
Change meaning	60%	-	57%
Remove meaning	0%	-	100%
Add meaning	100%	-	50%
Broaden meaning	0%	-	-
Narrow meaning	100%	-	-
None	-	100%	100%

```
title->link_text
WP_Customize_Upload_Control->WP_Customize_Media_Control
```

```
ACL_NO->ACL_NEVER
module_name->module_basename
```

Results Grammar Change


Wordpress

Drupal

phpBB

Precision of Grammar Change

Semantic change	Wordpress	Drupal	phpBB
Singular/Plural	100%	-	-
Verb conj change	-	-	-
Other POS	100%	-	100%
None	100%	100%	100%

$\overbrace{\texttt{tags}}^{\text{NNS}}\texttt{->}\ \overbrace{\texttt{tag}}^{\text{NN}}$

$\overbrace{\texttt{new}}^{\text{JJ}}\texttt{\_}\overbrace{\texttt{content}}^{\text{NN}}\ \texttt{->}\ \overbrace{\texttt{new}}^{\text{JJ}}\texttt{\_}\overbrace{\texttt{src}}^{\text{NN}}$

$\overbrace{\texttt{ACL}}^{\text{NN}}\texttt{\_}\overbrace{\texttt{NO}}^{\text{RB}}\texttt{->}\ \overbrace{\texttt{ACL}}^{\text{NN}}\texttt{\_}\overbrace{\texttt{NEVER}}^{\text{RB}}$

$\overbrace{\texttt{column}}^{\text{NN}}\texttt{\_}\overbrace{\texttt{type}}^{\text{NN}}\ \texttt{->}\ \overbrace{\texttt{orig}}^{\text{VBG}}\texttt{\_}\overbrace{\texttt{column}}^{\text{NN}}\texttt{\_}\overbrace{\texttt{type}}^{\text{NN}}$

Limitations of Detection

Construct validity:

- File renamings: thresholds 60%, CVS verging system
- Precision: human errors, subjectiveness
- Recall: small number of documented renamings

Internal validity:

- Calibration of thresholds, different results with different thresholds

External validity:

- Five open-source Java programs, different domain and size

Limitations of Classification

Construct validity:

- Precision and recall of detection
- Precision: human errors, subjectiveness

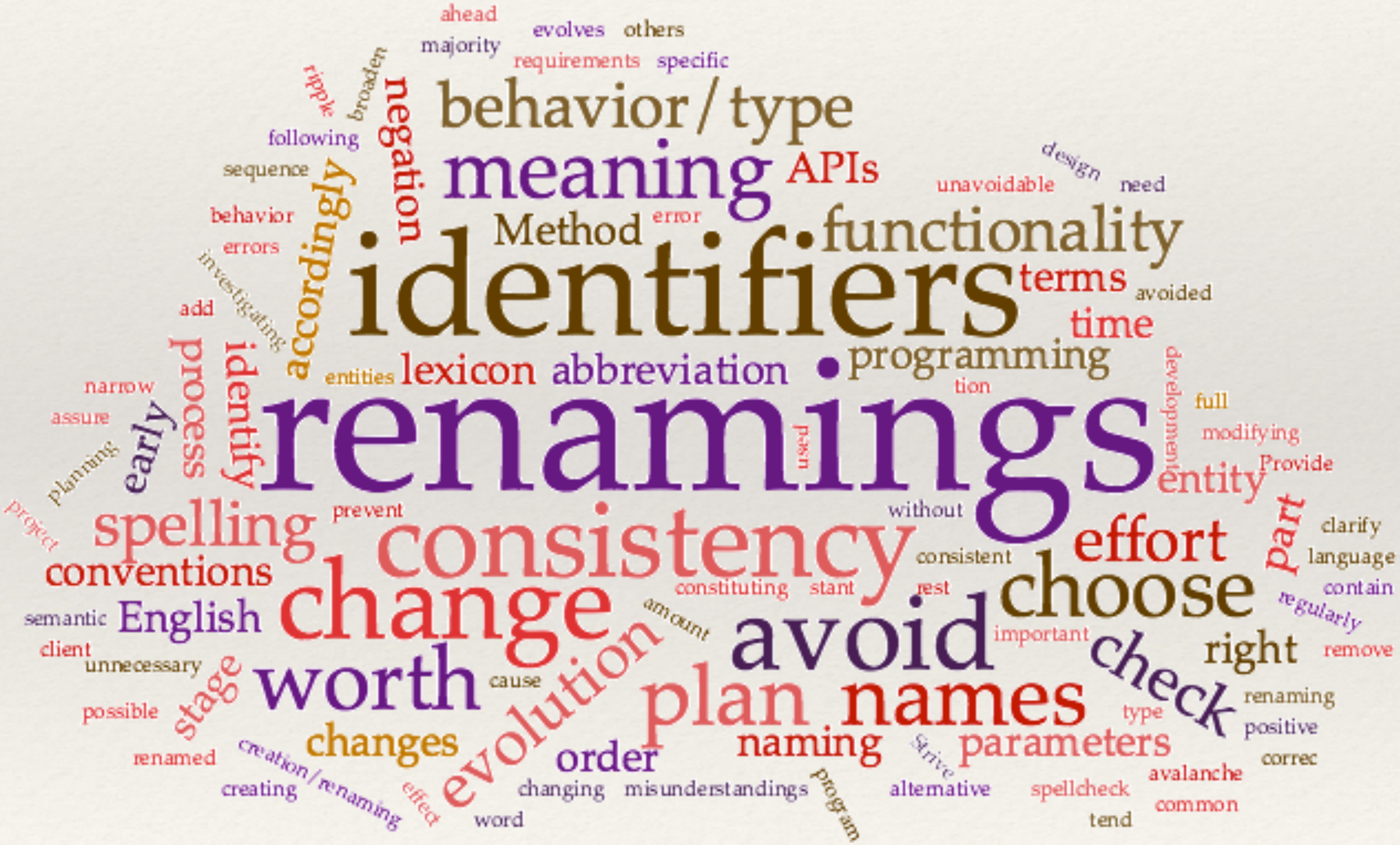
Internal validity:

- Use of threshold for term mapping, abbreviation and expansion

External validity:

- Generalization, Java and PHP, different trends

Lesson Learned



Conclusion

Goal: To understand when, why, and how developers rename identifiers.

- We know that renaming is quite a frequent activity during program evolution.
- It is mostly done when functionality of entities are changed and also during refactoring.
- Though sometimes there is an urge for renaming, it is avoided due to its cost and efforts.
- Developers tends to add and remove terms to rename identifiers, while keeping the part of speech intact.

Future Works

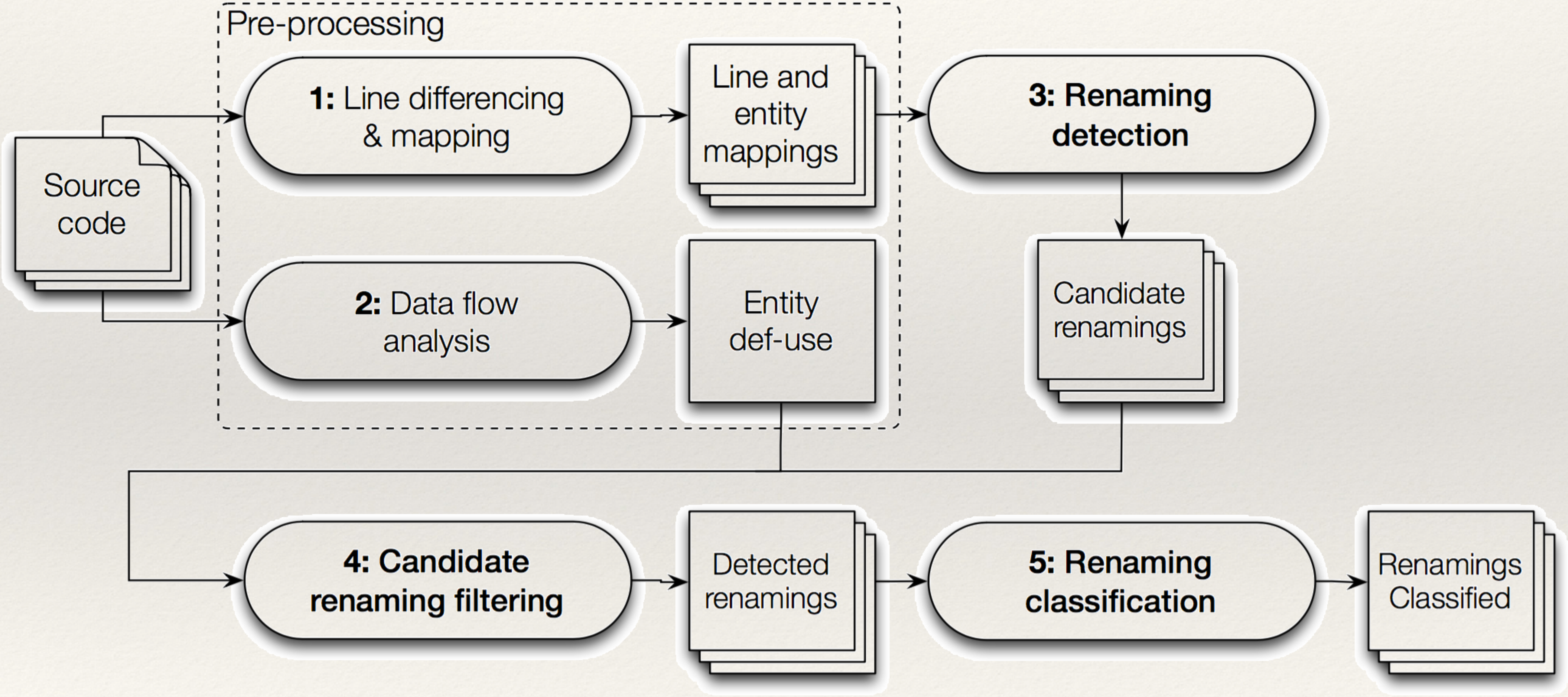
- Recommending a name for a new entity or an entity being renamed.
- Extends the study to other programming languages.
- Support automatic renamings in PHP programs.

Thank you :)

Examples of Renaming

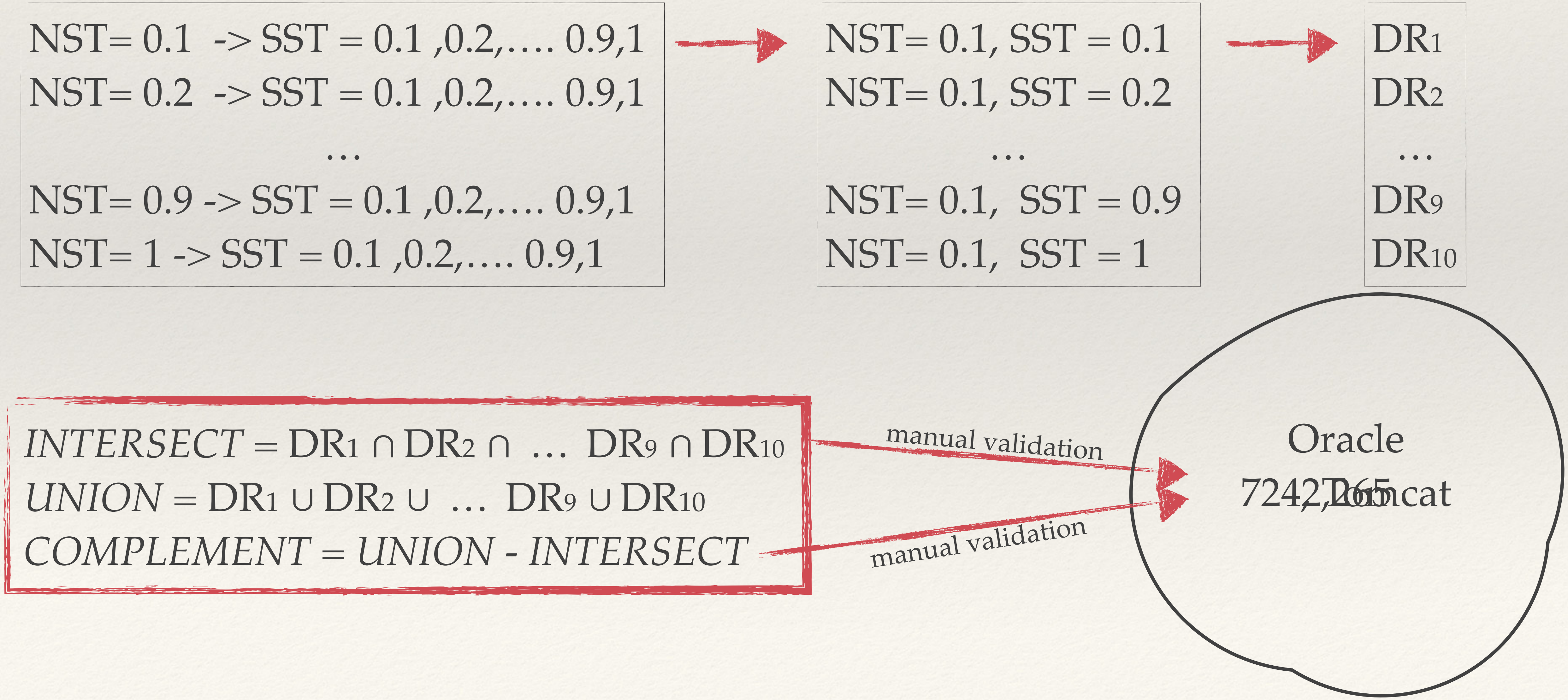
e -> t	parameter, Exception -> Throwable
g -> generalization	local var, MGeneralization -> Object
v -> list	local var, Vector -> List
sessState -> sessionState	local var, SessionState
length -> l	local var, int
jj_3R_70 -> jj_3R_69	method, private, boolean, final
verifyAXFR -> verifyStream	method, public, byte
rebuildTypesAffectedByMissingSecondaryTypes -> rebuildTypesAffectedBySecondaryTypes	method, protected, void
MicroContainerNotAdvisedAnnotationOverrideProxyAdvisorTestCase -> MicrocontainerAdvisedAnnotationOverrideProxyAdvisorTestCase	

Detection and Classification Approach



Thresholds for Detection

Declaration **S**imilarity **T**hreshold (**DST**) -> 0.7
Number of matched **S**tatement **T**hreshold (**NST**) -> [0,1] step +0.1
Statement **S**imilarity **T**hreshold (**SST**) -> for each fixed NST, [0,1] step +0.1



NST= 0.1 -> SST = 0.1 ,0.2,…. 0.9,1
NST= 0.2 -> SST = 0.1 ,0.2,…. 0.9,1
…
NST= 0.9 -> SST = 0.1 ,0.2,…. 0.9,1
NST= 1 -> SST = 0.1 ,0.2,…. 0.9,1

NST= 0.1, SST = 0.1
NST= 0.1, SST = 0.2
…
NST= 0.1, SST = 0.9
NST= 0.1, SST = 1

DR_1
DR_2
…
DR_9
DR_{10}

INTERSECT = $DR_1 \cap DR_2 \cap \ldots DR_9 \cap DR_{10}$
UNION = $DR_1 \cup DR_2 \cup \ldots DR_9 \cup DR_{10}$
COMPLEMENT = *UNION* - *INTERSECT*

manual validation

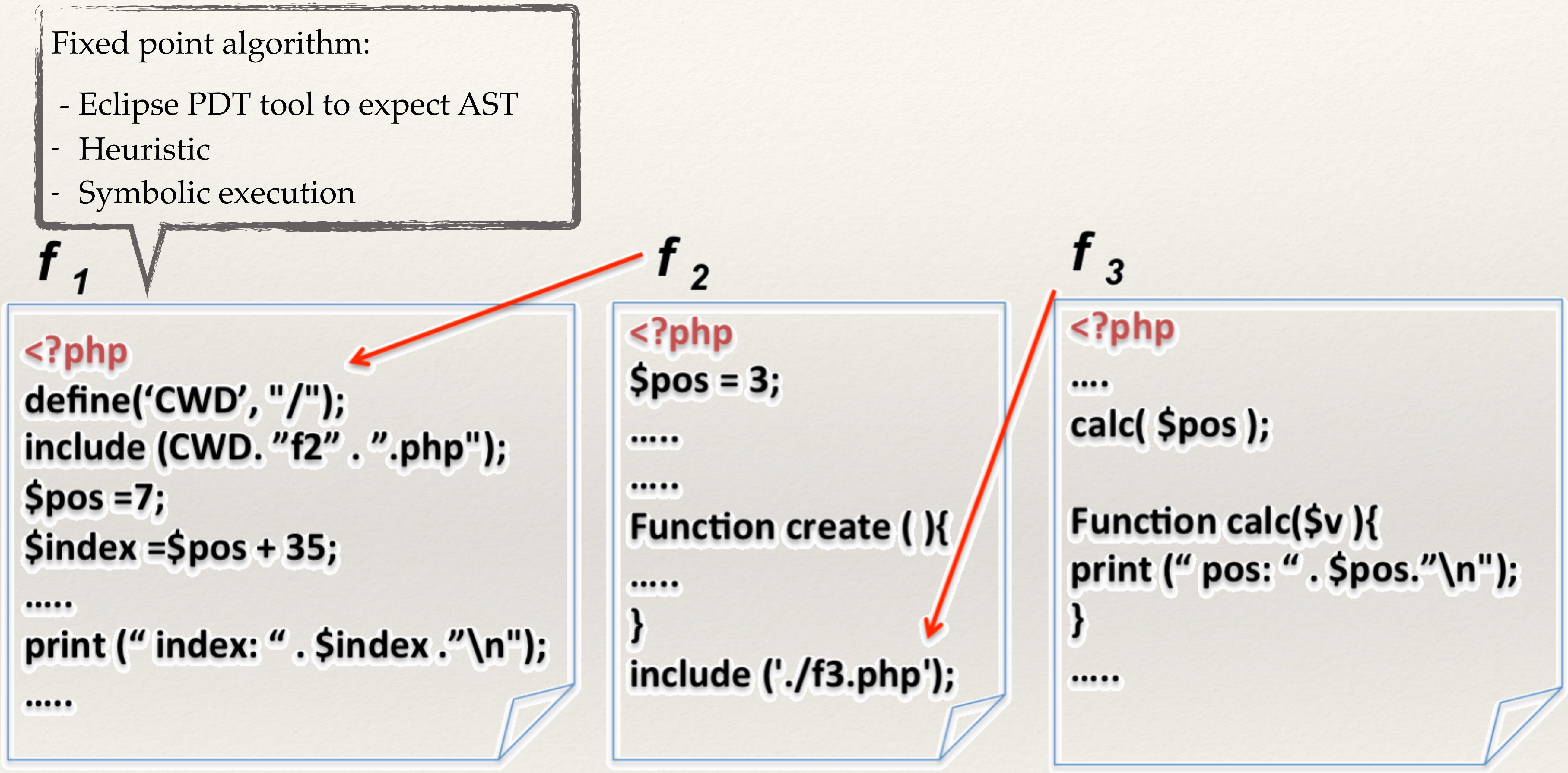
manual validation

Oracle
7242,265
Tomcat

Include Statements

- include(“./f1.php”)
- include_once (“./” . “f1.php”)
- require (PATH. “f1.php”)
- require_once (getRoot(). “f1.php”)

Include Resolution



Fixed point algorithm:

- Eclipse PDT tool to expect AST
- Heuristic
- Symbolic execution

f_1

```
<?php
define(‘CWD’, "/");
include (CWD. ”f2” . ”.php");
$pos =7;
$index =$pos + 35;
.....
print (“ index: “ . $index .”\n");
.....
```

f_2

```
<?php
$pos = 3;
.....
.....
Function create ( ){
.....
}
include ('./f3.php');
```

f_3

```
<?php
....
calc( $pos );

Function calc($v ){
print (“ pos: “ . $pos.”\n");
}
.....
```

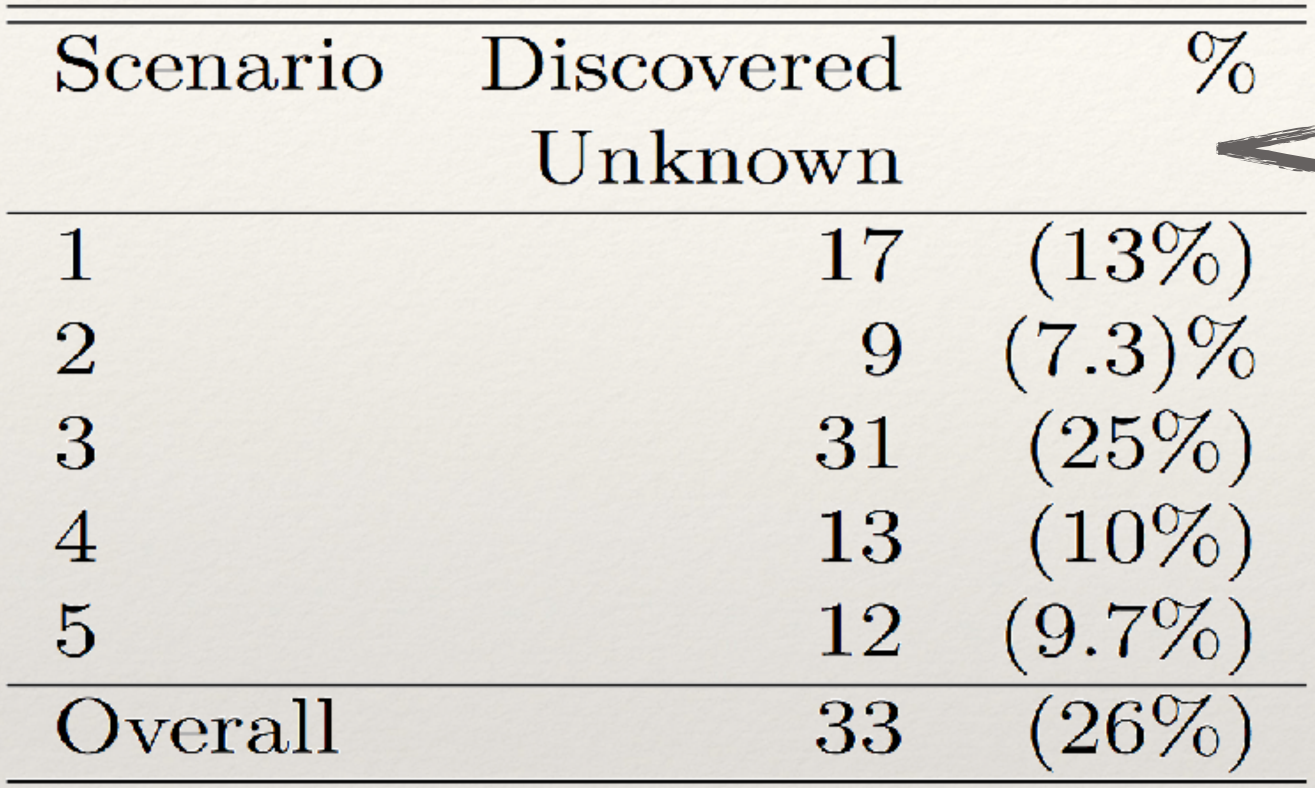
Experiment

Programs	Release	Includes statements	Unknown
Wordpress	3.6 – 3.7	629 - 649	37 - 37
Akismet	2.5.6 – 2.5.9	3 - 3	0 - 0
YARPP	3.5 - 4.4.1	17 - 26	16 - 23
Jetpack	2.7 - 2.3.5	95 - 126	37 - 63
NextGen Gallery	1.9.3 – 2.0.40	114 - 144	26 - 37
Contact Form 7	3.2 - 3.6	16 - 19	15 - 18
Google XML Sitemap	3.2.7 - 3.3.1	5 - 5	2 - 2
SEO by YOAST	1.1.7 - 1.4.22	22 - 42	18 - 35
W3 Total Cache	1.0.12 - 1.0.12	592 - 436	335 - 168
WP Sitemap Page	0.9.2.4 - 0.9.3	1 - 1	1 - 1
Google XML Sitemaps for qTranslate	3.2.7.1 - 3.3.1	6 - 6	2 - 2

Experiment

Programs	Includes statements	Unknown	Resolved
Wordpress	629 - 649	37 - 37	2 - 2
Akismet	3 - 3	0 - 0	0 - 0
YARPP	17 - 26	16 - 23	11 - 19
Jetpack	95 - 126	37 - 63	22 - 43
NextGen Gallery	114 - 144	26 - 37	14 - 15
Contact Form 7	16 - 19	15 - 18	11 - 13
Google XML Sitemap	5 - 5	2 - 2	2 - 2
SEO by YOAST	22 - 42	18 - 35	16 - 33
W3 Total Cache	592 - 436	335 - 168	290 - 135
WP Sitemap Page	1 - 1	1 - 1	1 - 1
Google XML Sitemaps for qTranslate	6 - 6	2 - 2	2 - 2

Limitation of Static Resolution

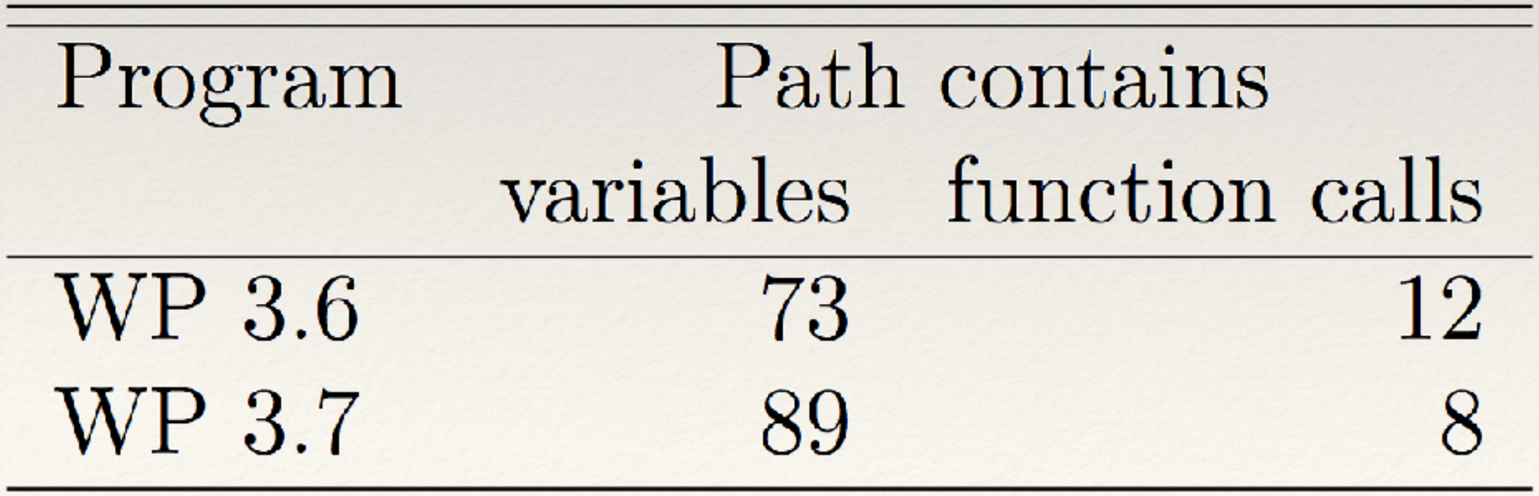


Scenario	Discovered Unknown	%
1	17	(13%)
2	9	(7.3)%
3	31	(25%)
4	13	(10%)
5	12	(9.7%)
Overall	33	(26%)

Dynamic analysis

-Use TXL to instrument the include statements

- Installed wordpress 3.6 with all 10 plugins
- Five simple scenarios
- Logged the actual files at run time



Program	Path contains	
	variables	function calls
WP 3.6	73	12
WP 3.7	89	8

Context and Motivation

Software lexicon:

- Identifiers
- Comments
- Literal

Importance of lexicon

- Program comprehension
- Traceability links
- Concept location

Lesson Learned

- Methods and parameters renamings are unavoidable due to evolution, i.e., constant changes in requirements.
- Using APIs without planning for change can cause ripple effect on the client lexicon.
- It is important to choose the naming conventions for each specific project in an early stage of the development process and following it consistently.
- It is worth taking the effort to identify the right order of terms constituting an identifier to clarify its meaning and avoid possible misunderstandings.
- To avoid the need for a sequence of renamings towards spelling error correction, it is worth taking the time to spellcheck the identifier name when creating or modifying an entity.

Lesson Learned

- It is worth investigating which one of the two, an abbreviation or its English alternative, is more common and thus should be used
- Identifiers that contain negation tend to be renamed towards positive names.
- The majority of semantic changes during renamings change, narrow, broaden, add, or remove a meaning to the identifier, as part of the evolution process and thus cannot be avoided.
- It is worth the effort to assure consistency between, on the one hand, the name of an entity, and, on the other hand, its functionality, type, or other entities.