EasyMerge

A NEW TOOL FOR CODE CLONES REFACTORING

H. QIN, S. PAN, Y. CHEN

Background

- A code fragment (CF) is any sequence of code lines (with or without comments). A CF is identified by its file name and begin-end line numbers in the original code base.
- A code fragment CF_2 is a clone of another code fragment CF_1 if they are similar by some given definition of similarity, that is, $f(CF_1) = f(CF_2)$ where f is the similarity function.
- Previous research shows that a significant fraction (between 7% and 23%) of the code in a typical software system has been cloned.
- By detecting, categorizing, and removing code clones we can produce easier to understand, cleaner, and more reusable code.

Clone Types

- Type-1: Identical code fragments except for variations in whitespace, layout and comments.
- Type-2: Syntactically identical fragments except for variations in identifiers, literals, types, whitespace, layout and comments.
- Type-3: Copied fragments with further modifications such as changed, added or removed statements, in addition to variations in identifiers, literals, types, whitespace and comments.
- Type-4: Two or more code fragments that perform the same computation but are implemented by different syntactic variants.

Clone Detection

Approach	Tool/author	P Platform	D External Dependencies	A Availability	U User Interface	O Output	I IDE Support	LP Language Paradigm	LS Language Support	R Clone Relation	G Clone Granularity	CT Clone Types	CA Comparison Algorithms	CU Comparison Granularity	CC Computational Complexity	PP Pre-/Post-Processing	H Heuristics/Thresholds	T Transformations (Table 8)	CR Code Representation (Table 9)	PA Program Analysis (Table 10)	E Empirical Validation	AR Availability of Results	S Subject Systems
Text-based	Johnson [55,54,53]	e	a	f	d	c	c	a	b	b	a	a	i	b	d	c	ad	cd	bd	c	d	b	g
	Duploc [41]	e	a	b	d	c	c	c	bcef	ad	a	ac	e	a	b	a	cd	b	bf	c	bc	b	g
	sif* [85]	e	a	f	d	a	c	a	b	b	f	ac	i	b	d	b	c	a	d	a	d	b	g
	DuDe [114]	e	a	e	a	c	c	c	be	a	a	ac	d	a	d	a	abc	ab	b	b	d	b	bcd
	SDD [78]	a	a	a	b	b	a	c	abe	a	a	ac	o	b	a	a	ac	a	b	c	d	b	be
	Marcus* [86]	e	b	f	d	a	c	a	b	b	a	acde	g	c	d	c	b	adf	h	c	b	b	g
	NICAD [99,104]	b	a	e	a	c	c	c	bde	b	bcd	abcd	o	a	b	a	bc	cghi	n	g	bc	a	abcde
	Nasehi [92]	e	b	f	b	b	c	b	e	a	b	abcd	o	k	d	a	b	h	p	g	d	b	g
	Simian [107]	d	a	b	a	a	c	cd	a	b	a	ab	q	a	d	c	e	e	f	a	d	b	b
Token-based	Dup [8]	b	a	e	d	c	c	c	be	a	a	ab	a	ad	a	b	ab	b	g	d	bc	b	bcd
	CCFinder(X) [59,58]	d	a	b	c	c	c	c	bcef	ad	a	abc	a	d	a	b	abd	eh	f	dg	bc	b	abf
	RTF [12]	c	a	e	d	a	c	c	be	b	a	ab	b	d	a	c	abd	bg	f	d	d	b	a
	CP-Miner [84]	e	b	e	d	d	c	c	bc	b	g	abc	f	e	d	b	cd	e	n	e	b	b	adef
	SHINOBI* [115]	c	a	f	b	b	b	c	cdi	b	a	ab	b	d	a	b	ab	e	d	l	d	b	g
	CPD [29]	a	b	a	c	c	c	c	bce	a	a	ab	q	d	d	c	c	g	p	d	d	b	b
	Clone Detective [37]	c	b	ab	b	b	b	b	ad	b	a	ab	q	d	a	c	a	e	f	d	d	b	g
	clones [14,72]	d	b	d	c	c	c	c	bdefi	ad	a	ab	a	ad	a	c	a	ah	fg	d	b	b	abcde
Tree-based	CloneDr [15]	c	b	cd	c	c	c	c	bcef	ad	a	abc	hl	f	b	c	b	i	j	e	bc	b	g
	Asta [42]	e	b	f	b	b	c	b	de	a	a	abc	q	f	d	c	bc	hi	j	e	b	b	g
	cdiff* [116]	e	a	f	d	b	c	a	b	e	f	abe	o	f	b	a	e	i	j	e	d	b	g
	cpdetector [72,43]	d	b	e	c	a	c	c	be	a	a	ab	a	d	a	b	ad	i	j	e	a	b	cd
	Deckard [52]	b	a	e	d	a	c	c	be	b	a	abc	h	f	b	b	bc	i	i	h	b	b	ab
	Tairas [111]	c	b	f	d	b	b	a	b	ad	b	ab	a	d	b	c	e	i	j	e	d	b	g
	CloneDetection [113]	c	b	f	d	d	c	b	ce	b	a	ab	f	e	b	c	c	i	l	e	d	b	b
	CloneDigger [20]	a	b	a	a	c	a	b	eg	a	a	abc	a	f	b	c	b	i	l	e	d	a	g
	C2D2 [74]	c	b	f	d	d	c	b	di	a	b	abc	n	d	d	c	ab	i	l	e	d	b	g
	Juillerat [57]	e	c	f	d	d	c	b	e	e	g	a	q	d	d	c	e	i	l	e	d	b	g
	SimScan [108]	d	b	b	c	c	ab	b	e	ad	a	abc	q	f	d	c	b	i	j	e	d	b	b
	ccdiml [16,14]	d	b	d	c	c	c	c	bcef	a	a	abc	l	f	b	c	b	i	j	e	b	b	g
Metrics-based	Kontogiannis [66]	e	c	f	d	d	c	a	b	a	bc	abcd	e	hi	b	c	b	i	i	e	b	b	g
	Mayrand [87]	e	b	f	d	d	c	a	b	a	b	abcd	q	i	b	c	b	d	i	e	bc	b	g
	Davey [31]	e	c	f	b	b	c	a	b	b	b	abcd	j	i	b	c	b	d	i	e	d	b	g
	Patenaude [93]	b	b	f	d	d	c	b	e	a	b	abcd	q	i	b	c	b	i	i	e	d	b	b
	Kontogiannis [67]	e	c	f	d	d	c	a	b	a	b	abcd	m	i	b	c	b	i	i	e	b	b	g
Graph-based	Duplix [75]	e	b	e	d	d	c	a	b	a	a	abcd	k	g	c	c	c	i	k	f	cd	b	g
	Komondoor [65]	c	b	e	d	d	c	a	b	ad	ab	abcd	k	g	c	b	e	i	k	f	d	b	g
	GPLAG* [81]	c	b	e	d	d	c	c	bce	a	b	abcd	k	g	c	c	bd	i	k	f	b	b	g
	Gabel [46]	c	b	e	d	d	c	c	bc	b	b	abcd	q	i	c	b	bcd	i	i	f	b	b	ad

	Citation	Scenario 1			Scenario 2				Scenario 3					Scenario 4			
		a	b	c	a	b	c	d	a	b	c	d	e	a	b	c	d
Text-based	Johnson [55,54]	●	●	◓	○	○	○	○	○	○	○	○	○	○	○	○	○
	Duploc [41]	●	●	○	○	○	○	○	◓	◓	○	○	◒	○	○	○	○
	sif [85]*	◒	◒	◒	○	○	⊙	⊙	⊙	⊙	○	○	○	⊙	⊙	⊙	○
	DuDe [114]	●	●	⊙	○	○	○	○	◓	◓	◒	◒	◒	○	○	○	○
	SDD [78]	◓	◓	◒	○	○	○	○	◓	◓	⊖	⊖	⊖	○	○	○	○
	Marcus [86]*	◒	○	◒	○	○	◒	⊖	◒	◒	○	○	○	◒	◒	◒	○
	Basic NICAD [99]	●	●	●	⊙	⊙	⊙	⊙	◓	◓	◓	◓	◓	⊙	⊙	⊙	○
	Full NICAD [104]	●	●	●	●	◓	◓	◓	◓	◓	◓	◓	◓	⊙	⊙	⊙	○
	Nasehi [92]	◒	◒	◒	◒	◒	◒	◒	◒	◒	⊖	⊖	⊖	○	○	○	◒
	Simian [107]	●	●	○	◒	◒	◒	○	○	○	○	○	○	○	○	○	○
Token-based	Dup [8]	●	●	○	●	○	◓	○	○	○	○	○	○	○	○	○	○
	CCFinder(X) [59,58]	◓	◓	◓	◓	◓	◓	○	⊙	⊙	○	○	○	○	○	○	○
	Gemini [112]*	◓	◓	◓	◓	◒	◓	○	◒	◒	◒	◒	◒	⊙	⊙	⊙	○
	RTF [12]	●	●	●	◓	◓	◓	○	○	○	○	○	○	○	○	○	○
	CP-Miner [84]	◓	◓	◓	◓	◓	◓	⊖	◓	◓	◒	◒	◒	⊙	⊙	⊙	○
	SHINOBI [115]*	◓	◓	◓	◓	◓	◓	○	⊙	⊙	○	○	○	○	○	○	○
	CPD [29]	●	○	●	◒	◒	◒	○	○	○	○	○	○	○	○	○	○
	Clone Detective [37]	◓	◓	◓	◓	◓	◓	○	○	○	○	○	○	○	○	○	○
	clones/iClones [14,72]	●	●	◓	●	◓	◓	○	○	○	○	○	○	○	○	○	○
Tree-based	CloneDr [15]	●	●	●	●	●	●	⊖	◒	◒	⊖	◒	◒	⊙	○	○	○
	Asta [42]	●	●	●	◒	◒	◒	◒	◒	◒	○	○	○	○	○	○	○
	cpdetector/clast [72,14]	●	●	●	◓	◓	◓	○	○	○	○	○	○	○	○	○	○
	Deckard [52]	◓	◓	◓	◓	◓	◓	○	◒	◒	◒	◒	◒	○	○	○	○
	Tairas [111]	◓	◓	◓	○	○	◒	○	○	○	○	○	○	○	○	○	○
	CloneDetection [113]	◓	◓	◓	◓	◓	◓	○	○	○	○	○	○	○	○	○	○
	CloneDigger [20]	●	●	●	◓	◓	◓	◓	◓	◓	○	○	○	○	○	○	○
	C2D2 [74]	◒	◒	◒	◒	◒	◒	◒	◒	◒	○	○	○	○	○	○	○
	Juillerat [57]	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○
	SimScan [108]	●	●	●	◓	◓	◓	○	⊙	⊙	⊙	⊙	⊙	○	○	○	○
	ccdiml [16,14]	●	●	●	●	●	●	○	◒	◒	⊖	●	◒	○	○	○	○
Metrics-based	Kontogiannis [66]	◓	◓	◓	◓	◓	◓	⊖	⊖	⊖	⊖	⊖	⊖	◒	◒	◒	⊖
	Mayrand [87]	◓	○	◓	◓	◓	◓	◒	◒	◒	⊖	⊖	⊖	◒	◒	⊖	○
	Dagenais [30]*	◒	◒	◒	◒	◒	◒	◒	◒	◒	⊙	⊙	⊙	◒	◒	⊙	○
	Merlo [89,90]	◓	◓	◓	◓	◓	◓	⊖	◒	◒	⊖	⊖	⊖	◒	◒	⊖	○
	Davey [31]	◓	◓	◓	◓	◓	◓	⊖	◒	◒	⊙	⊙	⊙	◒	⊖	⊖	○
	Patenaude [93]	◓	◓	◓	◓	◓	◓	⊖	⊖	⊖	⊖	⊖	⊖	⊙	⊙	⊙	○
	Kontogiannis [67]	◓	◓	◓	◓	◓	◓	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖
	Antoniol [1,2]	◓	◓	◓	◓	◓	◓	◒	◒	◒	⊖	⊖	⊖	◒	◒	⊖	○
Graph-based	Duplix [75]	◓	◓	◓	◓	◓	◓	◒	◒	◒	◒	◒	◒	◓	◒	⊖	○
	Komondoor [65]	◓	◓	◓	◓	◓	◓	⊖	⊖	⊖	⊙	⊙	⊙	◓	◓	⊖	○
	GPLAG [81]*	◓	◓	◓	◓	◓	◓	⊖	⊖	⊖	◒	◒	◒	◓	◓	⊖	◒
	Gabel [46]	◓	◓	◓	◓	◓	◓	◓	◓	◓	◒	◒	◒	◒	◒	◒	○

● *very well* ◓ *well* ◒ *medium* ⊖ *low* ⊙ *probably can* ○ *probably cannot* ○ *cannot.*

CloneDigger – Anti-Unification

- Anti-unification: let E_1 and E_2 be two terms. Term E is a generalization of E_1 and E_2 if there exist two substitutions σ_1 and σ_2 such that $\sigma_1(E) = E_1$ and $\sigma_2(E) = E_2$. The most specific generalization of E_1 and E_2 is called anti-unifier. The process of finding an anti-unifier is called anti-unification.
- The anti-unifier tree of two trees T_1 and T_2 is obtained by replacing some subtrees in T_1 and T_2 by special nodes, containing term placeholders which are marked with integers.
- Let U be the anti-unifier of two trees, the anti-unification distance between them is a sum of sizes of all substituting trees in their substitutions.

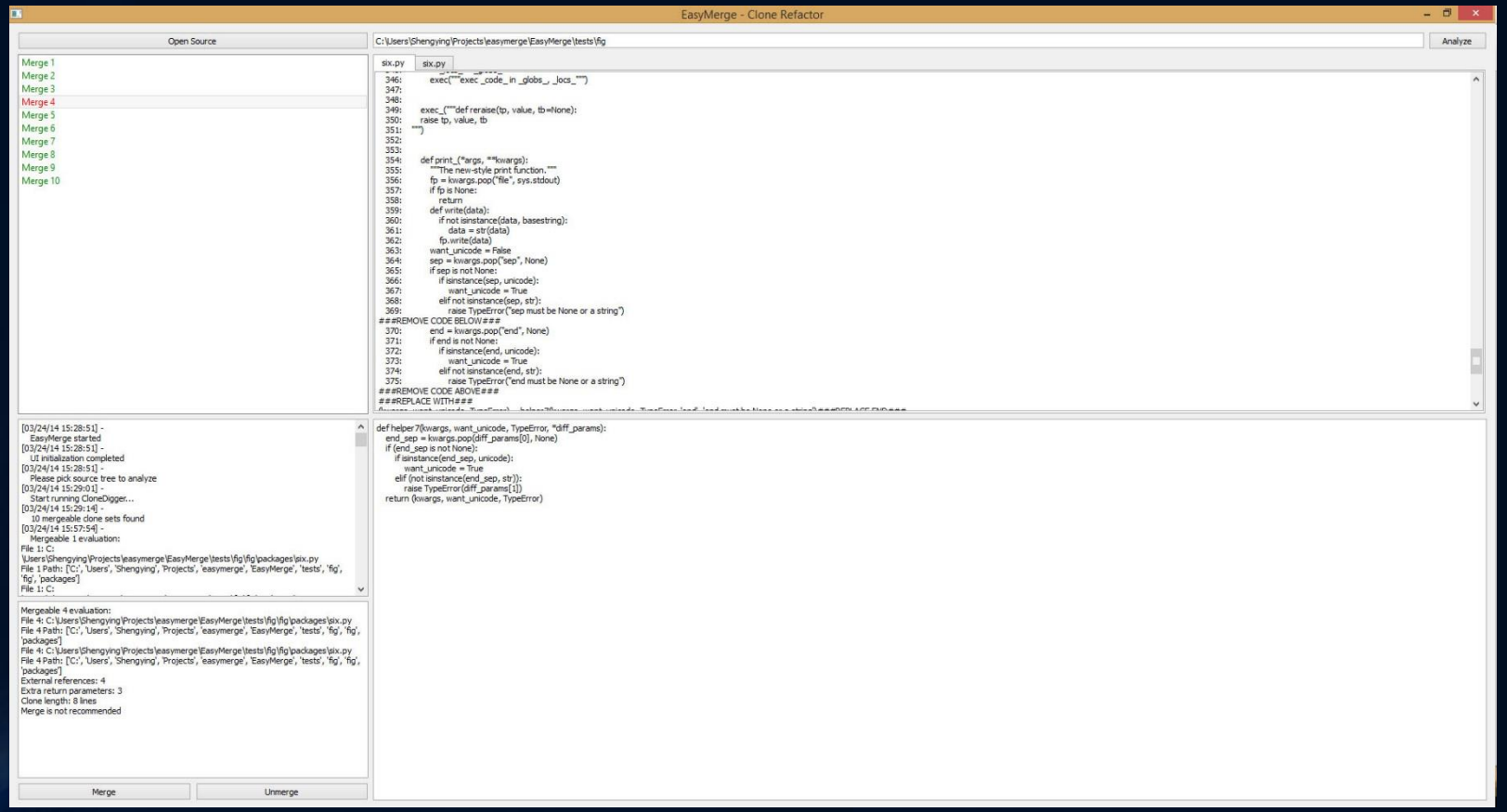
CloneDigger – Clone Detection Algorithm

- Build the AST for the program.
- Identify similar statements using anti-unification and partition them into clusters.
- Find identical sequences of cluster IDs. These are candidates to be reported as duplicate code fragments.
- Refine by examining the identified code sequences for overall similarity. In this phase, every pair of candidate sequences is checked for overall similarity at the statement level, again using anti-unification.

Clone Refactoring

- Extract Method
- Pull Up Method*
- Possible implications
  - External references such as variables, objects and function calls
  - Future references
- Related works: usage of refactoring guidance metrics

EasyMerge - UI



EasyMerge - Preprocessing

Get clone pairs from CloneDigger

- Threshold: Distance<=10, Size>=4, Distance<=Size

Filter which not appropriate to merge

- Class definitions, Import statements, Return statement

Merge clone pairs into clone sets

Remove overlapped clone sets

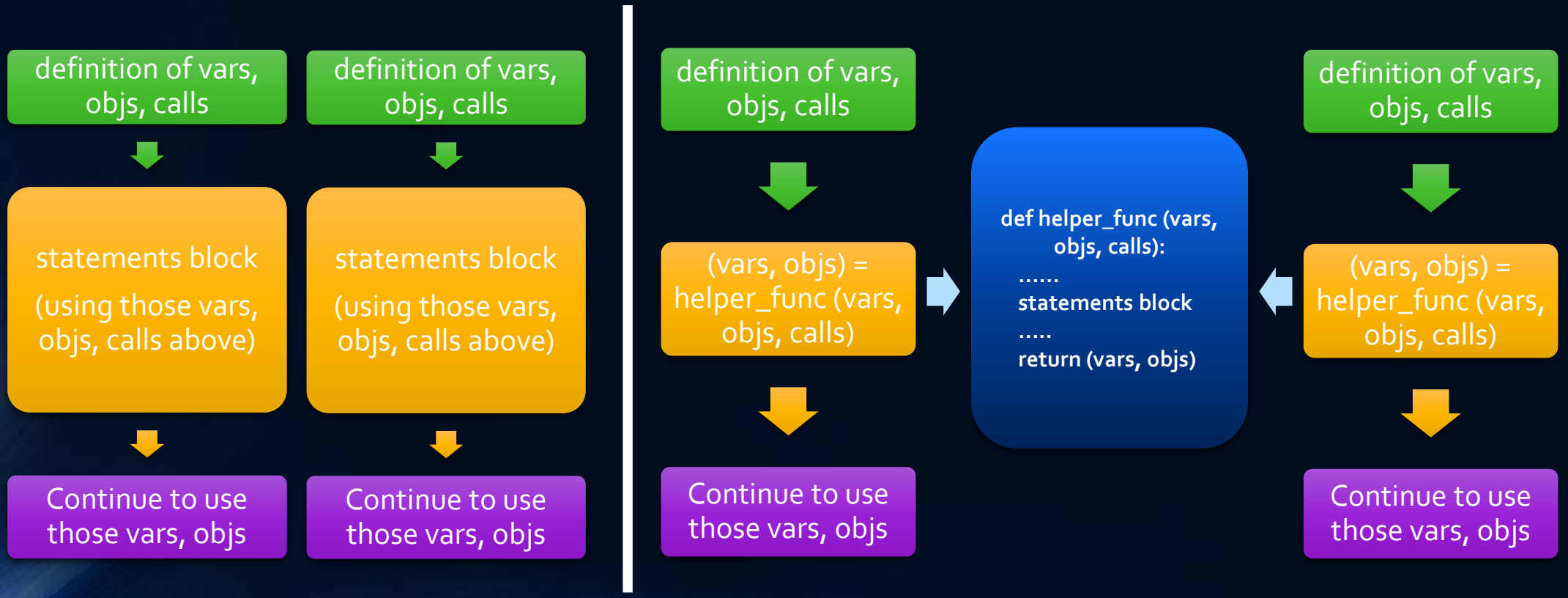
Separate Function Definitions with other Statements

EasyMerge - Classification

- Type1 clones vs. Type2/3/4 clones
- Clones in the same file vs. in different files
- Function definitions vs. Statements blocks

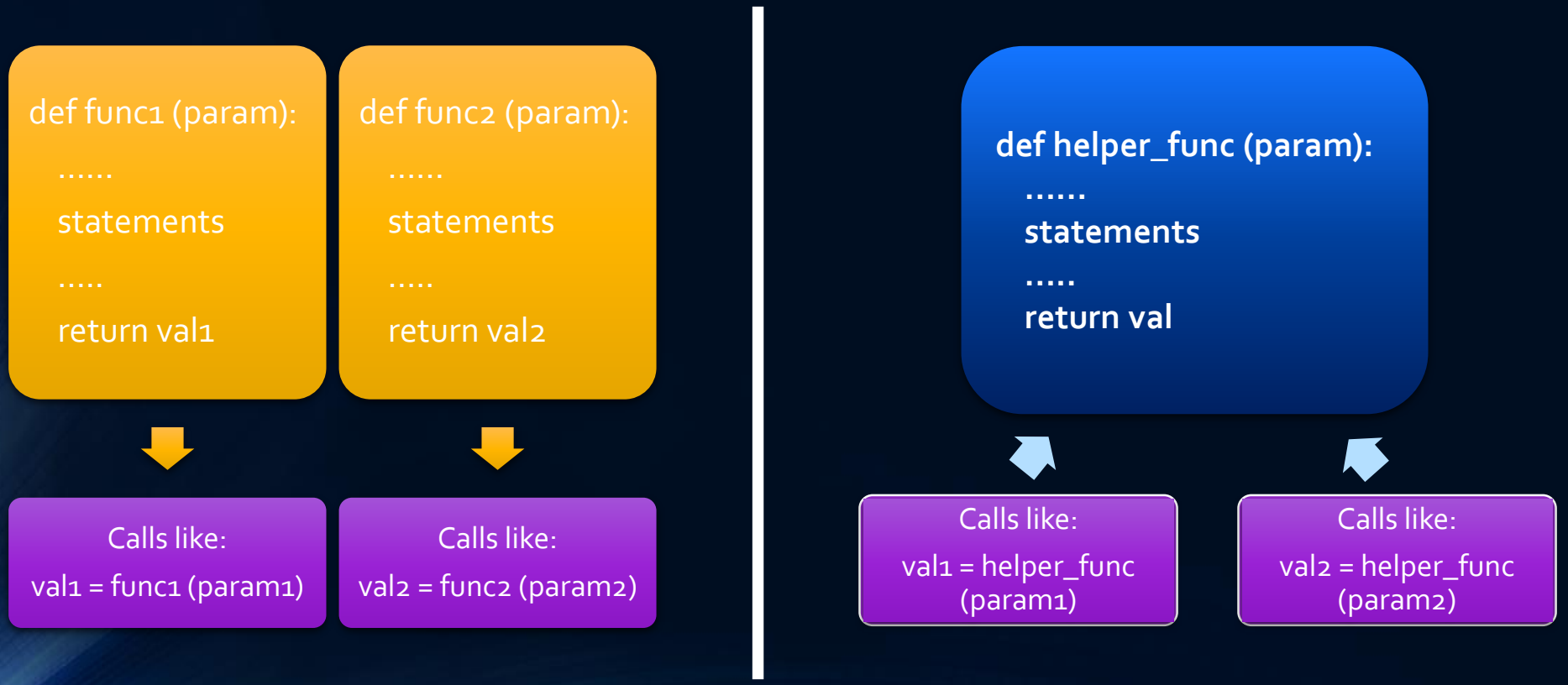
EasyMerge – Merging Objective

- Statements block



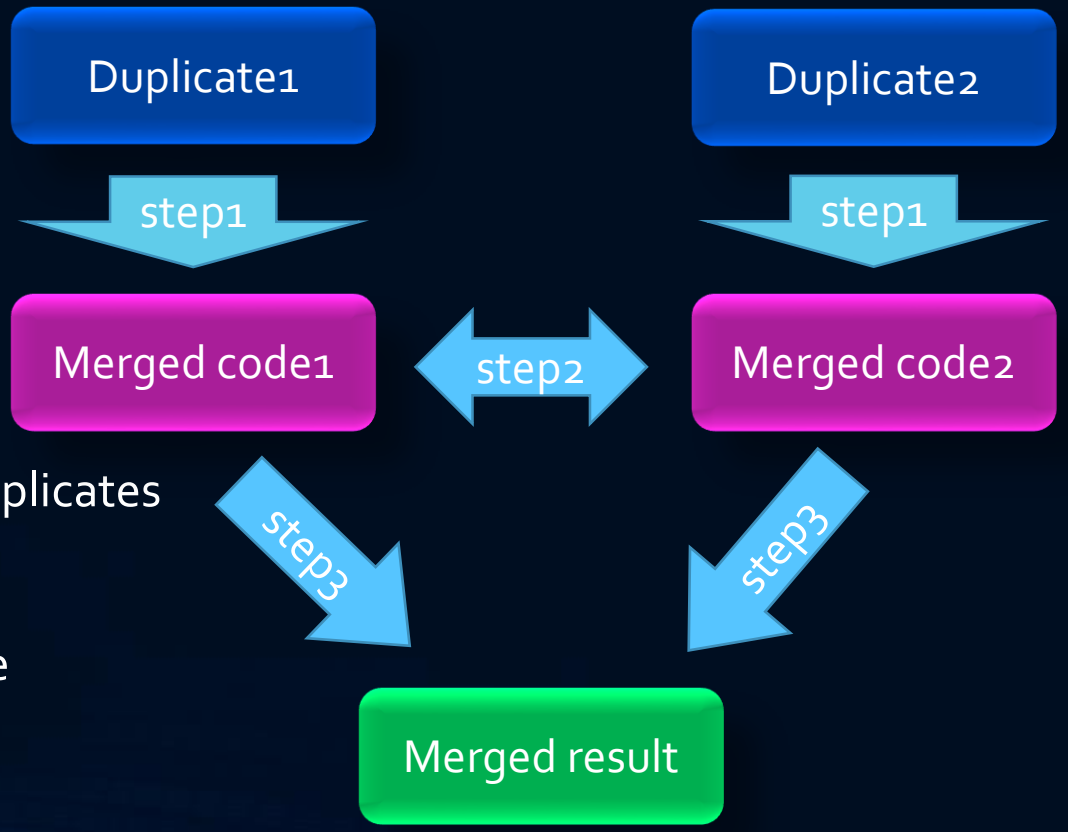
EasyMerge – Merging Objective

- Function definition



EasyMerge – Merging Procedure

- Preparation – get deeper information from AST
  - All the packages imported
  - All classes definitions and their scopes
  - All functions definitions and their scopes
  - All function-calls and the scopes they are in
- Merging Mechanism
  - Step1: Generate "merged codes" for each duplicates
  - Step2: Compare those "merged codes"
  - Step3: Merge those "merged codes" into one
  - Step4: Modify codes in each duplicates



EasyMerge – Merging Step1

Scan the clone code, find the variables, objects, function calls which are defined externally

Add those external variables, objects, function calls to the helper function's parameter list

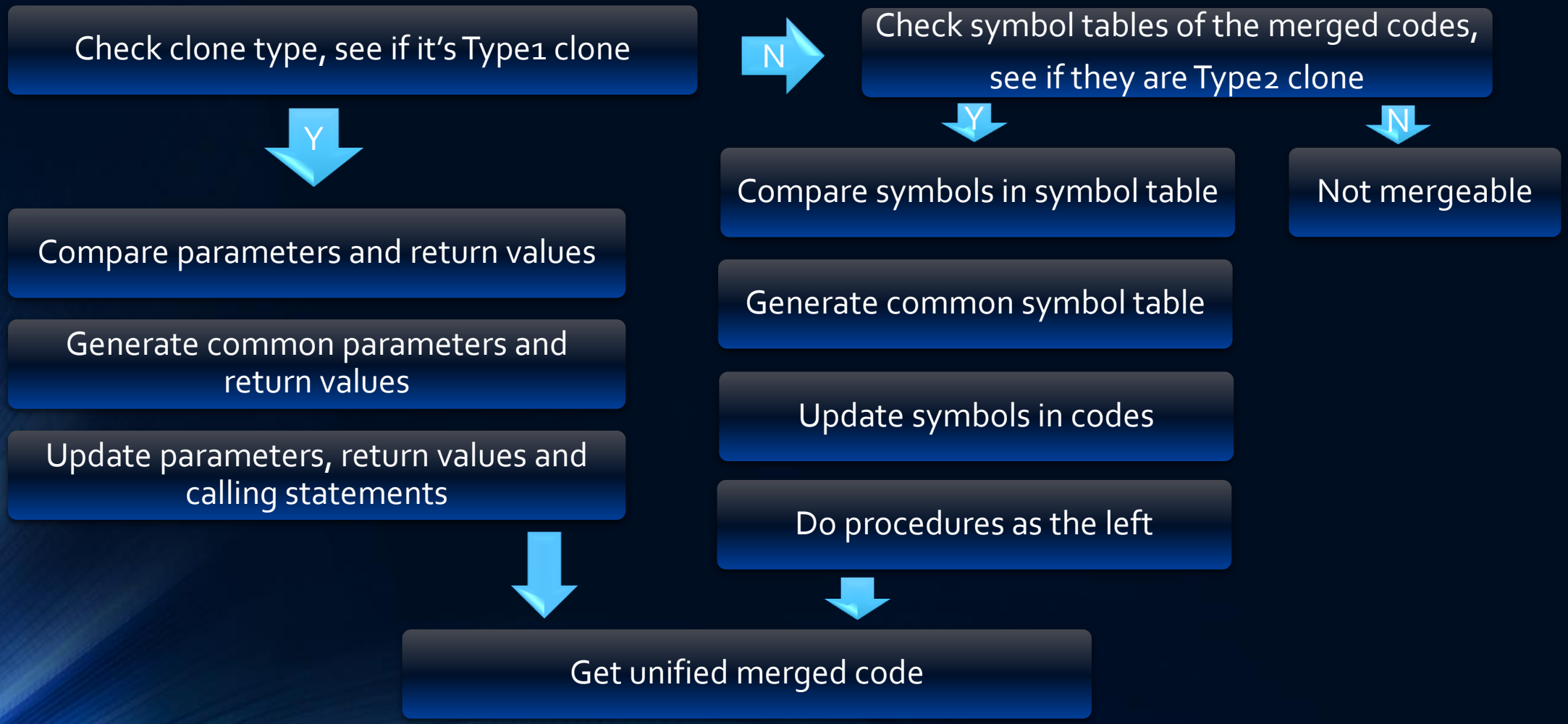
Scan the source file, find the variables, objects which are used after the clone code

Add those variables, objects to the return values of the helper function

Construct the helper function with signature line, body block and return statement

Generate new calling statements for each duplicates

EasyMerge – Merging Step2&3



EasyMerge – Recommendation

- What we can merge but we shouldn't merge
  - The duplicates are in different packages or libraries
  - Too many external references or return values
- Recommendation evaluation formula
  - $f(RE) = w_1 * L - w_2 * \mathrm{E} - w_3 * R - w_4 * D$
  - L = code length
  - E = number of external references
  - R = number of return values
  - D = "package/lib distance" between duplicates
  - $w_1 = 0.5;\ \ w_2 = 1;\ \ w_3 = 2;\ \ w_4 = 2$
- Recommending threshold
  - $f(RE) \leq 0;\ \ 0 < f(RE) \leq 8;\ \ f(RE) > 8$

Evaluation

- Environment
  - AMD FX 3.5GHz 6 cores CPU, 16GB RAM
  - Ubuntu Desktop 12.04, Python 2.7.5
- Test data
  - 5 most famous Python projects in Github
    - Scrapy, Pymongo, Reddit, Redis Python Client, Tornado
  - 6 randomly selected Python projects in Github
    - PyStruct, Tweepy, YouTube-dl, Beets, Fig, Wagtail

Experimental Results - Preprocessing

Preprocessing	# Lines	# Clone Sets	Set Size	Duplicate Lines	Line Repitition
CloneDigger	12859.5	1713.3	2	15.01%	8.88
After Filtering	12859.5	520.3	2	11.79%	4.37
Clustering	12859.5	128.5	2.85	11.79%	1.63
Removed Overlap	12859.5	78.7	2.37	9.76%	1
Spread Mix Typed	12859.5	85.7	2.41	9.67%	1

Random Selected Projects

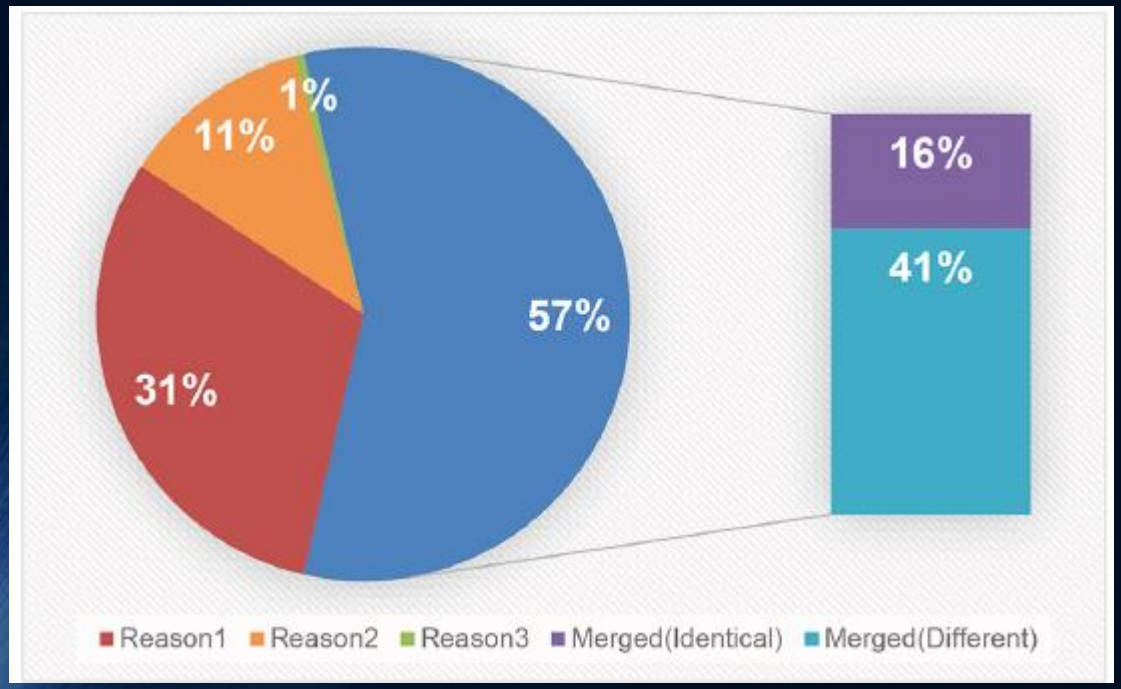
Preprocessing	# Lines	# Clone Sets	Set Size	Duplicate Lines	Line Repitition
CloneDigger	25807.2	1410.2	2	6.82%	7.69
After Filtering	25807.2	519.4	2	5.30%	3.61
Clustering	25807.2	146.2	2.56	5.30%	1.51
Removed Overlap	25807.2	91.4	2.19	4.32%	1
Spread Mix Typed	25807.2	111	2.18	4.26%	1

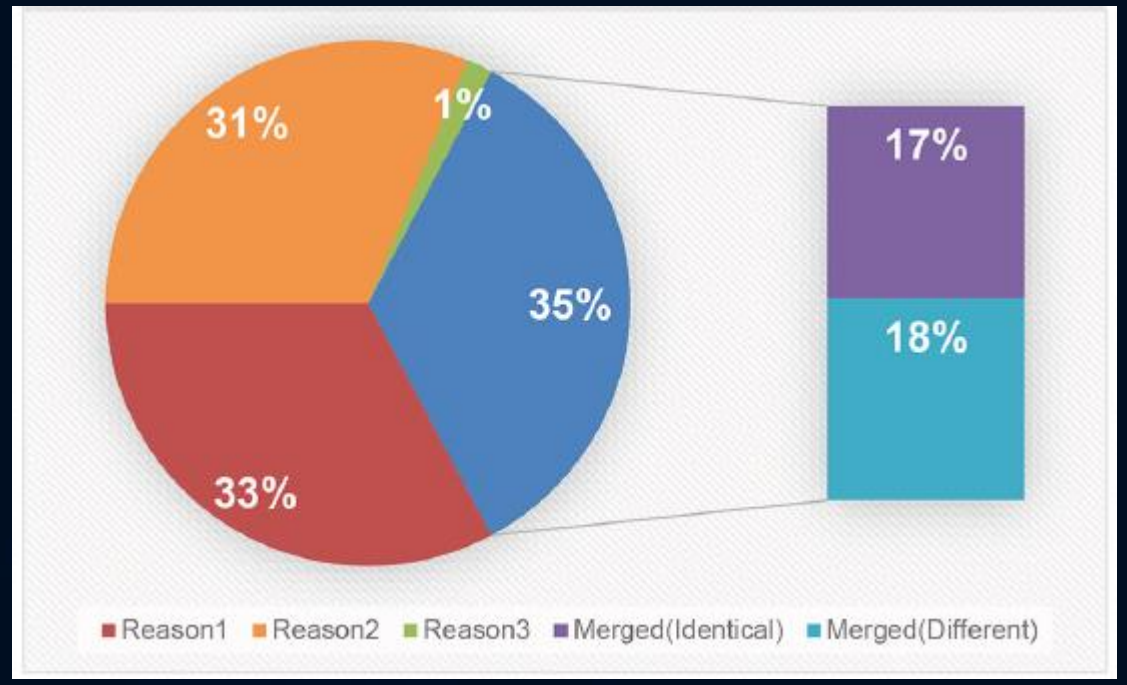
Famous Projects

Experimental Results - Classification

Classification	Random Projects (%)	Famous Projects (%)
Statements	70.0%	51.2%
Functions	30.0%	48.8%
Identical	16.9%	18.4%
Different	83.1%	81.6%
Same Files	56.4%	65.0%
Different Files	43.6%	35.0%

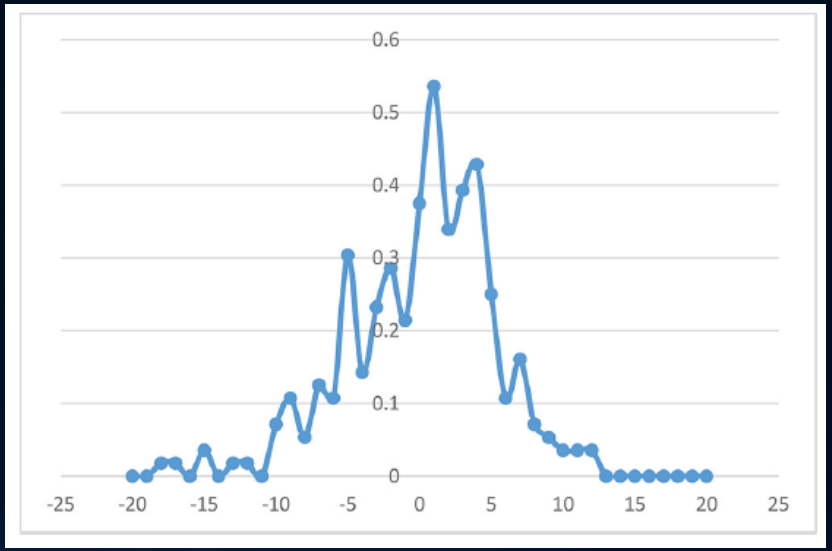
Experimental Results - Mergeability





Experimental Results – Merge results & Recommendation evaluation results

Merging	Random Projects	Famous Projects
Merged Lines %	3.51%	0.51%
Lines Reduction / Set	9.21	3.47
Distance / Set	5.55	2.92
External References	5.32	5.15



Conclusion & Future Work

- EasyMerge can merge most Type-1 clones and around half of Type-2 clones.
- A lot of seemingly simple scenarios contain subtle and tricky implications. We didn't have time to finish all features we planned.
- In the future, we can try to merge Type-3 clones. And to merge Type-4 clones, we need to replace CloneDigger with a better clone detection tool. And Type-4 clones will be really tricky to merge.
- Survey real developers to develop better recommendation formula.
- Add more intelligence to EasyMerge to handle test suites and independent libraries automatically.