School of
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

The Many Faces of Software Analytics

David Lo

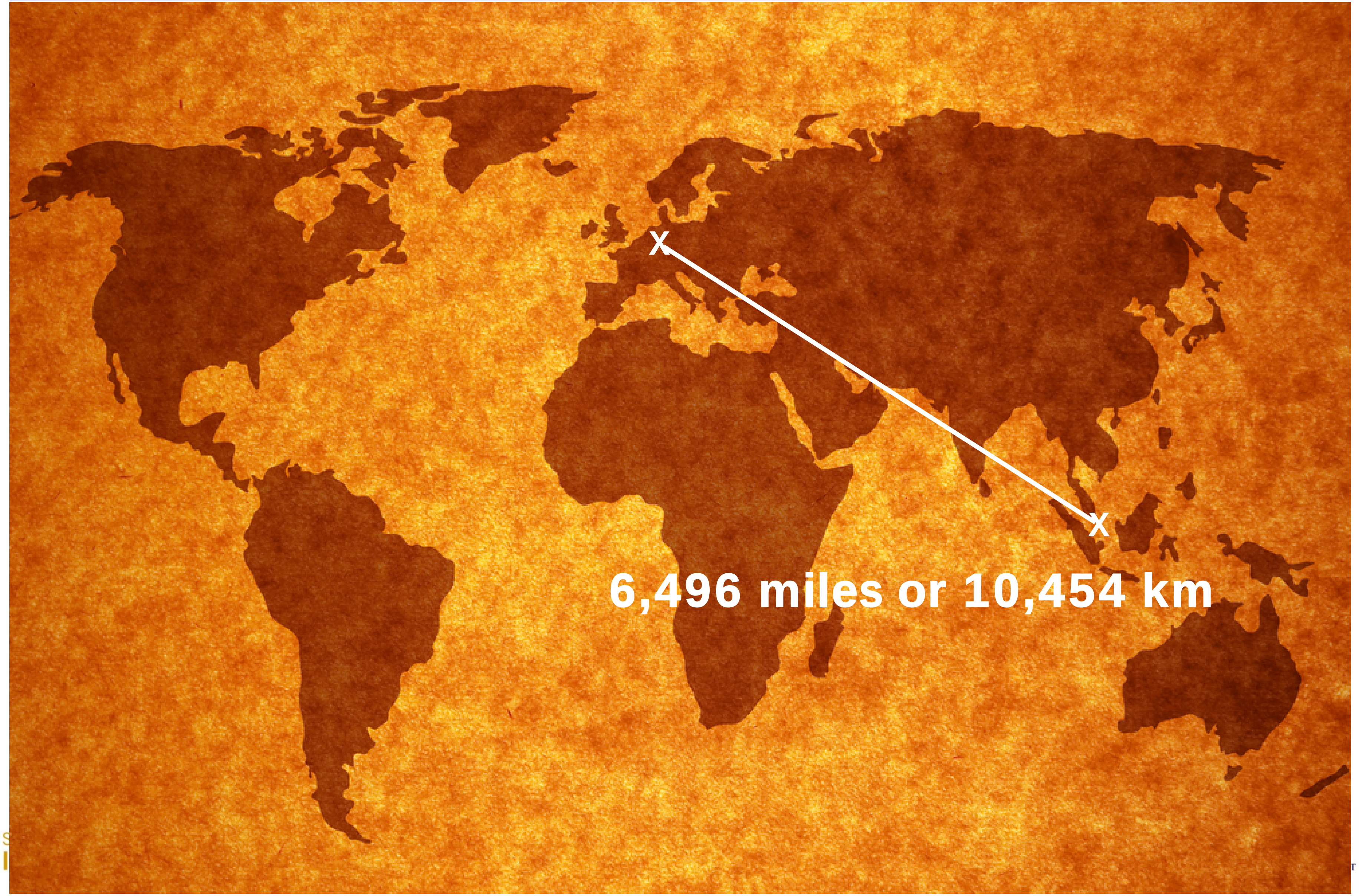
School of Information Systems

Singapore Management University

davidlo@smu.edu.sg

Talk at the University of Luxembourg, Dec 2014

A Brief Self-Introduction



A Brief Self-Introduction

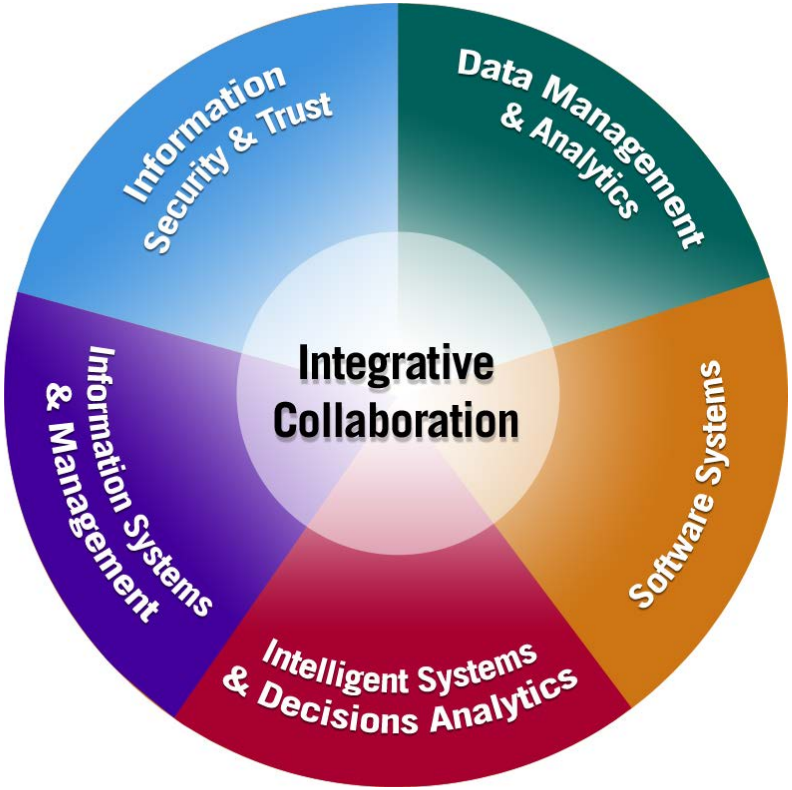
From Wikipedia

A Brief Self-Introduction

Singapore Management University

- Third university in Singapore
- Number of students:
  - 7000+ (UG)
  - 1000+ (PG)
- Schools:
  - Information Systems
  - Economics
  - Law
  - Business
  - Accountancy
  - Social Science

School of Information Systems



- Undergraduates: 1000+
- Master students: 100+
- Doctoral students: 50+

Our Research Group @ SMU

Our Research Group @ SMU

- 9 PhD Students
- 1 Visiting Professor
- 1 Research Engineer (Jan 2015)

Software Analytics

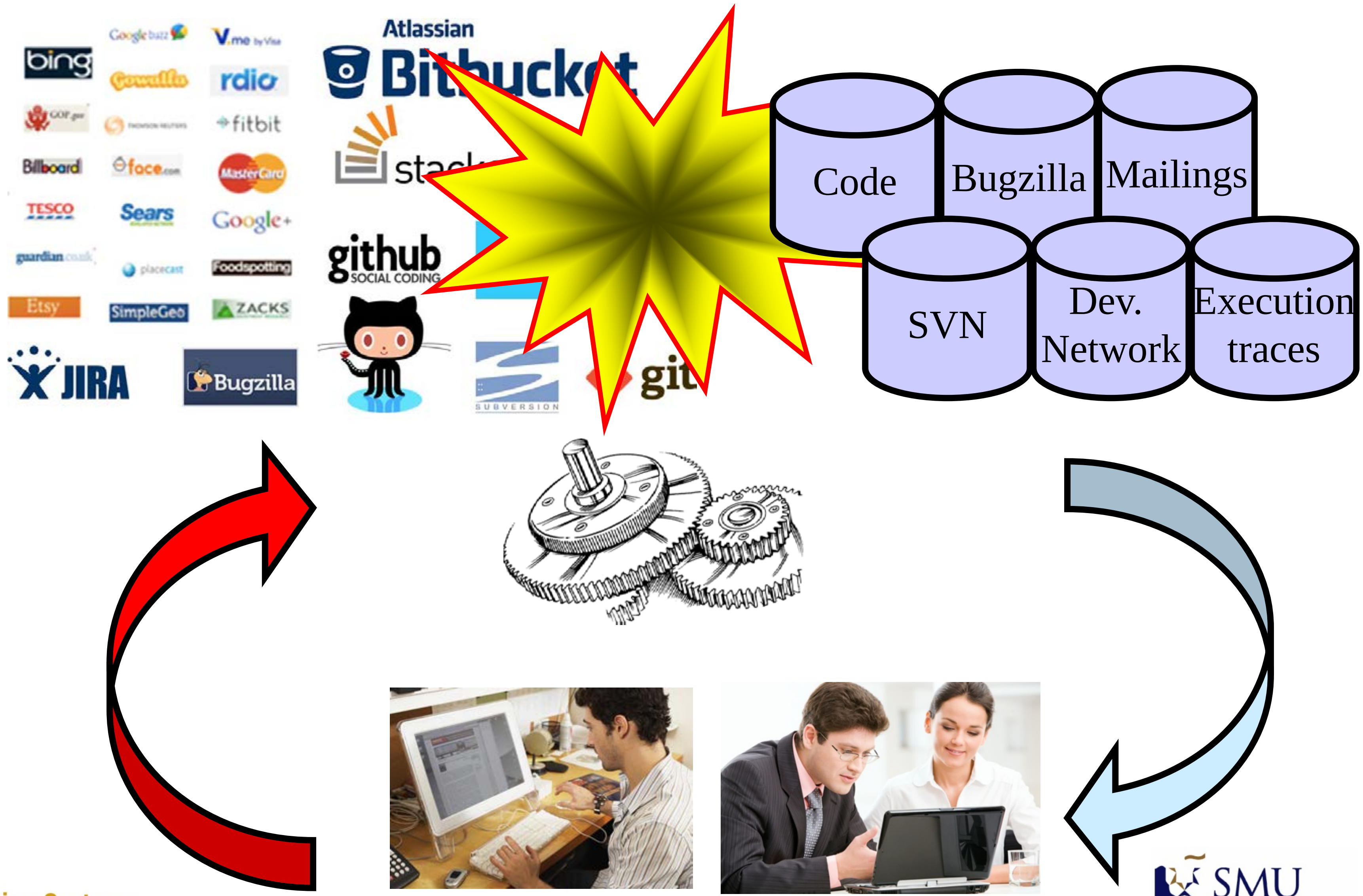
”Data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services”

(Zhang and Xie, 2012)

Software Analytics: Definition

- Analysis of a large amount of software data stored in various repositories in order to:
  - Understand software development process
  - Help improve software maintenance
  - Help improve software reliability
  - And more

Software Analytics



Research Directions: Software Analytics

Empirical Studies

Analytics for Coding & Collaboration

- Intelligent Multimodal Code Search
- Recommender for Libraries and Online Resources
- Coding and Collaboration With New Media

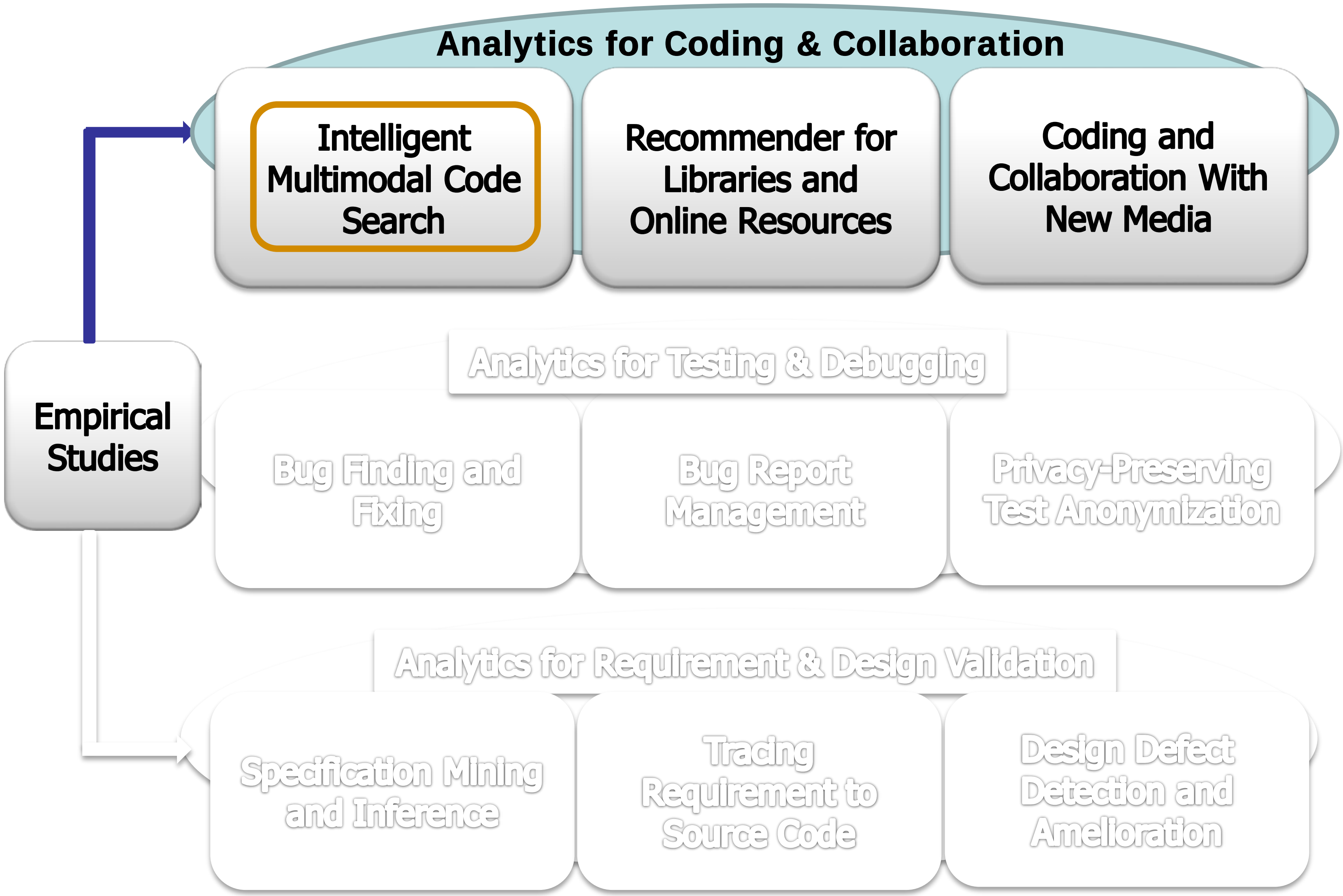
Analytics for Testing & Debugging

- Bug Finding and Fixing
- Bug Report Management
- Privacy-Preserving Test Anonymization

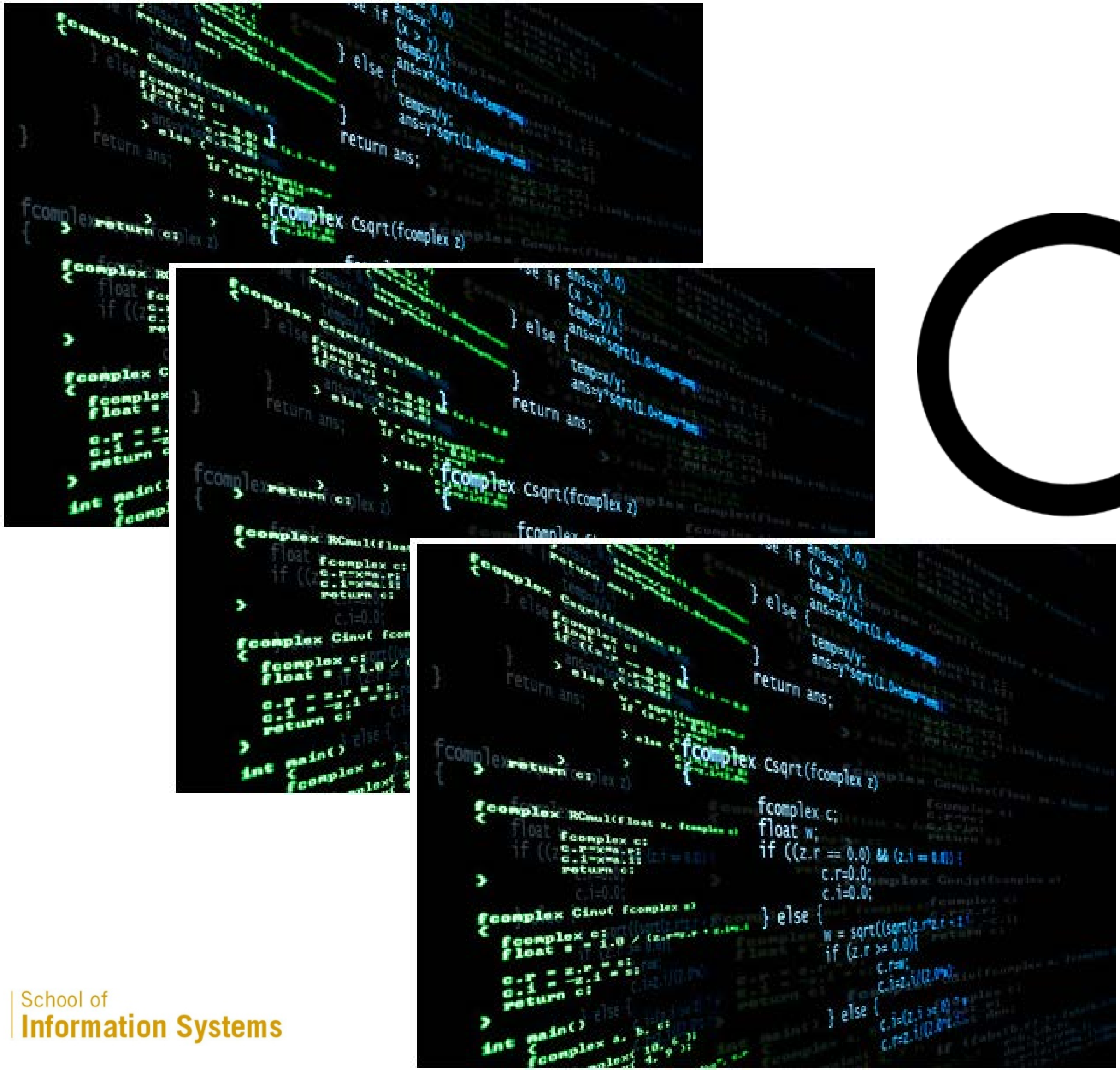
Analytics for Requirement & Design Validation

- Specification Mining and Inference
- Tracing Requirement to Source Code
- Design Defect Detection and Amelioration

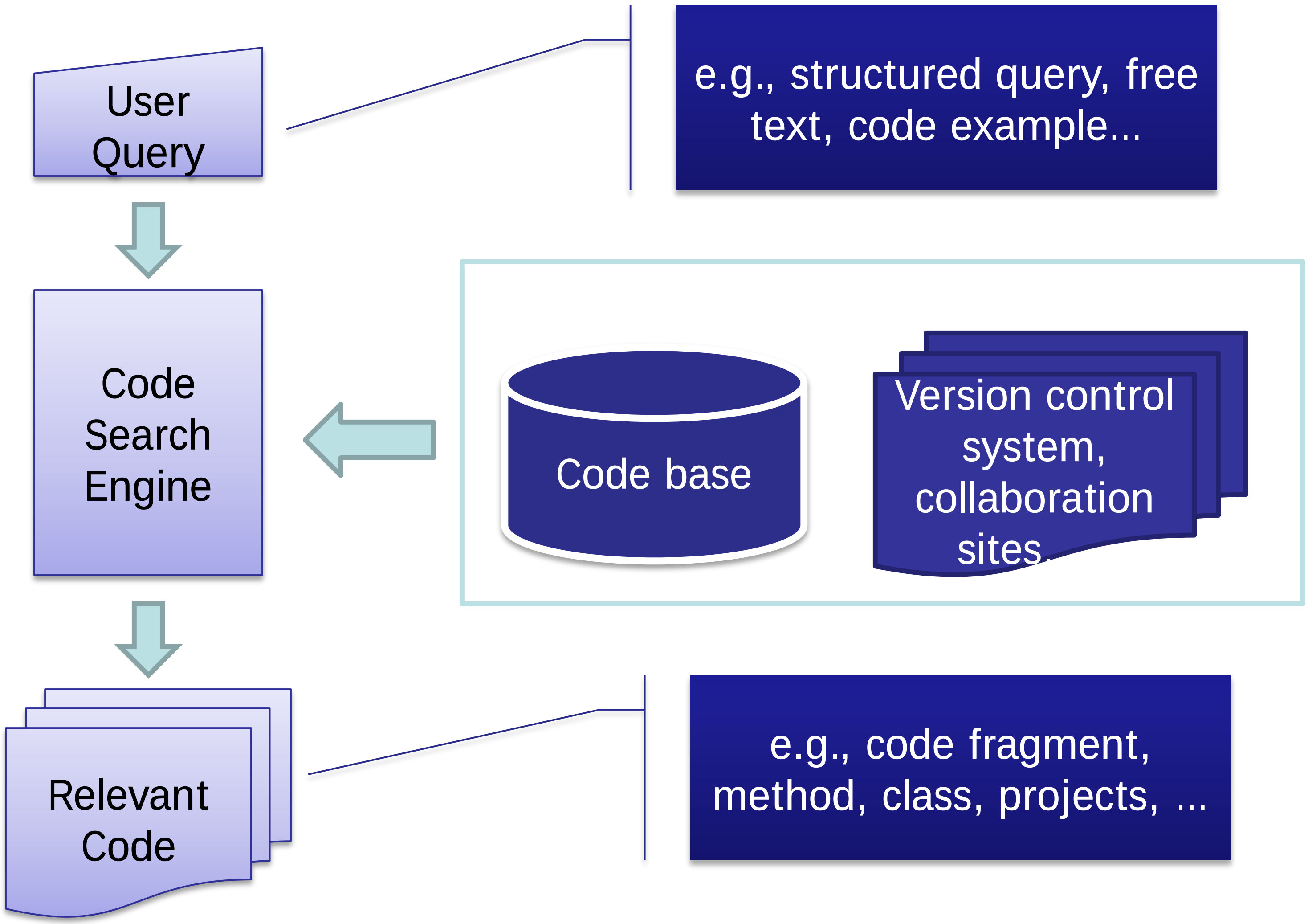
Our Past and Current Work



Intelligent Multi Modal Code Search



Intelligent Multi Modal Code Search



Intelligent Multimodal Code Search

Nodes: func A, func B, var C, var D;
Relations: C dataDepends A, D dataDepends B, D isFieldOf C;
Targets: D

Dependence Query Language

How do I load properties from an XML file?

Free Text

Code Search Engine

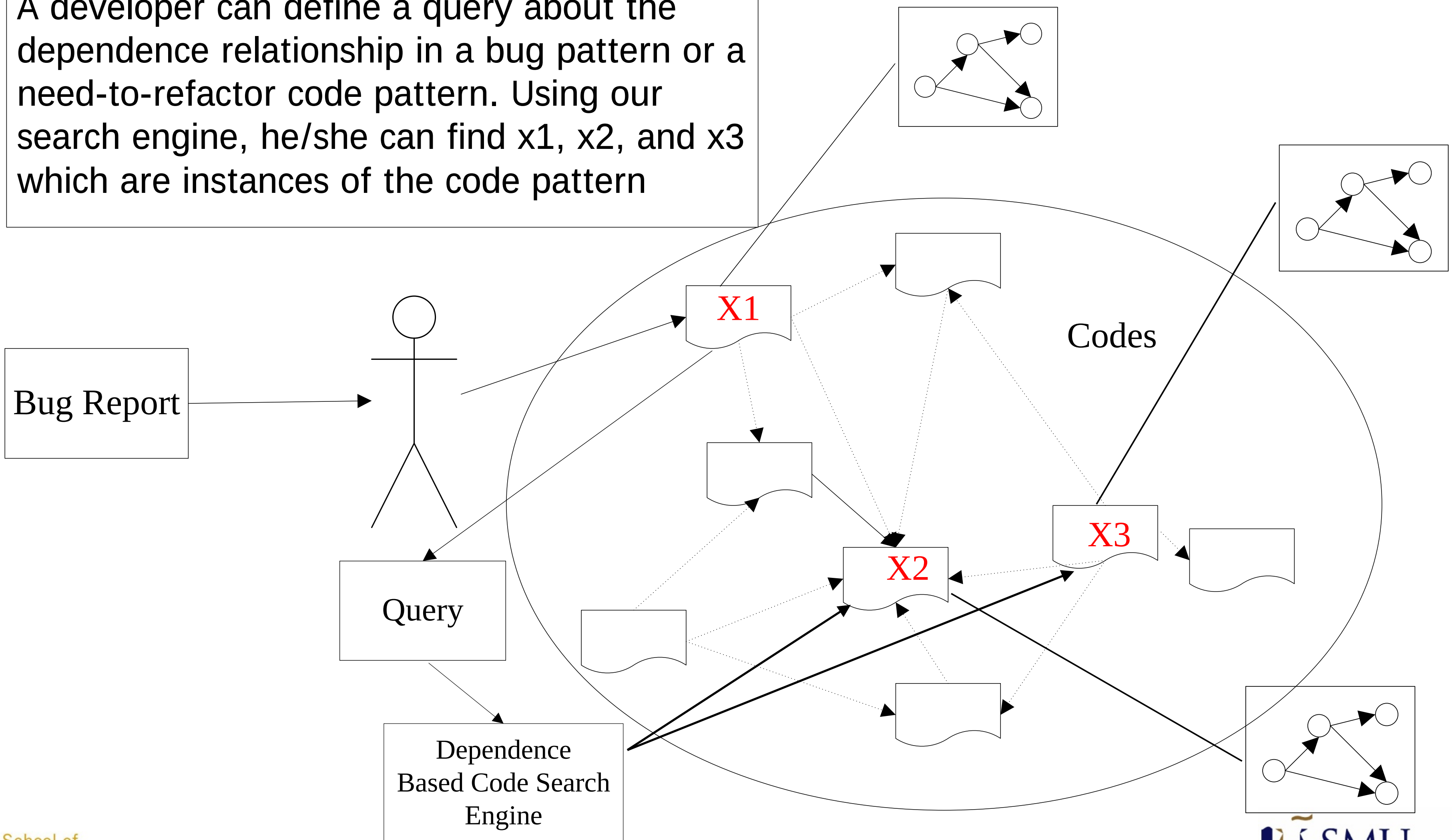
```
if ( tag -> check (a ,b ,c )){
tag ->a = a;
b = 0;
c = b ;
} else {
return a
}
```

```
if ( tag -> check (a ,b ,c )){
tag ->a = a;
b = 0;
c = b ;
} else {
return a
}
```

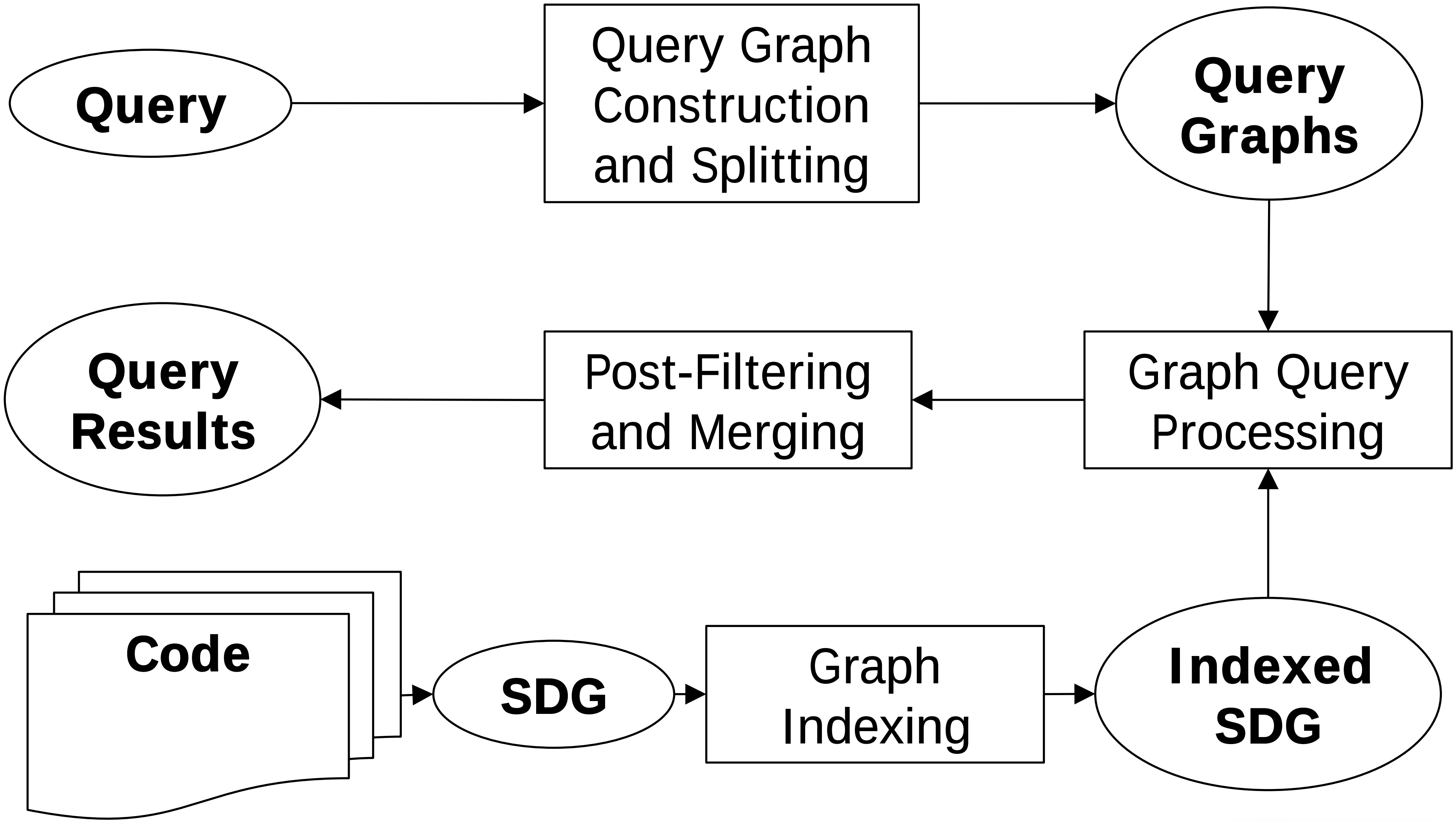
Code Examples

Structured Code Search (ASE10)

A developer can define a query about the dependence relationship in a bug pattern or a need-to-refactor code pattern. Using our search engine, he/she can find x1, x2, and x3 which are instances of the code pattern



Workflow of Our Approach



Dependence Query Language (DQL)

- Allows developers to describe a target
  - Involving several code elements
  - Including the dependencies between the elements

- Composed of 4 parts
  - Query identifier declarations [D]
  - Code element (node) constraints [N]
  - Relation constraints [R]
  - Desired target identifiers [T]

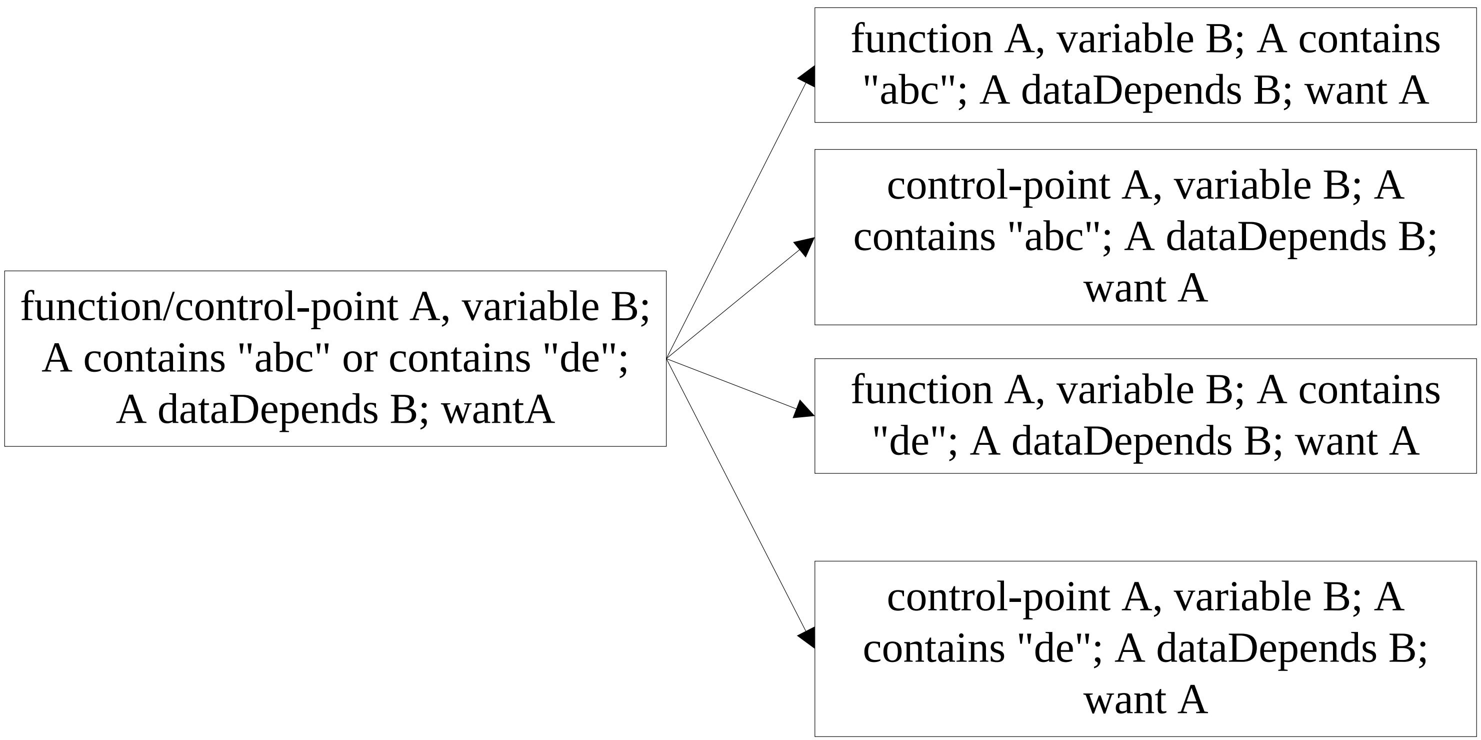
Dependence Query Language (DQL)

- Node Description [N]: Code element constraints
  - contains <Text>, inFile <FileName>, inFunction <FnName>, controlType <for/while/switch/if>, etc.

- Relation Description [R]: Relationship constraints
  - A *(transitively)* controls B, A calls B, A is data dependent on B
  - A is one step (*directly*) <depend-operation> on B
  - A textual contains B, etc.

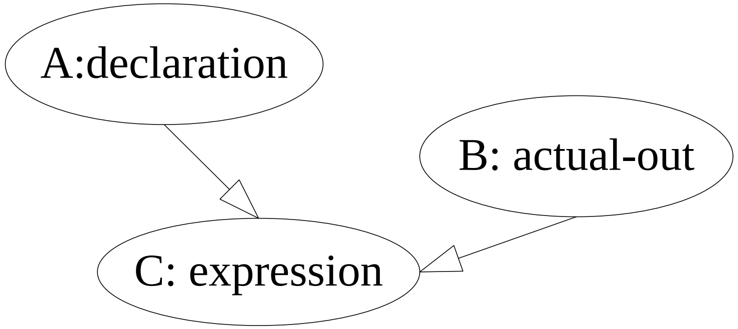
Query Splitting

- Split a query with disjunctions of conditions
- Result: Multiple queries with only conjunctions



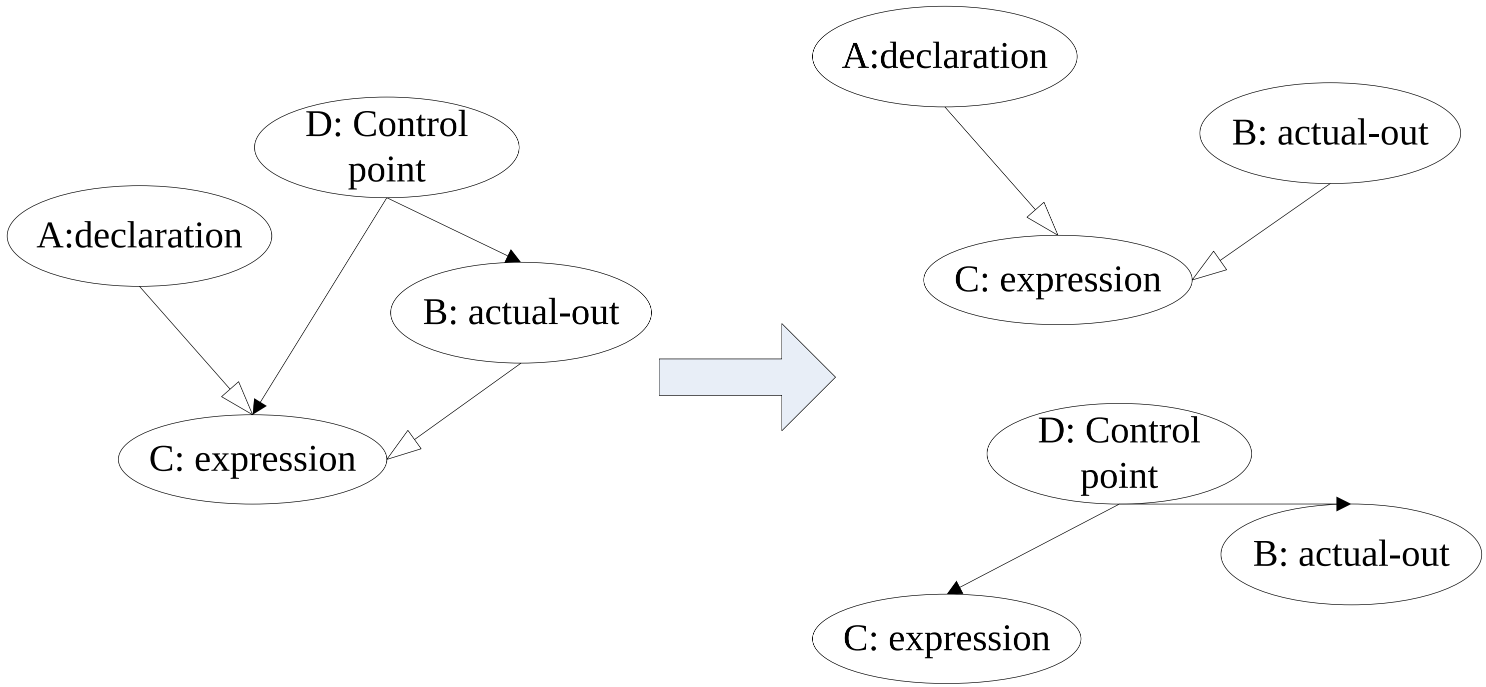
Query Graph Construction

- Query Declarations
  - Each identifier becomes a node in the query graph
- Relation Descriptions
  - Each dependence relation becomes an edge in the query graph



Query Graph Splitting

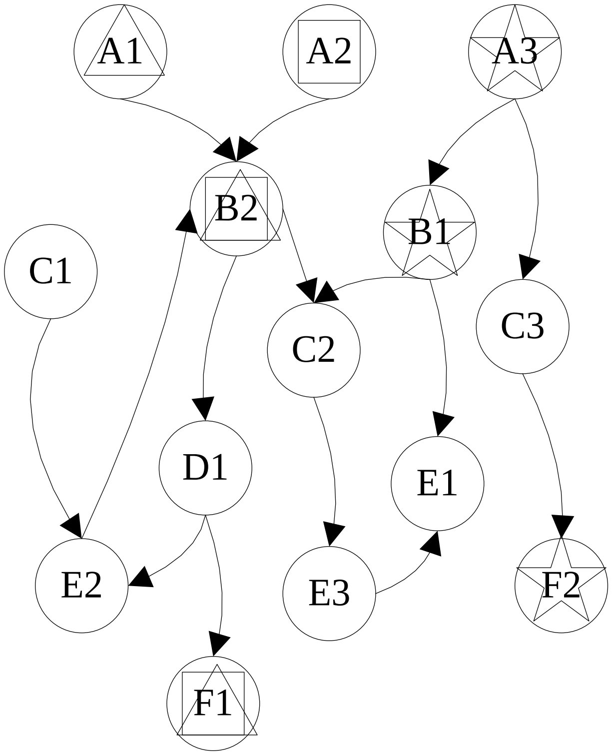
- Divide the query graph to two sub-graphs
  - Each only capture control OR data dependences



Graph Indexing and Query

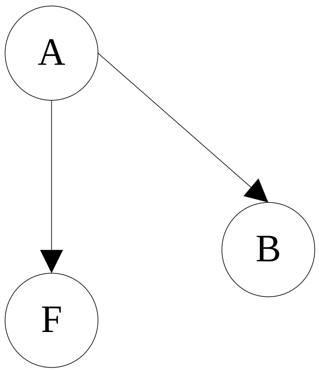
- Purpose:
  - Locate all instances of a given graph pattern in a large graph (Cheng et al., ICDE08)

Graph



(a)

Query



(b)

Three results found:
- triangle
- square
- star

Result Filtering & Merging

- Result Filtering
  - Textual conditions (e.g., textual contains)
  - Other relation descriptions

- Result Merging
  - Split 1: Disjunctions
  - Split 2: Data vs. Control Dependences
  - Need to union the sub-results

Evaluation

- Two open source projects
  - expat, gpsbabel

Project name	Description	Version	Size (LOC)
expat	XML handling library	2002-05-17	13
		2002-05-22	13
gpsbabel	GPS toolkit	2004-10-27	50
		2005-03-21	54

- Four software maintenance tasks
  - From pairs of snapshots from version histories
- Developer change = Gold standard

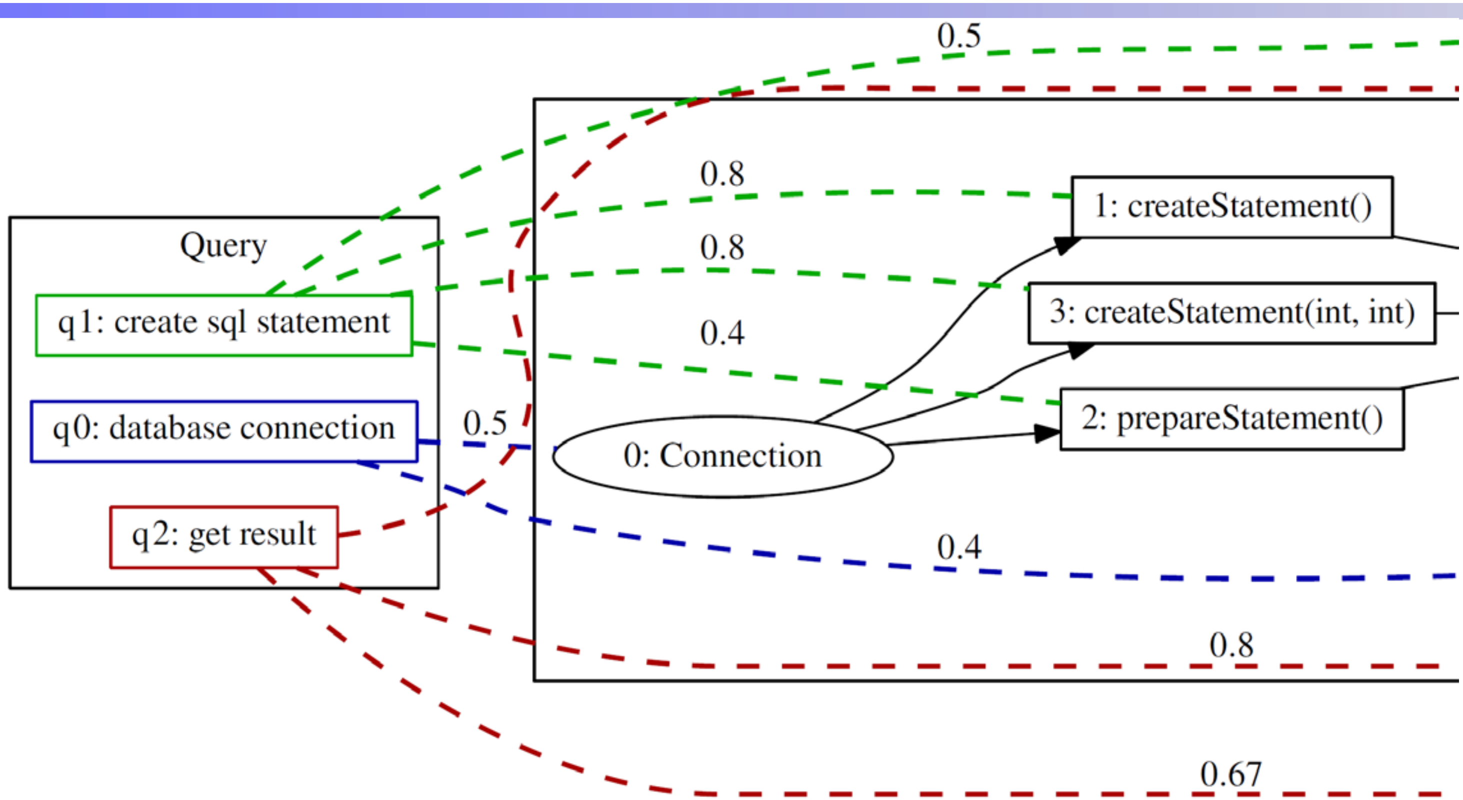
Overall Results: Accuracy

Task	#Targets	Text Search		Code Clone Detection		Our approach	
		FP	FN	FP	FN	FP	FN
1	2	526	0	0	2	36	0
2	8(186)	829(651)	0	0	8	200(22)	0
3	37	297	0	23	3	25	2
4	19	86	0	9	2	3	0

For task 2, the number in the bracket:

Adjusted numbers after considering correct locations that are not modified yet by developers

Free Text Code Search (FSE12)



Find optimum connected graph that meets user needs

Greedy subgraph search algorithm with shortest path indexing

	Portfolio			LRR		
	Prec.	**Rec.**	**F_1**	**Prec.**	**Rec.**	**F_1**
Group I	0.24	0.51	0.33	0.53	0.56	0.54
Group II	0.48	0.64	0.55	0.79	0.72	0.75

Example Based Code Search (ASEJ15)

Example 1:

```
if(c>3){
   c=getStr();
   c=ext();
}
```

Example 2:

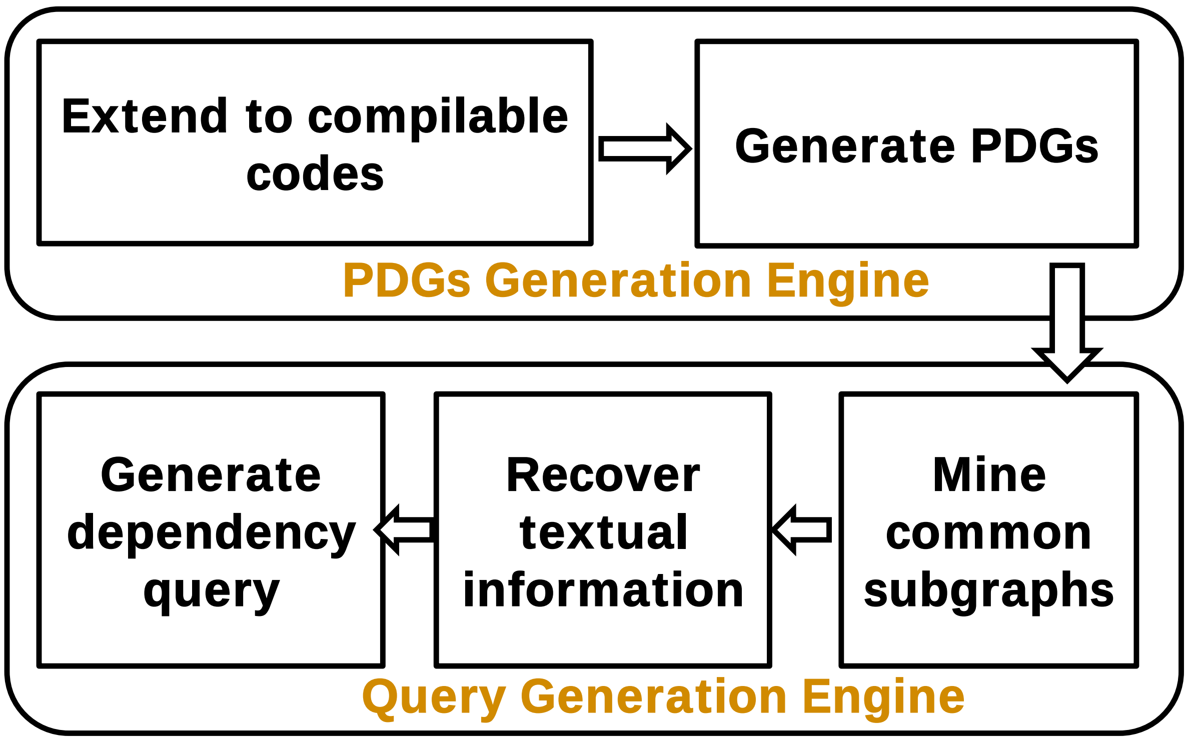
```
if(b>1){
   b=ext()+foo();
}
```

Node declarations: *ctrlPoint A, func B*;

Node descriptions: *A contains if, B contains ext*;

Relationship descriptions: *A oneStep controls B*;

Targets: *A,B*;



Lightweight type inference,
Closed subgraph mining

	Our	Manual
Prec.	0.684	0.584
Recall	0.721	0.767
F1	0.702	0.664

Coding & Collaboration

Intelligent Multimodal Code Search | **Recommender for Libraries and Online Resources** | **Coding and Collaboration With New Media** | **Empirical Studies**

Structured Code Search (ASE10)

Free Text Code Search (FSE12)

Example Based Code Search (ASEJ15)

Active Code Search (ASE14)

Structured + Topic Model (WCRE10)

Multi-Criteria Project Search (ICECCS13)

Similar Project Search (ICSM12)

Coding & Collaboration

Intelligent Multimodal Code Search | **Recommender for Libraries and Online Resources** | Coding and Collaboration With New Media | Empirical Studies

Recommending Related Libraries (WCRE13)

Recommending API Methods Given Feature Requests (ASE13)

Recommending Answer Posts (ASE11)

Coding & Collaboration

Intelligent Multimodal Code Search | Recommender for Libraries and Online Resources | **Coding and Collaboration With New Media** | Empirical Studies

Automated Content Categorization (ICPC14)

Recommending Tags to Contents (MSR13, ICSME14)

Recommending Best Answerers (QMC13)

Observatory of Tweets and Trends (ASE11)

Identification of Relevant Microblogs (ICSM12)

Developer Recommendation (WCRE11)

Project Success Estimation (CSMR13)

New Media Usages

MUD14
CSMR13
SAC13
MSR12

Coding Practice

PLOS13
COMPSAC13
CSMR13

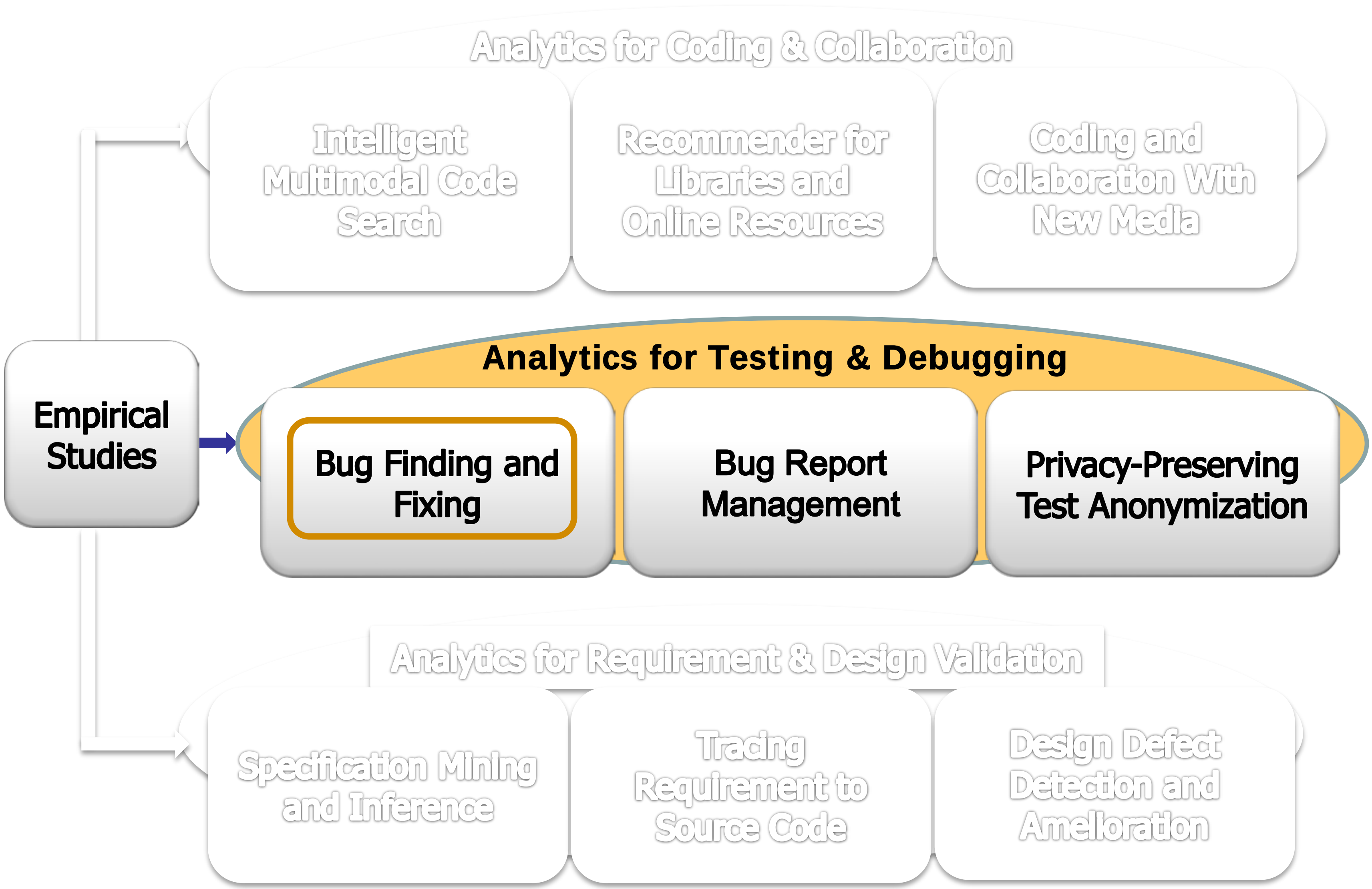
Collaboration Patterns

WCRE10

Software Diffusion

APSEC12

Our Past and Current Work



Bug Finding and Fixing are Hard !

- Software bugs cost the US Economy 59.5 billion dollars annually
  - Stated by the US National Institute of Standards and Technology in 2002

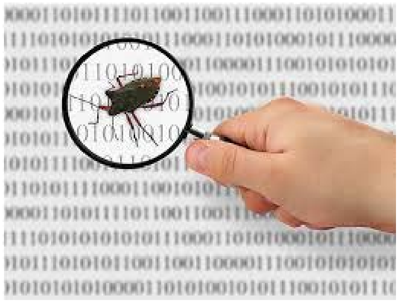
(Tassey, 2002)

- Software debugging is an expensive and time consuming task in software projects
  - Testing and debugging activities account 30-90% of the labor expended on a project

(Beizer, 1990)

Bug Finding Techniques

A buggy program



Analyze program

List of possible buggy program elements

Bug Finding Techniques

```
play.exceptions.TemplateExecutionException: / by zero
        at play.templates.Template.throwException(Template.java:262)
        at play.templates.Template.render(Template.java:227)
        at play.templates.Template$ExecutableTemplate.invokeTag(Template.java:359)
        at /app/views/Application/show.html.(line:21)
        at play.templates.Template.render(Template.java:207)
        at play.mvc.results.RenderTemplate.<init>(RenderTemplate.java:22)
        at play.mvc.Controller.renderTemplate(Controller.java:367)
        at play.mvc.Controller.render(Controller.java:393)
        at controllers.Application.show(Application.java:26)
        at play.utils.Java.invokeStatic(Java.java:129)
        at play.mvc.ActionInvoker.invoke(ActionInvoker.java:124)
        at Invocation.HTTP Request(Play!)
```

Failure

Bug 41588 - blank perspective screen

With no apparent reason, the entire perspective becomes blank. I have seen this problem occassionaly from release 1.0, and I have been hoping the problem would go away with newer releases. It still occurs frequently with 2.x.

Bug Report

Bug Finder

```
public void processFirstItem(ArrayList<Item> itemList) {
    if (!itemList.isEmpty()){
        itemList.get(0);
    }
public void processFirstStudent(ArrayList<Student> studList) {
    studList.get(0);
```

Anomaly

Spectrum-Based Fault Localization

Block ID	Program Element	T1	T2	T3, T4, …
1	double a, x; double ap, del, sum; int n; double temp; if (x <= 0.0)	●	●	
2	{return 0.0;}		●	
3	del = sum = 1.0 / (ap = a); for (n = 1; n <= ITMAX; ++n){	●		
4	sum += del *= x / ++ap; if (Abs(del) < Abs(sum) * EPS){	●		
5	/*BUGS: supposed to be:*/ /* temp = sum * exp(-x + a*log(x)-Lgamma(a))*/ temp = sum * exp(x + a*log(x)-Lgamma(a)); return temp;}}	●		
	Status of Test Case Execution	F	P	

Program spectra

Measuring suspiciousness

Block ID	Program Elements	Suspiciousness Scores: Ochiai	Klosgen	Pietatsky Shapiro

1	`double a, x;`
	`double ap, del, sum;`
	`int n;`
	`double temp;`
	`if ( x <= 0.0 )`
		0.82	0.31	-0.04

2	`{return 0.0;}`	0.39	0.06	0
3	`del = sum = 1.0 / (ap = a);` `for ( n = 1; n <= ITMAX; ++n ){`	0.93	0.34	-0.15
4	`sum += del *= x / ++ap;` `if ( Abs( del ) < Abs( sum ) * EPS ){`	0.93	0.34	-0.15

5	`/*BUGS: supposed to be:*/`
	`/*temp = sum * exp(-x + a*log(x)-LGamma(a))*/`
	`temp = sum * exp( x + a * log( x ) - LGamma(a));`
	`return temp; }}`
		0.93	0.34	0

e.g., spectrum-based fault localization

(Abreu et.al, TAICPART-MUTATION'07, Lucia et al., ICSM'10)

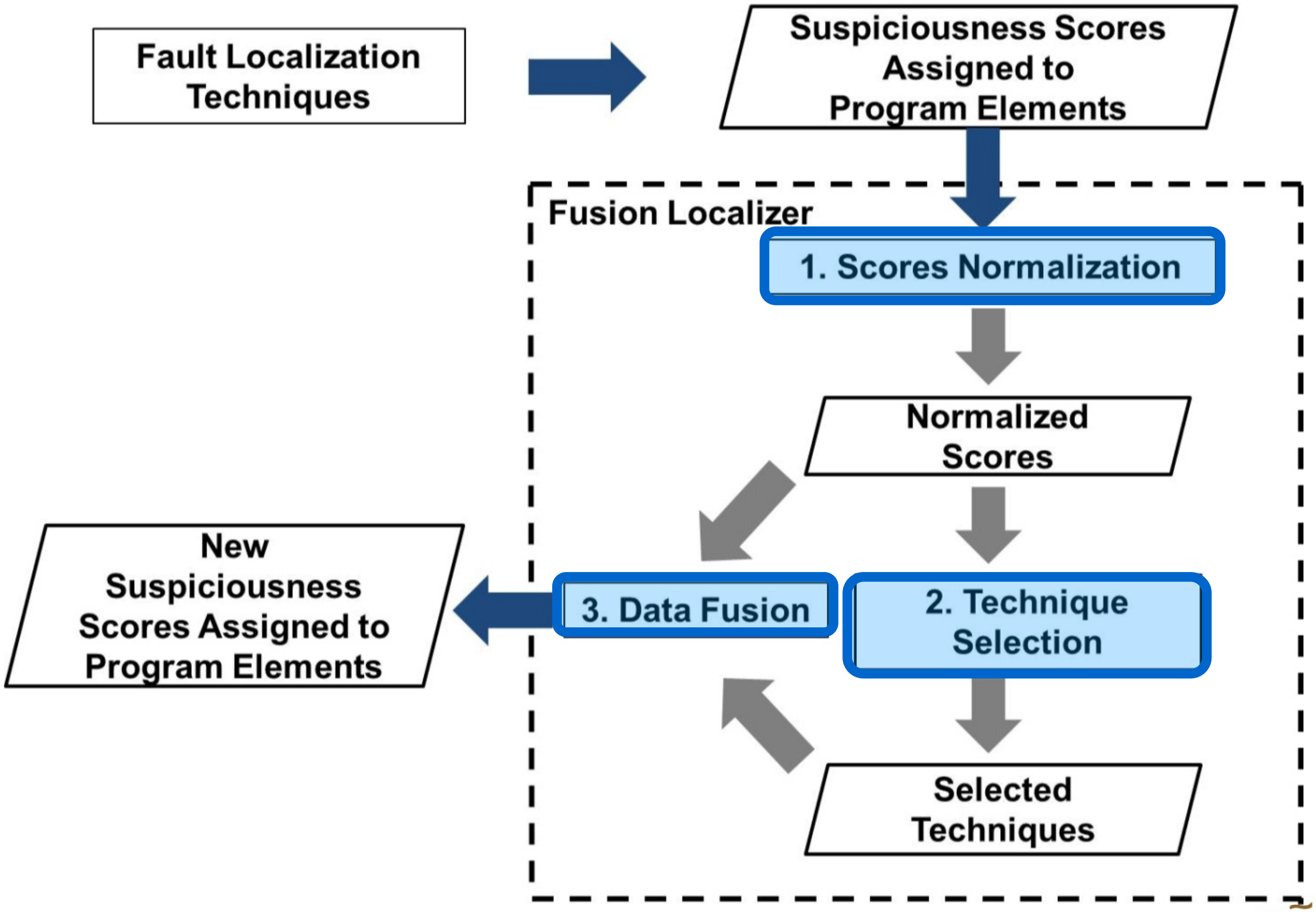
Motivation

There is no single fault localization techniques that is the best in all cases. (Lucia et al., JSEP, 2014)

Dataset	Version	Ochiai	Klosgen	Pietatsky Saphiro
space	v35	$\boxed{0.01}$	0.13	0.47
nanoxml	v2_b6	0.50	$\boxed{0.05}$	0.63
tcas	v23	0.56	0.56	$\boxed{0.04}$

Combine different techniques?

Fusion Localizer (ASE14)



Step 2. Techniques selection

A set of
fault localization techniques

Choosing the techniques to be fused

(A) Overlap-based selection

(B) Bias-based selection

Selected
fault localization techniques

Step 2. Techniques selection

(A) Overlap-based selection

- Based on the overlap ratio
- Select 50% of the least overlap techniques

(Wu, Data Fusion in Information Retrieval, 2012)

Step 2. Overlap-based selection

Technique	Top-K Most Suspicious Blocks
Ochiai	Block 2, Block 3, Block 4, Block 7, Block 8
Klosgen	Block 4, Block 5, Block 6, Block 7, Block 9
Piat. Shapiro	Block 1, Block 4, Block 5, Block 6, Block 8
Tarantula	Block 4, Block 5, Block 6, Block 8, Block 10

L_{all}	Block 1, Block 2, Block 3, Block 4, Block 5, Block 6, Block 7, Block 8, Block 9, Block 10
L_{Ochiai}	Block 2, Block 3

$$o\_rate_i = \frac{|L_{all}| - |L_i|}{|L_{all}|}$$

$$\text{Overlap Rate of Ochiai} = \frac{10 - 2}{10} = 0.8$$

Step 2. Technique selection

(B) Bias-based selection

- Based on the similarity score
- Bias = 1 – similarity score
- Select 50% of the most biased techniques

(Nuray and Can, Information Processing and Management, 2006)

Step 2. Bias-based selection

L_{all}

Block	Freq.
Block 1	1
Block 2	1
Block 3	1
Block 4	4
Block 5	3
Block 6	3
Block 7	2
Block 8	3
Block 9	1
Block 10	1

L_{Ochiai}

Block	Freq.
Block 1	0
Block 2	1
Block 3	1
Block 4	1
Block 5	0
Block 6	0
Block 7	1
Block 8	1
Block 9	0
Block 10	0

Technique	Top-K Most Suspicious Blocks
Ochiai	Block 2, Block 3, Block 4, Block 7, Block 8
Klosgen	Block 4, Block 5, Block 6, Block 7, Block 9
Piat. Shapiro	Block 1, Block 4, Block 5, Block 6, Block 8
Tarantula	Block 4, Block 5, Block 6, Block 8, Block 10

Cosine Similarity

$$Sim(L_i, L_{all}) = \frac{\sum_{j=1}^{m} L_j \times L_{all_j}}{\sqrt{\sum_{j=1}^{m} L_j^2} \times \sqrt{\sum_{j=1}^{m} L_{all_j}^2}}$$

$$Bias(L_i, L_{all}) = 1 - Sim(L_i, L_{all})$$

$$Sim(L_{Ochiai}, L_{all}) = \frac{1+1+4+2+3}{\sqrt{5} \times \sqrt{(1+1+1+16+9+9+4+9+1+1)}} = 0.6822$$

$$Bias(L_{Ochiai}, L_{all}) = 0.3178$$

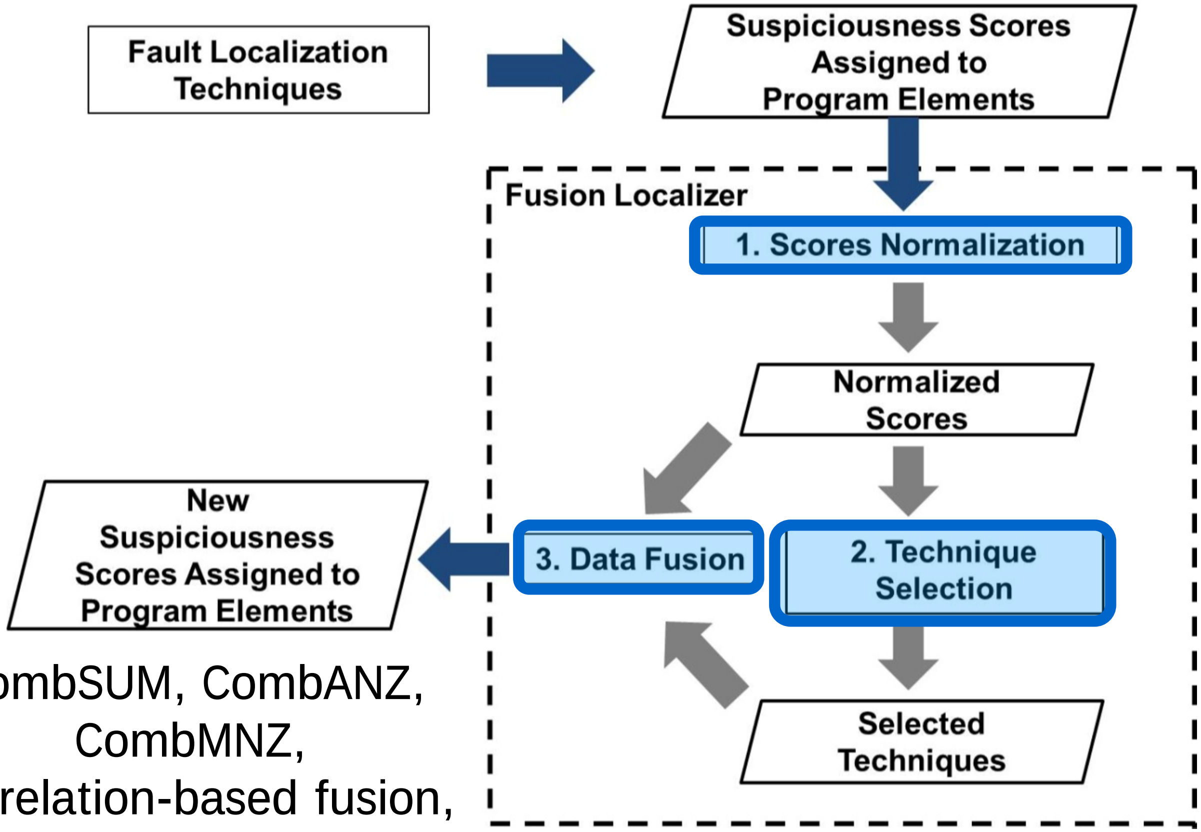
School of Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

Data fusion methods

- Score-based fusion
  1. CombSUM : Sum up all scores (Fox et al., NIST, 1994)
  2. CombANZ : Average of the non-zero scores (Fox et al., NIST, 1994)
  3. CombMNZ : Sum up all scores multiplied by the number of techniques that assign a non-zero score (Fox et al., NIST, 1994)
  4. Correlation-based method : CorrA, CorrB (Wu, "Data Fusion in Information Retrieval", 2012)
- Ranking-based fusion
  5. Borda Count : Sum up all ranking (Aslam and Montague, SIGIR, 2001)

Variants of Fusion Localizer



1. Zero-One Normalization
2. Reciprocal Ranking Normalization

1. Overlap-based
2. Bias-based

CombSUM, CombANZ, CombMNZ, Correlation-based fusion, BordaCount

Score Normalization, Technique Selection

Data Fusion

Dataset

Dataset	LOC	Num. of Buggy Version	Num. of Test Cases
print_token	478	5	4,130
print_token2	399	10	4,115
replace	512	31	5,542
schedule	292	9	2,650
schedule2	301	9	2,710
tcas	141	36	1,608
tot_info	440	19	1,051
space	6,218	35	13,585
NanoXML v1	3,497	6	214
NanoXML v2	4,007	7	214
NanoXML v3	4,608	9	216
NanoXML v5	4,782	8	216
XML security v1	21,613	6	92
XML security v2	22,318	6	94
XML security v3	19,895	4	84
Rhino	49k	11	20-152
Lucene	88k	9	1,072-1,154
Ant	264k	10	1,024-1,555

Total：230 Bugs

Avg. % of code inspected to localize all bugs

Technique	Average	Technique	Average
$F^{Zero-One,Overlap}_{CombANZ}$	21.36%	Naish2	24.63%
$F^{Zero-One,Bias}_{CombANZ}$	21.39%	GP13	24.78%
$F^{Zero-One,Bias}_{CombSUM}$	22.94%	Ochiai	25.29%
$F^{Zero-One,Overlap}_{CorrB\_Top50\%}$	23.11%	GP03	25.82%
$F^{Zero-One,Overlap}_{CombSUM}$	23.15%	Tarantula	26.77%
$F^{Zero-One,Overlap}_{CorrB\_Top10\%}$	23.23%	GP19	31.60%
$F^{Zero-One,Overlap}_{CombMNZ}$	23.31%	Naish1	34.40%
$F^{Zero-One,Bias}_{CorrB\_Top10\%}$	23.33%	GP02	39.48%
$F^{Zero-One,Bias}_{CorrB\_Top50\%}$	23.38%	Russel&Rao	42.48%
$F^{Zero-One,Overlap}_{CorrA\_Top10\%}$	23.56%	Binary	52.04%
$F^{Zero-One,Bias}_{CombMNZ}$	23.78%	Wong1	86.26%
$F^{Zero-One,Bias}_{CorrA\_Top10\%}$	23.78%		

Proportion/number of bugs localized

When 10% of blocks are inspected

Technique	%Bug
$F_{CombANZ}^{Zero-One,Overlap}$	46.96%
$F_{CombANZ}^{Zero-One,Bias}$	46.52%
Ochiai	42.17%
Naish2	36.96%
GP13	36.96%

When 10 blocks are inspected

Technique	Hit@10
$F_{CombANZ}^{Zero-One,Bias}$	91
$F_{CombANZ}^{Zero-One,Overlap}$	87
Ochiai	74
Naish2	73
GP13	72

Report-Directed Bug Finding (ICSME14)

ID	P	Status	Severity	Version	Summary
107*	P2	RESOLVED	enhancement	0.4	[BZ] Support integration with deadline
117					

107 Added generic column handling to allow +deadline to add the deadline column

$$\sum_{i=1}^{15} w_i \times VSM_i(b,f) + \sum_{J}^{J\in H,R,S} w_j \times score_J(b,f)$$

IR Composition + Genetic Algorithm + (History + Similar Report + Structure)

Project	Approach	Hit@1	Hit@5	Hit@10	MAP	MRR
AspectJ	$VSM_{natural}$	25 (8.7%)	43 (15.0%)	65 (22.3%)	0.05	0.13
	VSM[illegible]	[illegible]	[illegible]	[illegible]	0.07	0.16
	Am[illegible]	[illegible]	[illegible]	[illegible]	[illegible]3	0.54
	Am[illegible]	[illegible]	[illegible]	[illegible]	**[illegible]3**	**0.61**
Eclipse	VS[illegible]	[illegible]	[illegible]	[illegible]	[illegible]1	0.01
	VS[illegible]	[illegible]	[illegible]	[illegible]	[illegible]1	0.01
	Am[illegible]	[illegible]	[illegible]	[illegible]	[illegible]5	0.45
	Am[illegible]	[illegible]	[illegible]	[illegible]	**[illegible]9**	**0.48**
SWT	VS[illegible]	[illegible]	[illegible]	[illegible]	[illegible]1	0.24
	VS[illegible]	[illegible]	[illegible]	[illegible]	[illegible]23	0.26
	$AmaL$	61 (62.2%)	80 (81.6%)	**88 (89.8%)**	0.62	**0.71**
	$AmaL_{compo.}$	**62 (63.2%)**	**83 (82.6%)**	**88 (89.8%)**	**0.63**	**0.71**

SIG

On average, AmaLgam$_{\text{Composite}}$ improves AmaLgam by 6.8%, 8.0%, 5.0%, 14.4%, and 6.5% in terms of Hit@1, Hit@5, Hit@10, MAP, and MRR respectively

Anomaly-Directed Bug Finding (ICSE12)

Code Fragment 1	Code Fragment 2

Code Fragment 1

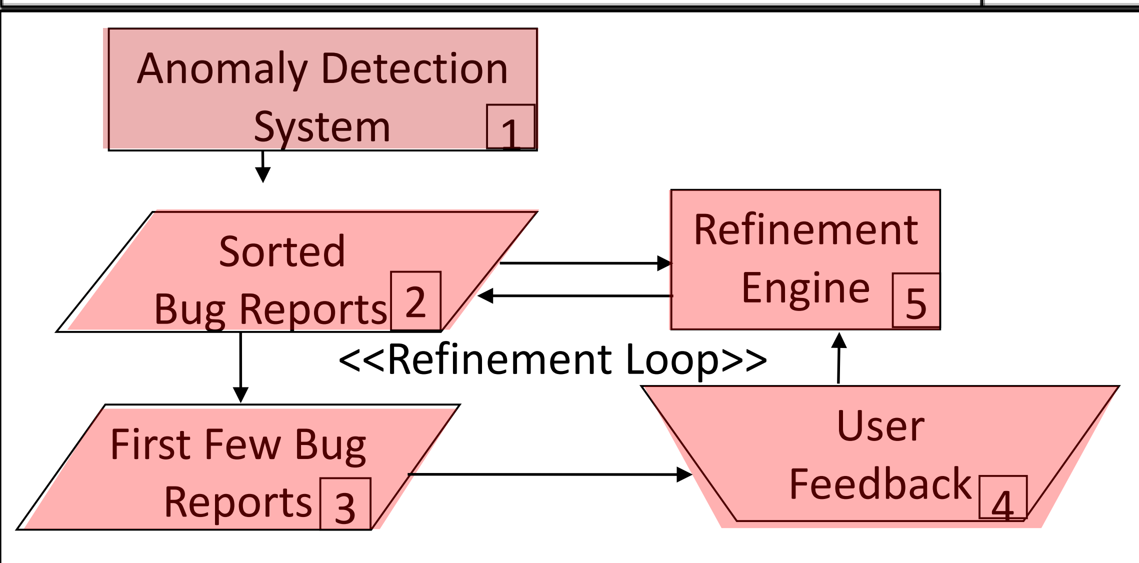
```
File: linux-2.6.19/fs/sysfs/inode.c

219: struct dentry * dentry = sd->s_dentry;
220:
221: if (dentry) {
        /* the following parts are detected as clones */
222:        spin_lock(&dcache_lock);
223:        spin_lock(&dentry->d_lock);
224:        if (!(d_unhashed(dentry) && dentry->d_inode)) {
225:                dget_locked(dentry);
226:                __d_drop(dentry);
227:                spin_unlock(&dentry->d_lock);
228:                spin_unlock(&dcache_lock);
229:                ......
```

Code Fragment 2

```
File: linux-2.6.19/drivers/infiniband/hw/ipath/ipath_fs.c

456: struct dentry *tmp;
457:
458: tmp = lookup_one_len(name, parent, strlen(name));
459:
460: spin_lock(&dcache_lock);
461: spin_lock(&tmp->d_lock);
462: if (!(d_unhashed(tmp) && tmp->d_inode)) {
463:         dget_locked(tmp);
464:         __d_drop(tmp);
465:         spin_unlock(&tmp->d_lock);
467:         spin_unlock(&dcache_lock);
468:         ......
```



Feature Extraction + Classification

Improve
Avg. % TP Found:
11% for Linux
87% for Eclipse
86% for ArgoUML

Testing & Debugging

Bug Finding and Fixing			Bug Report Management	Privacy-Preserving Test Anonymization	Empirical Studies

Failure-Directed	Report-Directed	Anomaly-Directed	Extensions	Automated Patching
ASE14 ICSM10-JSEP14 ICSM12 ASE11 HASE11 ISSTA09	ICSME14 ICSE12 ICPC14x2	CSMR-WCRE14 COMPSAC14 SAC14 ICSE12 RV11 ASE10 KDD09	<u>Eff. Estimate:</u> ISSRE14, ICSM13-EMSE15 <u>Reduce. Man. Eff.:</u> ASE12-ASEJ15 <u>Comm. Resource:</u> FSE14 ICSE14 <u>Post Mortem:</u> WCRE13	ASE12 ICSE12

Testing & Debugging

Bug Finding and Fixing | **Bug Report Management** | Privacy-Preserving Test Anonymization | Empirical Studies

Duplicate Detection

ASE12
CSMR12
ASE11
ICSE10

Report Prioritization

ICSM13-EMSE15
WCRE12
ICSM12

Report Categorization

COMPSAC14
ICECCS14
WCRE12

Report Assignment

WCRE13

Reopen Prediction

ASEJ15
CSMR13

Single-Data Release

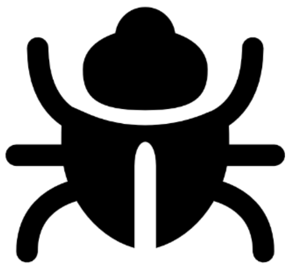
PLDI11

Multiple-Data Release

ASE12

Testing & Debugging

Bug Finding and Fixing | Bug Report Management | Privacy-Preserving Test Anonymization | **Empirical Studies**



Real Bugs

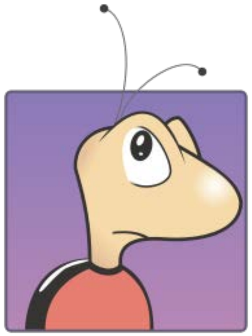
ASE12-ASEJ15
IEICE Trans14
SAC14
ISSRE12

Test Adequacy

APSEC14
CSMR-QSIC13

Fault Localization

ASE14
MSR12
ICSM13



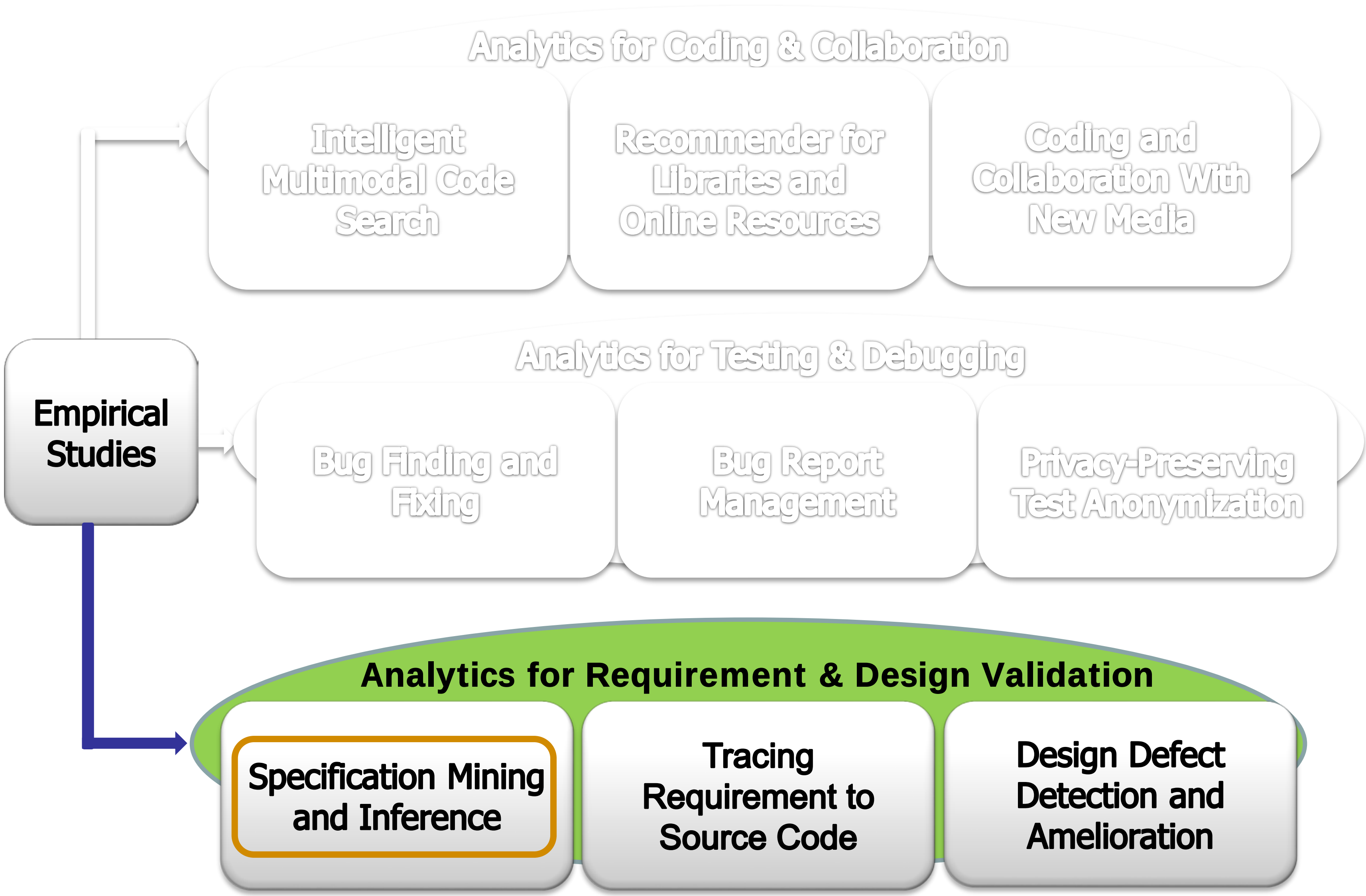
Bug Trackers

CSMR-WCRE14
ISSRE13

Bug Linking

CSMR13

Our Past and Current Work



Specification Mining and Inference

- Most bugs are caused due to semantic errors (Tan et al., ESEJ14)
  - Programs are not implemented according to requirements
- Developers often do not have the expertise or time to write formal specifications
- Viable solution: specification mining
  - Automated reverse engineering of specifications from programs

Specification Mining and Inference

Strong Properties

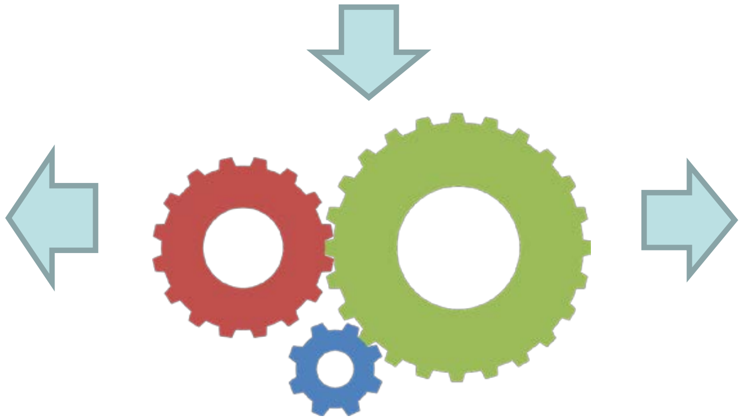
Likely invariants

Frequent patterns

Temporal rules

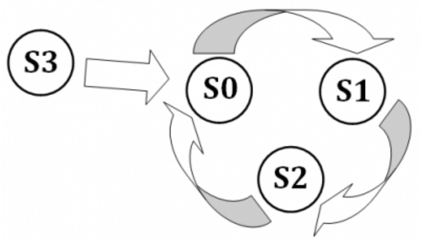
Live sequence charts

Execution Traces



Specification Miners

Unified Model



Finite State Machine

Message Sequence Graphs

Class Diagram

Mining Temporal Rules [JSEP08,SCP12,ICDE12]

- Aim:
  - Find temporal rules observed within a trace set:

    "Whenever a series of events occurs, eventually another series of events will also occur"
  - Among most widely used temporal logic expression for verification (Dwyer et al. ICSE'99).

LTL BNF Notation

$$
\begin{aligned}
rules &:= \quad G(prepost) \\
prepost &:= \quad event \rightarrow post | event \rightarrow XG(prepost) \\
post &:= \quad XF(event) | XF(event \wedge XF(post))
\end{aligned}
$$

Significance, Soundness and Completeness

- Distinguish Significant Rules via Statistical Notions
  - Support: The number of traces supporting the premise
  - Confidence: The likelihood of the premise being followed by the consequent

- Ensure Soundness and Completeness

  With respect to input traces and specified thresholds
- Sound

  All mined rules are statistically significant
- Complete

  All statistically significant rules are mined/represented

Scalability Challenge

Existing Method (Yang06)

Check all possible 2-event rules
(n x n of them)
for statistical significance

Need to check n^L rules for L-event rules

> 50^1000 operations
>
> vs.
>
> < 25 seconds

Our Method

Explore the search space depth first and identify significant ones

Employ a number of search space pruning strategies

Linear to the size of the output significant rules and the length of traces

Good results on standard benchmarks datasets

Specification Mining Strategies – I & II

$$\dfrac{\dfrac{\begin{array}{c}[Apriori - Support]\\ Rx = p \rightarrow c;\ Ry = q \rightarrow c\\ p \sqsubset q\\ sup(Rx) < min\_sup\end{array}}{sup(Ry) < min\_sup}}{Ry\ \ is\ \ not\ \ significant}$$

Rx: a -> z ; sup(Rx) < min_sup

Ry_s	a,b -> z a,b,c -> z a,c -> z a,b,d -> z	Non-significant

$$\dfrac{\dfrac{\begin{array}{c}[Apriori - Confidence]\\ Rx = p \rightarrow c;\ Ry = p \rightarrow d\\ c \sqsubset d\\ conf(Rx) < min\_conf\end{array}}{conf(Ry) < min\_conf}}{Ry\ \ is\ \ not\ \ significant}$$

Rx: a -> z ; conf(Rx) < min_conf

Ry_s	a -> b,z a -> b,c,z a -> c,z a -> b,d,z	Non-significant

Specification Mining Strategies - III

Detecting Redundant Rules

$$\frac{\begin{array}{c} Rx = p \rightarrow c;\, Ry = q \rightarrow d \\ p{+\!\!+}c \sqsubset q{+\!\!+}d \\ sup(Rx) = sup(Ry) \\ conf(Rx) = conf(Ry) \end{array}}{Rx\ \ is\ \ redundant}$$

Redundant rules are identified and removed early during mining process.

Rx: a -> b,c,d

Ry_s

$$\left.\begin{array}{c} \text{a -> b} \\ \text{a -> c} \\ \text{a -> b,c} \\ \text{a -> b,d} \\ \text{....} \end{array}\right\}$$

Redundant
iff
sup and conf are the same

Program Comprehension: JBoss App. Server

Premise	Consequent
TxManLocator.getInstance()	TransactionImpl.instanceDone()
TxManLocator.locate()	TxManager.getInstance()
TxManLocator.tryJNDI()	TxManager.releaseTransactionImpl()
TxManLocator.usePrivateAPI()	TransactionImpl.getLocalId()
TxManager.getInstance()	XidImpl.getLocalId()
TxManager.begin()	LocalId.hashCode()
XidFactory.newXid()	LocalId.equals()
XidImpl.getTrulyGlobalId()	TransactionImpl.unlock()
TransactionImpl.assocCurThread()	XidImpl.hashCode()
TransactionImpl.lock()	

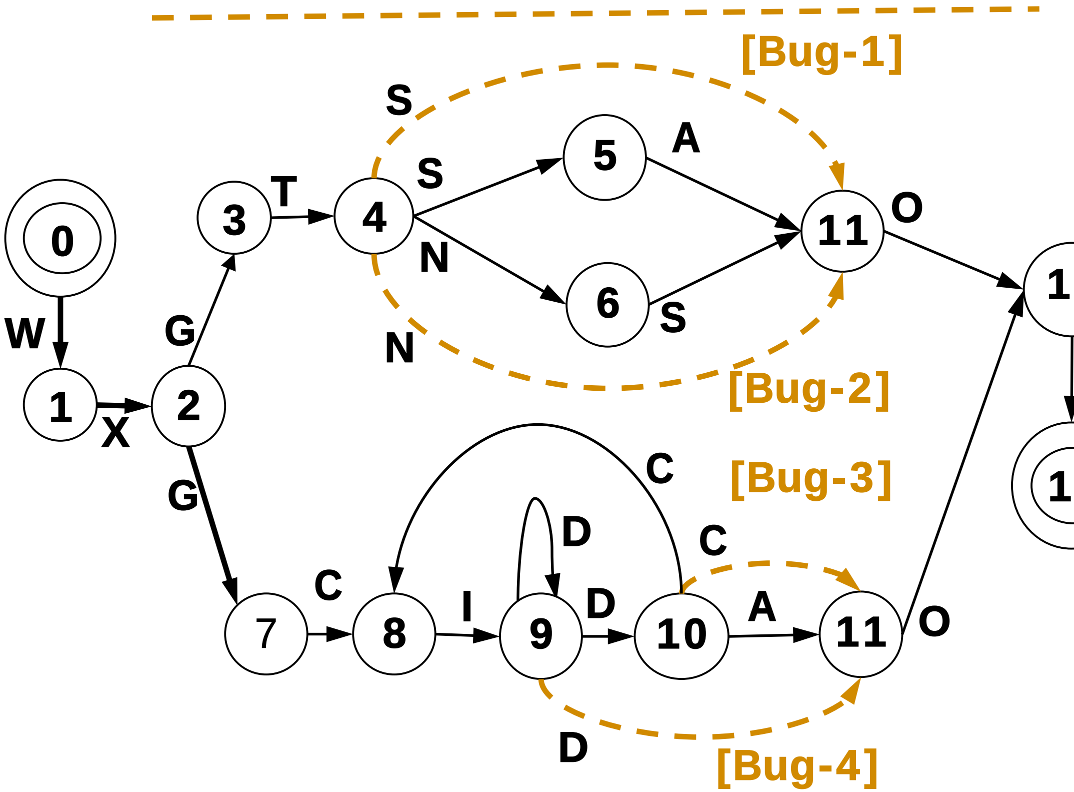
A series of transaction set up events (connection to server instance, transaction manager and implementation set up) is eventually followed with transaction termination events (transaction completion, resource release)

School of Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

Program Verification: VCS Application

Bug: Store (S) and rename (N) without appropriate next actions



Bug: Deletion (D) without log update

Mined Bug-Identifying Rules/Properties

<W;X;G;T;N> -> <S;O;Y>

[Bug-2]

<W;X;G;C;I;D> -> <A;O;Y>

[Bug-3] **[Bug-4]**

Library Usage Rules: Windows (WCRE09,SCP12)

- Collect traces from 10 Windows Applications:
  - Excell, OneNote, TextPad, VS.Net, Visio, WMPlayer, Virtual PC, Movie Maker, WordPad, Access

- Collect traces pertaining to:
  - Registry, Memory Management, GDI (Device Control and UI related API)
  - Produces several million events

Library Usage Rules: Windows

V HeapAlloc(,,); ->HeapFree(,,V);
V GlobalAlloc(,); -> GlobalFree(V);
V VirtualAlloc(,,); ->VirtualFree (,,V);
....
HeapFree(,,V); -P> V HeapAlloc(,,,);

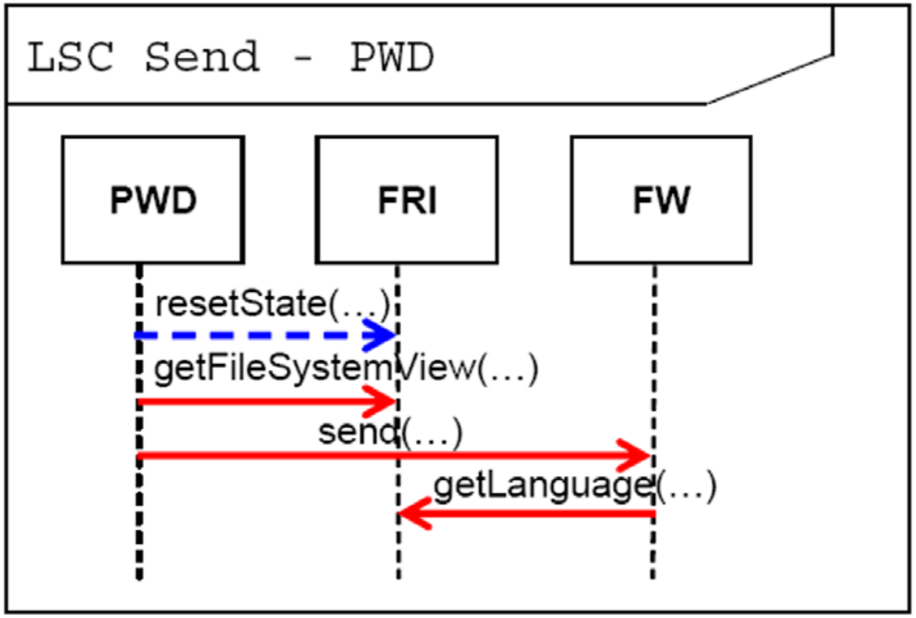
Detect double free, which is disallowed
"Calling HeapFree twice with the same pointer can cause heap corruption, resulting in subsequent calls to HeapAlloc returning the same pointer twice." [MSDN]

Library Usage Rules: Windows

RegCreateKeyExA(V,.) ->RegCloseKey(V);
Not all opened registry need to be closed
Predefined keys need not be closed

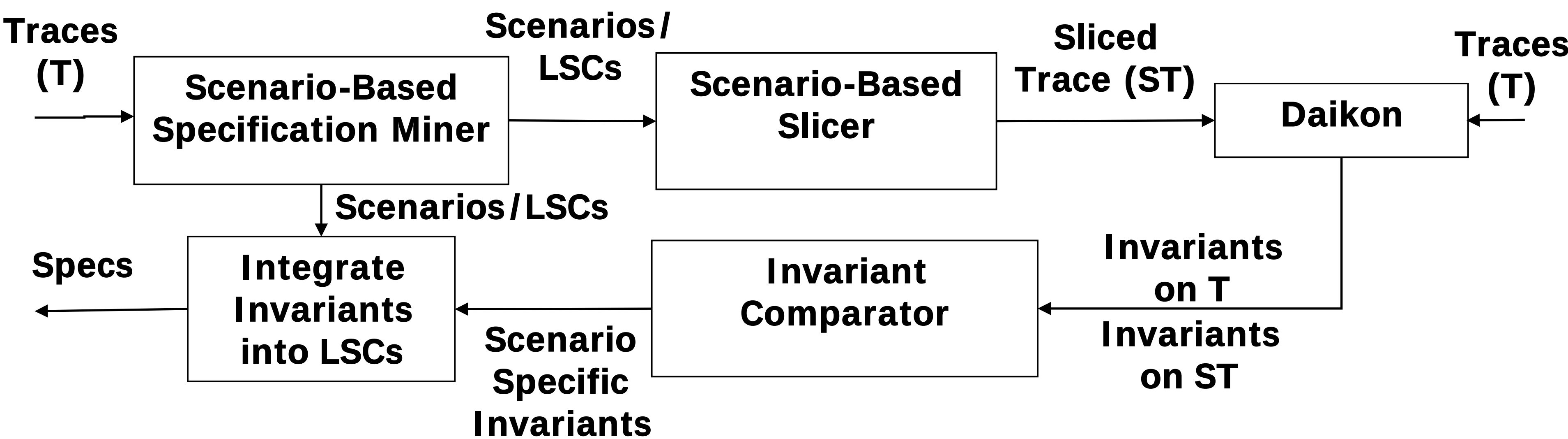
V CreateCompatDC(); -> DeleteDC(V);
V CreCompatBmap(,,);->DeleteObj (V);
V CreRectRgn(,,,)-> DeleteObj(V);
DeleteDC(V) –precede-> V CreCompDC()
SetBkColor(,V); -> V SetBkColor (,)
...

Mining Live Sequence Charts (ASE10,ASEJ12)



Method	Pre	Post
send(...)	code=257 subId="PWD"	subId="PWD" ...

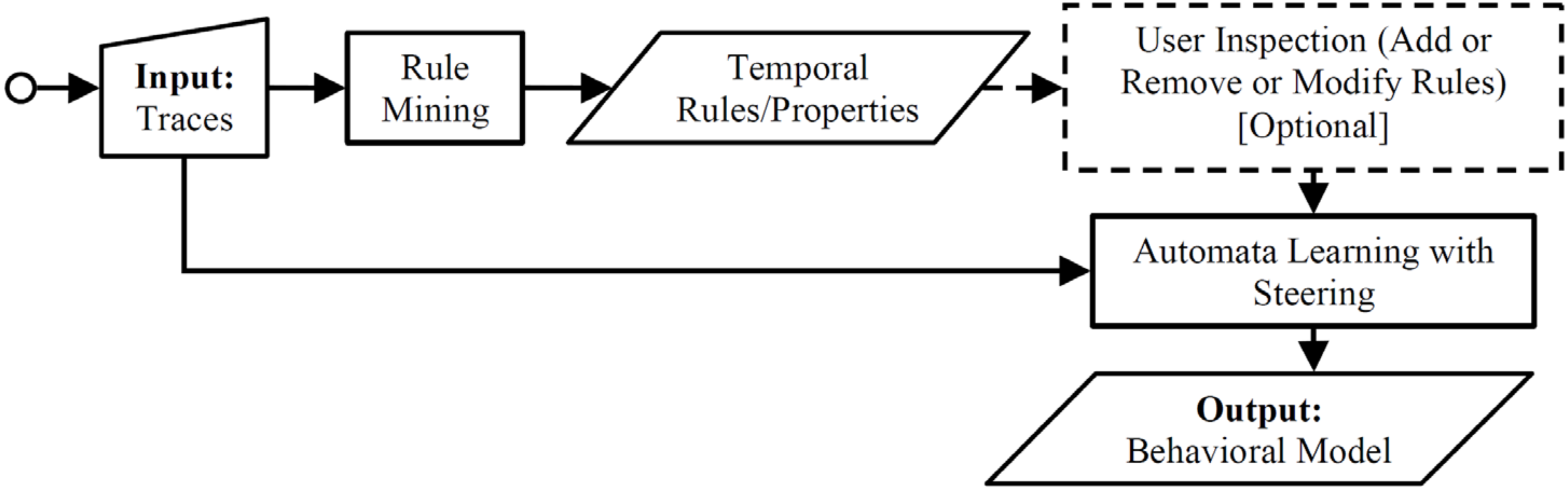
	CFTP	Jeti
Scenario Min.	53s	2s
Daikon (All/Sli)	163s/31s	77s/23s
Slicing	11s	3s



Mining Finite State Machines (FSE09)

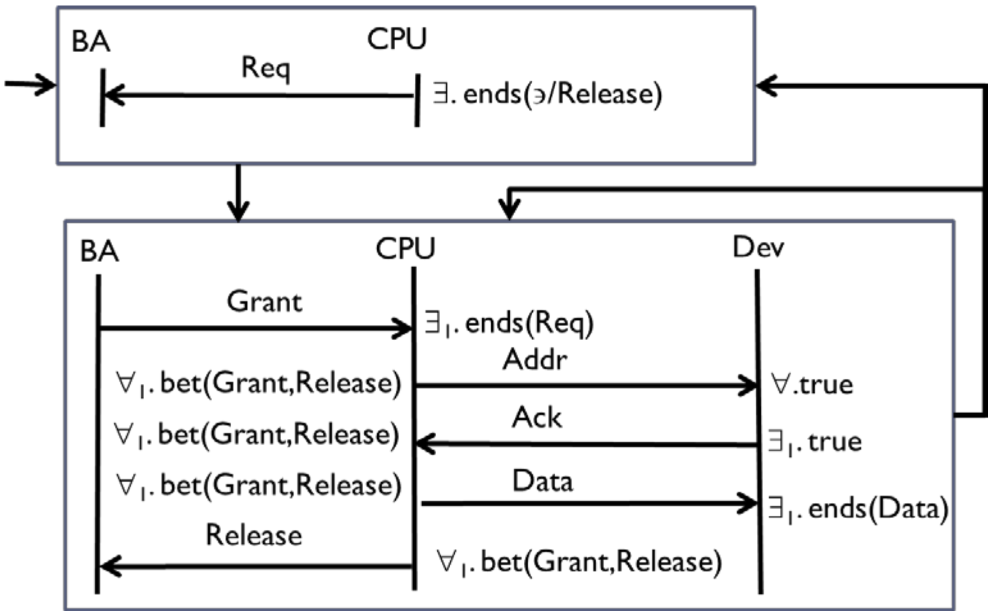
- FSM learner often overgeneralizes
  - Generates a prefix tree acceptor
  - Merge nodes (generalization)

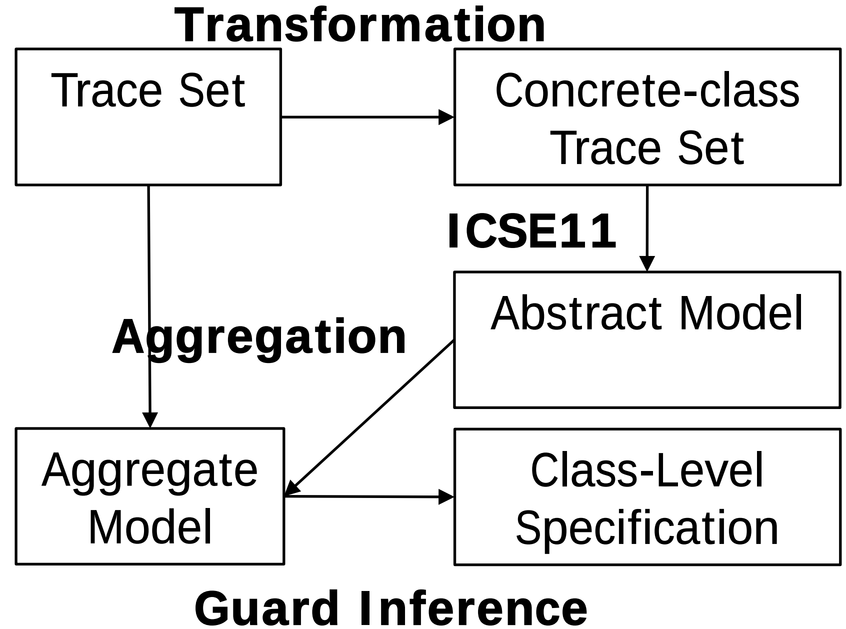
Identification of bad merges using mined temporal rules



System Model	|Evs.|	kTail			With Refinement		
		Precs.	Recall	Time	Precs.	Recall	Time
X11 Windowing Library	356.400	0.873	1.000	0.211	0.905	1.000	0.218
CVS Client	2121.000	0.169	0.970	0.557	1.000	0.970	0.616
WebSphere Business Processes	9317.080	1.000	0.999	1.453	1.000	0.999	1.528

Mining Message Sequence Graphs (ICSE12)



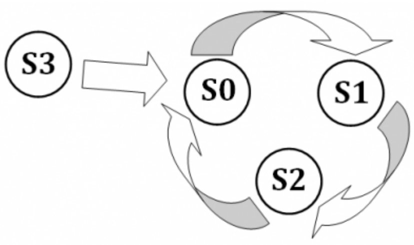


	Concrete			Symbolic		
	Prec.	*Recall*	*F1*	*Prec.*	*Recall*	*F1*
SIP	0.8	0.05	0.09	0.64	0.66	0.65
XMPP-Core	1.0	0.19	0.32	1.0	0.66	0.79
XMPP-MUC	0.61	0.36	0.45	0.67	0.63	0.65
CTAS	0.25	0.43	0.31	0.88	0.90	0.89

Requirement & Design

Specification Mining and inference | Tracing Requirement to Source Code | Design Defect Detection and Amelioration | Empirical Studies

Strong Properties



Unified Model

Inv.	Patterns	Rules	LSC	FSM	MSG	Class Diagram
ICSME14	ICDE09	SCP12	ASE13	FSE09	ICSE12	ICPC14
	KDD07	ICDE11	ICECCS11	FSE06	ICSE11	
		TKDE11	ASE10-ASEJ12			
		IS09	ASE09			
		WODA08	PASTE08			
		DASFAA08	ASE08			
		JSEP08	ASE07			

Requirement & Design

Specification Mining and inference

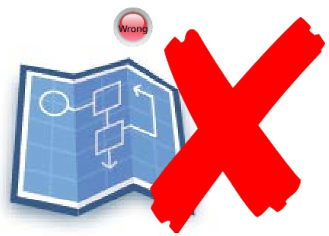
Tracing Requirement to Source Code

Design Defect Detection and Amelioration

Empirical Studies

Concern Location

ICSM13
WCRE11



Design Defect Detection In Tiered Architecture

SEKE11

Empirical Evaluation on Specification Miners

JSS12
WCRE06

Empirical Evaluation on Interestingness Measures

Future Directions

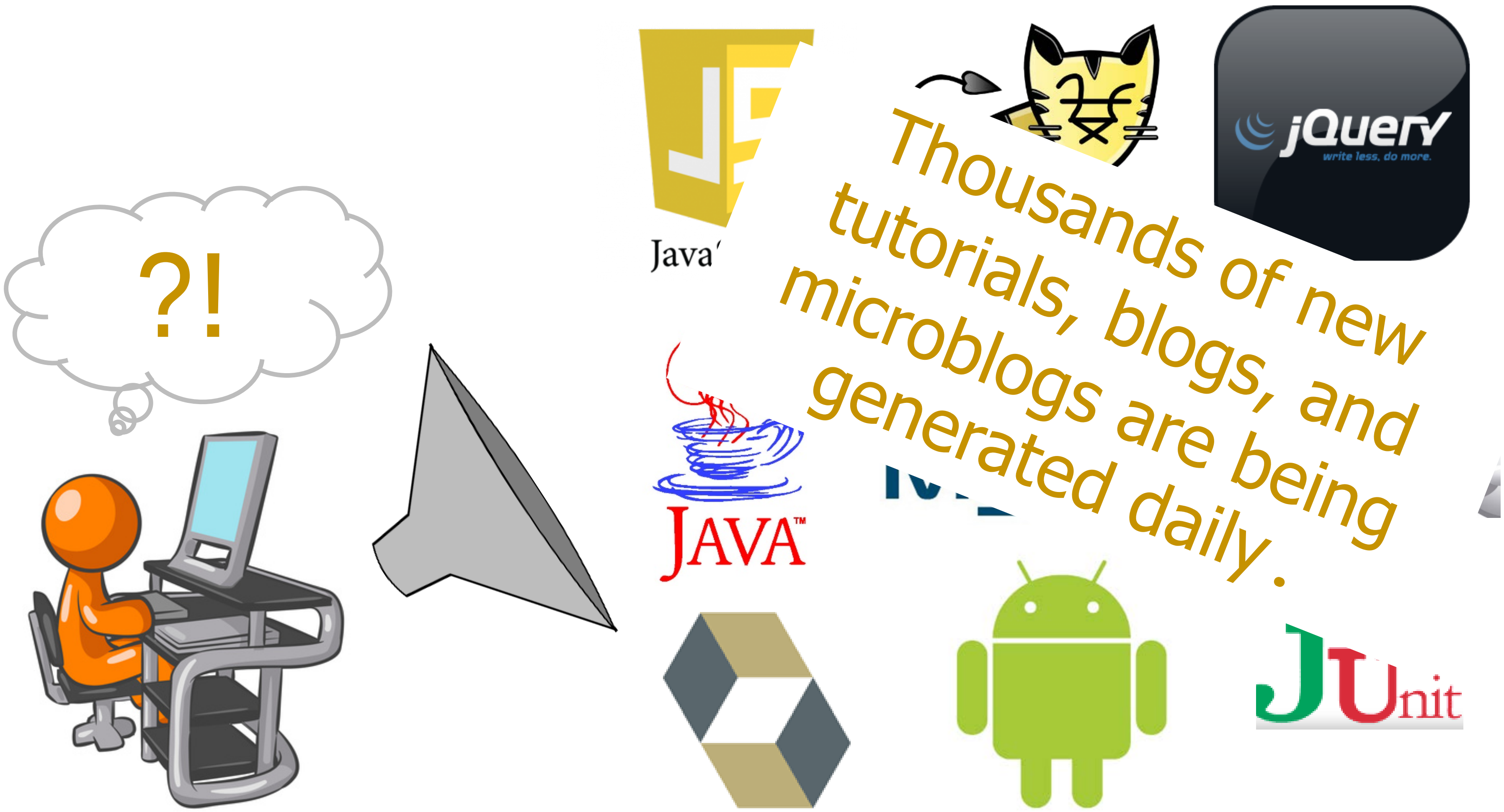
Big Data for Software Engineering

Millions of Projects
Millions of Bug Reports
Millions of Repositories
Millions of Blogs and Posts
Thousands of APIs
....

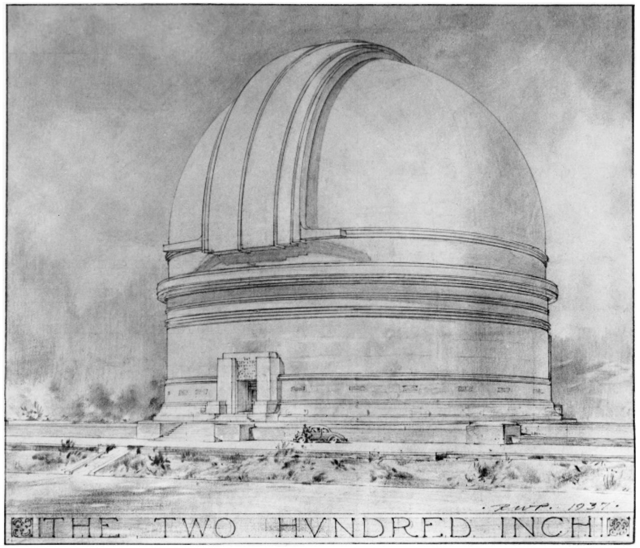
Wealth of Software Engineering Data

There is a wealth of information about what's new, what works, and what doesn't in the Web

Difficulty in “Making Sense” of Data



Our Vision: Personalized Observatory



- Highlights new developments, new solutions, and new pitfalls personalized to a target developer
- Gathers, groups, filters, and summarizes information obtained from various channels
- Automatically updates itself when relevant new information is released in the web

Proposed Process

Data Collection

Data Linking and Grouping

Data Matching and Filtering

Data Summarization

Automatic Update

Process: Data Collection (1)

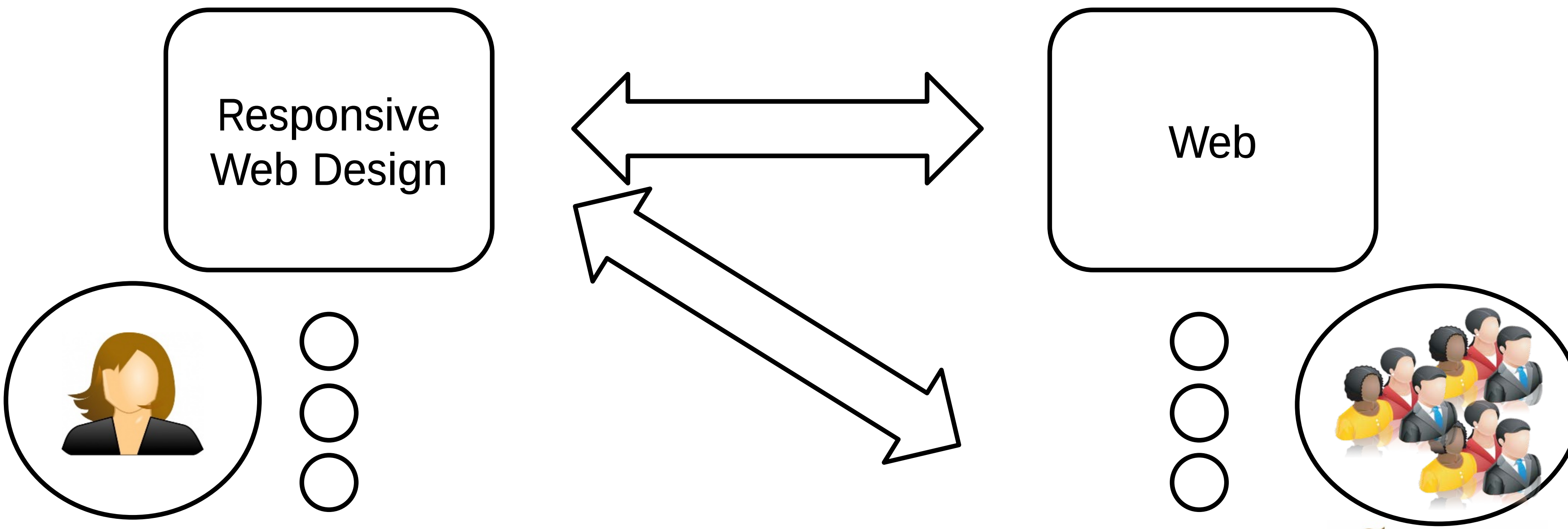
- Gather data from various information channels

Process: Grouping (2)

- Link related pieces of data together
- Group them to a higher level concept
- Approaches:
  - Topic modeling, Clustering

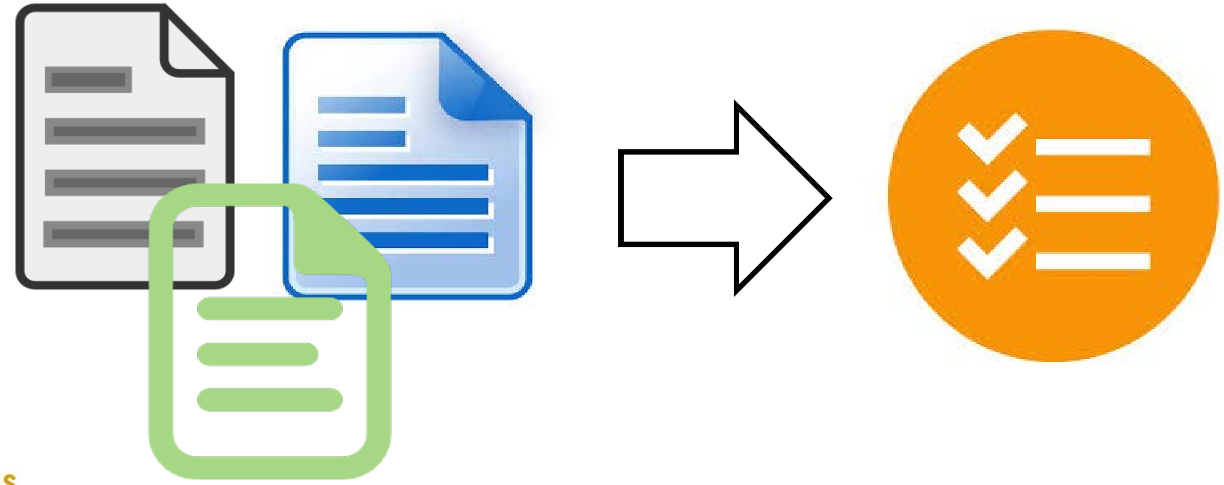
Process: Matching (3)

- Match user interest to community data
- Approaches:
  - Information retrieval approaches



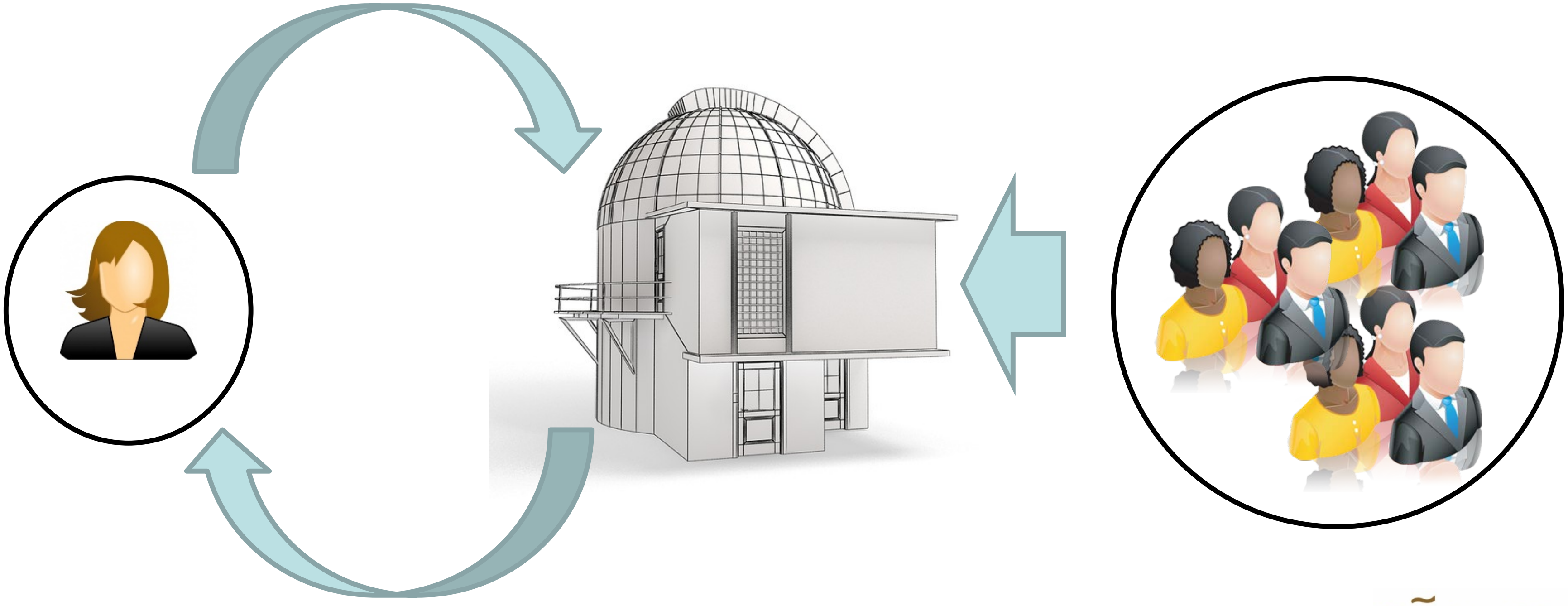
Process: Summarization (4)

- Motivation: A large collection of documents from the community might match user interests
  - Need to summarize them to a manageable size
- Approach: Text summarization approaches

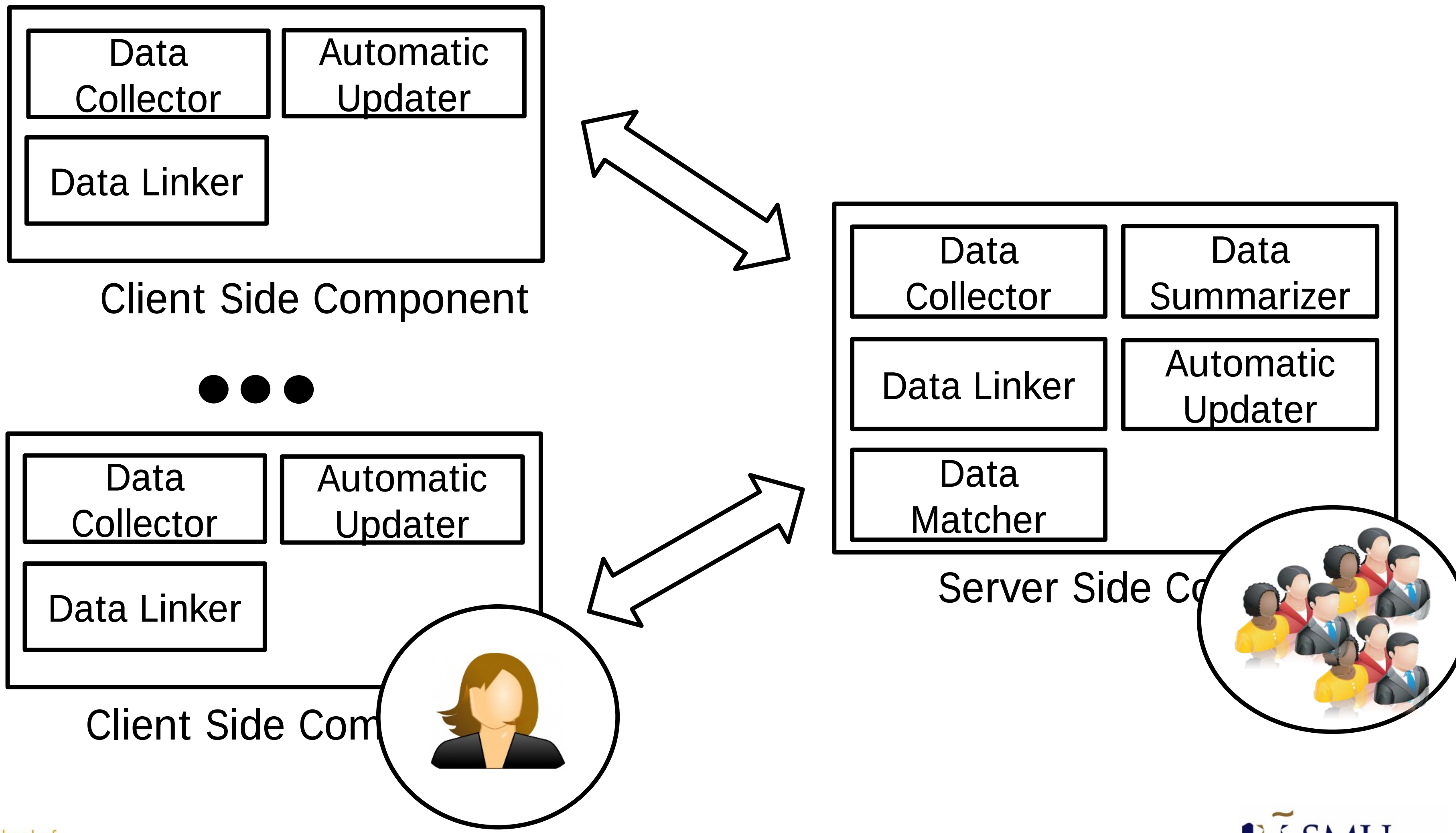


Process: Update (5)

- Continually update considering new user and community data



Proposed Infrastructure



Challenges: Vocabulary Mismatch

- Assumption: Related pieces of information are textual similar.
- Reality: Developer might use peculiar words that are not commonly used by others in the same community.
- How to bridge the differences in the vocabulary used by various developers?

Challenges: Privacy Concern

- Client component needs to send queries to server component
  - Includes developer personal data
- Raises privacy concern:
  - Can some private information be leaked?
  - Sensitive web data, source code, industry project, etc.
- How to minimize privacy leak while not reducing utility?

Challenges: Near Real-Time Update

- Huge amount of information being generated constantly on the web.
- Scale-up the server side component:
  - How to design efficient, incremental and parallel algorithms to collect, group, match, and summarize data?
- Reduce the size of queries being sent from clients to servers:
  - How to produce informative yet succinct queries?

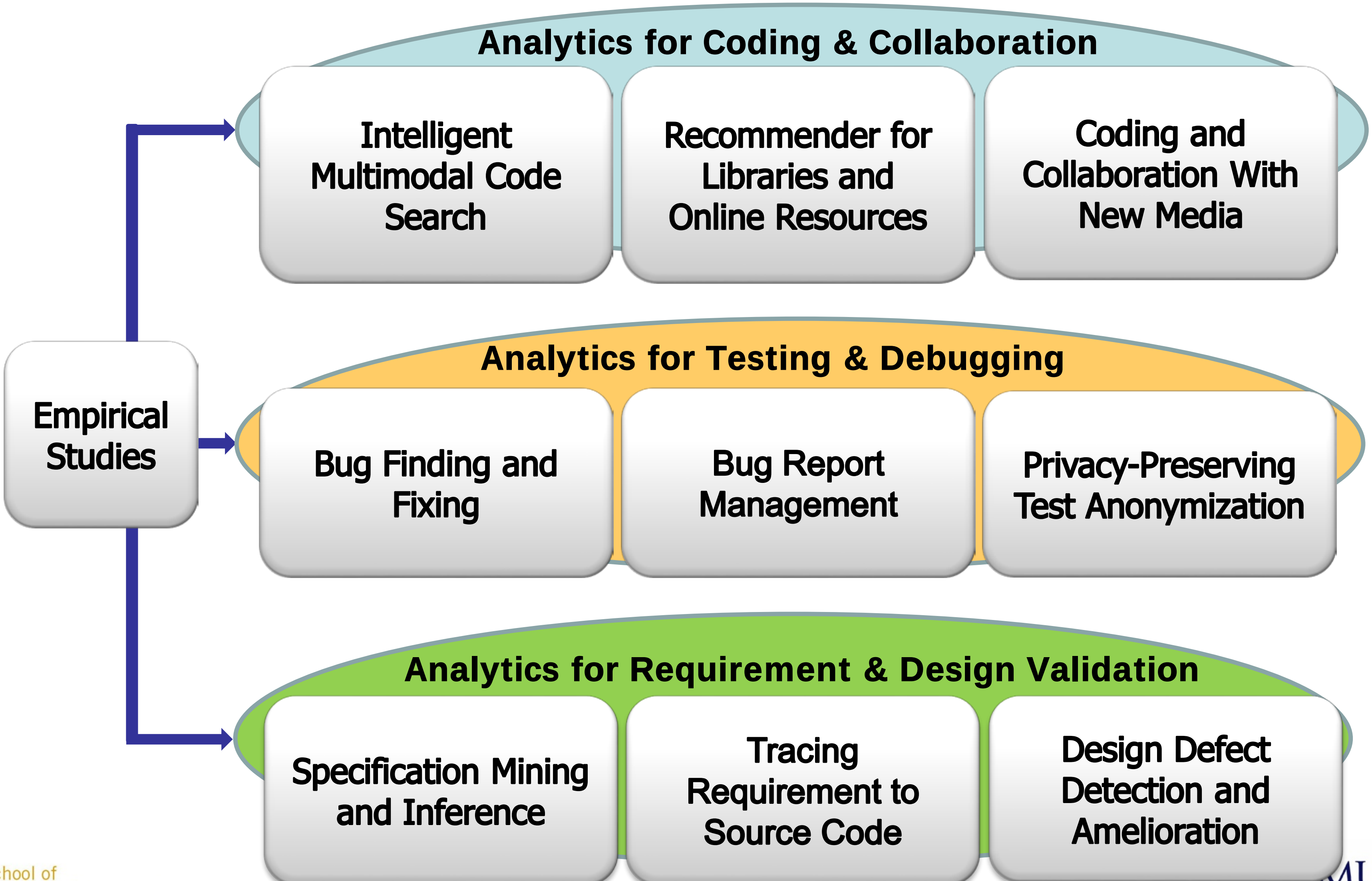
State of Research @ SMU

- Data Collection:
  - Observatory of trends in software related microblogs. ASE 2012
  - Automatic classification of software related microblogs. ICSM 2012

State of Research @ SMU

- Dealing with Vocabulary Mismatch:
  - Automated construction of a software-specific word similarity database. CSMR-WCRE 2014.
- Dealing with Privacy Concern:
  - kb-anonymity: a model for anonymized behaviour-preserving test and debugging data. PLDI 2011.

Summary: Software Analytics



Summary: Future Directions

Millions of Projects
Millions of Bug Reports
Millions of Repositories
Millions of Blogs and Posts
Thousands of APIs
....

Thank you!

Questions? Comments? Advice?

davidlo@smu.edu.sg