Integrating Defect Data, Code Review Data, and Version Control Data for Defect Analysis and Prediction

Tao C. Lee

DS Lab, EPFL
MS thesis defence, Lausanne, Switzerland
{tao.lee}@epfl.ch

September 11, 2013

Honeywell

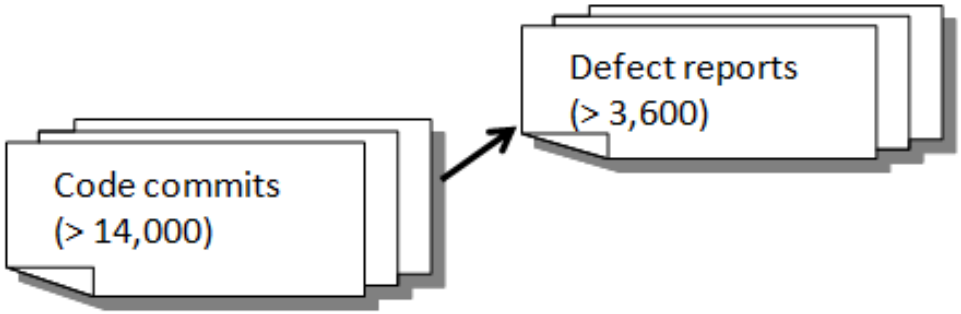
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

A software manager's view of defects

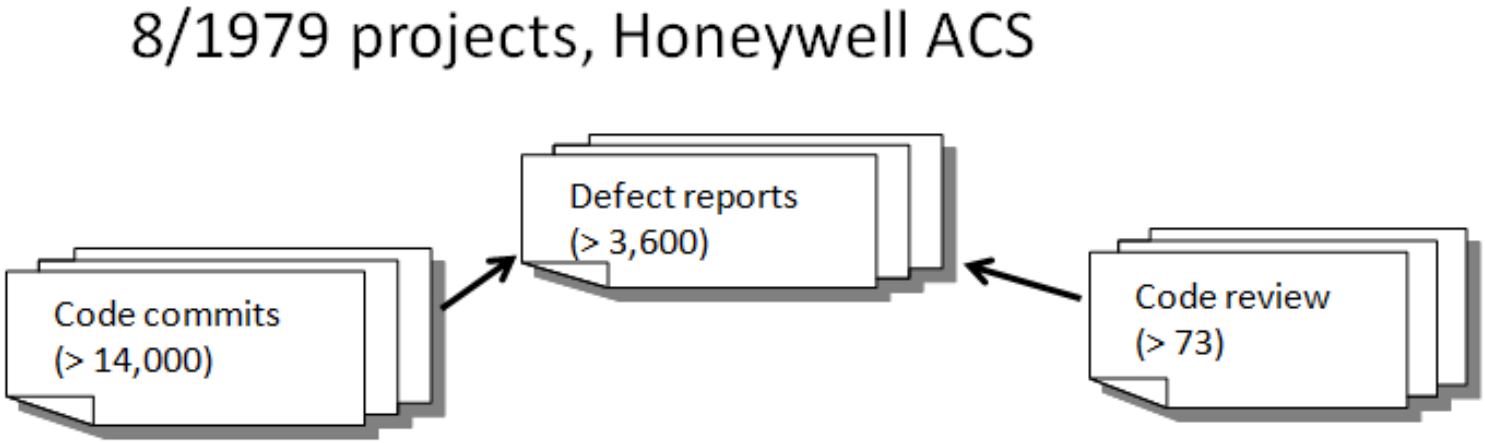
8/1979 projects, Honeywell ACS

A software manager's view of defects

8/1979 projects, Honeywell ACS

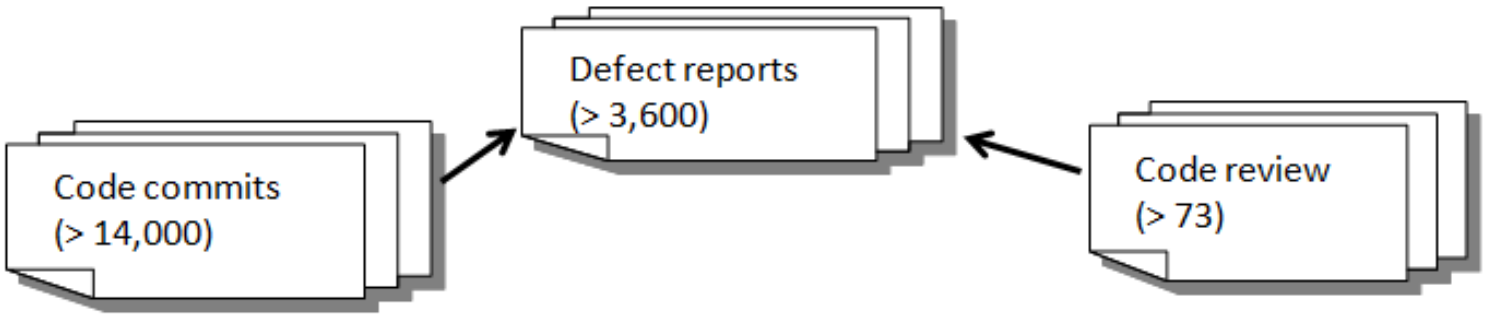


A software manager's view of defects



A software manager's view of defects

8/1979 projects, Honeywell ACS



How to make sense of this huge amount of information?

A software manager's view of defects

- Diversified projects: PLs (C, Java), applications (server, control)

A software manager's view of defects

- Diversified projects: PLs (C, Java), applications (server, control)
- Code reviews are rare

A software manager's view of defects

- Diversified projects: PLs (C, Java), applications (server, control)
- Code reviews are rare
- Defect types are often unknown

A software manager's view of defects

- Diversified projects: PLs (C, Java), applications (server, control)
- Code reviews are rare
- Defect types are often unknown
- Prediction of hot files of defects is of interest

A software manager's view of defects

- Diversified projects: PLs (C, Java), applications (server, control)
- Code reviews are rare
- Defect types are often unknown
- Prediction of hot files of defects is of interest
- Methodology must apply to large defect data of diversified projects

Outline

- Prior work

Outline

- Prior work
- Research contributions

Outline

- Prior work
- Research contributions
- Methodology of indexing defects by keywords

Outline

- Prior work
- Research contributions
- Methodology of indexing defects by keywords
- Applications of indexing defects by keywords

Outline

- Prior work
- Research contributions
- Methodology of indexing defects by keywords
- Applications of indexing defects by keywords
- Evaluation

Outline

- Prior work
- Research contributions
- Methodology of indexing defects by keywords
- Applications of indexing defects by keywords
- Evaluation
- Conclusions & future work

Prior work

Research with different emphases on defect data

Research	Method [a]	Defect type [b]	R [c]	C [d]
Chillarege92	M	function, interface, etc.	Y	N
Chou01	A	null, range, float, lock, etc.	N	Y
Nagappan05	M	N	N	Y
Tan07	A	copy-and-paste	N	Y
Mantyla09	M	functional, evolvable	Y	N
Yin11	A	fixes	N	Y

[a] Methods of defect finding: manual reviews or using automatic tools
[b] Defects are classified by types in research
[c] Code reviews are used in research
[d] Code commits are used in research

Prior work

Research with different emphases on defect data

Research	Method [a]	Defect type [b]	R [c]	C [d]
Chillarege92	M	function, interface, etc.	Y	N
Chou01	A	null, range, float, lock, etc.	N	Y
Nagappan05	M	N	N	Y
Tan07	A	copy-and-paste	N	Y
Mantyla09	M	functional, evolvable	Y	N
Yin11	A	fixes	N	Y

[a] Methods of defect finding: manual reviews or using automatic tools
[b] Defects are classified by types in research
[c] Code reviews are used in research
[d] Code commits are used in research

But these methods cannot solve our problem!

Research contributions

- Methodology of indexing defects by keywords

Research contributions

- Methodology of indexing defects by keywords
- Applications of indexing defects by keywords
  - What are the defect types?

Research contributions

- Methodology of indexing defects by keywords
- Applications of indexing defects by keywords
  - What are the defect types?
  - Do defects cluster in files?

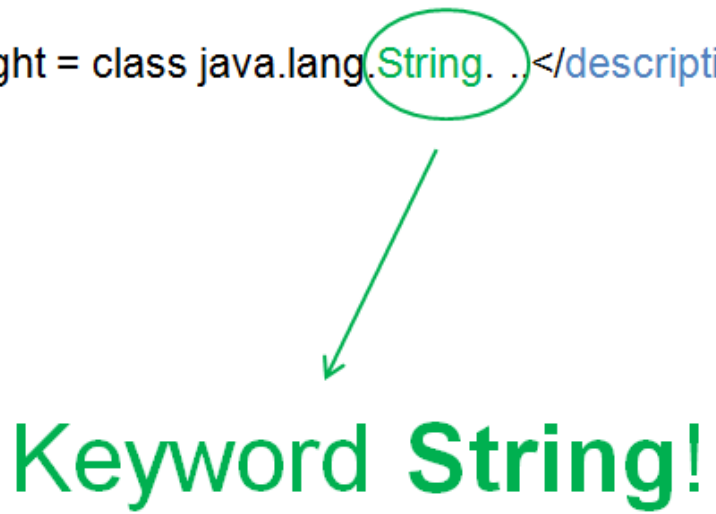
Research contributions

- Methodology of indexing defects by keywords
- Applications of indexing defects by keywords
  - What are the defect types?
  - Do defects cluster in files?
  - How to predict the hot files of defects?

Methodology of indexing defects by keywords

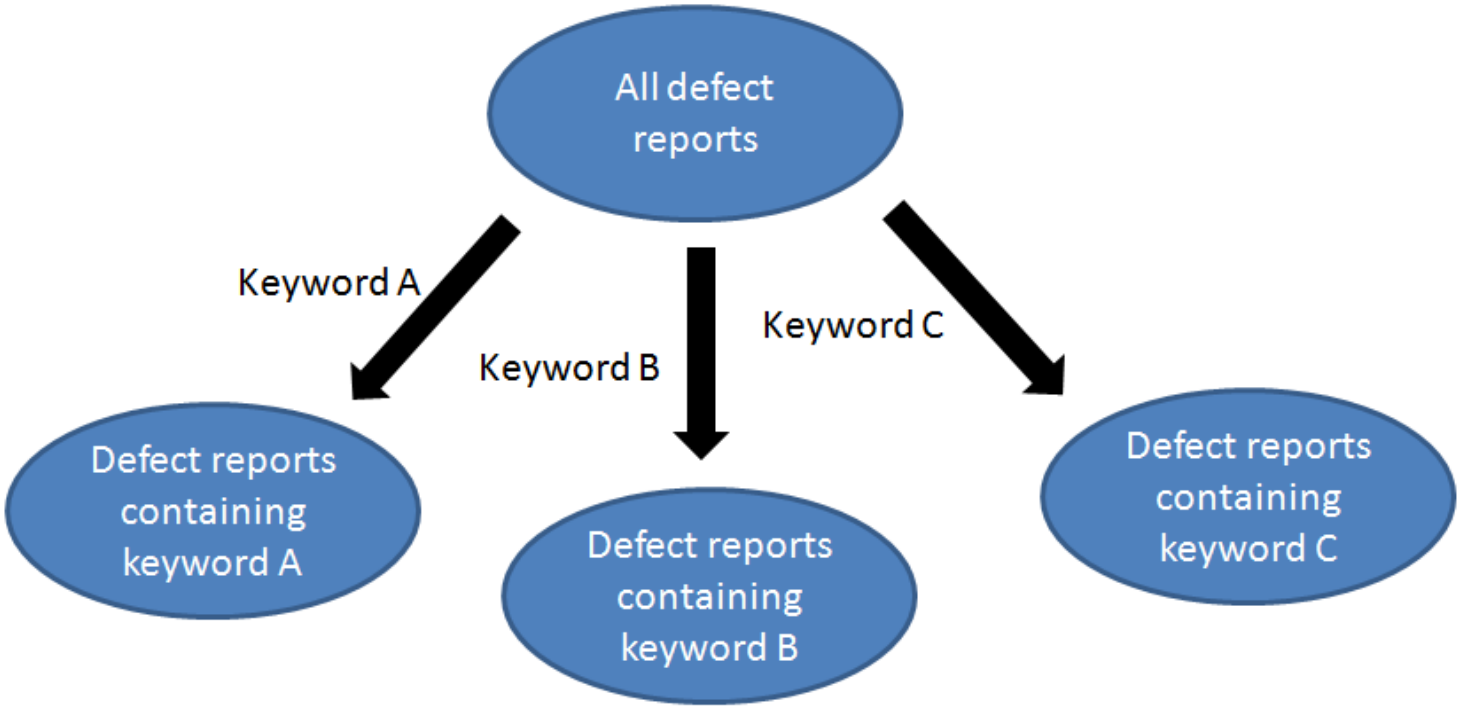
Keywords are found in defect reports ...

```
<item>
   <title> [AJE-105] Exception in Defect Scorecard </title>
   <summary> Exception in Defect Scorecard </summary>
   <description> … Currently left = class java.lang.Long, right = class java.lang.String. ..</description>
   <key id=" 215630 "> AJE-105 </key>
   <status id="6"> Closed </status>
   <resolution id="1"> Fixed </resolution>
   <assignee username="eXX">XXX</assignee>
   <reporter username="eYY">YYY</reporter>
   <comments>
     <comment> XXXX</comment>
     <comment> YYYY </comment>
   </comments>
</item>
```



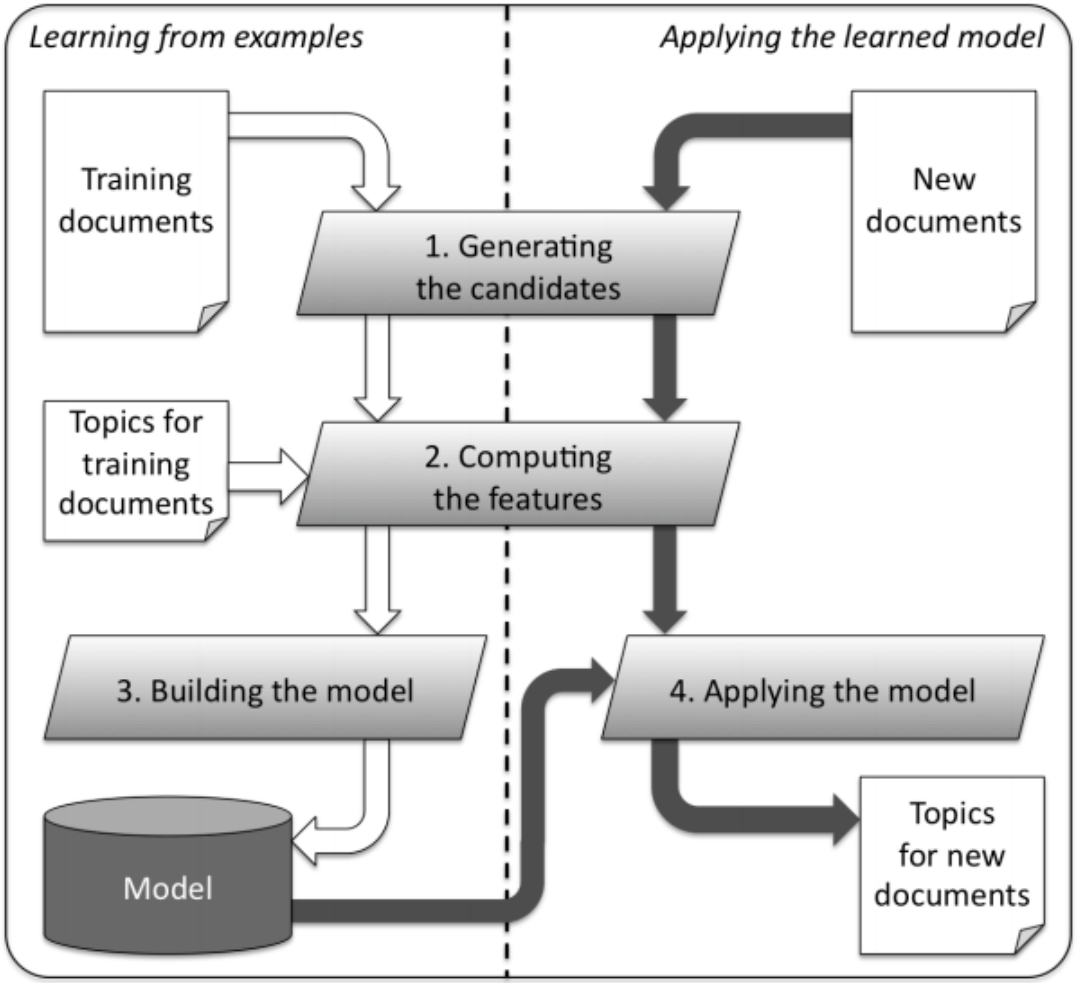
Methodology of indexing defects by keywords

Defect reports can be indexed by keywords ...



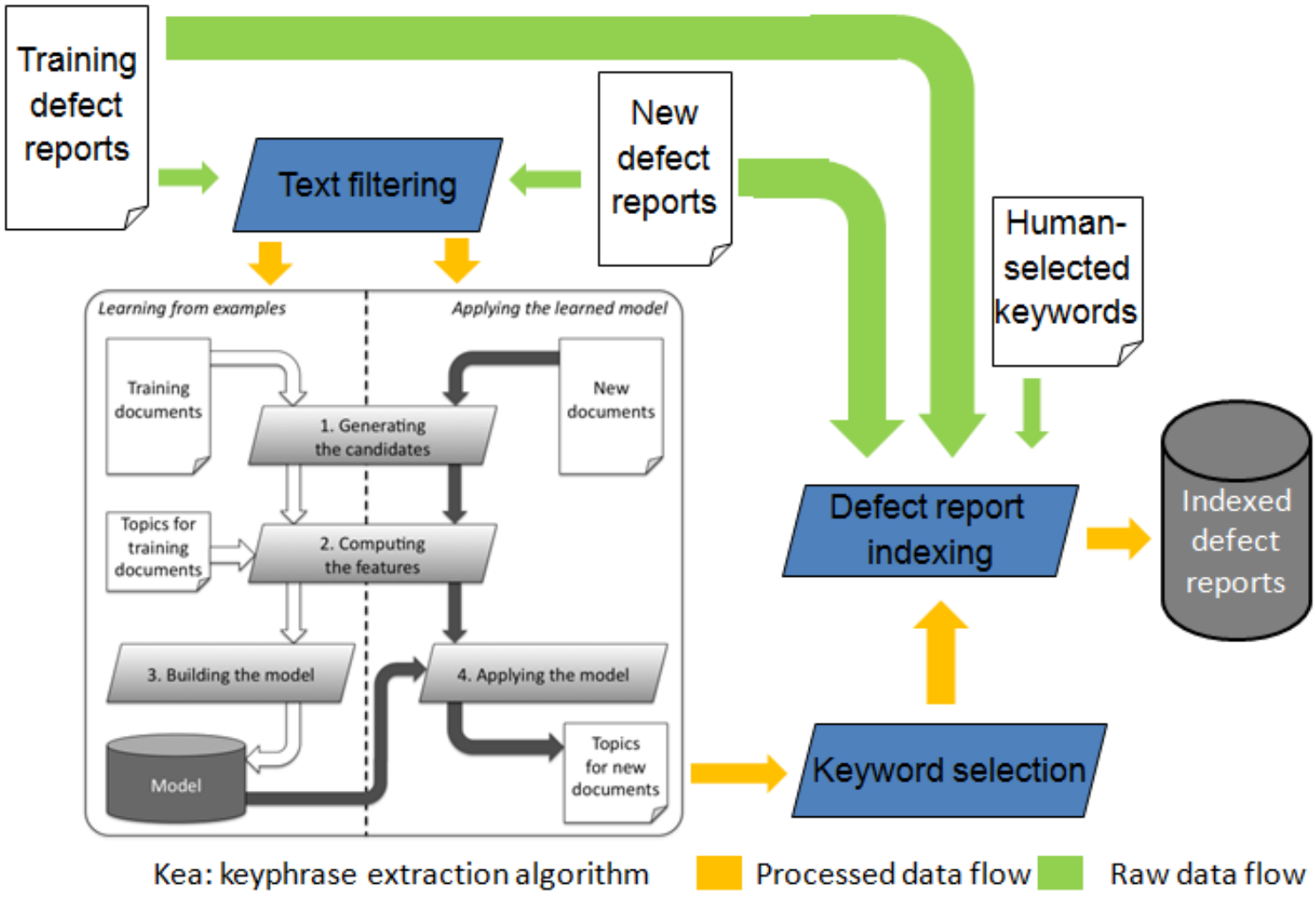
Methodology of indexing defects by keywords

Kea, a Keyphrase Extraction Algorithm [Medelyan09]



Methodology of indexing defects by keywords

Integration of Kea into the framework ...



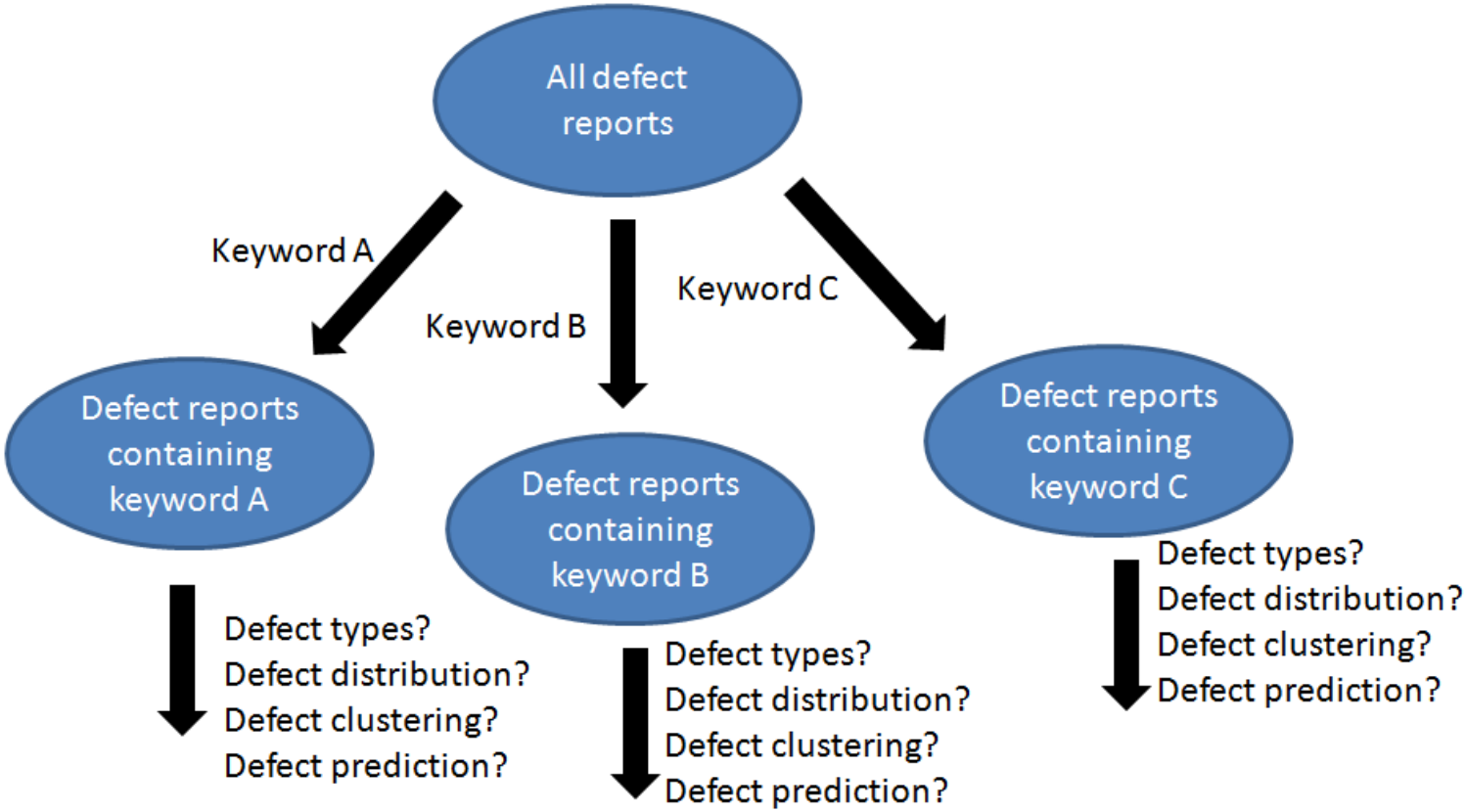
Methodology of indexing defects by keywords

An example of defects indexed by the keyword **memory**

Defects indexed by keywords	
Keyword	Defect summary
memory	Bad responce to HVAC Counter
	Rheem Furnace Model and Serial Number are cleared ...
	ECM Blower problem after User menu update
	no D4 alarm when memory card is invalid and data ...
	d1 alarm active even if memory card with correct ...
	Overflow of array index
	ESD Failure of Memory Card Connector
	RAM usage
	Bad SSD display when is fault clearing

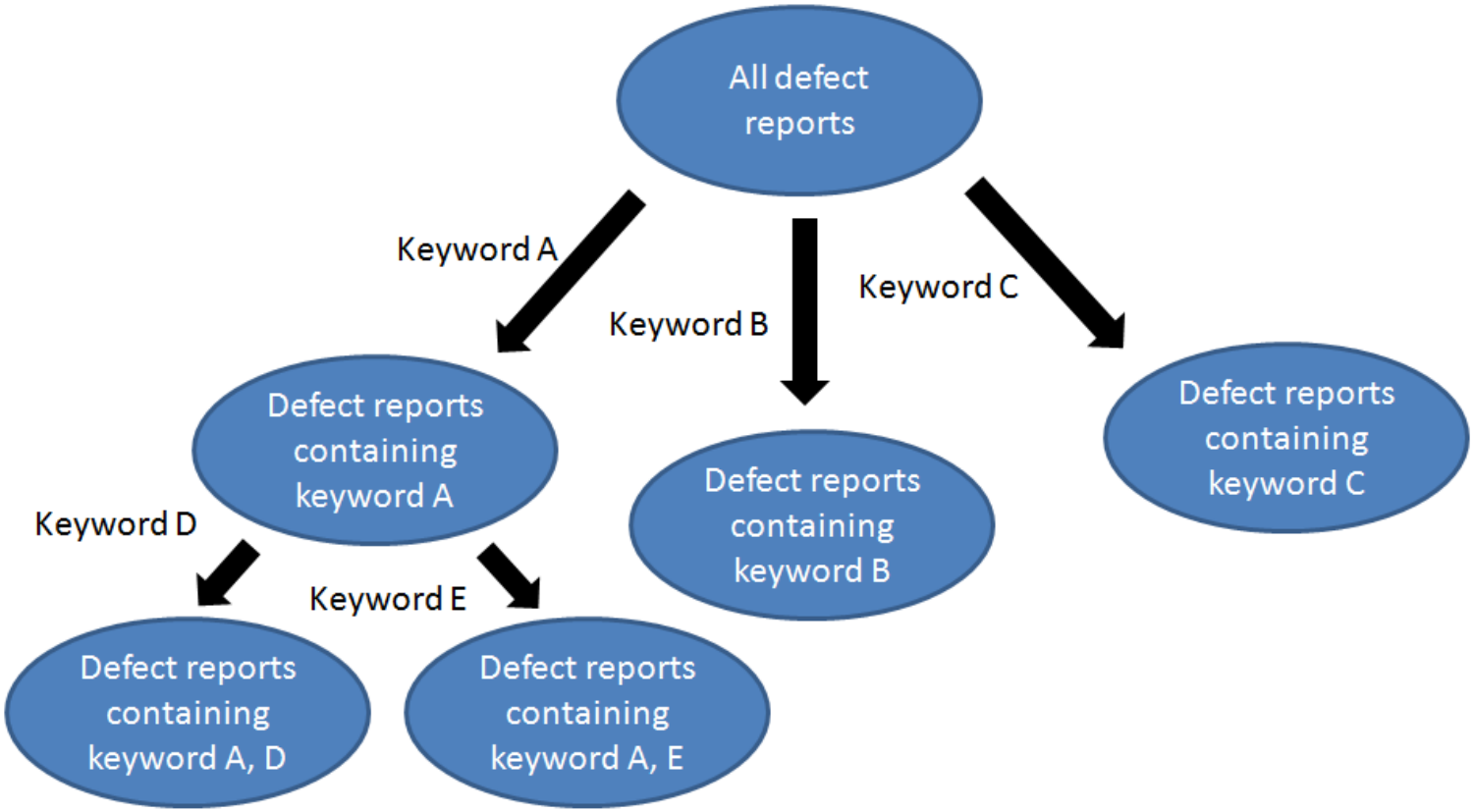
Applications of indexing defects by keywords

After defect indexing ...



What are the defect types?

Defect types are represented by (multiple) keywords ...



What are the defect types?

Algorithm 5 Inferring defect types by keywords
Input: a list of defect reports and a list of defect patterns
Output: a map of defect types with the associated list of defect reports

Require: x is a list of filtered defect reports, p is a list of defect patterns

```
1:  function InferringDefectTypesByKeywords(x, p)
2:      z ← new Map for storing indexed defects
3:      for x_i : x do
4:          maxFreq ← 0
5:          leastFirstPos ← -1
6:          defectType ← null
7:          for p_j : p do
8:              if ContainAllKeywords(x_i, p_j) then
9:                  fs ← ComputeGeneralizedFrequencyScore(p_j, x_i)
10:                 fps ← ComputeGeneralizedFirstPositionScore(p_j, x_i)
11:                 if maxFreq ≤ fs then          ▷ determine the dominant pattern
12:                     maxFreq ← fps
13:                     defectType ← p_j
14:                     leastFirstPos ← fps
15:                 else if fs = maxFreq then
16:                     if fps ≤ leastFirstPos then
17:                         defectType ← p_j
18:                         leastFirstPos ← fps
19:         if defectType ≠ null then
20:             z ← z.put(defectType, x_i)
21:     return z ▷ z is a map of defect types with the associated list of defect reports
```

Defect clustering

$$Fraction80Metric = \frac{\text{number of files containing 80\% of defects}}{\text{total number of files}}$$

number of files are counted from the files with the most defects.

How to predict the hot files of defects?

$$BugPredMetric = \sum_{i=1}^{N} \frac{1}{1 + e^{slope \cdot t_i}} \text{ [Google11]}$$

where

slope is a configuable parameter with default value 12.

N is the number of commits of a file.

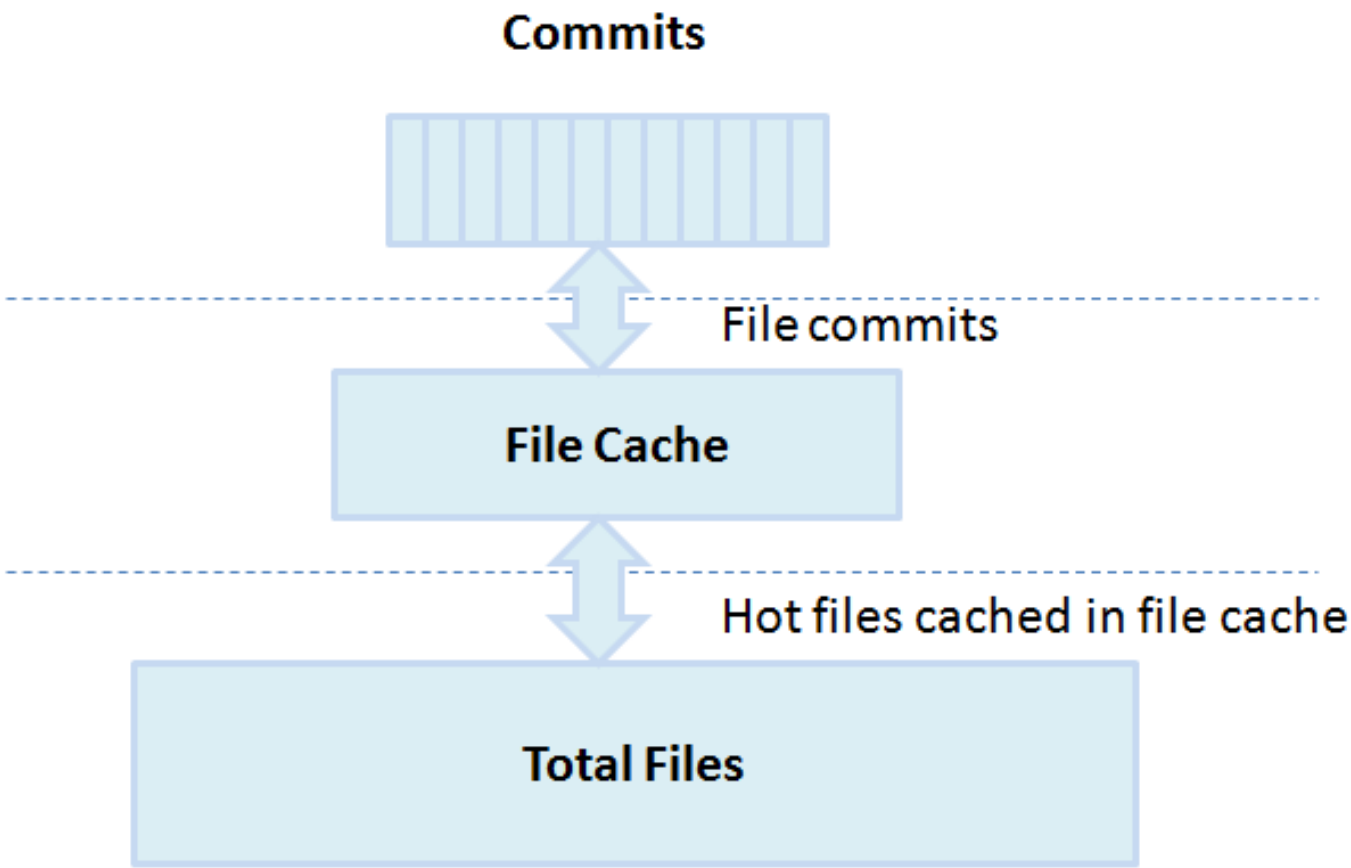
t_i is the normalized timestamp of the ith commit,
with the oldest timestamp $= 1$ and now $= 0$.

$$\text{defect hit rate} = hit/(hit + miss)$$

hit/*miss* is the number of times a committed file is (/not) in the file cache

How to predict the hot files of defects?

File cache simulation



How to predict the hot files of defects?

Algorithm 10 Defect hot spots prediction in files
Input: a list of file commits, a file cache size, a slope parameter
Output: computed defect hit rate of the file cache replacement policy using a known bug prediction metric, a list of hot files in the file cache

Require: x is a list of file commits, c is the size of file cache, s is the slope parameter

```
1: function DefectHotSpotsPredictionInSourceFiles(x, c, s)
2:     hit ← 0
3:     miss ← 0
4:     defectHitRate ← 0.0
5:     cache ← New Set for storing files
6:     x ← SortCommitsInChronologicalOrder(x)
7:     for x_i : x do
8:         f ← ListOfCommitedFiles(x_i)
9:         for f_j : f do
10:            if cache contains f_j then
11:                hit ← hit + 1
12:            else
13:                miss ← miss + 1
14:            h ← UpdateCommitedFiles(f_j)
15:            r ← RankFilesByBugPredScore(h, s)   ▷ Compute BugPredMetric for
    each file, and sort the files in descending order
16:            cache ← UpdateFileCacheByScore(r, c) ▷ Keep the top c files in cache
17:    defectHitRate ← hit / (hit+miss)
18:    Output(cache)                                ▷ output the files in the file cache
19:    return hitRate
```

Evaluation

Project	Description	PL	Defects	Commits
Project X	a JIRA server extension	Java	423	3235
Project Y	Gas ignition controls	C	300	2977

Project X

Project	Description	PL	Defects	Commits
Project X	a JIRA server extension	Java	423	3235
Project Y	Gas ignition controls	C	300	2977

Project X

Table: Fraction of defects by defect types for project X (total 423 defects)

Defect type (keyword)	Fraction of defects
layout	1.42%
string	1.89%
javascript	1.89%
reproduce	4.49%
message	5.91%
configuration	6.15%
display	9.93%
panel	11.58%
dialog	13.71%
report	15.13%
Unindexed defects	27.19%

Project X

Table: *Fraction80Metric* by defect types for project X

Defect type (keyword)	Number of files	*Fraction80Metric*
All defects	1375	42.76%
layout	17	35.29%
string	9	55.56%
javascript	15	20.00%
reproduce	17	29.41%
message	35	28.57%
configuration	16	31.25%
display	65	24.62%
panel	42	30.95%
dialog	51	41.18%
report	56	48.21%

Project X

Table: Top-10 hot files for project X

atlassian-plugin.xml
ContourProjectManagerImpl.java
component-wise-issue-chart.js
changeHistory.js
ComponentWiseIssueReportResource.java
defect-resolution-time-chart.js
DefectResolutionTimeReportResource.java
chart.js
ComponentWiseIssueReport.java
ChangeHistoryTableImpl.java

Project X

Table: Defect hit rate by defect types for project X

file cache size = 20, and slope = 12		
Defect type (keyword)	Files	Defect hit rate
All defects	1375	30.66%
layout	17	100.00%
string	9	100.00%
javascript	15	100.00%
reproduce	17	100.00%
message	35	95.61%
configuration	16	100.00%
display	65	90.60%
panel	42	90.08%
dialog	51	86.23%
report	56	91.10%

Project X

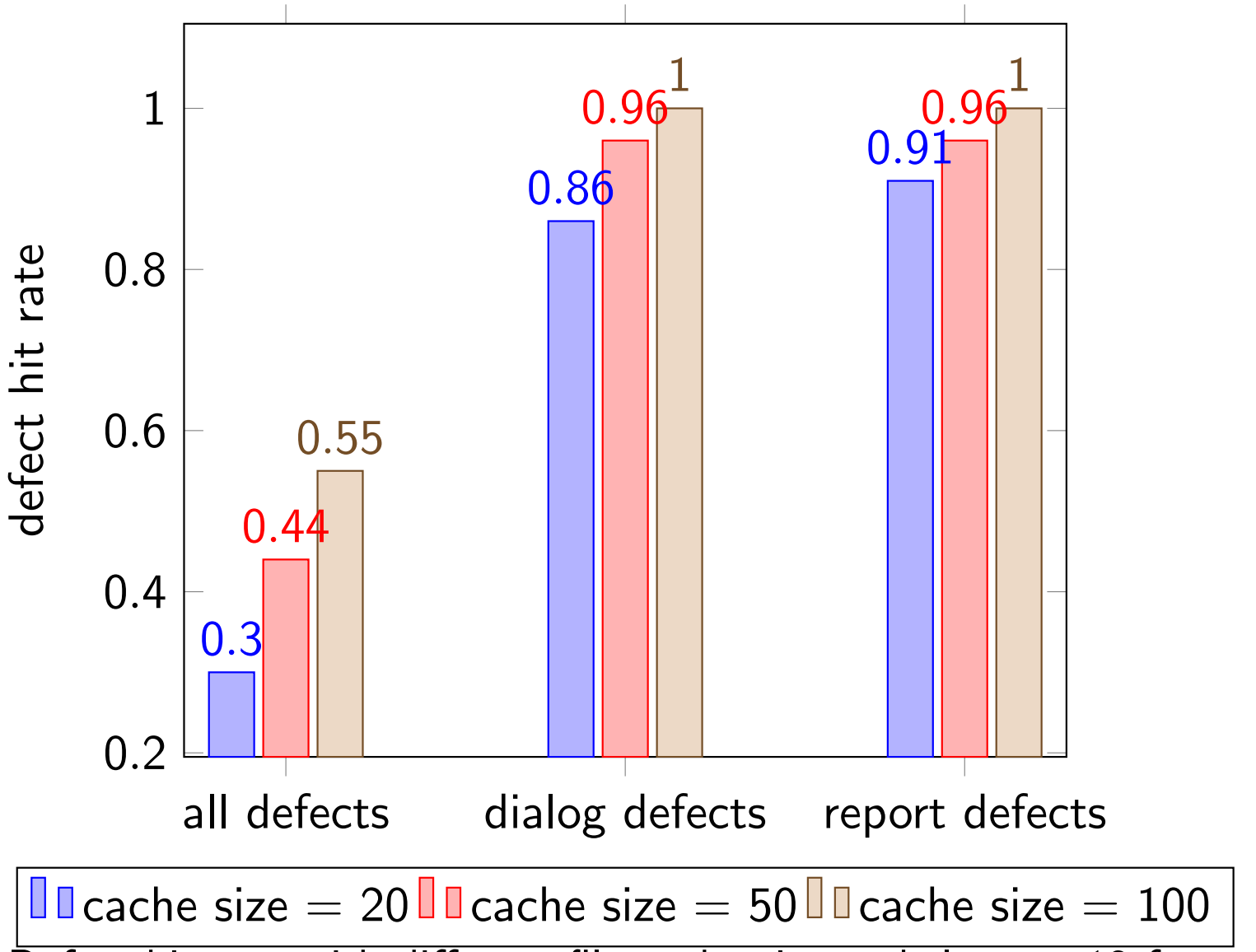


Figure: Defect hit rate with different file cache size and slope = 12 for project X

Project Y

Project	Description	PL	Defects	Commits
Project X	a JIRA server extension	Java	423	3235
Project Y	Gas ignition controls	C	300	2977

Project Y

Table: Fraction of defects by defect types for project Y (total 300 defects)

Defect type (keyword)	Fraction of defects
print	0.33%
string	1.00%
memory	1.67%
fan	2.67%
cool	3.00%
power	5.67%
lock	6.33%
circulator	7.00%
alarm	11.67%
heat	22.33%
Unindexed defects	38.33%

Project Y

Table: *Fraction80Metric* by defect types for project Y

Defect type (keyword)	Number of files	*Fraction80Metric*
All defects	2736	7.27%
string	25	56.00%
fan	30	73.33%
cool	50	64.00%
power	18	33.33%
lock	53	52.83%
circulator	76	44.74%
alarm	64	34.38%
heat	104	55.77%

Project Y

Table: Top-10 hot files for project Y

application.c
ign_interface.c
osdata.h
IgnSupport.c
RN_ObjectProc.c
LED_UserInterface.c
error.c
appscheduler.c
ModelDataCheck.c
RN_FaultsSetup.h

Project Y

Table: Defect hit rate by defect types for project Y

file cache size = 20, and slope = 12		
Defect type (keyword)	Files	Defect hit rate
All defects	1375	22.37%
string	25	88.25%
memory	3	100.00%
fan	30	86.59%
cool	50	90.04%
power	18	100.00%
lock	53	85.79%
circulator	76	81.32%
alarm	64	87.78%
heat	104	69.71%

Project Y

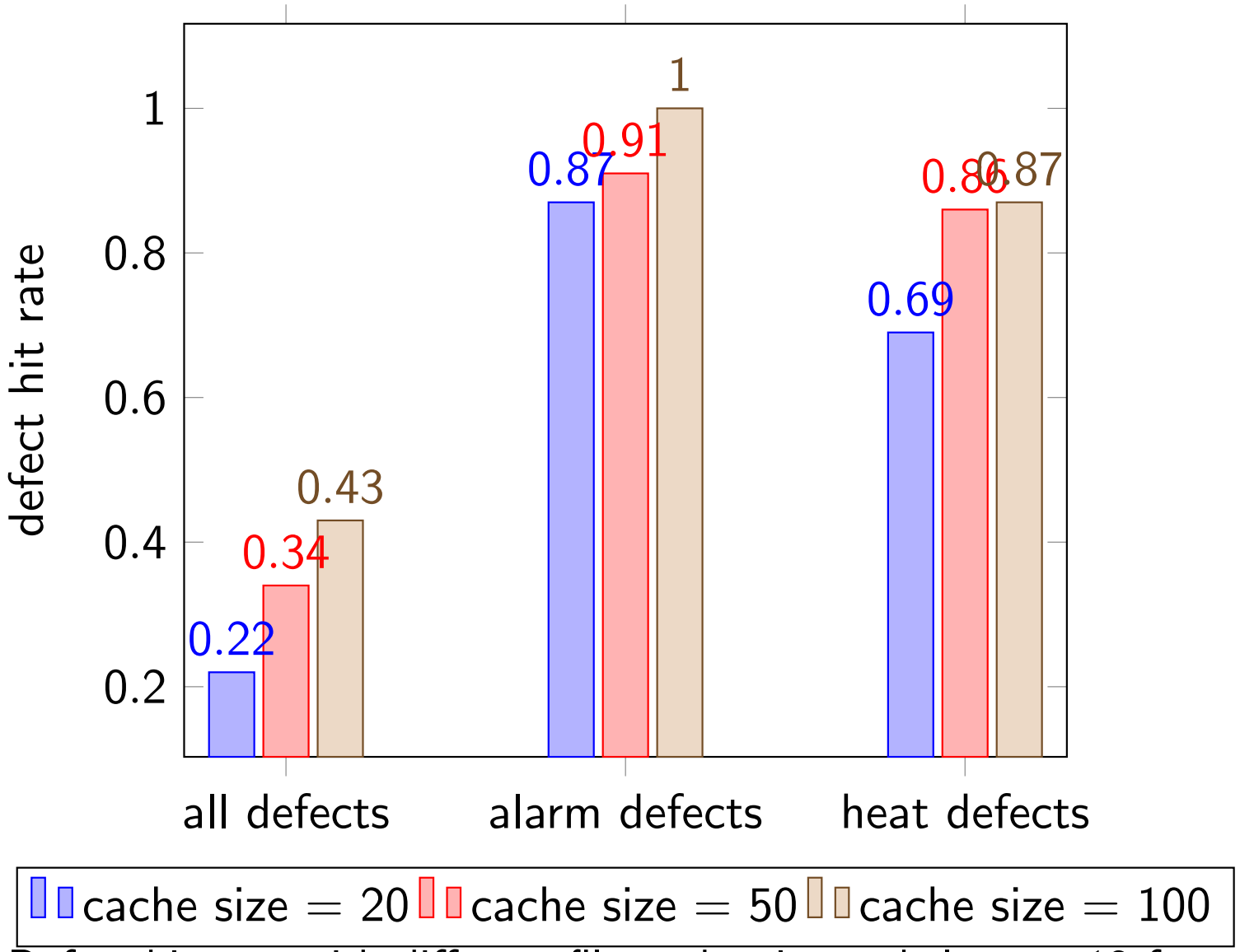


Figure: Defect hit rate with different file cache size and slope = 12 for project Y

Discussion

What are the defect types? Infer them by keywords.

Project	Defect types (keywords)
Project X	**layout**, **string**, **javascript**, **reproduce**, **message**, **configuration**, **display**, **panel**, **dialog**, **report**, <10% non-functional defects indexed, 27% defects unindexed.
Project Y	**print**, **string**, **memory**, **fan**, **cool**, **power**, **lock**, **circulator**, **alarm**, **heat**, <10% non-functional defects indexed, 38% defects unindexed.

Discussion

Do defects cluster in files? YES!

Project	Defect clustering
Project X	For all defect types 20%-50% of files contained more than 80% of defects.
Project Y	For most defect types 20%-50% of files contained more than 80% of defects.

Discussion

What are the top files to review?

Table: Top files to review for projects X and Y

Project	Hot files of defects
Project X	atlassian-plugin.xml
	ContourProjectManagerImpl.java
	component-wise-issue-chart.js
	changeHistory.js
	ComponentWiseIssueReportResource.java
Project Y	application.c
	ign_interface.c
	osdata.h
	IgnSupport.c
	RN_ObjectProc.c

Conclusions

Methodology of indexing defects by keywords

- New approach to defect classification

Conclusions

Methodology of indexing defects by keywords

- New approach to defect classification
- Scales to large dataset

Conclusions

Methodology of indexing defects by keywords

- New approach to defect classification
- Scales to large dataset
- Applies to diversified projects (PLs, applications)

Conclusions

Methodology of indexing defects by keywords

- New approach to defect classification
- Scales to large dataset
- Applies to diversified projects (PLs, applications)

Applications of indexing defects by keywords

- Effective inference of a wide range of defect types

Conclusions

Methodology of indexing defects by keywords

- New approach to defect classification
- Scales to large dataset
- Applies to diversified projects (PLs, applications)

Applications of indexing defects by keywords

- Effective inference of a wide range of defect types
- Type-based clustering analysis of defects

Conclusions

Methodology of indexing defects by keywords

- New approach to defect classification
- Scales to large dataset
- Applies to diversified projects (PLs, applications)

Applications of indexing defects by keywords

- Effective inference of a wide range of defect types
- Type-based clustering analysis of defects
- Type-based prediction of the hot files of defects

Future Work

Advanced indexing techniques

- Dictionary-based, semantics-aware indexing?

Future Work

Advanced indexing techniques

- Dictionary-based, semantics-aware indexing?
- How to integrate user tagging?

Future Work

Advanced indexing techniques

- Dictionary-based, semantics-aware indexing?
- How to integrate user tagging?

Towards dynamic defect analysis

- Would the dominant defect type vary through time?

Future Work

Advanced indexing techniques

- Dictionary-based, semantics-aware indexing?
- How to integrate user tagging?

Towards dynamic defect analysis

- Would the dominant defect type vary through time?
- What would be the implications of defect evolution to testing?

Future Work

Advanced indexing techniques

- Dictionary-based, semantics-aware indexing?
- How to integrate user tagging?

Towards dynamic defect analysis

- Would the dominant defect type vary through time?
- What would be the implications of defect evolution to testing?

Integrating defect data online: JIRA plugin

- How to optimize the performance of the tool?

Future Work

Advanced indexing techniques

- Dictionary-based, semantics-aware indexing?
- How to integrate user tagging?

Towards dynamic defect analysis

- Would the dominant defect type vary through time?
- What would be the implications of defect evolution to testing?

Integrating defect data online: JIRA plugin

- How to optimize the performance of the tool?
- How to collect more complete defect dataset?

Thank you, questions please.

{tao.lee}@epfl.ch

Honeywell

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE