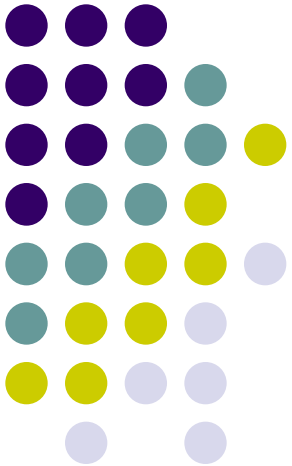
ASE2008

Automated Aspect Recommendation through Clustering-Based Fan-in Analysis

Danfeng Zhang, Yao Guo, Xiangqun Chen

Institute of Software, Peking University

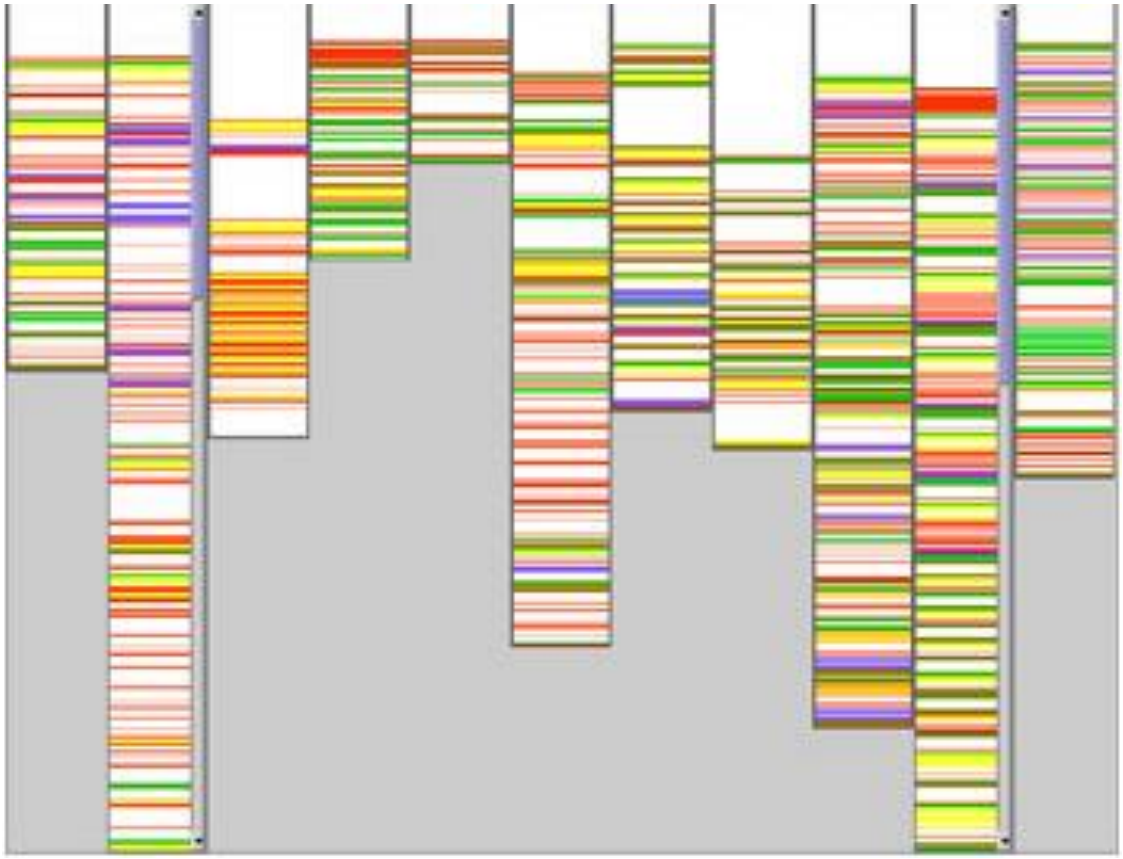


Talk Outline

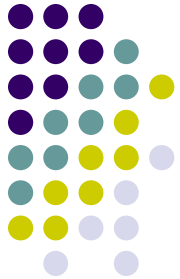
- Background
- Motivation
- Clustering-Based Fan-in Analysis (CBFA)
- Evaluation
- Conclusion

Crosscutting Concern (CCC)

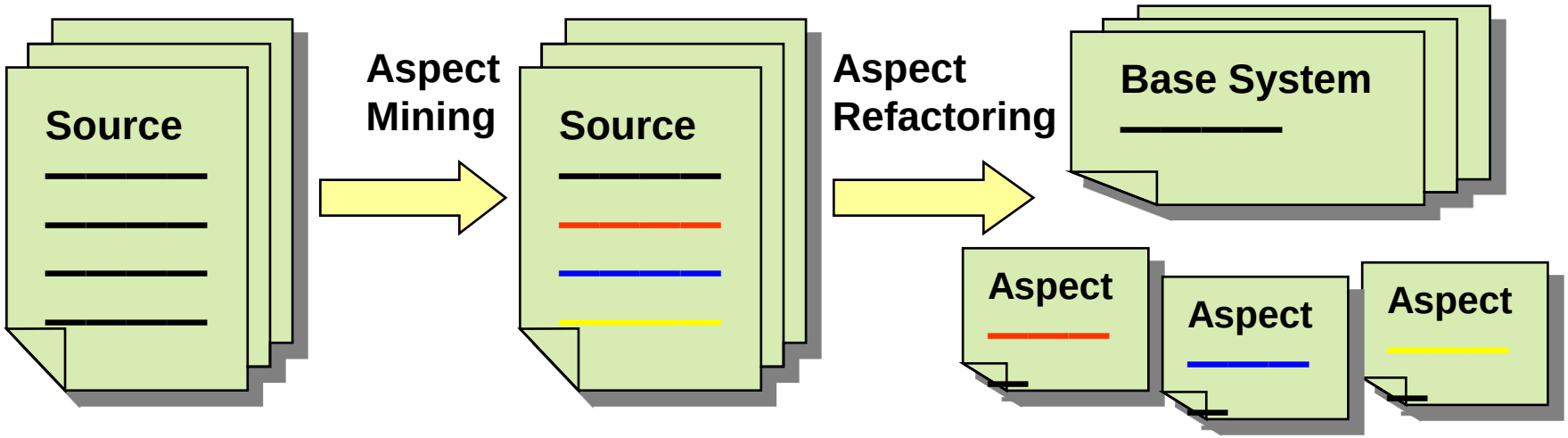
- CCCs in ASML (a software component consisting of 19,000 lines of C code) [M. Bruntink *et. al.* 2004]



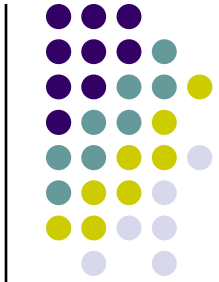
Aspect-Oriented Programming



- To encapsulate the CCCs into Aspects
- ***Aspect Mining*** → Refactoring

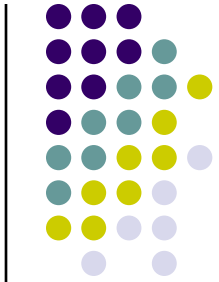


Background



- Goal: ***Apply AOP to the Linux system***
- Our previous work: a case study of aspect mining in Linux
  - Applied several existing approaches to identify the CCCs in Linux [APSEC 2007]
  - Techniques evaluated: fan-in analysis, clone detection
- This paper: Clustering-Based Fan-in Analysis
  - A new aspect mining approach to improve mining results
  - Applicable for both C and Java

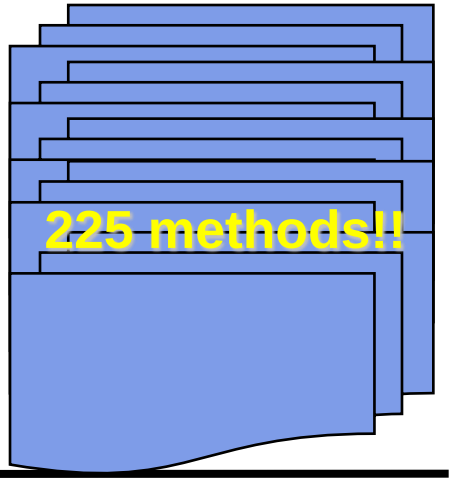
Motivation



- Fan-in analysis [M. Marin *et. al*, 2004]
  - Key idea
    - CCCs are usually implemented using ***single methods***, which may be called from ***numerous places*** in the code
    - Frequently called methods are likely to be a CCC
  - Fan-in value of a method *m*
    - The number of distinct method bodies that can invoke *m*
  - Return methods whose fan-in is larger than *a predefined threshold* as the mining results
    - A threshold of 10 is suggested

Performance of fan-in analysis

Method Name	Fan-in
atomic inc	41
atomic dec	20
atomic_set	15
atomic_read	13
ATOMIC_INIT	11
atomic_add	7
atomic_dec_and_test	7
atomic_add_negative	3
atomic_sub	2
atomic_sub_and_test	1
atomic_inc_and_test	1

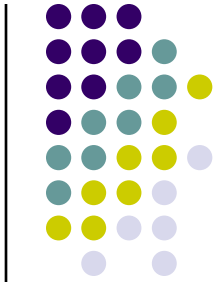


Threshold : 10

- Require huge effort to group a concern
- Tend to miss small fan-in ones

Atomic Lock Concern

Our solution

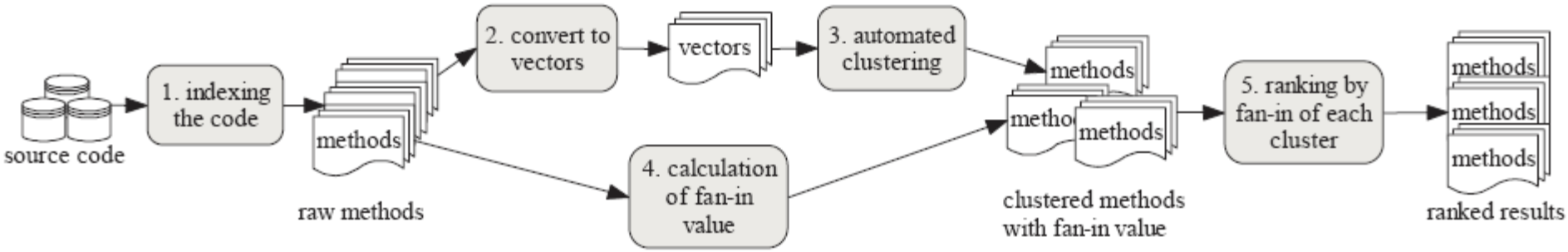


- Clustering-Based Fan-in Analysis (CBFA)

- Key Approaches
  - A new clustering based mining technique to group the method automatically
    - Incorporated text mining mechanisms from the AI field
  - A new ranking metric (*cluster fan-in*) to provide better aspect recommendation
    - instead of using cluster sizes as in most existing approaches

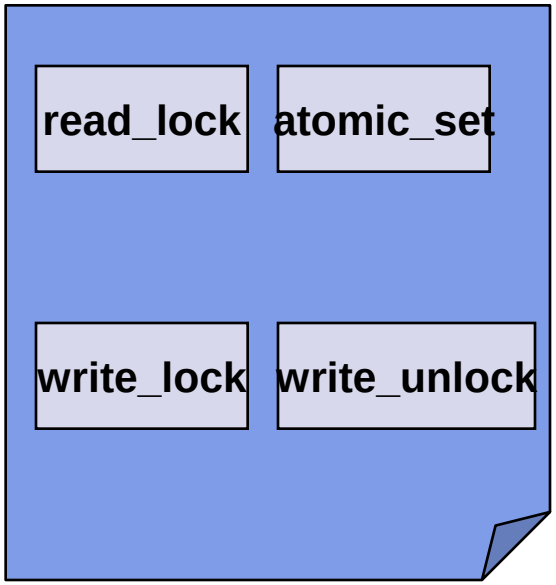
Clustering Based Fan-in Analysis (CBFA)

- Technique overview
  - Method retrieval
  - Vector representation
  - Clustering
  - Fan-in value calculation
  - Ranking and return final results



Method Retrieval

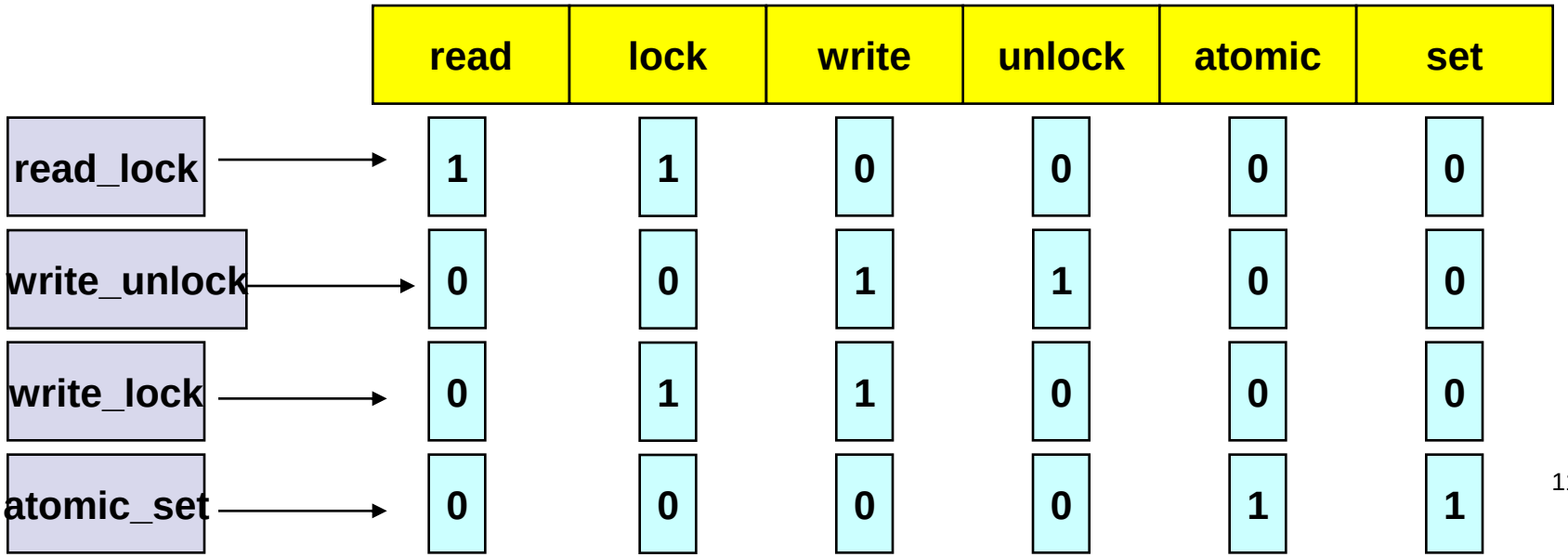
- Only method names (including function-like macros in C) need to be retrieved.



Vector Representation

- Convert each method name into a vector
  - Split into tokens (base on naming convention)

    read_lock → read lock; nextFigure → next figure
  - Use all available tokens as dimensions
  - The corresponding field is set to 1 if the method name contains a certain word



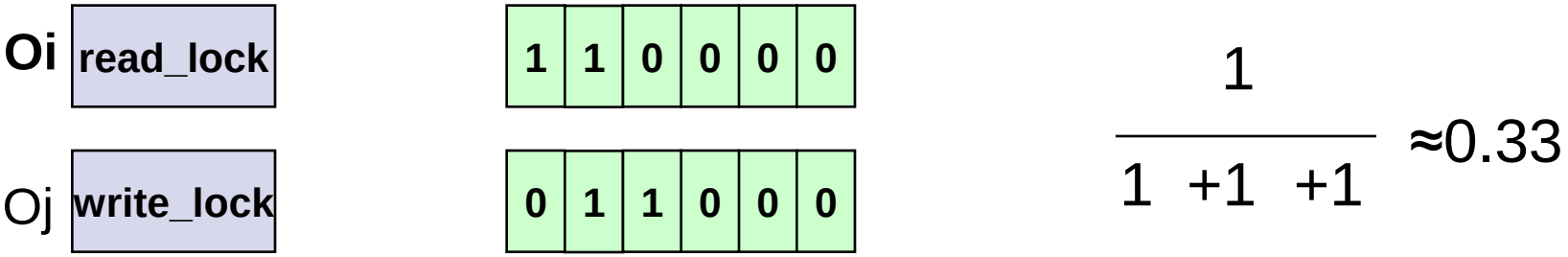
Clustering

- Many existing similarity metrics
  - Euclid distance
  - Cosine distance
  - …
- However,
  - They normally treat ‘0’s and ‘1’s equally
  - Our model is asymmetric
    - Many ‘1’ in common → similar
    - Many ‘0’ in common → meaningless

Clustering

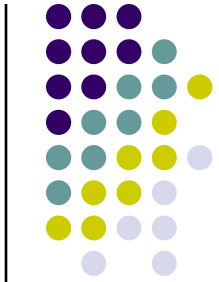
- Similarity Criteria used in our approach
  - *Jaccard Coefficient*

$$J(O_i, O_j) = \frac{N_{ij}}{N_i + N_j + N_{ij}}$$



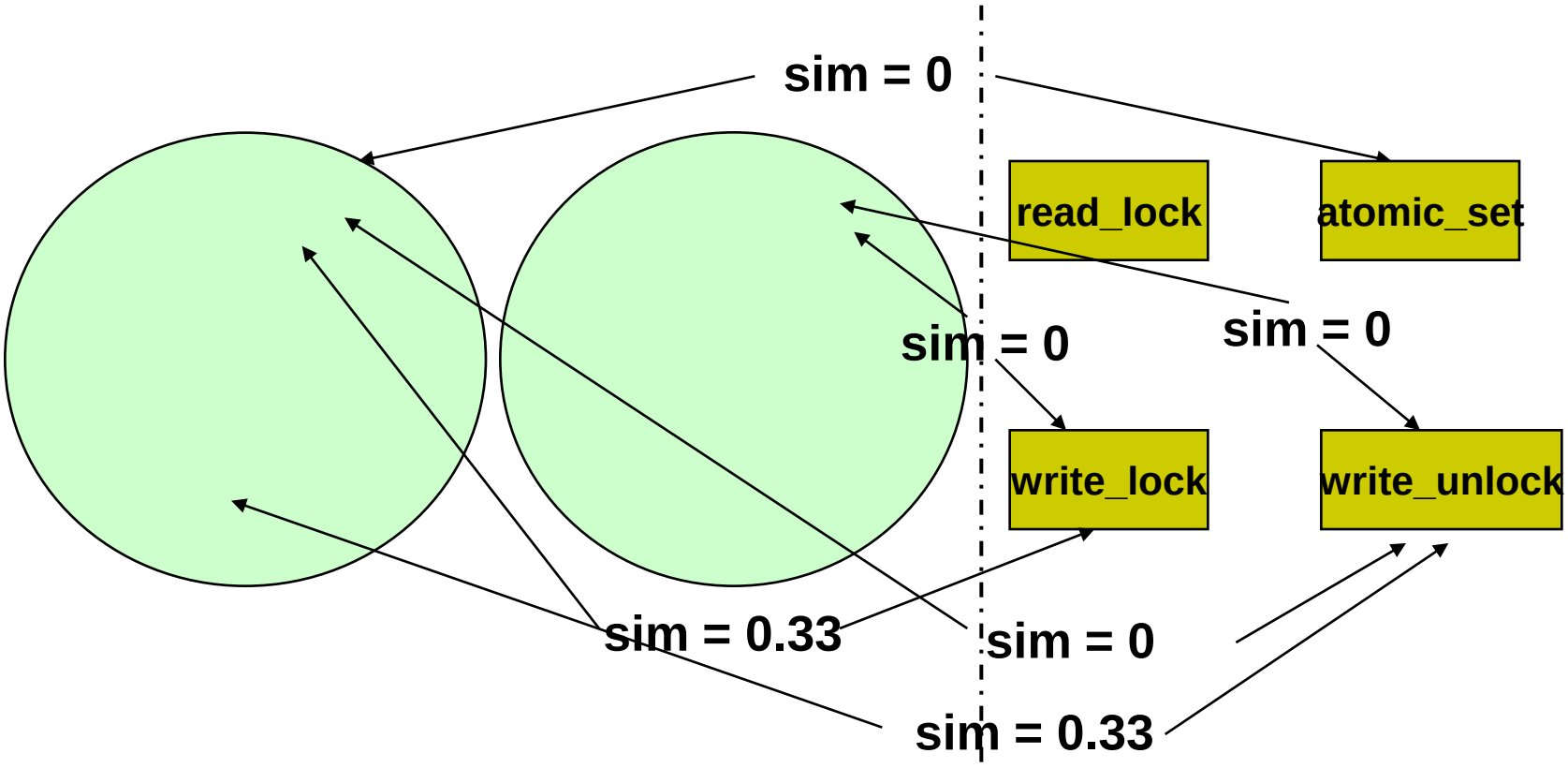
$$\frac{1}{1 + 1 + 1} \approx 0.33$$

Clustering



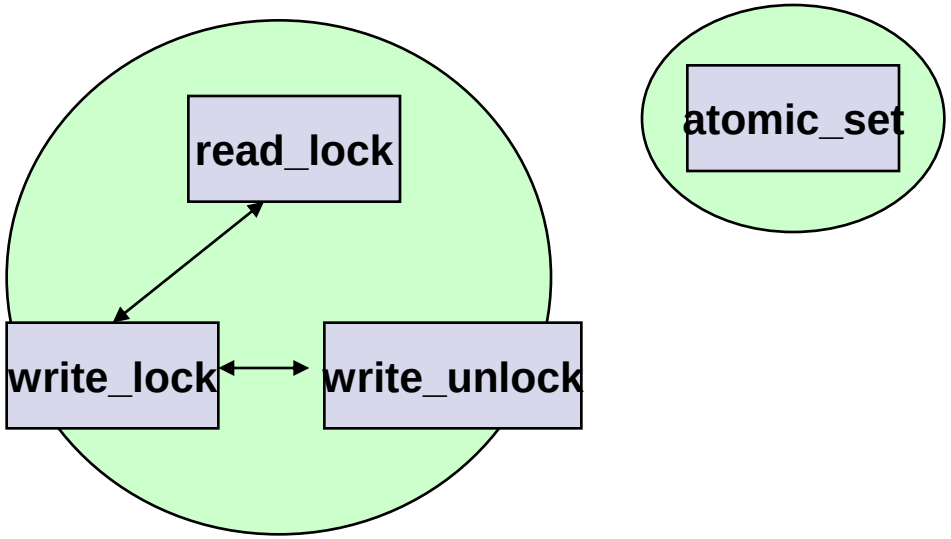
- Also many existing algorithms
  - k-means
  - Hierarchical Agglomerative Clustering Algorithm (HACA)
  - …
- Problem
  - Hard to decide the optimal cluster numbers in advance
- Our approach
  - a simple *heuristic* approach
  - Set *simMin=0.3* (minimal similarity that two methods are grouped)

Clustering - Example



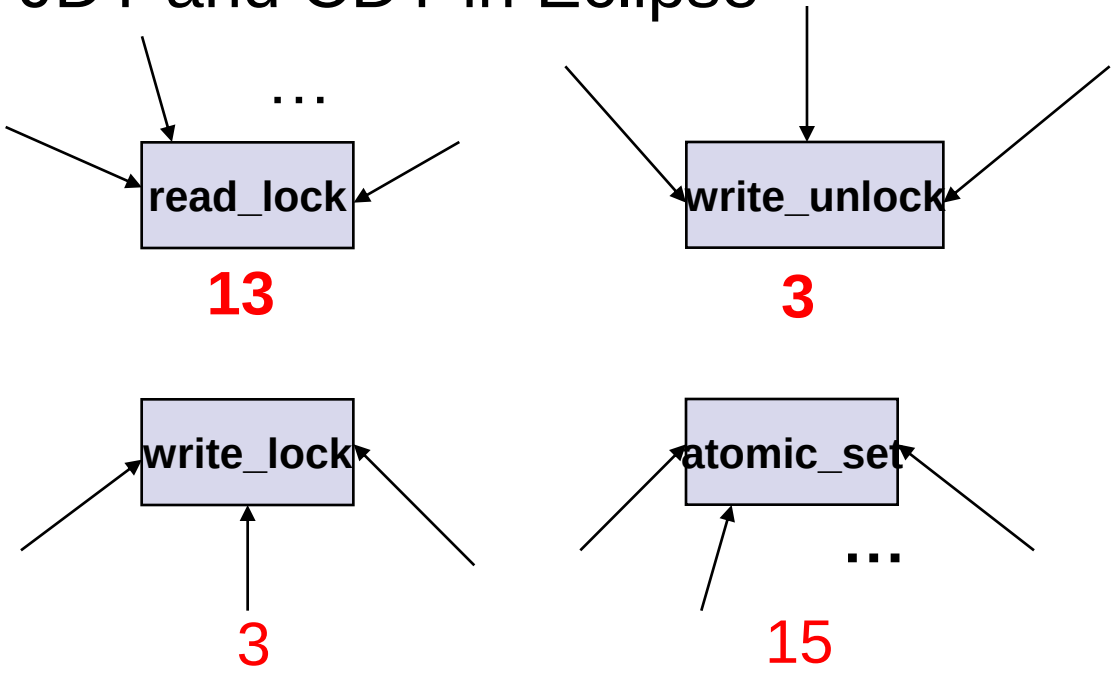
Clustering

- Properties of our clustering approach
  - Similar methods are automatically grouped into same clusters
  - Dissimilar, but related ones can also be automatically grouped



Fan-in Value Calculation

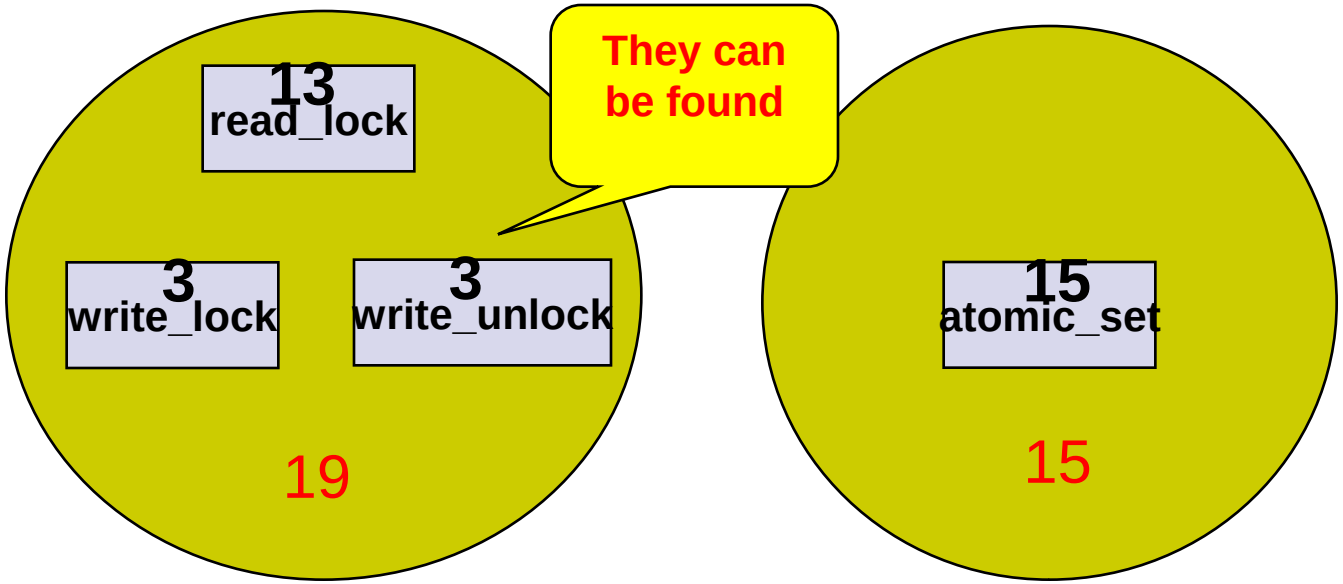
- *Java*: the definition used in original fan-in analysis
- *C*: consider function-like macros as well as functions
- The calculation is straightforward with the help of JDT and CDT in Eclipse



Ranking

- Fan-in value is still a good metric
  - Stands for “popularity” and “significance”
- We are concerned with the “popularity” of a ***concern***
- Rank them by ***cluster fan-in***

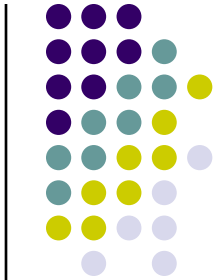
$$F(C) = \sum f(m_i), m_i \in C$$



Evaluation

- Metrics
  - Concern Coverage
    - The rate of methods in a certain concern can be found
  - True Positives
    - The rate of methods that are truly related to a CCC in the recommendation results
  - Concern Coverage is more important
- Systems
  - Java: *JHotDraw 5.4b* (12K LOC)
  - C: *Linux 2.4.18* (84K LOC)

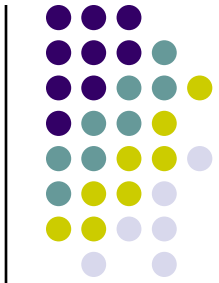
Techniques Compared



- Fan-in analysis
  - The publicly available tool *FINT* is used [M. Marin et. al. 2004]

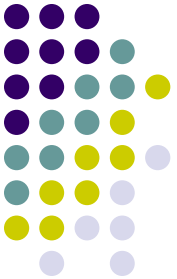
- Identifier analysis [T. Tourwe et. al. 2004]
  - Also a mining approach provides grouped results
  - Filter out clusters whose size is smaller than a certain threshold (normally 10)
  - We implemented a prototype tool ourselves

Techniques Compared



- Dynamic analysis [P. Tonella et. al. 2004]
  - Key idea
    - Use the trace file to group related methods
  - The *Dynamo* aspect mining tool is used

Top-Down Approach



- Performance on several well-known CCCs
  - JHotDrow

CCC	Related methods
Undo(7)	*undo, redo, execute, pushUndo, popUndo, pushRedo, popRedo*
Observer (5)	*addFigureChangeListener, removeFigureChangeListener, willChange, changed, figureChanged*
Iterator(6)	*next, hasNext, nextFigure, hasNextFigure, nextHandle, hasNextHandle*
Visitor(4)	*visit, visitFigure, visitHandle, visitFigureChangeListener*
Persistence (16)	*read, readStorable, readString, readInt, readLong, readColor, readDouble, readBoolean, write, writeStorable, writeString, writeInt, writeLong, writeColor, writeDouble, writeBoolean*

Top-Down Approach

- Performance on several well-known CCCs
  - Synchronization concerns in Linux

Mechanisms	Related functions
atomic operation (11)	**ATOMIC_INIT atomic_read atomic_set** *atomic_add atomic_sub atomic_dec atomic_add_negative atomic_sub_and_test atomic_inc atomic_dec_and_test atomic_inc_and_test*
spin lock (11)	*spin_lock spin_trylock spin_unlock* **spin_lock_init spin_is_locked spin_lock_irqsave spin_lock_irq spin_lock_irqrestore spin_unlock_irq spin_unlock_irqsave spin_unlock_irqrestore**
read/write spin lock (15)	*read_lock write_lock* **read_unlock write_unlock read_lock_irq read_lock_irqsave read_unlock_irq read_unlock_irqstore write_lock_irq write_lock_irqsave write_unlock_irq write_unlock_irqrestore write_trylock rw_lock_init rw_is_locked**
big kernel lock (5)	*lock_kernel unlock_kernel* **kernel_locked release_kernel_lock reacquire_kernel_lock**

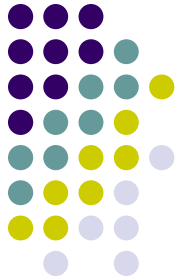
Top-Down Approach

- Results in JHotDraw

Concern	Concern Coverage				True Positives			
	CBFA	Fan	Iden	Dyn	CBFA	Fan	Iden	Dyn
Undo	86%	43%	86%	57%	100%	N/A	50%	64%
Observer	80%	100%	60%	40%	86%	N/A	73%	62%
Iterator	100%	83%	0%	0%	100%	N/A	N/A	NA
Visitor	100%	0%	0%	75%	86%	N/A	N/A	50%
Persistence	100%	37%	100%	44%	80%	N/A	75%	70%
Average	93%	53%	49%	43%	90%	74%	66%	62%

Reason
The size of Iterator
Is only 6

Recommendation Quality



- CBFA rank clusters using *"Cluster Fan-in"*
  - Most current approaches using cluster size as the ranking metric

- An example: How many groups a user needs to examine before finding all 5 CCCs in JHotDraw
  - CBFA: covered in top 42 clusters
  - Identifier analysis: needs to look at 151 groups

Bottom-Up Approach

- To analyze the capability of CBFA to find other CCCs
  - Top 10 recommendations of CBFA are presented and compared to other approaches
  - Only concern coverage is shown

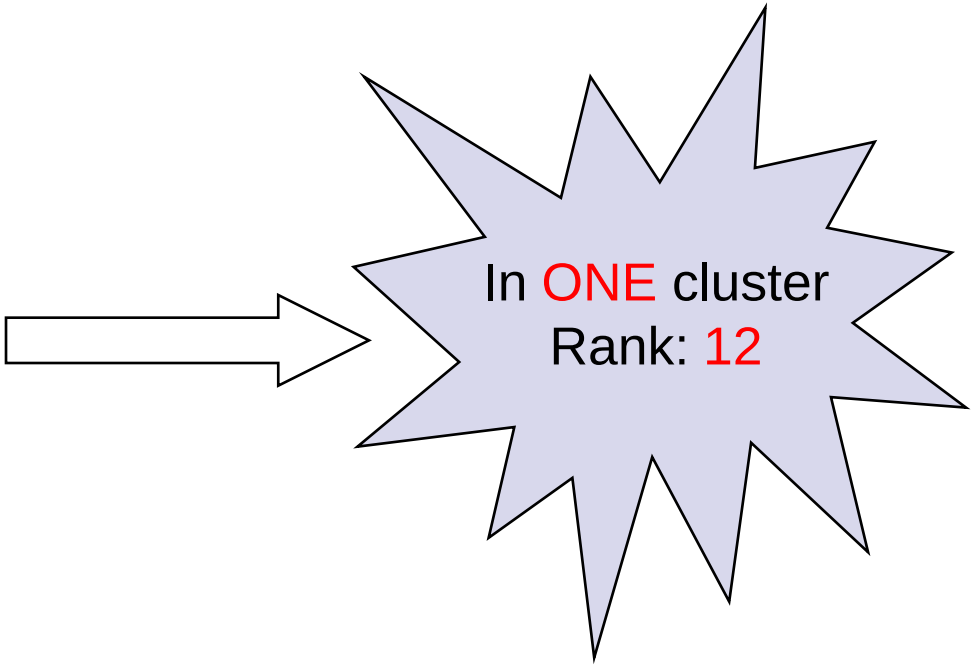
Bottom-Up Approach

- Results in JHotDraw

Concern	Concern Coverage			
	CBFA	Fan	Iden	Dyn
composition	100%	100%	100%	0%
mouse	87%	27%	100%	29%
zoom	100%	0%	0%	0%
factory method	100%	2%	100%	2%
iterator	100%	83%	0%	0%
persistance	100%	37%	100%	44%
undo	86%	43%	86%	57%
manage handle	75%	0%	50%	100%
observer	80%	100%	60%	40%
draw	92%	12%	96%	4%
Average	92%	40%	70%	28%

Example Revisited

Method Name	Fan-in
atomic inc	41
atomic dec	20
atomic_set	15
atomic_read	13
ATOMIC_INIT	11
atomic_add	7
atomic_dec_and_test	7
atomic_add_negative	3
atomic_sub	2
atomic_sub_and_test	1
atomic_inc_and_test	1



Atomic Lock Concern

Conclusion

- An new automated aspect mining approach: CBFA
  - Automatically group methods related to the same crosscutting concern together
  - Recommend aspects based on the *cluster fan-in ranking metric*
  - Applied to two real-life systems
    - Improves aspect mining coverage significantly
    - Provides better recommendation

Thank You !