/* iComment: Bugs or Bad Comments? */

Lin Tan
Ding Yuan
Gopal Krishna
Yuanyuan (YY) Zhou

OPERA group
University of Illinois
at Urbana-Champaign

Motivation

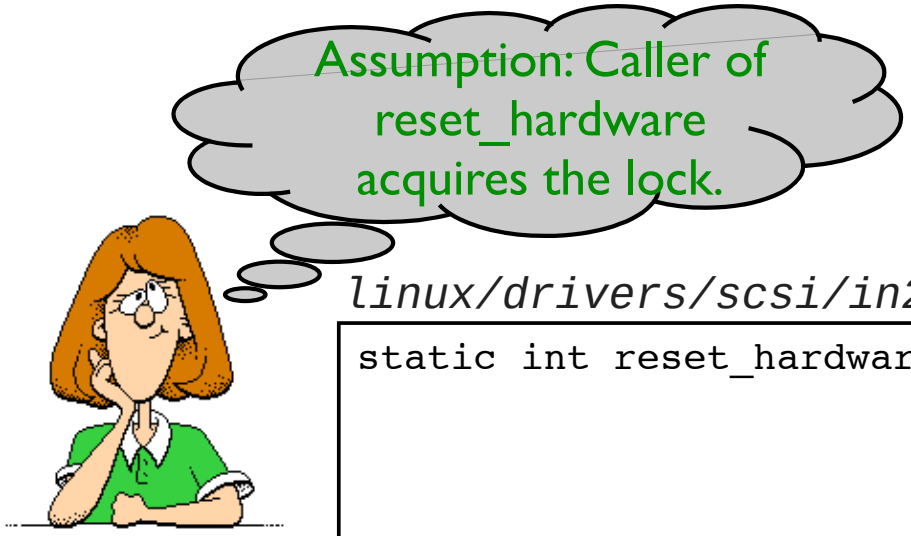
- Software bugs affect reliability.

Motivation

- Software bugs affect reliability.
  - Many due to mismatches between code and programmers' assumptions.

Motivation

- Software bugs affect reliability.
  - Many due to mismatches between code and programmers' assumptions.

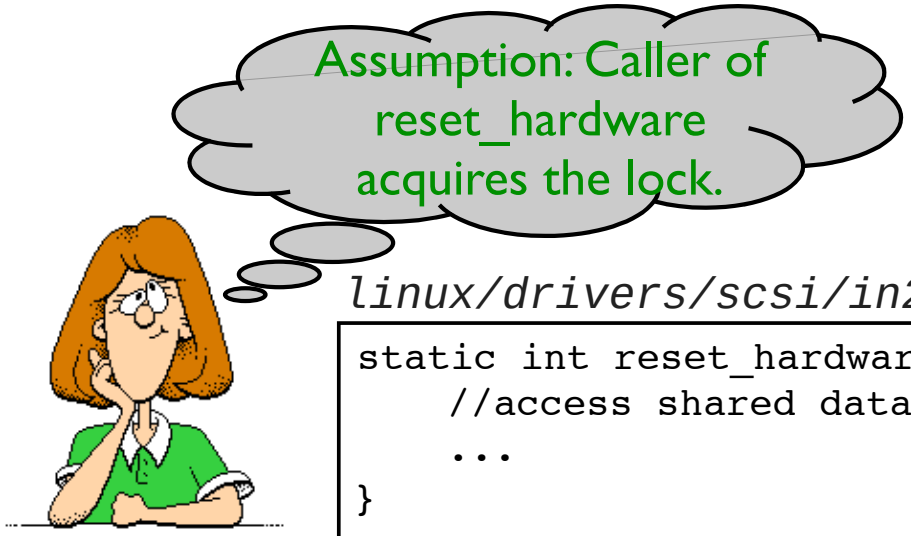


linux/drivers/scsi/in2000.c:

```
static int reset_hardware(…) {...
```

Motivation

- Software bugs affect reliability.
  - Many due to mismatches between code and programmers' assumptions.

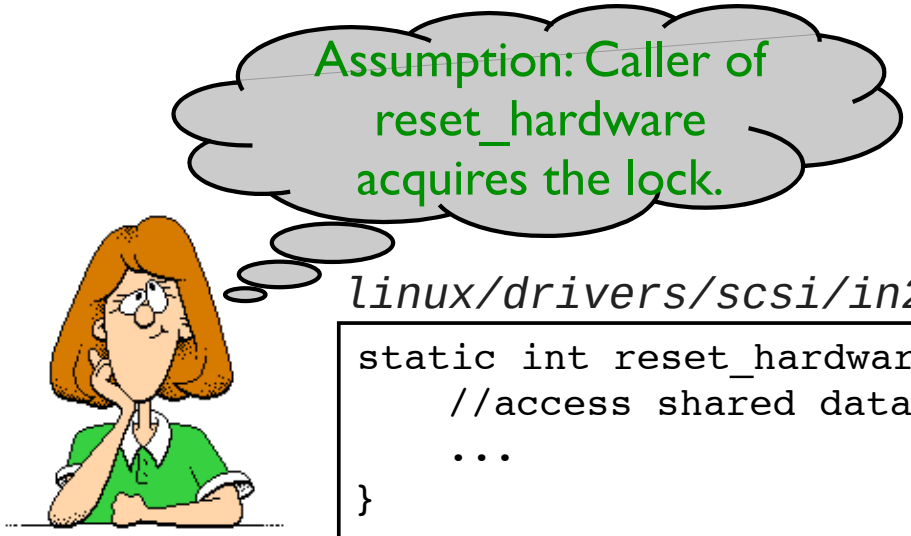


linux/drivers/scsi/in2000.c:

```
static int reset_hardware(…) {...
    //access shared data.
    ...
}
...
```

Motivation

- Software bugs affect reliability.
  - Many due to mismatches between code and programmers' assumptions.

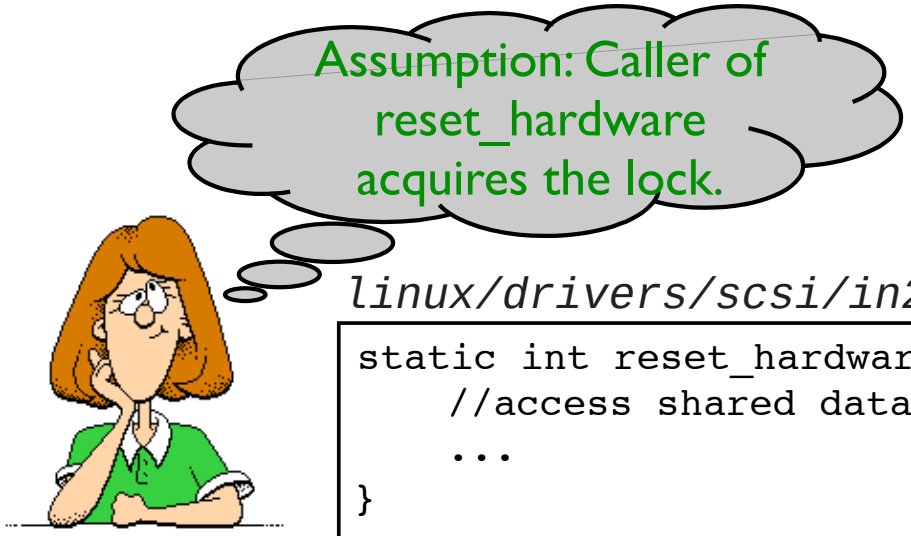


linux/drivers/scsi/in2000.c:

```
static int reset_hardware(…) {...
    //access shared data.
    ...
}
...
static int in2000_bus_reset(…)
{...
    reset_hardware(…);
    ...
}
```

Motivation

- Software bugs affect reliability.
  - Many due to mismatches between code and programmers' assumptions.



linux/drivers/scsi/in2000.c:

```
static int reset_hardware(…) {...
    //access shared data.
    ...
}
...
static int in2000_bus_reset(…)
{...
    reset_hardware(…);
    ...
}
```

No lock acquisition => A bug!

Prevalence of Comments

- Program comments express assumptions.

Prevalence of Comments

- Program comments express assumptions.

linux/drivers/scsi/in2000.c:

```
/* Caller must hold instance lock!*/
static int reset_hardware(…) {…}
```

Prevalence of Comments

- Program comments express assumptions.

```
linux/drivers/scsi/in2000.c:
 /* Caller must hold instance lock!*/
 static int reset_hardware(…) {…}
```

- Millions lines of comments exist in software.

Software	Linux	Mozilla
Lines of code (excluding copyright notices and blank lines)	5.0M	3.3M
Lines of Comment (excluding copyright notices and blank lines)	1.0M	0.51M

Prevalence of Comments

- Program comments express assumptions.

```
linux/drivers/scsi/in2000.c:
 /* Caller must hold instance lock!*/
 static int reset_hardware(…) {…}
```

- Millions lines of comments exist in software.

Software	Linux	Mozilla
Lines of code (excluding copyright notices and blank lines)	5.0M	3.3M
Lines of Comment (excluding copyright notices and blank lines)	1.0M	0.51M

- Comments are not fully utilized yet.
  - Ignored by compilers and bug detection tools.

OPERA

Code vs. Comments

Code	Comment	Implication
Precise	Imprecise	Comments are harder to analyze.

Code vs. Comments

Code	Comment	Implication
Precise	Imprecise	Comments are harder to analyze.
Can be tested	Can NOT be tested	Comments may become less reliable as software evolves.

OPERA

Code vs. Comments

Code	Comment	Implication
Precise	Imprecise	Comments are harder to analyze.
Can be tested	Can NOT be tested	Comments may become less reliable as software evolves.
Harder to understand	Easier to understand	Likely to read comments. Wrong comments mislead programmers.

OPERA

Code vs. Comments

Code	Comment	Implication
Precise	Imprecise	Comments are harder to analyze.
Can be tested	Can NOT be tested	Comments may become less reliable as software evolves.
Harder to understand	Easier to understand	Likely to read comments. Wrong comments mislead programmers.

- Many assumptions are difficult to infer from source code alone.
  - Inferring from source code alone may fail
    - for cases that no (or only a few) places of the code follow the assumption.

Code vs. Comments

Code	Comment	Implication
Precise	Imprecise	Comments are harder to analyze.
Can be tested	Can NOT be tested	Comments may become less reliable as software evolves.
Harder to understand	Easier to understand	Likely to read comments. Wrong comments mislead programmers.

- Many assumptions are difficult to infer from source code alone.
  - Inferring from source code alone may fail
    - for cases that no (or only a few) places of the code follow the assumption.
- Use comment-code redundancy to detect comment-code mismatches.

Possibility (1): Bugs

- Mismatches indicate:
  - Possibility (1): Bugs
    - Due to time-constraints or other reasons.
    - Old code is not updated according to a new assumption.

Possibility (1): Bugs

- Mismatches indicate:
  - Possibility (1): Bugs
    - Due to time-constraints or other reasons.
    - Old code is not updated according to a new assumption.

A bug automatically detected by iComment:

```
linux/drivers/ata/libata-core.c:
/* LOCKING: caller. */
void ata_dev_select(…) {…}

…
int ata_dev_read_id(…) {
    …
    ata_dev_select(…);
    …
}
```

Possibility (1): Bugs

- Mismatches indicate:
  - Possibility (1): Bugs
    - Due to time-constraints or other reasons.
    - Old code is not updated according to a new assumption.

A bug automatically detected by iComment:

```
linux/drivers/ata/libata-core.c:
/* LOCKING: caller. */
void ata_dev_select(…) {…}

…
int ata_dev_read_id(…) {
    …
    ata_dev_select(…);
    …
}
```

Assumption in Comment.

Possibility (1): Bugs

- Mismatches indicate:
  - Possibility (1): Bugs
    - Due to time-constraints or other reasons.
    - Old code is not updated according to a new assumption.

A bug automatically detected by iComment:

```
linux/drivers/ata/libata-core.c:
/* LOCKING: caller. */
void ata_dev_select(…) {…}

…
int ata_dev_read_id(…) {
    …
    ata_dev_select(…);
    …
}
```

Assumption in Comment.

No lock is held before calling ata_dev_select.

Possibility (1): Bugs

- Mismatches indicate:
  - Possibility (1): Bugs
    - Due to time-constraints or other reasons.
    - Old code is not updated according to a new assumption.

A bug automatically detected by iComment:

```
linux/drivers/ata/libata-core.c:
/* LOCKING: caller. */
void ata_dev_select(…) {…}

…
int ata_dev_read_id(…) {
    …
    ata_dev_select(…);
    …
}
```

Assumption in Comment.

No lock is held before calling ata_dev_select.

Mismatch! The bug is already confirmed by Linux developers after we reported it.

Possibility (2): Bad Comments

- Possibility (2): Bad comments - can cause new bugs
  - Comments are not updated accordingly.

Possibility (2): Bad Comments

- Possibility (2): Bad comments - can cause new bugs
  - Comments are not updated accordingly.

A bad comment automatically detected by iComment:

```
mozilla/security/nss/lib/ssl/sslsnce.c:
/* Caller must hold cache lock when calling
this. */
static sslSessionID * ConvertToSID(…) {…}
…
static sslSessionID *ServerSessionIDLookup(…)
{   …
    UnlockSet(cache, set);
    …
    sid = ConvertToSID(…);
    …
}
```

Possibility (2): Bad Comments

- Possibility (2): Bad comments - can cause new bugs
  - Comments are not updated accordingly.

A bad comment automatically detected by iComment:

```
mozilla/security/nss/lib/ssl/sslsnce.c:
/* Caller must hold cache lock when calling
this. */
static sslSessionID * ConvertToSID(…) {…}
…
static sslSessionID *ServerSessionIDLookup(…)
{   …
    UnlockSet(cache, set);
    …
    sid = ConvertToSID(…);
    …
}
```

Assumption in Comment.

Possibility (2): Bad Comments

- Possibility (2): Bad comments - can cause new bugs
  - Comments are not updated accordingly.

A bad comment automatically detected by iComment:

```
mozilla/security/nss/lib/ssl/sslsnce.c:
/* Caller must hold cache lock when calling
this. */
static sslSessionID * ConvertToSID(…) {…}
…
static sslSessionID *ServerSessionIDLookup(…)
{   …
    UnlockSet(cache, set);
    …
    sid = ConvertToSID(…);
    …
}
```

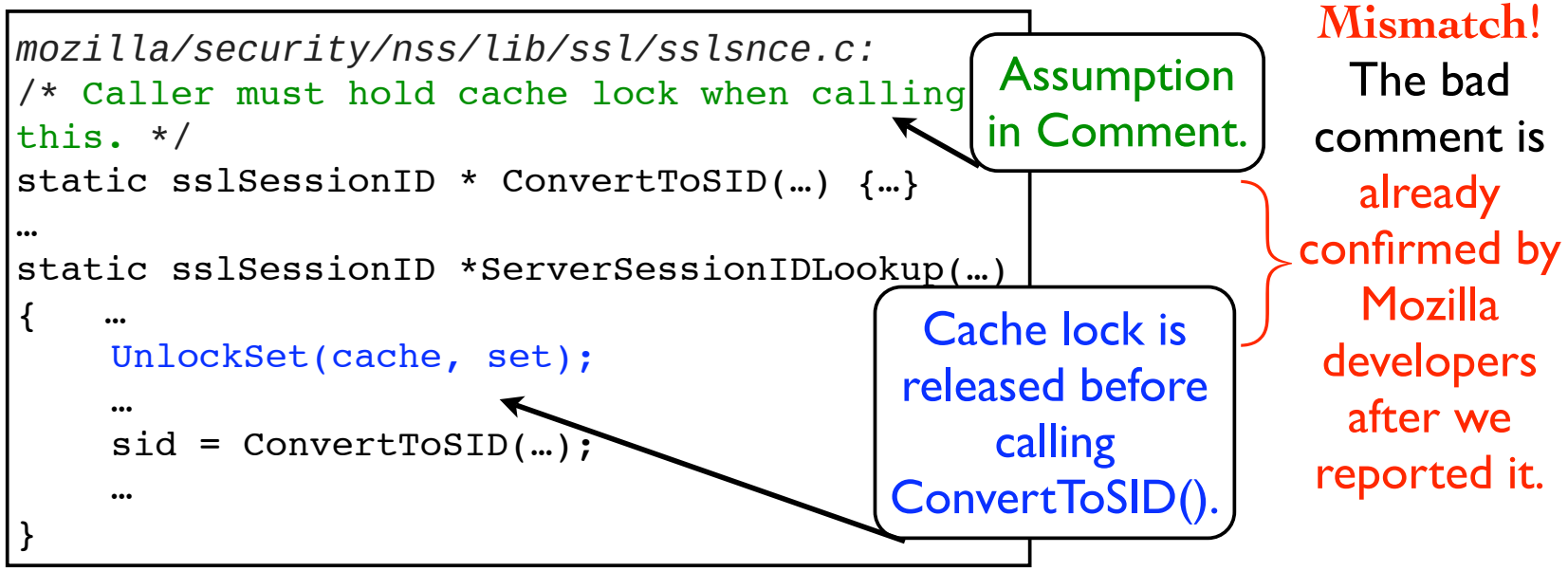
Assumption in Comment.

Cache lock is released before calling ConvertToSID().

Possibility (2): Bad Comments

- Possibility (2): Bad comments - can cause new bugs
  - Comments are not updated accordingly.

A bad comment automatically detected by iComment:

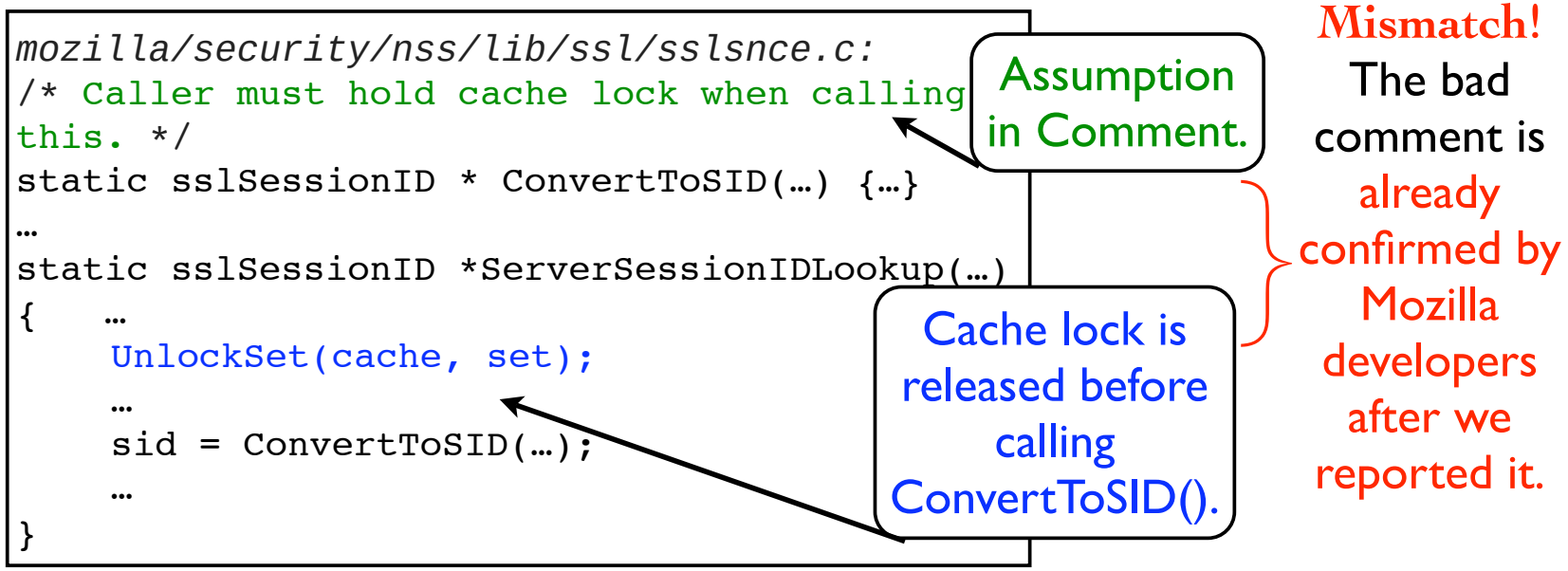


Mismatch! The bad comment is already confirmed by Mozilla developers after we reported it.

Possibility (2): Bad Comments

- Possibility (2): Bad comments - can cause new bugs
  - Comments are not updated accordingly.

A bad comment automatically detected by iComment:



- Our paper contains bad comment examples that already caused new bugs.

Challenges

- Goal: Detect comment-code inconsistencies.

Challenges

- Goal: Detect comment-code inconsistencies.

- Challenges of understanding comments written in natural language
  - Various ways to paraphrase natural language
    - /* We need to acquire the write IRQ lock before calling ep_unlink(). */
    - /* Lock must be acquired on entry to this function. */
    - /* Caller must hold instance lock! */

Challenges

- Goal: Detect comment-code inconsistencies.
- Challenges of understanding comments written in natural language
  - Various ways to paraphrase natural language
    - /* We need to acquire the write IRQ lock before calling ep_unlink(). */
    - /* Lock must be acquired on entry to this function. */
    - /* Caller must hold instance lock! */
- Use Natural Language Processing (NLP) techniques?

NLP alone is not enough.

- NLP only analyzes sentence structures.

1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

NLP alone is not enough.

- NLP only analyzes sentence structures.

Caller	must	hold	instance	lock

1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

NLP alone is not enough.

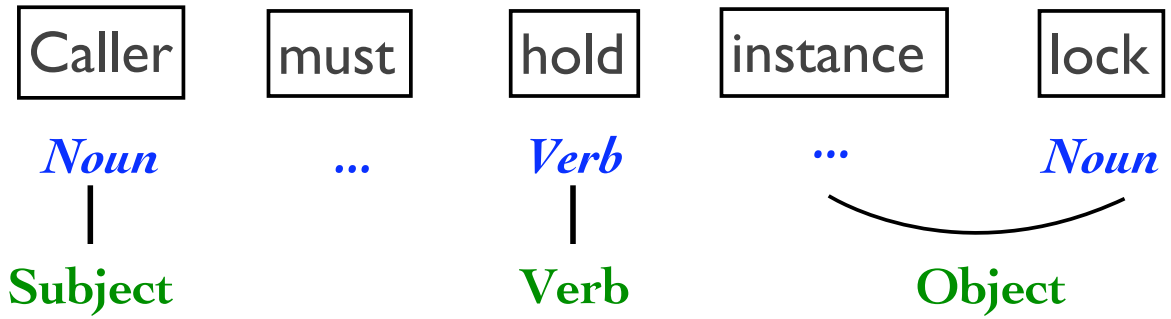
- NLP only analyzes sentence structures.

Caller	must	hold	instance	lock
Noun	*...*	*Verb*	*...*	*Noun*

1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

NLP alone is not enough.

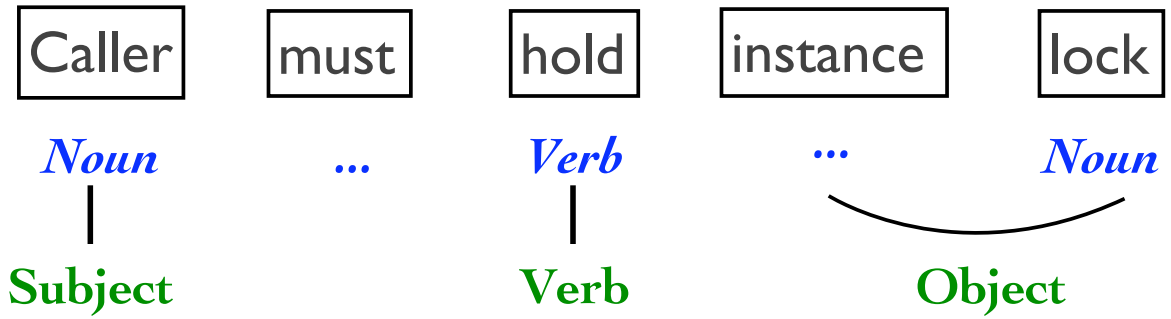
- NLP only analyzes sentence structures.



1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

NLP alone is not enough.

- NLP only analyzes sentence structures.

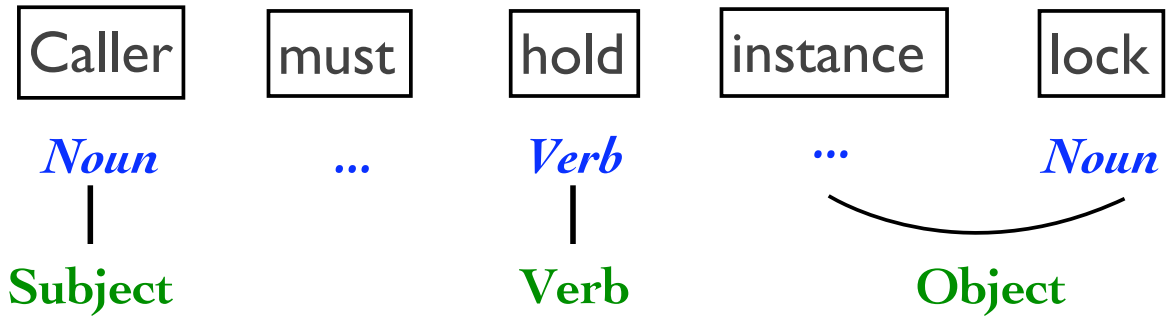


1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

- NLP is far from "understanding" natural language text.

NLP alone is not enough.

- NLP only analyzes sentence structures.

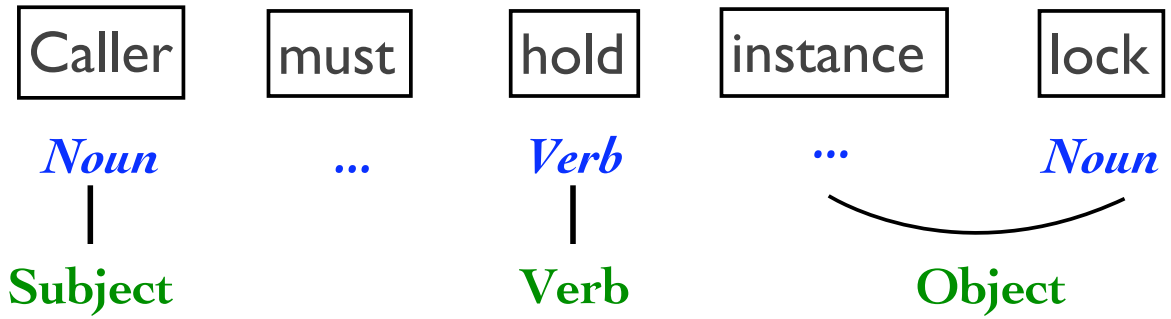


1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

- NLP is far from "understanding" natural language text.
- Many comments are not even grammatically correct.

NLP alone is not enough.

- NLP only analyzes sentence structures.



1. POS Tagging (acc: 97%)
2. Chunking (acc: 90%)
3. Semantic Role Labeling (acc: 70%)

- NLP is far from "understanding" natural language text.
- Many comments are not even grammatically correct.
- Almost impossible to automatically analyze any arbitrary comments.

Idea & Contributions

- Took the first step to automatically analyze comments written in natural language to check for mismatches
  - Combine Natural Language Processing (NLP), Machine Learning, Statistics, and Program Analysis

- Automatically extracted 1832 rules and detected 60 *new* bugs and bad comments (19 confirmed by developers)
  - 2 topics, lock-related and call-related.
  - Latest versions of 4 large software projects, Linux, Mozilla, Apache and Wine.

Outline

- Motivation, Challenges & Contributions
- Our Approach
  - Analyze comments written in natural language
  - Detect comment-code inconsistencies
- Methodology & Results
- Related work
- Conclusions

What to Analyze?

- What information is useful to extract?
- What information can be checked against code?

What is useful to extract?

- Two types of comments (examples from Linux):
  - Explain code segment: /* Set the clock rate */
  - Express assumptions/rules: /* Caller must hold instance lock! */

- We focus on rule-containing comments.
  - Likely to be inconsistent with code.
  - Likely to mislead programmers to introduce bugs.

OPERA

What can be checked?

- Not everything in comments can be checked.
- Checking can only be done topic by topic.
  - Race detectors - race bugs
  - Purify, Valgrind, etc - memory bugs
- So our comment analysis is topic by topic.
  - A general framework allowing users to choose the topic, such as lock and call-from.

OPERA

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.

OPERA

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.
1	<Lock L> must NOT be held before entering <Function F>.

OPERA

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.
1	<Lock L> must NOT be held before entering <Function F>.
2	<Lock L> must be held in <Function F>.
2	<Lock L> must NOT be held in <Function F>.

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.
1	<Lock L> must NOT be held before entering <Function F>.
2	<Lock L> must be held in <Function F>.
2	<Lock L> must NOT be held in <Function F>.

lock related

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.
1	<Lock L> must NOT be held before entering <Function F>.
2	<Lock L> must be held in <Function F>.
2	<Lock L> must NOT be held in <Function F>.
3	<Function A> must be called from <Function B>
3	<Function A> must NOT be called from <Function B>
...	...

lock related

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.
1	<Lock L> must NOT be held before entering <Function F>.
2	<Lock L> must be held in <Function F>.
2	<Lock L> must NOT be held in <Function F>.
3	<Function A> must be called from <Function B>
3	<Function A> must NOT be called from <Function B>
...	...

lock related

call related

Rule Template Examples

ID	Rule Template Examples
1	<Lock L> must be held before entering <Function F>.
1	<Lock L> must NOT be held before entering <Function F>.
2	<Lock L> must be held in <Function F>.
2	<Lock L> must NOT be held in <Function F>.
3	<Function A> must be called from <Function B>
3	<Function A> must NOT be called from <Function B>
...	...

lock related

call related

- L, F, A and B are rule parameters.
- See our paper for many other templates supported.
- Many other templates can be added.

Extracting Target Comments

OPERA

- Statistics & NLP
  - Topic keyword filtering - automatic

Extracting Target Comments

- Statistics & NLP
  - Topic keyword filtering - automatic

Take lock as the topic:

```
#A: /* return -EBUSY if a lock is held. */
#B: /* Lock must be held on entry to this function. */
#C: /* Caller must acquire instance lock! */
#D: /* Mutex locked flags */
...
```

Extracting Target Comments

- Statistics & NLP
  - Topic keyword filtering - automatic

Take lock as the topic:

```
#A: /* return -EBUSY if a lock is held. */
#B: /* Lock must be held on entry to this function. */
#C: /* Caller must acquire instance lock! */
#D: /* Mutex locked flags */
...
```

Extracting Target Comments

- Statistics & NLP
  - Topic keyword filtering - automatic
  - Correlated word filtering - automatic

Take lock as the topic:

```
#A: /* return -EBUSY if a lock is held. */
#B: /* Lock must be held on entry to this function. */
#C: /* Caller must acquire instance lock! */
#D: /* Mutex locked flags */
...
```

Extracting Target Comments

- Statistics & NLP
  - Topic keyword filtering - automatic
  - Correlated word filtering - automatic

$$cosine(A,B) = \frac{P(A,B)}{\sqrt{P(A)P(B)}}$$

See our paper for details.

Take lock as the topic:

#A: /* return -EBUSY if a lock is held. */
#B: /* Lock must be held on entry to this function. */
#C: /* Caller must acquire instance lock! */
#D: ~~/* Mutex locked flags */~~
...

Extracting Target Comments

- Statistics & NLP
  - Topic keyword filtering - automatic
  - Correlated word filtering - automatic

$$cosine(A,B) = \frac{P(A,B)}{\sqrt{P(A)P(B)}}$$

See our paper for details.

Linux	hold	acquire	call	unlock	protect
Mozilla	hold	acquire	unlock	protect	call

Take lock as the topic:

#A: /* return -EBUSY if a lock is held. */
#B: /* Lock must be held on entry to this function. */
#C: /* Caller must acquire instance lock! */
#D: ~~/* Mutex locked flags */~~
...

Classifying Comments

- Machine Learning & NLP
- Automatically classify comments to different templates (give each comment a unique label)

Classifying Comments

- Machine Learning & NLP
- Automatically classify comments to different templates (give each comment a unique label)

#A: /* return -EBUSY if a lock is held. */

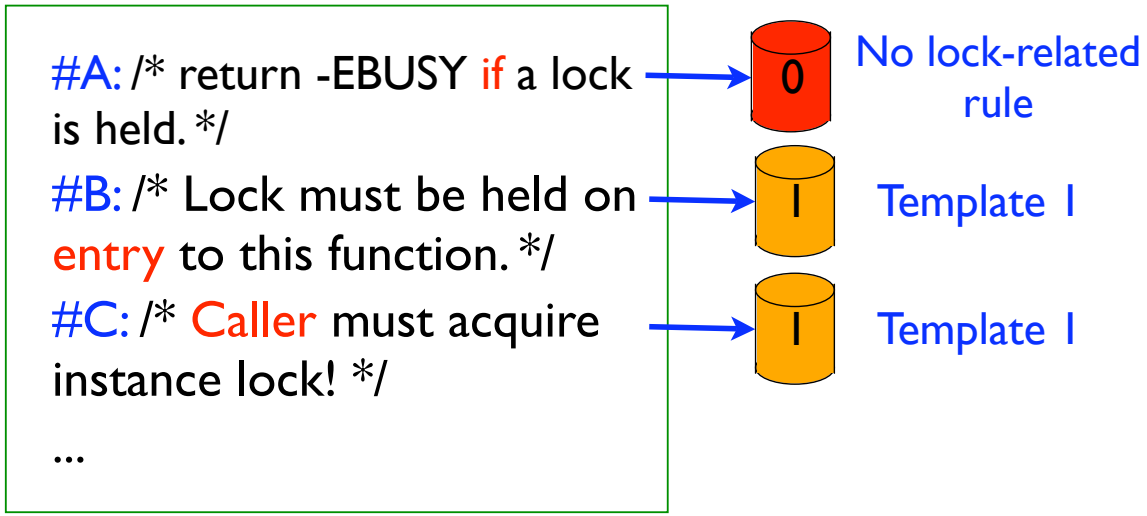
#B: /* Lock must be held on entry to this function. */

#C: /* Caller must acquire instance lock! */

...

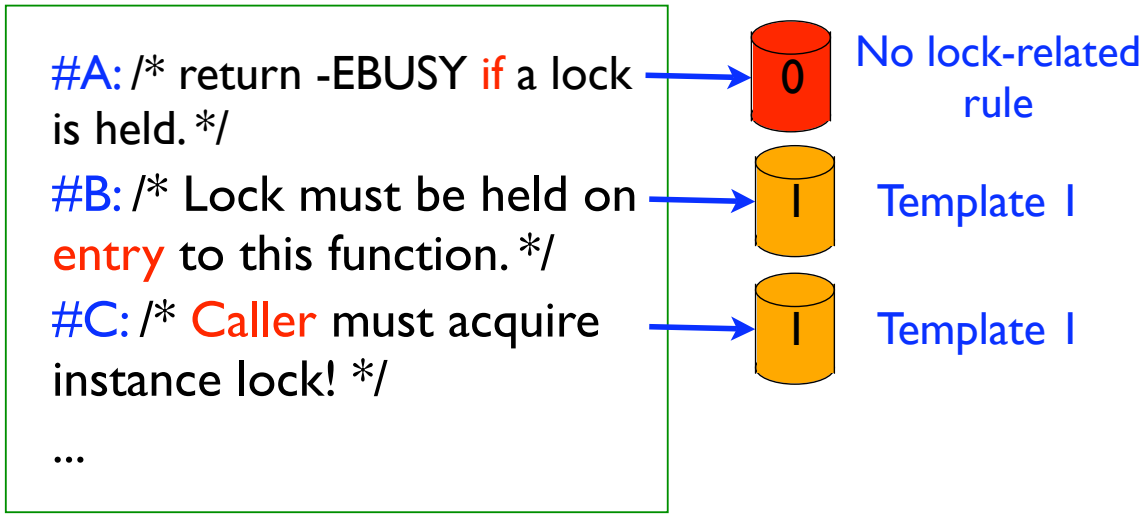
Classifying Comments

- Machine Learning & NLP
- Automatically classify comments to different templates (give each comment a unique label)



Classifying Comments

- Machine Learning & NLP
- Automatically classify comments to different templates (give each comment a unique label)
  - Core technique: Use learning classifier automatically built from a small set of manually labeled comments



OPERA

Decision Tree

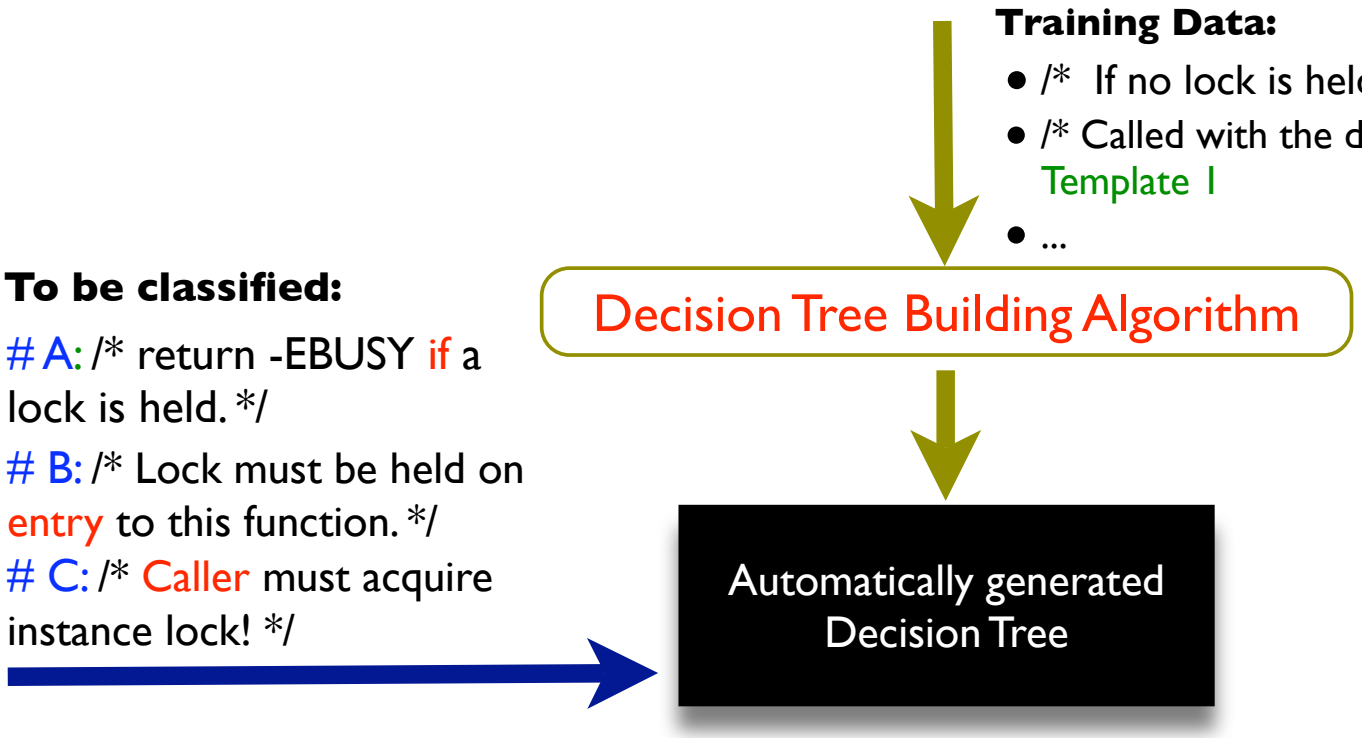
Training Data:

- /* If no lock is held, zap it. */ - NO rule
- /* Called with the device lock held. */ - Template 1
- ...

Decision Tree Building Algorithm

Automatically generated Decision Tree

Decision Tree



Decision Tree

To be classified:

A: /* return -EBUSY if a lock is held. */

B: /* Lock must be held on entry to this function. */

C: /* Caller must acquire instance lock! */

Automatically generated Decision Tree

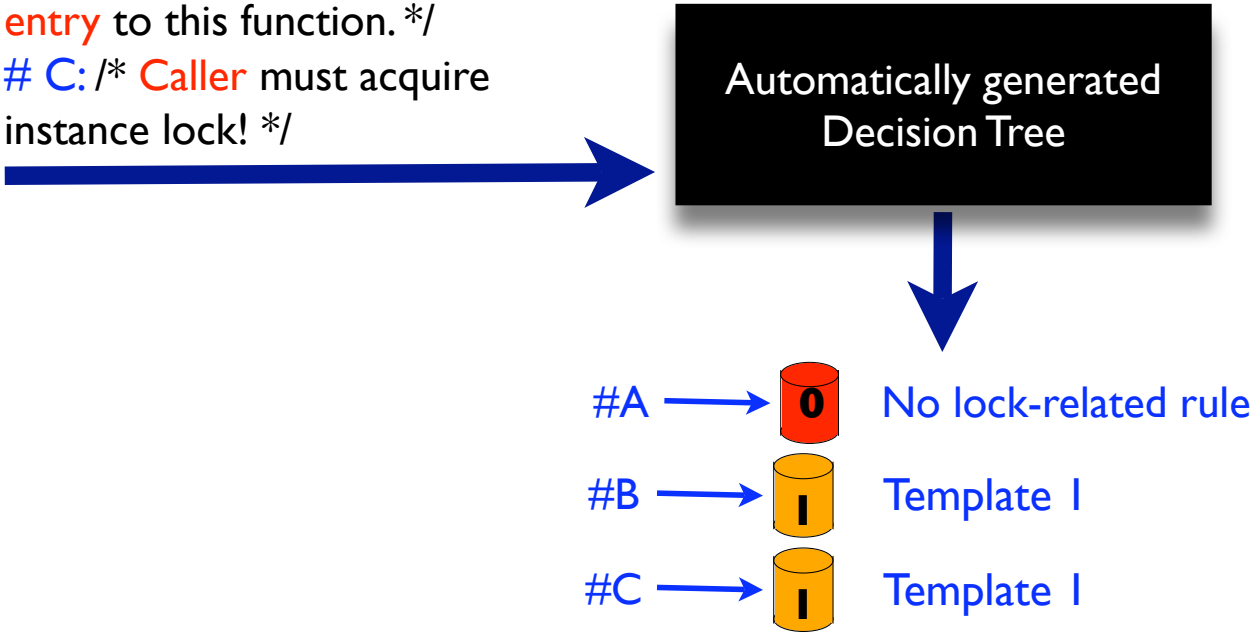
Decision Tree

To be classified:

A: /* return -EBUSY if a lock is held. */

B: /* Lock must be held on entry to this function. */

C: /* Caller must acquire instance lock! */



Decision Tree

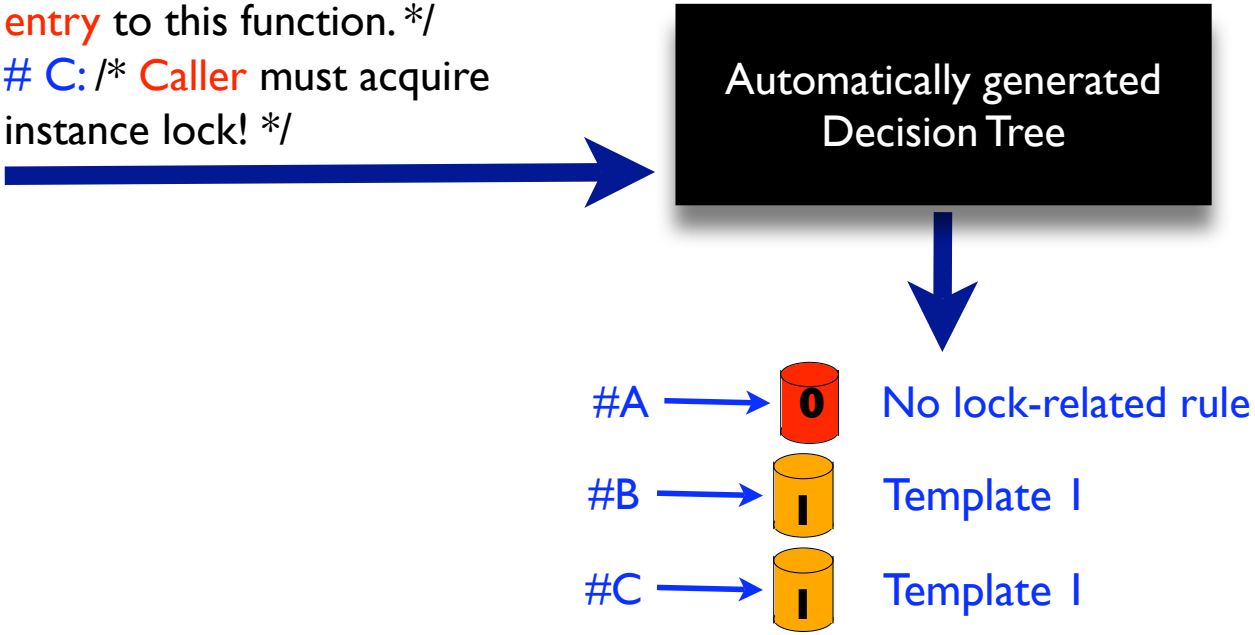
Feature selection is important.

To be classified:

A: /* return -EBUSY if a lock is held. */

B: /* Lock must be held on entry to this function. */

C: /* Caller must acquire instance lock! */



General-purpose Training

- The training is optional for the users
  - Done by us before releasing iComment (only once per topic).

General-purpose Training

- The training is optional for the users
  - Done by us before releasing iComment (only once per topic).
  - Feasible because:
    - Programmers share wording and phrasing (confirmed by our correlated word results)
    - Cross-software training results show decision trees trained on one software can classify comments from other software with high accuracy (~89%)

General-purpose Training

- The training is optional for the users
  - Done by us before releasing iComment (only once per topic).
  - Feasible because:
    - Programmers share wording and phrasing (confirmed by our correlated word results)
    - Cross-software training results show decision trees trained on one software can classify comments from other software with high accuracy (~89%)
- Took only about 2 hours to manually classify comments of 2 topics for Linux, Mozilla, Apache and Wine

Generating Rules

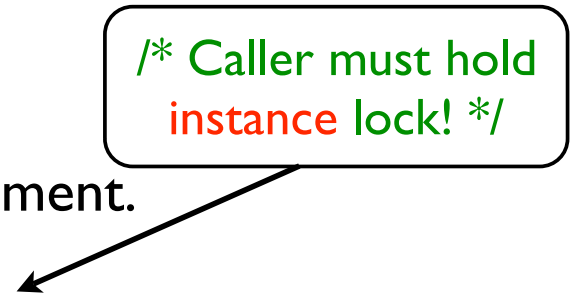
- NLP & Program Analysis
- What are the parameters?

Generating Rules

- NLP & Program Analysis
- What are the parameters?
  - The function name is right after the comment.

Generating Rules

- NLP & Program Analysis
- What are the parameters?
  - The function name is right after the comment.
  - The lock name is the object of the verb.



Generating Rules

- NLP & Program Analysis
- What are the parameters?
  - The function name is right after the comment.
  - The lock name is the object of the verb.
- Is the rule positive or negative?
  - Positive if the verb is not modified by a negation word.

/* Caller must hold instance lock! */

Rule Checker

- Use static analysis for checking
  - Flow-sensitive, and context sensitive
  - Simple point-to analysis
- Mismatch report ranking
  - Support
  - Violation

Outline

- Motivation, challenges & contributions
- Our Approach
  - Analyze comments written in natural language
  - Detect comment-code inconsistencies
- Methodology and Results
- Related work
- Conclusions

OPERA

Methodology

Software	LOC	LOM	Language	Description
Linux	5.0M	1.0M	C	OS
Mozilla	3.3M	.51M	C&C++	Browser Suite
Wine	1.5M	.22M	C	Program to Run WinApp on Unix
Apache	.27M	.057M	C	Web Server

- Latest versions of 4 large software projects
- 2 topics: lock-related and call-related
- 18% of comments are used for training on average.
  - Our training sensitivity analysis provides guidance on how much training data to use (find detailed results in our paper).

Overall Results

Software	Mismatches	Bugs	BadCom	FP	Rules
Linux	51 (14)	30 (11)	21 (3)	32	1209
Mozilla	6 (5)	2 (1)	4 (4)	3	410
Wine	2	1	1	3	149
Apache	1	0	1	0	64
Total	60 (19)	33 (12)	27 (7)	38	1832

Overall Results

Software	Mismatches	Bugs	BadCom	FP	Rules
Linux	51 (14)	30 (11)	21 (3)	32	1209
Mozilla	6 (5)	2 (1)	4 (4)	3	410
Wine	2	1	1	3	149
Apache	1	0	1	0	64
Total	60 (19)	33 (12)	27 (7)	38	1832

- Automatically detected 60 new bugs and bad comments

Overall Results

Software	Mismatches	Bugs	BadCom	FP	Rules
Linux	51 (14)	30 (11)	21 (3)	32	1209
Mozilla	6 (5)	2 (1)	4 (4)	3	410
Wine	2	1	1	3	149
Apache	1	0	1	0	64
Total	60 (19)	33 (12)	27 (7)	38	1832

- Automatically detected 60 new bugs and bad comments
  - 19 new bugs and bad comments already confirmed by the corresponding developers.

Overall Results

Software	Mismatches	Bugs	BadCom	FP	Rules
Linux	51 (14)	30 (11)	21 (3)	32	1209
Mozilla	6 (5)	2 (1)	4 (4)	3	410
Wine	2	1	1	3	149
Apache	1	0	1	0	64
Total	60 (19)	33 (12)	27 (7)	38	1832

- Automatically detected 60 new bugs and bad comments
  - 19 new bugs and bad comments already confirmed by the corresponding developers.
- Major causes of false positives
  - Mostly caused by inaccuracy from checking
  - Incorrectly generated rules

Training Accuracy

- Accuracy = the percentage of correctly labeled comments
- Software-specific training accuracy (lock-related)

Linux	Mozilla	Wine	Apache
90.8%	91.3%	96.4%	100%

Other measures, such as Kappa and Macro-F score, show similar results.
Accuracies for call-related comments are similar.

Training Accuracy

- Accuracy = the percentage of correctly labeled comments
- Software-specific training accuracy (lock-related)

Linux	Mozilla	Wine	Apache
90.8%	91.3%	96.4%	100%

Other measures, such as Kappa and Macro-F score, show similar results.
Accuracies for call-related comments are similar.

- Cross-software training accuracy (lock-related)

Training SW	Mozilla	Wine	Apache
Linux	81.5%	78.6%	83.3%
Linux+Mozilla	/	89.3%	88.9%

- Training can be done by us before releasing iComment to analyze users' software.

Related Work

- Extracting rules from source code and execution behaviors [SOSP01 & OSDI06 Engler et. al., Daikon, ...]:
  - Our approach complements these techniques.
- Annotation Language [Microsoft SAL, Java annotations, Splint, SafeDrive, Sparse, ...]:
  - Not as expressive: usability
  - Not widely adopted vs. millions lines of comments already exist.
- Automatic document generation from comments [C# XML comments, JavaDoc, Doxygen, RDoc, ...]:
  - Do NOT analyze the natural language part
  - Share similar challenges of analyzing unstructured comments.

Conclusions

- Comment-code inconsistencies hurt software quality and reliability.
- First work to automatically analyze comments written in natural language for mismatch detection
  - iComment automatically extracted 1832 rules on 2 topics and detected 60 *new* bugs and bad comments (19 confirmed by developers)
- More work in this direction!
  - Analyze other system documents in natural language

Acknowledgments

Professor Stefan Savage (Shepherd)
Anonymous Reviewers
NSF Research Grants
DOE Research Grants
Microsoft Gift Grants
Intel Gift Grants
NSF Student Travel Scholarship

Thank you! Questions?

Lin Tan
Ding Yuan
Gopal Krishna
Yuanyuan (YY) Zhou

OPERA group
University of Illinois
at Urbana-Champaign