Listening to Programmers
/* -- Taxonomies and Characteristics of Comments in Operating System Code */

Lin Tan

Yoann Padoleau, Lin Tan, Yuanyuan Zhou
University of Illinois, Urbana-Champaign
Motivation

• Many innovations to improve software quality & productivity:
  • PL, IDE, bug detection tools, annotation languages, ...
  • Valgrind, Splint, Linux’s Sparse, Microsoft SAL, TagSEA, Mylyn, ...
Motivation

• Many innovations to improve software quality & productivity:
  • PL, IDE, bug detection tools, annotation languages, ...
  • Valgrind, Splint, Linux’s Sparse, Microsoft SAL, TagSEA, Mylyn, ...

• Helpful to know what developers want
Comments Reveal Needs

- User studies:
  - Examples: CMU, Microsoft Research HIP
  - Challenges: Hard to collect representative data
Comments Reveal Needs

• User studies:
  • Examples: CMU, Microsoft Research HIP
  • Challenges: Hard to collect representative data

• Our novel observation:
  • Comments reveal what developers want.
Potential of Comments (I)

- Developers want to express code relationships

```c
opensolaris/sun/io/ms.c:
timeout_id_t msd_timeout_id; /* id returned by timeout() */
```
Potential of Comments (I)

- Developers want to express code relationships
- Motivate language support & bug detection tools

```c
opensolaris/sun/io/ms.c:
timeout_id_t msd_timeout_id; /* id returned by timeout() */
```
Potential of Comments (I)

- Developers want to express code relationships
- Motivate language support & bug detection tools
- Motivate IDE features for better navigation capability
Potential of Comments (II)

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

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

static int in2000_bus_reset(...){ ...
    reset_hardware(...); ... }
```
Potential of Comments (II)

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

static int in2000_bus_reset(...){ ...
    No lock acquisition ⇒ A bug!
    reset_hardware(...); ...
}
```
Potential of Comments (II)

Developers want to express assumptions/intentions.

Motivate bug detection tools [TanSOSP’07]
Potential of Comments (II)

- Developers want to express assumptions/intentions.
- Motivate bug detection tools [TanSOSP’07]

See our paper for more examples.
## Prevalence of Comments

<table>
<thead>
<tr>
<th>Software</th>
<th>Linux</th>
<th>FreeBSD D</th>
<th>OpenSolaris</th>
<th>Mozilla</th>
<th>MySQL</th>
<th>Eclipse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Comments</td>
<td>1.2M</td>
<td>0.6M</td>
<td>1.1M</td>
<td>1.2M</td>
<td>0.3M</td>
<td>1.7M</td>
</tr>
</tbody>
</table>

(kernel only for OSs, excluding blank lines, including copyright notices)

- Millions lines of comments (23-30%) exist.
- Various languages: C, C++, Java
- Written by thousands of developers or more
Our Contributions

• First comprehensive comment study on semantics:
  • Manually examine 2100 randomly sampled comments from 6 large popular software projects (in C, C++, Java)
  • Many findings and implications:
    • Provide guidance to the design of tools/languages
    • New comment taxonomies & analysis tools
  • Available at http://ece.uwaterloo.ca/~lintan/CComment
Outline

• Motivation

• Methodology & Taxonomies

• OS Comments: Findings and Implications

• Non-OS Comments: Similarities & Differences

• Related Work & Conclusions
Our OS Comment Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>OpenSolaris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>5.2M</td>
<td>2.4M</td>
<td>3.7M</td>
</tr>
<tr>
<td>Lines of Comments</td>
<td>1.2M</td>
<td>0.6M</td>
<td>1.1M</td>
</tr>
<tr>
<td>% of Comments</td>
<td>23.1%</td>
<td>25.0%</td>
<td>29.7%</td>
</tr>
</tbody>
</table>

(kernel only, excluding blank lines, including copyright notices)
## Our OS Comment Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>OpenSolaris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>5.2M</td>
<td>2.4M</td>
<td>3.7M</td>
</tr>
<tr>
<td>Lines of Comments</td>
<td>1.2M</td>
<td>0.6M</td>
<td>1.1M</td>
</tr>
<tr>
<td>% of Comments</td>
<td>23.1%</td>
<td>25.0%</td>
<td>29.7%</td>
</tr>
<tr>
<td># of Comments</td>
<td>729,923</td>
<td>289,413</td>
<td>380,111</td>
</tr>
<tr>
<td>(delimited by /* */ or //)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(kernel only, excluding blank lines, including copyright notices)
Our OS Comment Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>OpenSolaris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>5.2M</td>
<td>2.4M</td>
<td>3.7M</td>
</tr>
<tr>
<td>Lines of Comments</td>
<td>1.2M</td>
<td>0.6M</td>
<td>1.1M</td>
</tr>
<tr>
<td>% of Comments</td>
<td>23.1%</td>
<td>25.0%</td>
<td>29.7%</td>
</tr>
<tr>
<td># of Comments</td>
<td>729,923</td>
<td>289,413</td>
<td>380,111</td>
</tr>
<tr>
<td>(delimited by /* */ or //)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample Size</td>
<td>350</td>
<td>350</td>
<td>350</td>
</tr>
</tbody>
</table>

(kernel only, excluding blank lines, including copyright notices)

- Randomly sampled 1050 comments
- Causing reasonably small margin of error
Our OS Comment Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>OpenSolaris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>5.2M</td>
<td>2.4M</td>
<td>3.7M</td>
</tr>
<tr>
<td>Lines of Comments</td>
<td>1.2M</td>
<td>0.6M</td>
<td>1.1M</td>
</tr>
<tr>
<td>% of Comments</td>
<td>23.1%</td>
<td>25.0%</td>
<td>29.7%</td>
</tr>
<tr>
<td># of Comments</td>
<td>729,923</td>
<td>289,413</td>
<td>380,111</td>
</tr>
<tr>
<td>(delimited by /* */ or //)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample Size</td>
<td>350</td>
<td>350</td>
<td>350</td>
</tr>
</tbody>
</table>

(kernel only, excluding blank lines, including copyright notices)

- Randomly sampled 1050 comments
- Causing reasonably small margin of error

Java & C++ code later
Dimensions of Taxonomies

- Who
- What
- When
- Where
Dimensions of Taxonomies

- What can be utilized & how much?
- How to use these comments?
Dimensions of Taxonomies

- What can be utilized & how much?
- How to use these comments?

Studied 309 unique OS developers.
Classification Process

• Iterative process

• Double verification

• A tool to help
  • Automatically extract author, time and related entities
Threats to Validity

- Focused on OS/C code (only 3 C++/Java projects)
Threats to Validity

- Focused on OS/C code (only 3 C++/Java projects)
- Sampled 2100 comments
Threats to Validity

• Focused on OS/C code (only 3 C++/Java projects)
• Sampled 2100 comments
• Same amount of comments from each software
Threats to Validity

- Focused on OS/C code (only 3 C++/Java projects)
- Sampled 2100 comments
- Same amount of comments from each software
- Subjectivity
Threats to Validity

- Focused on OS/C code (only 3 C++/Java projects)
- Sampled 2100 comments
- Same amount of comments from each software
- Subjectivity
- Outdated comments
Outline

- Motivation
- Methodology & Taxonomies
  - OS Comments: Findings and Implications
  - Non-OS Comments: Similarities & Differences
- Related Work & Conclusions
Exploitable Comments

- At least 52.6 ±2.9% or ~736,109 comments in the 3 OSs:
  - Could be leveraged by existing or to-be-proposed techniques
  - Could guide the design of language features, IDE features, annotation languages and bug detection tools
Finding 1: Integers

- 22.1% of the exploitable comments clarify the usage and meaning of integers and integer macros.

```c
1. #define E1000_MCC 0x0401C  /* Multiple Collision Count - R/cr */
   #define E1000_RCTL 0x00100  /* Rx Control - RW */
```
Finding 1: Integers

- 22.1% of the exploitable comments clarify the usage and meaning of integers and integer macros.

1. #define **E1000_MCC** 0x0401C /* Multiple Collision Count - R/cr */
   #define **E1000_RCTL** 0x00100 /* Rx Control - RW */

```
E1000_READ_REG (hw, E1000_MCC);
E1000_WRITE_REG (hw, E1000_MCC);
```
Finding 1: Integers

22.1% of the exploitable comments clarify the usage and meaning of integers and integer macros.

1. #define E1000_MCC 0x0401C /* Multiple Collision Count - R/ccl */
   #define E1000_RCTL 0x00100 /* Rx Control - RW */

E1000_READ_REG (hw, E1000_MCC);
E1000_WRITE_REG (hw, E1000_MCC);

Const doesn’t solve the problem!
Finding 1: Integers

- 22.1% of the exploitable comments clarify the usage and meaning of integers and integer macros.

1. `#define E1000_MCC 0x0401C /* Multiple Collision Count - R/clr */`
   `#define E1000_RCTL 0x00100 /* Rx Control - RW */`

E1000_READ_REG (hw, E1000_MCC);

E1000_WRITE_REG (hw, E1000_MCC);

Const doesn’t solve the problem!

2. `#define EMU_DOCK_MINOR_REV 0x26 /* 0000xxx 3 bit ... Minor rev */`

3. `int mem; /* memory in 128 MB units */`
Finding 1: Integers

- 22.1% of the exploitable comments clarify the usage and meaning of integers and integer macros.

1. `#define E1000_MCC 0x0401C /* Multiple Collision Count - R/clr */
   #define E1000_RCTL 0x00100 /* Rx Control - RW */`

   E1000_READ_REG (hw, E1000_MCC);

   E1000_WRITE_REG (hw, E1000_MCC);

   Const doesn’t solve the problem!

2. `#define EMU_DOCK_MINOR_REV 0x26 /* 0000xxx 3 bit ... Minor rev */`

3. `int mem; /* memory in 128 MB units */`

- Should pay more attention to integers and integer macros
  - Domain specific languages, extended types, bug detection tools, ...
Finding 2: Code Relationships

• 16.8% of exploitable comments are about code relationships.
Finding 2: Code Relationships

- 16.8% of exploitable comments are about code relationships.

1. /* See comment in struct sock definition to understand ... */
Finding 2: Code Relationships

- 16.8% of exploitable comments are about code relationships.

1. /* See comment in struct sock definition to understand */
2. /* Called from mmioctl_page_retire */
Finding 2: Code Relationships

- 16.8% of exploitable comments are about code relationships.

1. /* See comment in struct sock definition to understand ... */
2. /* Called from mmioctl_page_retire ... */
3. /* Call scsi_free before mem_free since ... */
Finding 2: Code Relationships

- 16.8% of exploitable comments are about code relationships.

1. /* See comment in struct sock definition to understand */
2. /* Called from mmioctl_page_retire */
3. /* Call scsi_free before mem_free since */

/* WARNING: If you change any of these defines, make sure to change */
Finding 2: Code Relationships

- 16.8% of exploitable comments are about code relationships.

1. /* See comment in struct sock definition to understand ... */
2. /* Called from mmioctl_page_retire ... */
3. /* Call scsi_free before mem_free since ... */

/* WARNING: If you change any of these defines, make sure to change ... */

- Exploit such comments to provide better navigation capabilities
- Inspire techniques to express code relationships and evolution
Finding 3: Locking

- 4.7% of the non-trivial comments are synchronization/lock related.

```c
1. /* Caller must hold instance lock! */
   static int reset_hardware(...) {...}
```
Finding 3: Locking

• 4.7% of the non-trivial comments are synchronization/lock related.

1. /* Caller must hold instance lock! */
   static int reset_hardware(...) {
   ...}

2. /* Locking key to struct socket:
   *(a) constant after allocation, no locking required.
   *(b) ...
   *(h) locked by global mutex so_global_mtx. ...
   */
   struct socket {
   short so_type; /* (a) generic type, see socket.h */
   ...
   so_gen_t so_gencnt; /* (h) generation count */
   }
Finding 3: Locking

- 4.7% of the non-trivial comments are synchronization/lock related.

1. /* Caller must hold instance lock! */
   static int reset_hardware(...) {...}

2. /* Locking key to struct socket:
   * (a) constant after allocation, no locking required.
   * ...
   * (h) locked by global mutex so_global_mtx. ...
   */
   struct socket {
     short so_type; /* (a) generic type, see socket.h */

     so_gen_t so_gencnt; /* (h) generation count */
   }

Finding 3: Locking

- 4.7% of the non-trivial comments are synchronization/lock related.

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

2. /* Locking key to struct socket:
   * (a) constant after allocation, no locking required.
   * ...
   * (h) locked by global mutex so_global_mtx. ...
   */
   struct socket {
     short so_type; /* (a) generic type, see socket.h */
     ...
     so_gen_t so_gencnt; /* (h) generation count */
   }

- Design easy-to-use annotations to express lock-related concerns
Finding 4: Annotation Convertible

• At least 10.7% of the exploitable comments can be expressed by existing annotations languages.

• Linux’s Sparse, Microsoft’s SAL, Sun’s Lock_Lint, Splint, Deputy

```c
opensolaris/intel/io/acpica/resources/rscalc.c:
/* ... AmlBufferLength - Size of AmlBuffer ... */
ACPI_STATUS AcpiRsGetListLength (  
    UINT8    *AmlBuffer,  
    UINT32   AmlBufferLength, ...)  
```
Finding 4: Annotation Convertible

• At least 10.7% of the exploitable comments can be expressed by existing annotations languages.

• Linux’s Sparse, Microsoft’s SAL, Sun’s Lock_Lint, Splint, Deputy

```c
opensolaris/intel/io/acpica/resources/rscalc.c:
/* ... AmlBufferLength - Size of AmlBuffer ... */
ACPI_STATUS AcpiRsGetListLength ((UINT8 *AmlBuffer,
UINT32  __ecount(AmlBuffer) AmlBufferLength, ...)```
Finding 4: Annotation Convertible

- At least 10.7% of the exploitable comments can be expressed by existing annotations languages.
- Linux’s Sparse, Microsoft’s SAL, Sun’s Lock_Lint, Splint, Deputy

```c
opensolaris/intel/io/acpica/resources/rscalc.c:
/* ... AmlBufferLength - Size of AmlBuffer ... */
ACPI_STATUS AcpiRsGetListLength (  
    UINT8                        *AmlBuffer,  
    UINT32 __ecount(AmlBuffer)    AmlBufferLength, ...)  
```

- Automatically convert these comments into annotations
## Open vs. Closed Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>OpenSolaris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FreeBSD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenSolaris</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Open vs. Closed Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>23.1%</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>25%</td>
</tr>
<tr>
<td>OpenSolaris</td>
<td>29.7%</td>
</tr>
</tbody>
</table>
# Open vs. Closed Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploitable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>23.1%</td>
<td>55.7%</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>25%</td>
<td>51.7%</td>
</tr>
<tr>
<td>OpenSolaris</td>
<td>29.7%</td>
<td>50.3%</td>
</tr>
</tbody>
</table>
## Open vs. Closed Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploitable</th>
<th>Integers</th>
<th>Relationship</th>
<th>Locking</th>
<th>Annotation Convertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>23.1%</td>
<td>55.7%</td>
<td>26.7%</td>
<td>19.5%</td>
<td>5.1%</td>
<td>8.2%</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>25%</td>
<td>51.7%</td>
<td>21.5%</td>
<td>14.9%</td>
<td>5.5%</td>
<td>12.7%</td>
</tr>
<tr>
<td>OpenSolaris</td>
<td>29.7%</td>
<td>50.3%</td>
<td>17.6%</td>
<td>15.9%</td>
<td>3.4%</td>
<td>11.4%</td>
</tr>
</tbody>
</table>
Open vs. Closed Source

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploitable</th>
<th>Integers</th>
<th>Relationship</th>
<th>Locking</th>
<th>Annotation Convertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>23.1%</td>
<td>55.7%</td>
<td>26.7%</td>
<td>19.5%</td>
<td>5.1%</td>
<td>8.2%</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>25%</td>
<td>51.7%</td>
<td>21.5%</td>
<td>14.9%</td>
<td>5.5%</td>
<td>12.7%</td>
</tr>
<tr>
<td>OpenSolaris</td>
<td>29.7%</td>
<td>50.3%</td>
<td>17.6%</td>
<td>15.9%</td>
<td>3.4%</td>
<td>11.4%</td>
</tr>
</tbody>
</table>

- OpenSolaris (started as closed software) exhibits similar characteristics from its open source counterparts.
- Complement the results of previous study [SpinellisICSE’08]
- Our findings are general across different OSs.
Outline

- Motivation
- Methodology & Taxonomies
- OS Comments: Findings and Implications
  - Non-OS Comments: Similarities & Differences
- Related Work & Conclusions
## Non-OS Comment Source

<table>
<thead>
<tr>
<th>Software Type</th>
<th>Linux</th>
<th>FreeBSD</th>
<th>OpenSolaris</th>
<th>Mozilla</th>
<th>MySQL</th>
<th>Eclipse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>OS</td>
<td>OS</td>
<td>OS</td>
<td>Browser</td>
<td>DB Server</td>
<td>IDE</td>
</tr>
<tr>
<td>Language</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C/C++</td>
<td>C/C++</td>
<td>Java</td>
</tr>
<tr>
<td>Lines of Code</td>
<td>5.2M</td>
<td>2.4M</td>
<td>3.7M</td>
<td>4.5M</td>
<td>1.2M</td>
<td>6.1M</td>
</tr>
<tr>
<td>Lines of Comments</td>
<td>1.2M</td>
<td>0.6M</td>
<td>1.1M</td>
<td>1.2M</td>
<td>0.3M</td>
<td>1.7M</td>
</tr>
<tr>
<td>Sample Size</td>
<td>350</td>
<td>350</td>
<td>350</td>
<td>350</td>
<td>350</td>
<td>350</td>
</tr>
</tbody>
</table>

- Randomly sampled 1050 comments
Non-OS Comment Findings

<table>
<thead>
<tr>
<th>Software</th>
<th>OS</th>
<th>Non-OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Mostly similar, but OS has more locking & integer comments
- See our paper for other differences
Non-OS Comment Findings

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>23.1-29.7%</td>
</tr>
<tr>
<td>Non-OS</td>
<td>21.8-28.5%</td>
</tr>
</tbody>
</table>

- Mostly similar, but OS has more locking & integer comments
- See our paper for other differences
Non-OS Comment Findings

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploitable</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>23.1-29.7%</td>
<td>52.6±2.9%</td>
</tr>
<tr>
<td>Non-OS</td>
<td>21.8-28.5%</td>
<td>57.5±2.9%</td>
</tr>
</tbody>
</table>

- Mostly similar, but OS has more locking & integer comments
- See our paper for other differences
Non-OS Comment Findings

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploitable</th>
<th>Relationship</th>
<th>Annotation Convertible</th>
<th>Locking</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>23.1-29.7%</td>
<td>52.6±2.9%</td>
<td>16.8%</td>
<td>10.7%</td>
<td>4.1%</td>
</tr>
<tr>
<td>Non-OS</td>
<td>21.8-28.5%</td>
<td>57.5±2.9%</td>
<td>21.2%</td>
<td>13.7%</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

- Mostly similar, but OS has more locking & integer comments
- See our paper for other differences
## Non-OS Comment Findings

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploitable</th>
<th>Relationship</th>
<th>Annotation Convertible</th>
<th>Locking</th>
<th>Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>23.1-29.7%</td>
<td>52.6±2.9%</td>
<td>16.8%</td>
<td>10.7%</td>
<td>4.1%</td>
<td>22.1%</td>
</tr>
<tr>
<td>Non-OS</td>
<td>21.8-28.5%</td>
<td>57.5±2.9%</td>
<td>21.2%</td>
<td>13.7%</td>
<td>1.7%</td>
<td>6.3%</td>
</tr>
</tbody>
</table>

- Mostly similar, but OS has more locking & integer comments
- See our paper for other differences
Non-OS Comment Findings

<table>
<thead>
<tr>
<th>Software</th>
<th>Comments</th>
<th>Exploit</th>
<th>Relation</th>
<th>Annotation Convertible</th>
<th>Locking</th>
<th>Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>23.1-29.7%</td>
<td>52.6±2.9%</td>
<td>16.8%</td>
<td>10.7%</td>
<td>4.1%</td>
<td>22.1%</td>
</tr>
<tr>
<td>Non-OS</td>
<td>21.8-28.5%</td>
<td>57.5±2.9%</td>
<td>21.2%</td>
<td>13.7%</td>
<td>1.7%</td>
<td>6.3%</td>
</tr>
</tbody>
</table>

- Mostly similar, but OS has more locking & integer comments
- See our paper for other differences
- Exceptions are not always used.
- Still use comments to explain the exception types

```c
/* return 1 if ACK, 0 if NAK, -1 if error. */
static int slhci_transaction(...) { ... }
```
Related Work

• **Comment studies**
  [WoodfieldICSE’81], [Etzkorn’99], [Stamelos’02], [Warren’02], [YingMSR05], [Marin’05], [JiangMSR’06], [Fluri’07], [StoreyICSE’08], ...

• **Usefulness of comments for program understanding**
  [WoodfieldICSE’81]

• **Impact of already commented code** [Marin’05]

• **TODO comments** [YingMSR05], [StoreyICSE’08]
Conclusions

• Comments reveal interesting findings, which guide the design of new tools and languages:
  • Abusive use of integers
  • Lack of expressing power on code relationship and evolution
  • Many can be expressed by existing annotation languages

• New taxonomies, tools & examined comments
  • Available at http://ece.uwaterloo.ca/~lintan/CComment

More findings and implications in our paper