Motivation

How can we improve Software reliability?

- Programming Language Extensions
- New Development Tools
- Annotation Languages
How can we improve Software reliability?

- Programming Language Extensions
- New Development Tools
- Annotation Languages
- Better Programming Languages
What do programmers need?

Unfortunately many of these innovations are not fully leveraged by programmers.
Studying comments can help...

... improving programming languages.

```c
struct st_drivetype {
    char* name;
    int len;
};

const struct st_drivetypes[] = {
    "Unisys...", // name
    15, // length
};
```
Studying comments can help...

- improving programming languages.

```c
struct st_drivetype {
    char* name;
    int len;
};

const struct st_drivetypes[] = {
    "Unisys...", // name
    15, // length
};

This led to the GCC Designator Extensions.

```c
struct st_drivetypes st = {
    .name = "Unisys...",
    .len = 15
};
```
Methodology

- Linux
  - LOC 5.2M
  - Comments: 1.2M (23.1%)

- FreeBSD
  - LOC 2.4M
  - Comments: 0.6M (25%)

- OpenSolaris
  - LOC 3.7M
  - Comments: 1.1M (29.7%)

Challenges:

1. Challenge 1: Understand the content of the comment
2. Challenge 2: No taxonomy based on comment content (yet)
3. Issue 1: No general conclusions about comments in software
4. Issue 2: Subjectivity
5. Issue 3: Fixed amount of comments
Methodology

- **Linux**
  - LOC 5.2M
  - Comments: 1.2M (23.1%)

- **FreeBSD**
  - LOC 2.4M
  - Comments: 0.6M (25%)

- **OpenSolaris**
  - LOC 3.7M
  - Comments: 1.1M (29.7%)

350 Comments per OS selected at random
Methodology

- **Linux**
  - LOC 5.2M
  - Comments: 1.2M (23.1%)

- **FreeBSD**
  - LOC 2.4M
  - Comments: 0.6M (25%)

- **opensolaris**
  - LOC 3.7M
  - Comments: 1.1M (29.7%)

- 350 Comments per OS selected at random
- Challenge 1: Understand the content of the comment
- Challenge 2: No taxonomy based on comment content (yet)
- Issue 1: No general conclusions about comments in software
- Issue 2: Subjectivity
- Issue 3: Fixed amount of comments
Methodology

- Linux
  - LOC 5.2M
  - Comments: 1.2M (23.1%)
- FreeBSD
  - LOC 2.4M
  - Comments: 0.6M (25%)
- OpenSolaris
  - LOC 3.7M
  - Comments: 1.1M (29.7%)

- 350 Comments per OS selected at random
- Challenge 1: Understand the content of the comment
- Challenge 2: No taxonomy based on comment content (yet)
Methodology

- **LOC 5.2M**  
  Comments: 1.2M (23.1%)

- **LOC 2.4M**  
  Comments: 0.6M (25%)

- **LOC 3.7M**  
  Comments: 1.1M (29.7%)

- 350 Comments per OS selected at random
- Challenge 1: Understand the content of the comment
- Challenge 2: No taxonomy based on comment content (yet)
- Issue 1: No general conclusions about comments in software
Methodology

- Linux: LOC 5.2M, Comments: 1.2M (23.1%)
- FreeBSD: LOC 2.4M, Comments: 0.6M (25%)
- OpenSolaris: LOC 3.7M, Comments: 1.1M (29.7%)

- 350 Comments per OS selected at random
- Challenge 1: Understand the content of the comment
- Challenge 2: No taxonomy based on comment content (yet)
- Issue 1: No general conclusions about comments in software
- Issue 2: Subjectivity
Methodology

- **Linux**
  - LOC 5.2M
  - Comments: 1.2M (23.1%)

- **FreeBSD**
  - LOC 2.4M
  - Comments: 0.6M (25%)

- **openSUSE**
  - LOC 3.7M
  - Comments: 1.1M (29.7%)

- 350 Comments per OS selected at random
- Challenge 1: Understand the content of the comment
- Challenge 2: No taxonomy based on comment content (yet)
- Issue 1: No general conclusions about comments in software
- Issue 2: Subjectivity
- Issue 3: Fixed amount of comments
Taxonomies of Comments

- What
- Where
- When
- Content
- Time
- Evolution
- Entity
- Subsystem
- Author
- Beneficiary

Comments
Taxonomies of Comments

- **Content (§4, §5)**
  - Type
  - Interface
  - Code Relationship
  - PastFuture
  - Meta
  - Explanation

- **Beneficiary (§4, §5)**
  - Programming Language
  - Bug Detection Tool
  - Annotation Language
  - Code/Text Editor
  - Version Control System
  - Other

- **Entity (§6.1)**
  - Macro
  - Function Definition
  - Field
  - Variable
  - If
  - Function Call
  - Loop
  - Assignment
  - ...

- **Subsystem (§6.1)**
  - Core
  - Architecture
  - File System
  - Drivers
  - Network
  - Other
Exploitable Comments

Exploitable Toplevel Content Categories

- Type
- Interface
- Code Relationship
- PastFuture
## Exploitable Toplevel Content Categories

- Type
- Interface
- Code Relationship
- PastFuture

## Exploitable Comment

Can potentially be used by existing work or inspire new work.
Exploitable Comments

<table>
<thead>
<tr>
<th>Exploitable Toplevel Content Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>Code Relationship</td>
</tr>
<tr>
<td>PastFuture</td>
</tr>
</tbody>
</table>

Exploitable Comment

Can potentially be used by existing work or inspire new work.

52.6 ± 2.9% Comments in the three OSs belong to these four top level content Categories.
**F1:** 22.1% of the exploitable comments describe the usage and meaning of integers and integer macros.
F1: 22.1% of the exploitable comments describe the usage and meaning of integers and integer macros.

- Bits and Bytes

```c
#define IXGB_GPTCL 0x02108 /* Good Packets Transmitted Count */
```
F1: 22.1% of the exploitable comments describe the usage and meaning of integers and integer macros.

- Bits and Bytes

  ```c
  #define IXGB_GPTCL 0x02108 /* Good Packets Transmitted Count */
  ```

- Error Returns

  ```c
  /* return 1 if ACK, 0 if NAK, -1 if error */
  static int slhci_transaction(...) {
  ... 
  }
F2: 16.8% of the exploitable comments specify or emphasize some particular code relationship.
F2: 16.8% of the exploitable comments specify or emphasize some particular code relationship.

- Data Flow
  ```c
  bool vdev_nowritecache;
  /* true if flushwritecache failed */
  ```
F2: 16.8% of the exploitable comments specify or emphasize some particular code relationship.

- **Data Flow**
  
  ```c
  bool vdev_nowritecache;
  /* true if flushwritecache failed */
  ```

- **Control Flow**
  
  ```c
  switch (i) {
    case 0:  printf("0");  break;
    case 1:  printf("1");  break;
    default:  /* Not reached */
  }
  ```
**F5:** At least 10.7% of the exploitable comments can be expressed via annotation languages.
Summary

- Comments are written when programmers have no other way to express their intentions
- Analyzed 1050 comments from three Operating Systems
- 52.6% Comments are exploitable comments
- 10.7% of the exploitable comments can be expressed via annotation languages
Paper
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5070533

CComment: http://opera.ucsd.edu/CComment/

Deputy: http://deputy.cs.berkeley.edu/

Splint: http://www.splint.org/

Sparse: http://sparse.wiki.kernel.org

Article on Lock_Lint:
http://developers.sun.com/solaris/articles/locklint.html
Comment Age & Location

(a) Location

(b) Absolute Age
Non OS Study

Study based on...

- Eclipse (Java)
- MySQL (C, C++)
- Firefox (C, C++)

<table>
<thead>
<tr>
<th>Software</th>
<th>Com%</th>
<th>Exp%</th>
<th>F2</th>
<th>F3</th>
<th>F5</th>
<th>F4</th>
<th>F1</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>5.6%</td>
<td>10.7%</td>
<td>4.7%</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>3.8%</td>
<td>13.7%</td>
<td>1.7%</td>
<td>6.3%</td>
</tr>
</tbody>
</table>