

User Interaction Issues in Defect Detection Tools

Jon Pincus
PPRC, Reliability Tools
Microsoft Research

My first program analysis tool

- Implemented in Prolog
- Input: syntax tree for the program
 - (manually translated into Prolog representation)
- Output: complexity metrics
- Sample session:
> compute_metrics(Program, Metrics)
no.

August 16, 2001

Jon Pincus, Microsoft

Some more recent background: PREfix

- C/C++ defect detection via static analysis
- Powerful inter-procedural analysis
 - Incomplete
 - Unsound
 - Useful in practice
- Typically run as part of a centralized build
- V1.0: command-line tool
 - ... no significant adoption
- V2.0 ... V4.0: whole system (UI, DB)
 - ... broad adoption

August 16, 2001

Jon Pincus, Microsoft

Even more recent background: PREfast

- Lightweight, "desktop" defect detection
- Simple intra-procedural analyses
- Compared to PREfix:
 - fewer defects
 - higher noise
 - fast
- V1.0: UI, XML log; fast adoption
- Infrastructure: plugins
 - New defects (e.g., I18N)
 - Platform for other tools

August 16, 2001

Jon Pincus, Microsoft

PREfast as an experiment

- What had we learned from PREfix?
 - Which defects people care about
 - Build integration techniques
 - User interaction techniques
 - Scalable, powerful analyses
- Replace the analyses with weak, fast ones
 - Leverage build integration, user interaction, knowledge
- Result: surprisingly well accepted

(yeah, yeah, I know, it's not a controlled experiment ...)

August 16, 2001

Jon Pincus, Microsoft

"Analysis is necessary, but by no means sufficient"

- Actual analysis is only a small part of any "program analysis tool".
 - In PREfix, < 10% of the "code mass"
- No matter what the power of the analyses, make sure to consider other aspects as well

August 16, 2001

Jon Pincus, Microsoft

Stepping back a little ...

- Why do people use a tool? If
 - it helps them get their work done ...
 - ... more efficiently than they would otherwise
 - ... without making them look (or feel) bad.

User interaction is key to all of these points

Aside: See Alan Cooper's books, e.g. *About Face*

August 16, 2001

Jon Pincus, Microsoft

Successful tools

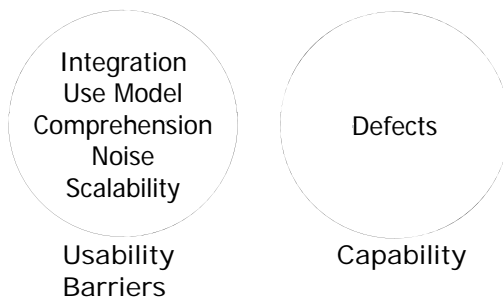
- Provide value ...
- ... at sufficiently low cost
- ... and work in the target environment
 - ... which typically means with "average" developers

User interaction is once again key

August 16, 2001

Jon Pincus, Microsoft

From an internal presentation: Progress in "static analysis"



August 16, 2001

Jon Pincus, Microsoft

Apologies in advance

- My examples are by no means exhaustive
- I'm a tools junkie, not a research junkie
 - Please don't be offended if I fail to cite you!
- Examples are typically drawn from tools I've worked on or used
 - It's just familiarity, not any belief that these are better!

August 16, 2001

Jon Pincus, Microsoft

Key areas of user interaction for defect detection tools

- Specifying what properties to check
- Controlling the analysis
- Dealing with the results of the analysis
 - Viewing individual defects
 - Dealing with noise
 - Managing sets of defects

August 16, 2001

Jon Pincus, Microsoft

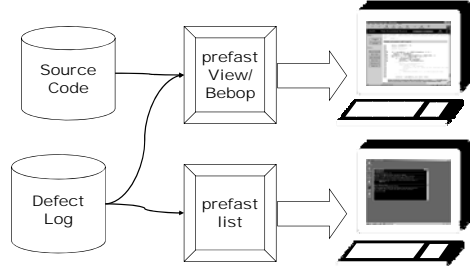
This presentation's focus

- Static analysis tools
- Dealing with the results of the analysis
 - Viewing individual defects
 - Dealing with noise
 - (Managing sets of defects?)

August 16, 2001

Jon Pincus, Microsoft

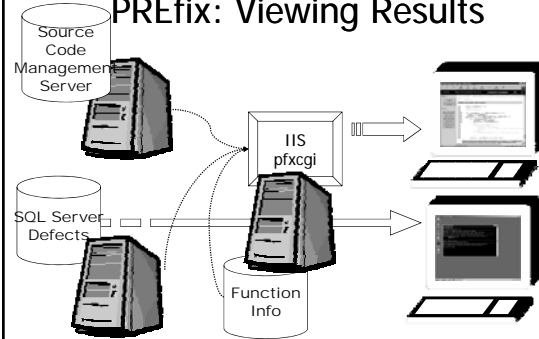
PREfast, SLAM, ESP: Viewing Results



August 16, 2001

Jon Pincus, Microsoft

PREfix: Viewing Results



August 16, 2001

Jon Pincus, Microsoft

Viewing individual defects

- An interesting question:

Why don't static analysis tools report results as clearly and concisely as the compiler?

August 16, 2001

Jon Pincus, Microsoft

Sample compiler-reported defect

```

void f(char *p)
{
    float f;
    f = p;
}

example.c(4) : error C2115: '=' : incompatible types
  
```

August 16, 2001

Jon Pincus, Microsoft

Sample compiler-reported defect

```

#include <set>
std::set<const int> set;

xmemory(57) : error C2535: 'const int *__thiscall
std::allocator<int const >::address(const int &) const': member
function already defined or declared
    xmemory(54) : see declaration of 'address'
    xtree(51) : see reference to class template instantiation
'std::allocator<int const >' being compiled
    set(33) : see reference to class template instantiation
'std::_Tree<int const ,int const ,struct std::set<int const ,struct
std::less<int const >,class std::allocator<int const > >::_Xfn,struct
std::less<int const >,class std::allocator<int const > >' being compiled
    constint.cpp(2) : see reference to class template instantiation
'std::set<int const ,struct std::less<int const >,class
std::allocator<int const > >' being compiled
  
```

August 16, 2001

Jon Pincus, Microsoft

Sample compiler-reported defect

```

SymbolInfo *Find(ISymbolPtr id);
SymbolInfo *Find(char *name);
main() {
    const char* n = "a constant";
    Find(n);
}

comp.h(690) : error C2227: left of '->QueryInterface'
must point to class/struct/union
comp.h(75) : see reference to function template
instantiation 'long __thiscall __com_ptr_t<class
_com_IID<struct ISymbol,&struct __s_GUID
_GUID_bd136bac_431b_4d58_af70_9cec395c3828>
>::QueryInterface(const char *const & )' being compiled
  
```

August 16, 2001

Jon Pincus, Microsoft

Sample compiler-reported defect

```
void f(char *p)
{
float f;
f = p;
}
```

example.c(4) : error C2115: '=' : incompatible types

August 16, 2001

Jon Pincus, Microsoft

Sample compiler-reported defect

```
#include <set>
std::set<const int> set;
```

```
xmemory(57) : error C2535: 'const int * __thiscall
std::allocator<int const >::address(const int &) const': member
function already defined or declared
xmemory(54) : see declaration of 'address'
xtree(51) : see reference to class template instantiation
'std::allocator<int const >' being compiled
set(33) : see reference to class template instantiation
'std::_Tree<int const ,int const ,struct std::set<int const ,struct
std::less<int const >,class std::allocator<int const > >::Kfn,struct
std::less<int const >,class std::allocator<int const > >' being compiled
constint.cpp(2) : see reference to class template instantiation
'std::set<int const ,struct std::less<int const >,class
std::allocator<int const > >' being compiled
```

August 16, 2001

Jon Pincus, Microsoft

Sample compiler-reported defect

```
SymbolInfo *Find(ISymbolPtr id);
SymbolInfo *Find(char *name);
main() {
const char* n = "a constant";
Find(n);
}
```

```
comip.h(690) : error C2227: left of '->QueryInterface'
must point to class/struct/union
comip.h(75) : see reference to function template
instantiation 'long __thiscall _com_ptr_t<class
_com_IID<struct ISymbol,&struct __s_GUID
_GUID_bdl36bac_431b_4d58_af70_9cec395c3828>
>::QueryInterface(const char *const &)' being compiled
```

August 16, 2001

Jon Pincus, Microsoft

Viewing individual defects

- A more relevant question – and answer
- Why don't static analysis tools (including the compiler) report complex defects as clearly and concisely as easy ones?
- Because they're complex, of course!

August 16, 2001

Jon Pincus, Microsoft

Sample defect #1

```
int f(int i)
{
int n;
if (i == 0)
n = 1;
return n;
}
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #1: compiler warning

```
int f(int i)
{
int n;
if (i == 0)
n = 1;
return n;
}
```

```
uwmsrsil.c(6) : warning C4701: local variable 'n'
may be used without having been initialized
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #1: PREFIX warning

```
int f(int i)
{
int n;
if (i == 0)
    n = 1;
return n;
}
uwmsrsil.c(6):warning 1: using uninitialized memory 'n'
uwmsrsil.c(3) : stack variable declared here
Problem occurs when the following condition is true:
uwmsrsil.c(4) : when 'i != 0' here
Path includes 2 statements on the following lines:
    4 6
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #1: PREfast/PREFIX GUI

August 16, 2001

Jon Pincus, Microsoft

Sample defect #1: Comments

- Compiler warning doesn't give enough info
 - *When* is n uninitialized?
 - As a result, often not fixed
- Key techniques
 - Additional detailed information
 - Path
 - Link to additional information
- There's an information management problem
 - Even in simple example, output is very verbose

August 16, 2001

Jon Pincus, Microsoft

Sample defect #2

```
extern int phase_of_moon(void);

/* initialize the buffer; or return failure */
static int initialize(char *buff)
{
if (phase_of_moon())
    return 0;
buff[0] = 0;
return 1;
}
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #2 (cont.)

```
void uwmsrsi2(void)
{
char buff[100];
initialize(buff);
if (buff[0] != 0)
    return;
/* ... */
}
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #2 : PREFIX message

```
void uwmsrsi2(void)
{
char buff[100];
initialize(buff);
if (buff[0] != 0)
    return;
/* ... */
}

uwmsrsi2.c(17) : warning 1: using uninitialized memory
'(buff)[0]'
problem occurs in function 'uwmsrsi2'
uwmsrsi2.c(15) : stack variable declared here
Path includes 2 statements on the following lines:
    16 17
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #2: PREFIX GUI

August 16, 2001

Jon Pincus, Microsoft

Sample defect #2: Comments

- Where's the defect?
 - The failure to initialize **buff** before calling **initialize**?
 - **initialize**'s failure to initialize **buff** in the error case?
 - The fact that **initialize** can fail?
 - The failure to check the return value of **initialize**?
 - What if the caller of **uwmsrsi2** has previously verified that **phase_of_moon() == 0**?
- Key technique:
 - Navigation across functions
 - (But how does the user know where to look?)
- Is there a way to present this in a single screen?

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3

```
...  
CHAR buff[MAX_PATH];  
GetWindowsDirectory(buff, sizeof(buff));  
SetCurrentDirectory(buff, sizeof(buff));
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: original PREFIX message

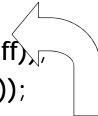
```
...  
CHAR buff[MAX_PATH];  
GetWindowsDirectory(buff, sizeof(buff));  
SetCurrentDirectory(buff, sizeof(buff));  
  
Warning: using uninitialized memory buff[0]  
Problem occurs in call to SetCurrentDirectory
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: explanation

```
...  
CHAR buff[MAX_PATH];  
GetWindowsDirectory(buff, sizeof(buff));  
SetCurrentDirectory(buff, sizeof(buff));
```



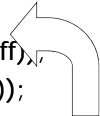
*GetWindowsDirectory can fail,
in which case it doesn't initialize its output*

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: explanation

```
...  
CHAR buff[MAX_PATH];  
GetWindowsDirectory(buff, sizeof(buff));  
SetCurrentDirectory(buff, sizeof(buff));
```



*GetWindowsDirectory can fail,
in which case it doesn't initialize its output
But most people don't think of this, so
think the message is noise*

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: revised PREfix message

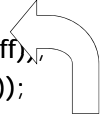
...
CHAR buff[MAX_PATH];
GetWindowsDirectory(buff, sizeof(buff));
Warning: Failure to check return value
SetCurrentDirectory(buff, sizeof(buff));

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: explanation

...
CHAR buff[MAX_PATH];
GetWindowsDirectory(buff, sizeof(buff));
SetCurrentDirectory(buff, sizeof(buff));



*GetWindowsDirectory can fail,
So the return value must be checked
But most people don't BELIEVE this, so still
think the message is noise*

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: re-revised PREfix message

...
CHAR buff[MAX_PATH];
GetWindowsDirectory(buff, sizeof(buff));
Warning: Failure to check return value
GetWindowsDirectory can fail in low-memory
situations
SetCurrentDirectory(buff, sizeof(buff));

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: What I'd really like to see

*PREfix' display mechanisms don't yet support this,
but I'd rather see something like ...*

CHAR buff[MAX_PATH];
GetWindowsDirectory(buff, sizeof(buff));
Warning: Failure to check return value
GetWindowsDirectory can fail in low-memory situations
SetCurrentDirectory(buff, sizeof(buff));
Warning: using uninitialized memory buff[0]
Problem occurs in call to SetCurrentDirectory
Due to previous failure to check return value

August 16, 2001

Jon Pincus, Microsoft

Sample defect #3: Comments

- Key technique:
 - Iterative, detailed, defect-specific info
- But can the tool know all of that up front?
 - I tend to think not – instead, provide hooks for people to provide this information

August 16, 2001

Jon Pincus, Microsoft

PREfast “defect description”

- An XML description of each defect, with
 - Brief description (mandatory; everything else is optional)
 - Additional details
 - Effect of the defect
 - Hypothesis about cause (phrased as question)
 - Severity
 - One or more examples (erroneous and corrected code)
 - Documentation (as XHTML, RTF, or reference)
 - Help URL for more information
 - Owner's e-mail address
 - GUID

August 16, 2001

Jon Pincus, Microsoft

Sample defect #4

```
void uwmsrsi4(LPCTSTR in) {
TCHAR buff[100];
_tcsncpy(buff, in, sizeof(buff));
/* ... */
}
```

TCHAR is typedef'ed as either char or wchar_t, depending on whether UNICODE is defined
_tcsncpy expands to either strncpy or wcsncpy

August 16, 2001

Jon Pincus, Microsoft

Sample defect #4 PREFIX 3.5 message

```
void uwmsrsi4(LPCTSTR in) {
TCHAR buff[100];
_tcsncpy(buff, in, sizeof(buff));
/* ... */
}
uwmsrsi4.c(10) : warning 23: bounds error
(overflow): 'buff'
used as parameter 1 (dest) of call to 'wcsncpy'
size of buffer 'buff' is 200 bytes
reference is 399 bytes from start of buffer
uwmsrsi4.c(9) : stack variable declared here
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #4 PREFIX 4.0 message

```
void uwmsrsi4(LPCTSTR in) {
TCHAR buff[100];
_tcsncpy(buff, in, sizeof(buff));
}
```

```
uwmsrsi4.c(10) : warning 51: using number of bytes
instead of number of characters for 'buff'
used as parameter 1 (dest) of call to 'wcsncpy'
size of buffer 'buff' is 200 bytes
reference is 399 bytes from start of buffer
uwmsrsi4.c(9) : stack variable declared here
problem occurs when the following condition is true:
uwmsrsi4.c(10) : when 'wcslen(in) >= 200'
during call to 'wcsncpy' here
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #4: Comments

- Key technique:
 - Additional precision in the warning message
- The fact that it's a *stack* buffer is important for prioritizing the warning
 - There are well-known techniques for exploiting stack buffer overruns

August 16, 2001

Jon Pincus, Microsoft

Sample defect #5

```
#define CHECK(hr) \
{if (! SUCCEEDED(hr)) goto bail; }

int uwmsrsi_macro(void) {
int i;
CHECK(initialize(&i));
return ++i;

bail:
return 0;
}
```

August 16, 2001

Jon Pincus, Microsoft

Sample defect #6

```
int uwmsrsi_loop(void) {
int arr[5];
int i, total = 0;

arr[0] = 3; arr[1] = 1;
arr[3] = 4; arr[4] = 2;

for (i = 0; i < 5; i++)
total += arr[i] * i;
return total;
}
```

August 16, 2001

Jon Pincus, Microsoft

Additional complexities

- Templates
- Value derivation
- Recursion
- Long functions
- Source code changes

August 16, 2001

Jon Pincus, Microsoft

What about more complex properties?

- Race conditions
- Deadlocks
- Arbitrary properties

August 16, 2001

Jon Pincus, Microsoft

SLAM “debugger” UI

August 16, 2001

Jon Pincus, Microsoft

Summarize vs. navigate?

- Ideally, want to summarize and hide unimportant details
 - But which details are important?
- Developers often think sequentially ...
 - ... even when this is less efficient

August 16, 2001

Jon Pincus, Microsoft

Noise

- Noise = “messages people don’t care about”
 - (not just “bogus” messages)
- Usually, noise is worse than missing a defect

Too much noise

=> people won’t use the tool

== missing *all* the defects

August 16, 2001

Jon Pincus, Microsoft

Noise can result from

- Incorrect requirements
- Integration issues
- Usability issues (e.g., unclear messages)
- Parser incompatibilities
- Analysis inaccuracies
- ...

August 16, 2001

Jon Pincus, Microsoft

Dealing with noise

- Improving analysis is usually not sufficient
 - Sometimes, it's necessary
- Successful user interaction techniques:
 - Prioritization
 - History
 - Filtering
 - Improving presentation, navigation
 - Providing more detail

August 16, 2001

Jon Pincus, Microsoft

Message Prioritization

- Which messages correspond to defects that will actually be fixed?
- "Rank": a synthetic metric of a message's "goodness"
 - Better-ranking messages are more likely to identify defects that will actually get fixed
- Multiple dimensions:
 - Severity of consequences
 - Likelihood that message is correct
 - Comprehensibility of message
 - ...

August 16, 2001

Jon Pincus, Microsoft

Noise and history

- Noise naturally increases over time
 - People fix the real defects
- A history mechanism avoids these problems
 - Distinguish newly-occurring messages
 - Goal: avoid re-examining noise messages
 - "Fuzzy" notion of equality survives code changes
- Code modification is also possible

August 16, 2001

Jon Pincus, Microsoft

Noise Example

- In 1999, 30% of PREFIX-reported bugs in the "RAID" bug tracking system were being misdiagnosed
 - Real defects rejected as bogus
 - Unnecessary fixes for false messages
- Why? A surprising reason:
 - The URL describing a PREFIX message wasn't self-descriptive, so couldn't be stored in RAID
 - Most people weren't even seeing the PREFIX UI!
- Solution:
 - Make the URL's self-descriptive (duh)
 - Automatically store specific information and the URL as part of the RAID entry
- Result: misdiagnosis rate dropped below 5%

August 16, 2001

Jon Pincus, Microsoft

Approaches to sets of defects

- List
- GUI
 - List
 - Tabular
 - Tree
 - Filtering, Sorting, ...
- Database and query language
- Statistical analyses

August 16, 2001

Jon Pincus, Microsoft

Example: sets of defects

- Textual compiler output
- Compiler output in a dev environment
 - (emacs, Visual Studio, ...)
- Typo.pl: tabular view
- PREFIX: tree/list, with filtering and sorting
 - Note: PREFIX 4.0 moves to a tabular view
- PREFIXfast: tabular, filtering

August 16, 2001

Jon Pincus, Microsoft

Observations

- Others have looked at this problem
 - Spreadsheets
 - OLAP/Data mining
 - "Information agent" work
- Improvements here are very highly leveraged
 - Reducing the number of defects to examine by an order of magnitude is much easier than reducing the time to examine a single defect by an order of magnitude ...
- Prioritization and statistical analysis can make a big difference

August 16, 2001

Jon Pincus, Microsoft

User Interaction Issues in Defect Detection Tools

Jon Pincus
PPRC, Reliability Tools
Microsoft Research