Recovering System Specific Rules from Software Repositories

Chadd Williams
Jeff Hollingsworth

Problem

- How much do you know about your 10 year old code base?
  - didn't someone rewrite the matrix objects?
    - how do you transform an image now?
- Implicit rules build up over time
  - little or no documentation
  - failure to understand implicit rules causes bugs
    - 32% of bugs detected during maintenance[1]
- We can discover implicit rules by looking at code changes

[1] Matsumura, T., Monden, A., Matsumoto, K., The Detection of Faulty Code Violating Implicit Coding Rules, IWPSE '02

Implicit Rule

- Function Usage Pattern
  - how functions are invoked with respect to each other in the source code
  - describe *relationships* between functions
  - static analysis - intraprocedural

```
HDC hdc = BeginPaint( hwnd, &ps );
if( hdc )
    DrawIcon( hdc, x, y, hIcon );
EndPaint( hwnd, &ps );
```

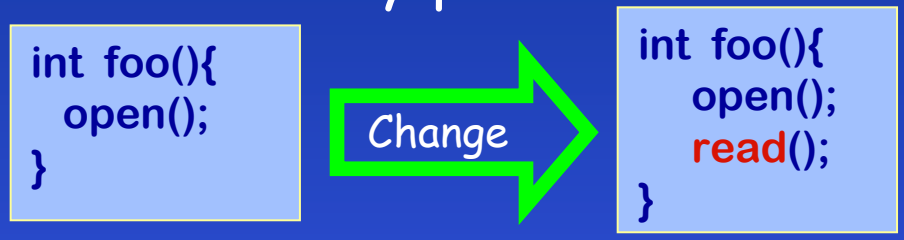
Called After

```
mdi =  HeapAlloc(GetProcessHeap());
if (!mdi)
   HeapFree(GetProcessHeap(), 0, cs);
```

Conditionally Called After

Function Usage Pattern Miner

- Find new instances of relationships
  - where that instance was not found in the revision immediately prior

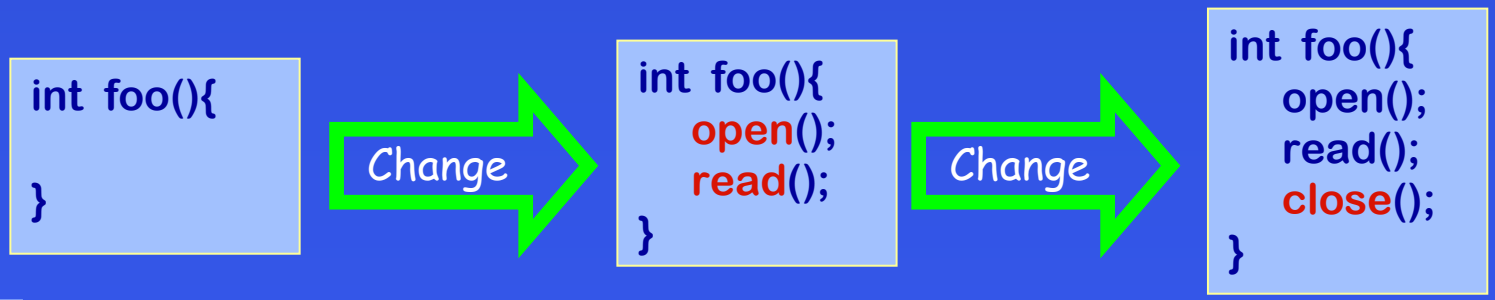


new *instance* of read() *called after* open()

- Preliminary filtering heuristic
  - function calls within 10 source lines of code
    - many APIs contain functions that are called in quick succession
    - error handling is near error producing function

Classification of Mined Data

- Each mined instance is classified by how it entered the source code:
  - both of the function calls were added
    - instance added in full
  - one function call was added
    - the added function completed the pairing
    - bug fix? refactoring?
  - neither of the function calls were added
    - deleted code? control flow change?



Rating Mined Relationships

- Determine support and confidence for each mined relationship
  - *confidence* of foo() -> bar()
    - in what percent of instances that start with foo(), is foo() follow by bar() ?
  - *support* of foo() -> bar()
    - what percent, of all instances found, are foo() -> bar() ?
  - present a sorted list to the user
    - sort on support then confidence

Preliminary Case Study

- Mined Wine CVS repository
  - 15,666 unique relationships added > 9 times
  - 862 unique relationships added > 99 times
- What relationships are found in CVS?
  - how was it added to the source code?
  - compare to relationships in the latest version of the source code
- How can this help us find bugs?
- Can we mine data for a specific API?

How do the Top 25 of the lists differ?

- Most similar to latest version
  - added both function calls
    - sum of differences in ranking: 91
    - items unique to one list: 8
- Least similar to latest version
  - added one function call
    - sum of differences in ranking: 41
    - items unique to one list: 28

Relationships found in the Latest Version of the Source Code

Called After Relationship		COUNT
fprintf	fprintf	12671
VariantChangeTypeEx	VariantChangeTypeEx	6700
GetProcAddress	GetProcAddress	3605
HeapFree	GetProcessHeap	3577
printf	printf	3098
HeapAlloc	GetProcessHeap	2851
memcmp	memcmp	2294
GetProcessHeap	GetProcessHeap	1985
GetProcAddress	VariantChangeTypeEx	1747
GetDlgItem	GetDlgItem	1742

Relationships Created By Adding One Function Call

Called After Relationship		COUNT
fprintf	fprintf	2606
RtlFreeHeap	GetProcessHeap	1782
RtlAllocateHeap	GetProcessHeap	1251
HeapFree	GetProcessHeap	1200
GetProcAddress	GetProcAddress	1100
HeapAlloc	GetProcessHeap	816
GetProcessHeap	RtlFreeHeap	768
GetProcessHeap	HeapFree	480
memcmp	memcmp	342
GetProcessHeap	GetProcessHeap	233

What relationships were found?

- EnterCriticalSection -> LeaveCriticalSection
  - in latest version: 939 times
- How were the instances created?
  - add both function calls: 1,277 times
  - add one function call: 5 times
  - added one function but did not complete the pairing: 82 times
    - **78 of these uncompleted pairings were because of the 10 line heuristic**

```
EnterCriticalSection( &(This->lock) );
uRef = ++(This->ref);
if (This->driver)
    IDsCaptureDriver_AddRef(This-
>driver);
LeaveCriticalSection( &(This->lock) );
```

How can this help us find bugs?

- Profile of a bug plagued relationship
  - created often by adding one function call
  - rarely created by adding two function calls
- Possible bug
  - TREEVIEW_UpdateScrollBars -> TREEVIEW_Invalidate
  - update the scroll bars after adding items
  - invalidate the Treeview so it gets redrawn

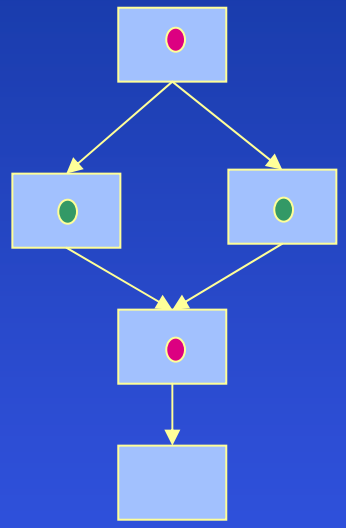
```
for ( Each Item In the List ) {
   TREEVIEW_DrawItem(infoPtr, hdc, wineItem);
}

TREEVIEW_UpdateScrollBars (infoPtr);
. . .
return;
```

Mining Relationships for an API

- What relationships are found between functions declared in an API?
- msiquery.c - database access API
  - two sets of functions:
    - Msi*Foo*(, LPCSTR,) and MSI_*Foo*(, LPCWSTR,)
  - MsiDatabaseOpenViewA -> MsiViewExecute
  - MSI_DatabaseOpenViewW -> MSI_ViewExecute
- Heap access functions
  - HeapAlloc(GetProcessHeap(), . . .)
  - HeapAlloc() -> HeapFree()

Future Work

- Apply our tool to more projects
  - projects that use a common external library
- Track removed usage patterns
- Better filtering heuristic
  - control flow based
  - data flow based
- How do we use the patterns we find?
  - documentation
  - feed patterns to static source code checkers to find violations



```
hdc = BeginPaint( hwnd, &ps );
if( hdc )
    DrawIcon( hdc, x, y, hIcon );
EndPaint( hwnd, &ps );
```

Backup Slides

How do the Top 25 of the lists differ?

- Difference metric
  - distance between rankings of common items
  - number of items unique to each list
- Most similar to latest version
  - Added both function calls
    - sum of differences in ranking: 50
    - items unique to one list: 18
- Least similar to latest version
  - Added one function call
    - sum of differences in ranking: 12
    - items unique to one list: 48

Source Code Change History

- We can discover implicit rules by looking at code changes
  - every change is committed
  - changes highlight misunderstood code
  - changes highlight new code
- Studying each commit gives fine-grain knowledge
  - how quickly does a rule emerge?
  - how fast is a rule adopted?
  - how often is it used later?

Debug functions in Wine

- Many of the relationships involve a debug statement
  - overwhelmed the rest of the results
  - filtered from the data
  - future work:
    - what can we determine about the proper use of debug statements?

```
if (RegOpenKeyA(HKEY, name, &key)) {
   RegCloseKey(key);
    TRACE(message);
}
```

Relations highlighted by CVS mining

- Data Flow Functionality
  - GetDlgItem -> EnableWindow

```
case WM_USER:
    EnableWindow (GetDlgItem (…), FALSE);
    EnableWindow (GetDlgItem (…), FALSE);
    EnableWindow (GetDlgItem (…), FALSE);
    SetFocus (GetDlgItem (hwnd, IDC_TOOLBARBTN_LBOX));
    return TRUE;
```

Conditionally Called After

- 3,872 unique patterns added 10 or more times
- Error handling code
  - conditionally report error
  - which functions need errors handled
- Debug code
  - conditionally call a debug function

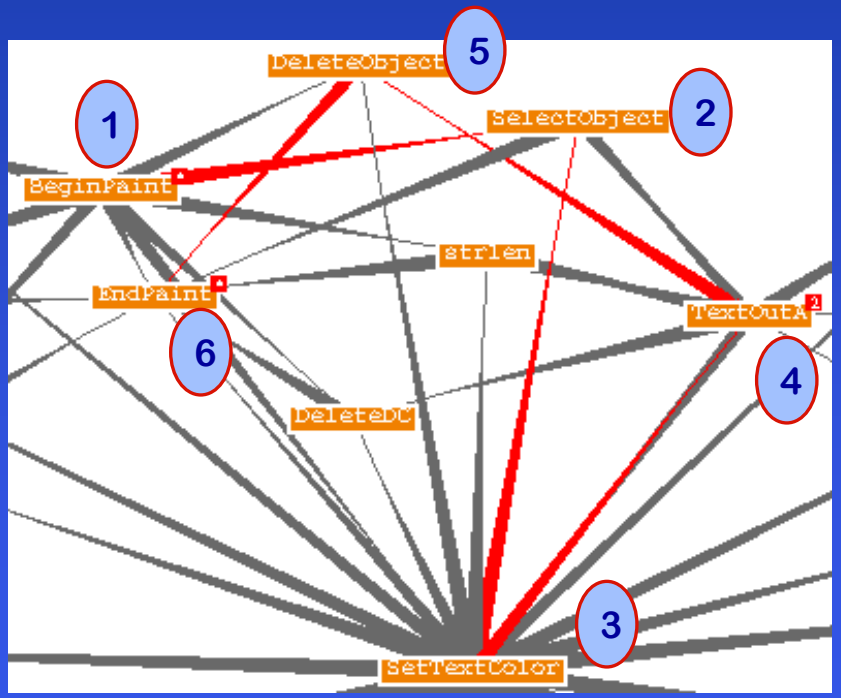
```
if (!(hModule = LoadLibraryExA(fileName, 0, LLDF)))
    WINE_ERR("LoadLibraryExA (%s) failed, %ld\n", fileName, GetLastError());
```

Transitive Patterns

- *called after* may be a transitive pattern
  - only a binary pattern
  - allow larger patterns to be built
  - may need to add more context information

Patterns Identified

SelectObject	*called after*	BeginPaint
SetTextColor	*called after*	SelectObject
TextOutA	*called after*	SetTextColor
DeleteObject	*called after*	TextOutA
EndPaint	*called after*	DeleteObject



Chains of relationships

- Search through the relationships
  - relationships created by adding two functions
  - find relationships of high confidence and support such that:

GetDlgItem() -> GetDlgItem ()

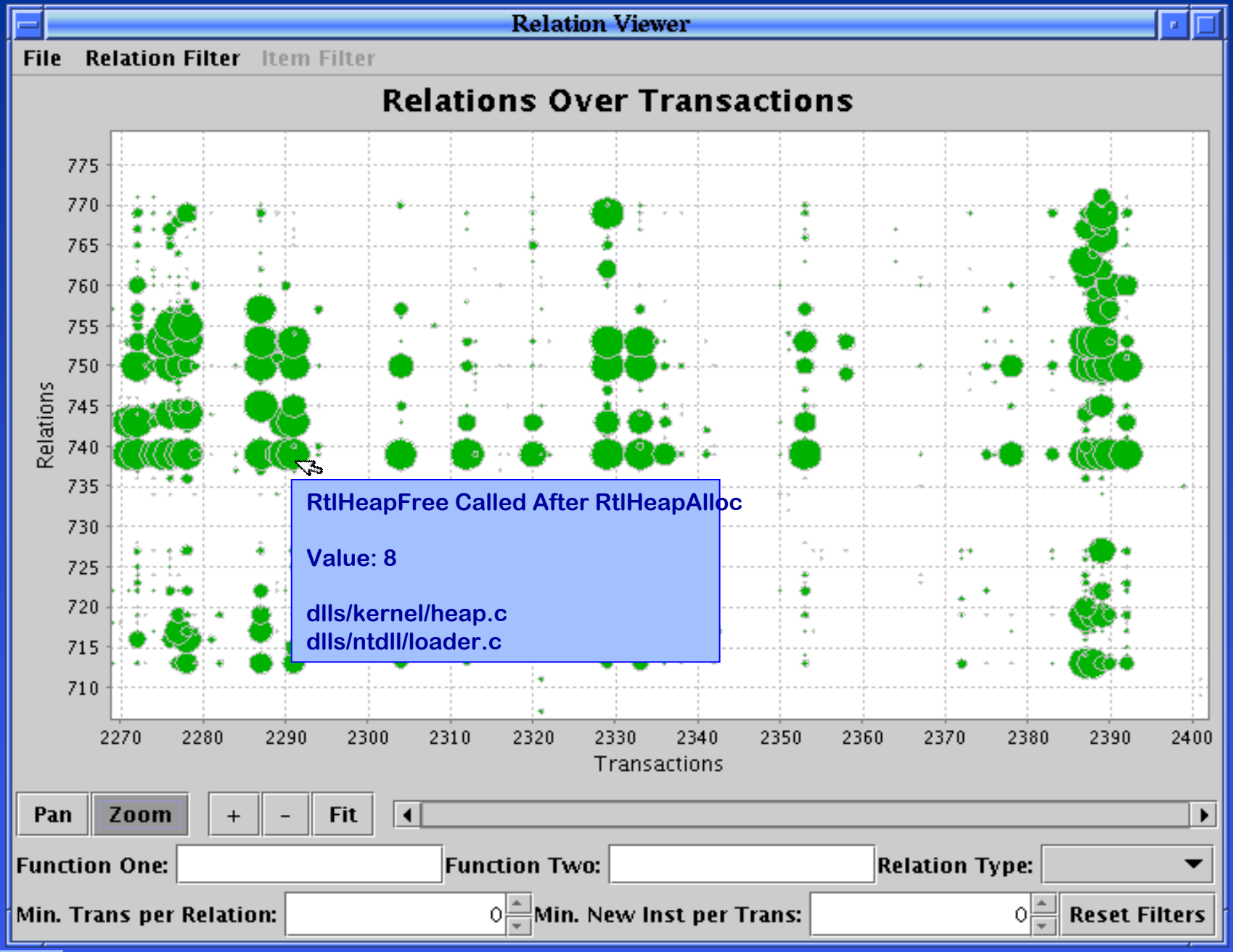
GetDlgItem() -> EnableWindow () **EnableWindow () -> GetDlgItem()**

GetDlgItem() -> EnableWindow () -> GetDlgItem()

```
case WM_USER:
    EnableWindow (GetDlgItem (…), FALSE);
    EnableWindow (GetDlgItem (…), FALSE);
```

- Data flow functionality
  - LoadCursorA -> RegisterClassA
    - in latest version: 42 times
    - add both function calls: 43 times

```
wClass.hCursor = LoadCursorA (...);
RegisterClassA (&wClass);
```


Relation Viewer
File
Relation Filter
Item Filter
Relations Over Transactions
Relations
775
770
765
760
755
750
745
740
735
730
725
720
715
710
2270
2280
2290
2300
2310
2320
2330
2340
2350
2360
2370
2380
2390
2400
Transactions
RtlHeapFree Called After RtlHeapAlloc
Value: 8
dlls/kernel/heap.c
dlls/ntdll/loader.c
Pan
Zoom
+
-
Fit
Function One:
Function Two:
Relation Type:
Min. Trans per Relation:
0
Min. New Inst per Trans:
0
Reset Filters