

LOCATING REGRESSION BUGS

Amiram Yehudai

Shmuel Tyszberowicz

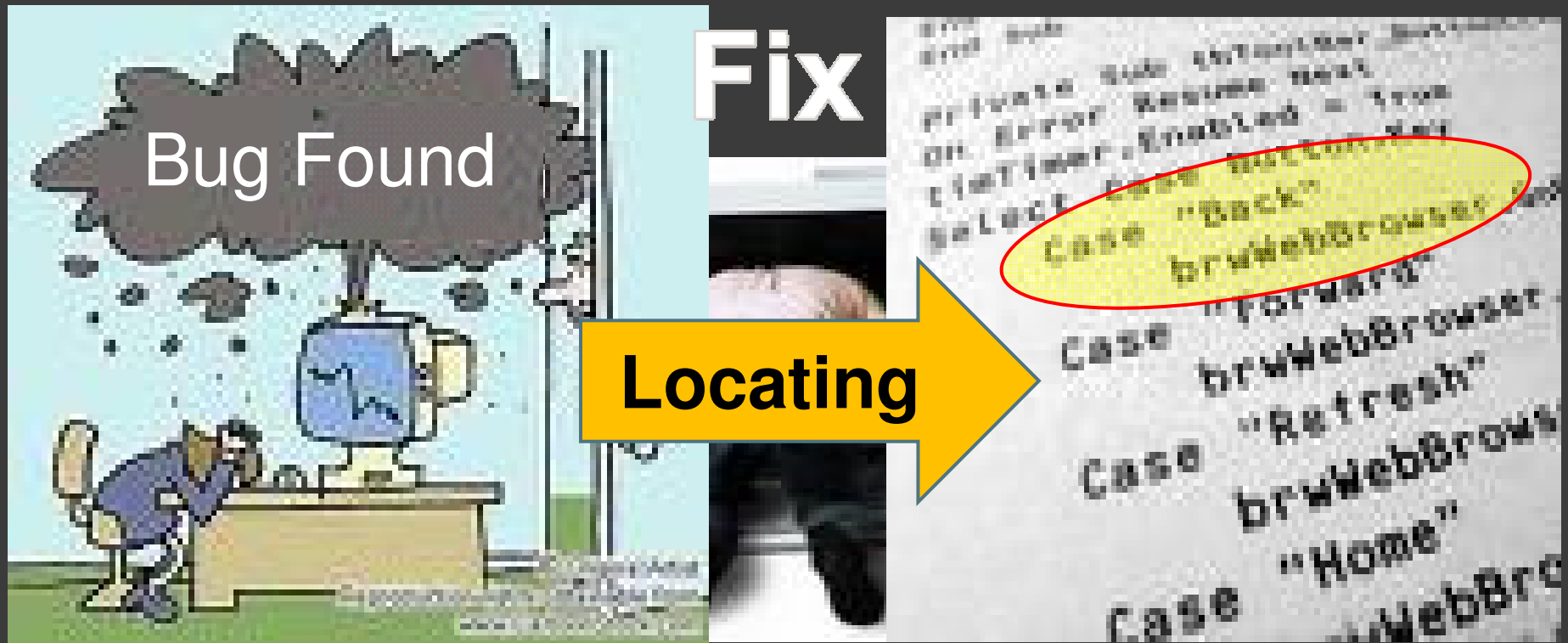
Dor Nir

Outline

- ⦿ Problem introduction.
- ⦿ Regression bug – definition.
- ⦿ Proposed solution.
- ⦿ Experimental results.
- ⦿ Future work.

Problem Introduction

- Given **existence** of a bug, we want to locate the place in the source code that causes the bug.



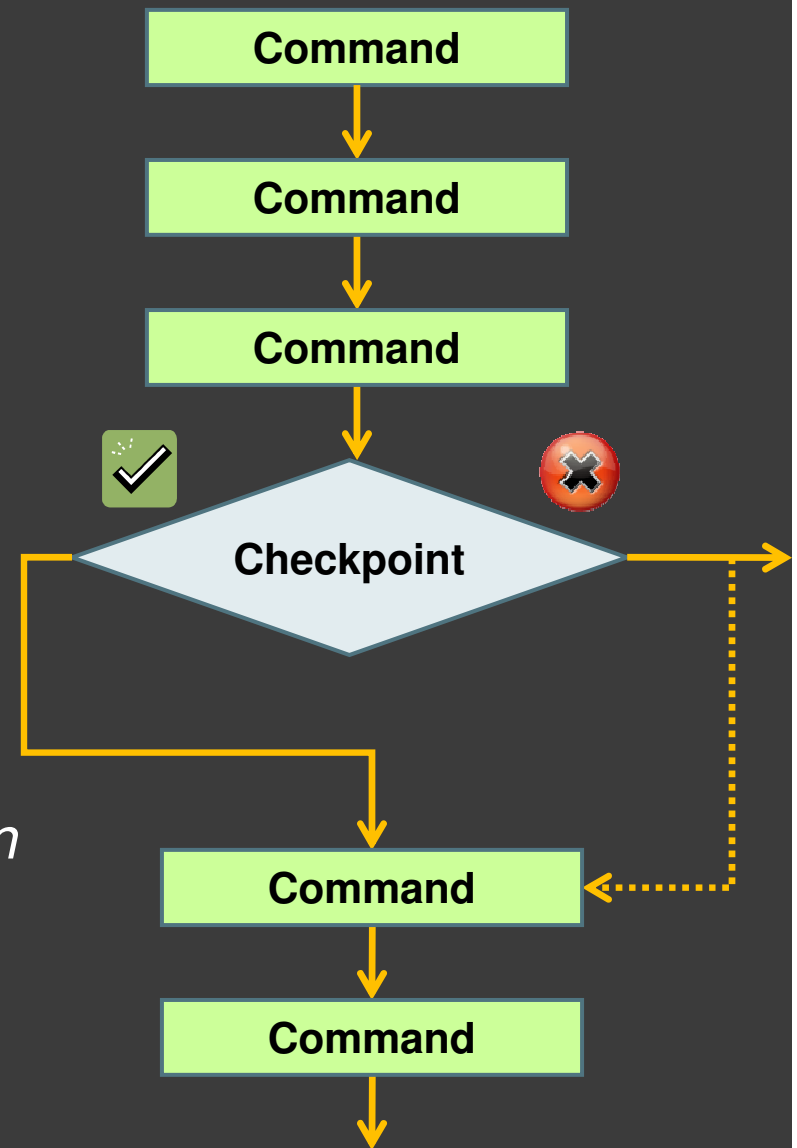
Test case structure

● Command:

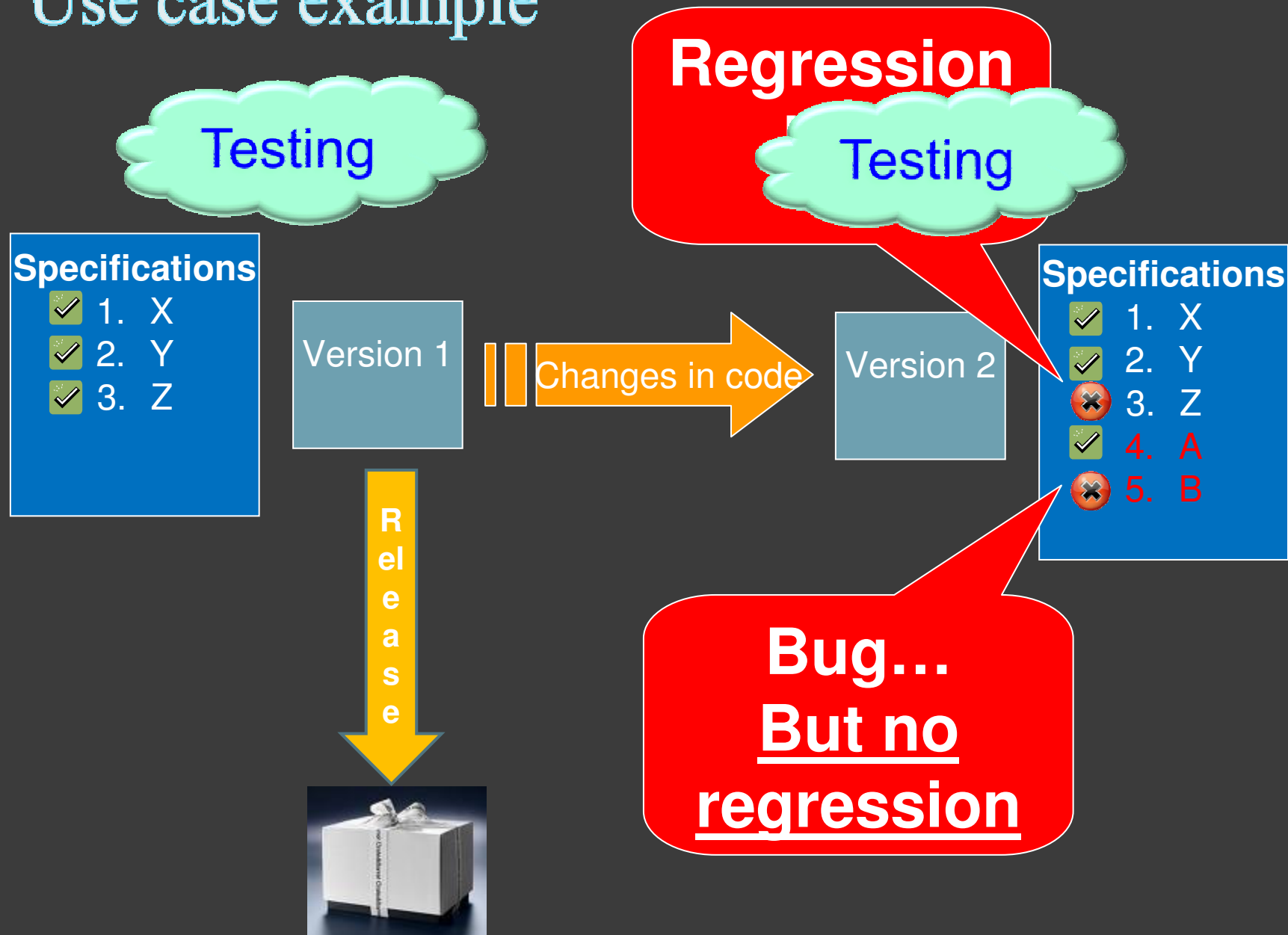
- *Click on the button.*
- *Enter a text to an edit box.*
- *Insert a record to DB.*
- ...

● Checkpoint:

- *Check that the button is enabled.*
- *Check that a certain text is shown in the edit box.*
- *Check the value in a record of a DB.*
- ...



Use case example



Regression bug definition

- Regression bugs occur whenever software functionality that **previously worked** as desired **stops working**, or no longer works as planned.
- Typically regression bugs occur as an **unintended** consequence of program changes.

Where is it?



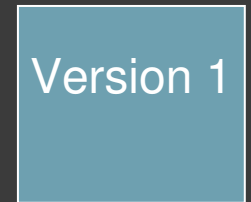
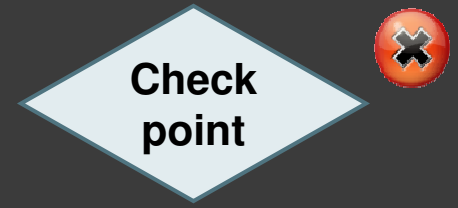
- What is the cause for the regression bug?



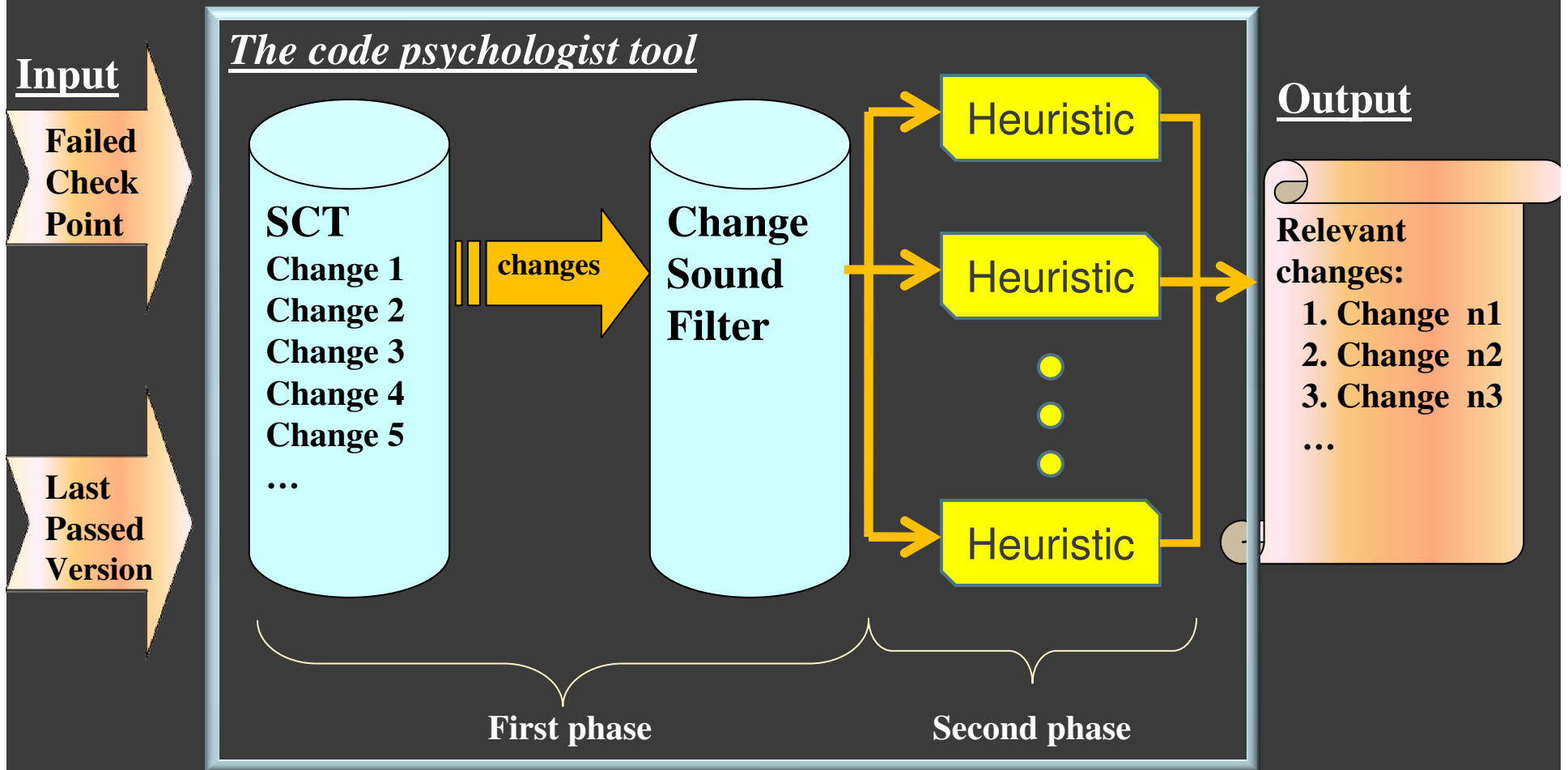
- What is the change that causes the regression bug?

Problem definition

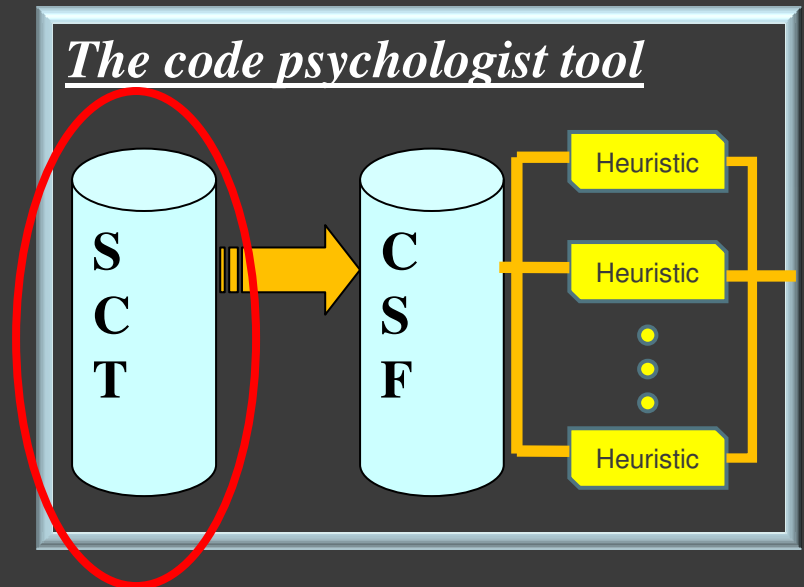
- **C** – A checkpoint that failed when running a test-case.
- **V** - last version of the AUT where checkpoint C **still passed** when running the test-case.
- We want to find in the source code of the AUT the locations $p_1, p_2 \dots p_n$ that caused C to fail.



CodePsychologist tool

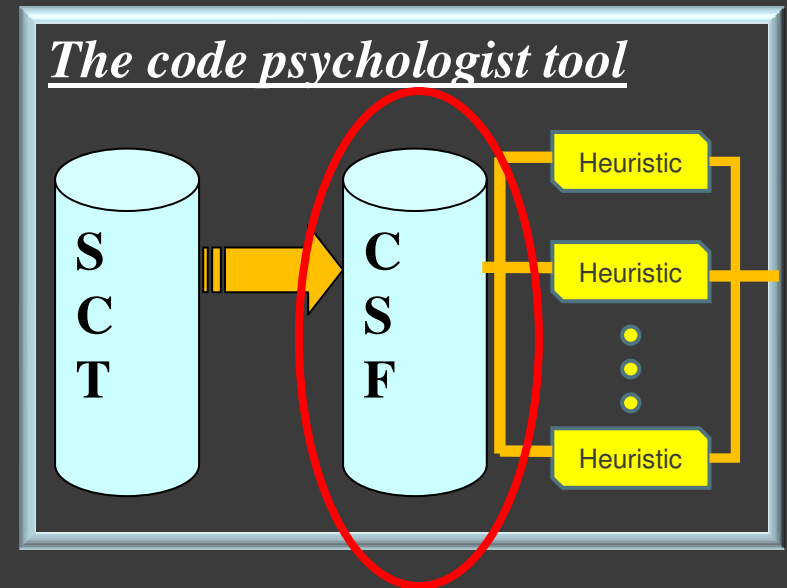


Source Control Tool



- Data Base of source code.
- Very common in software development.
- Check-in / Check-out operation.
- History of versions.
- Differences between versions.
- Retrieve changes submitted after version V.
- Amount of retrieved changes can be large.

Changes Sound Filter



- Retrieving relevant changes.
- Soundness— The output of the CSF must contains the changes that cause the regression bug.

CSF (Continue)

Mapping table:

Tests

Check text in
message box

File t.xml was
Created successfully

“SELECT NAMES from
Table1” is not empty

Source code

Windows.cpp



errMessages.cpp



File.cs



IO.cs



C:\code\windows



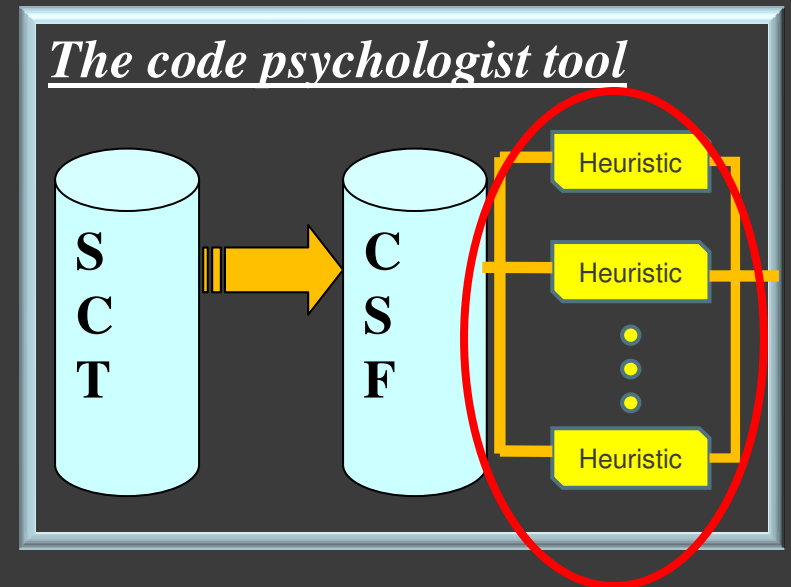
DB project



CSF (Continue)

- Filtering refactoring changes.
- Changes in comments.
- Using profiler information to filter irrelevant changes.
 - Code that was not executed could not cause the regression.

Second phase



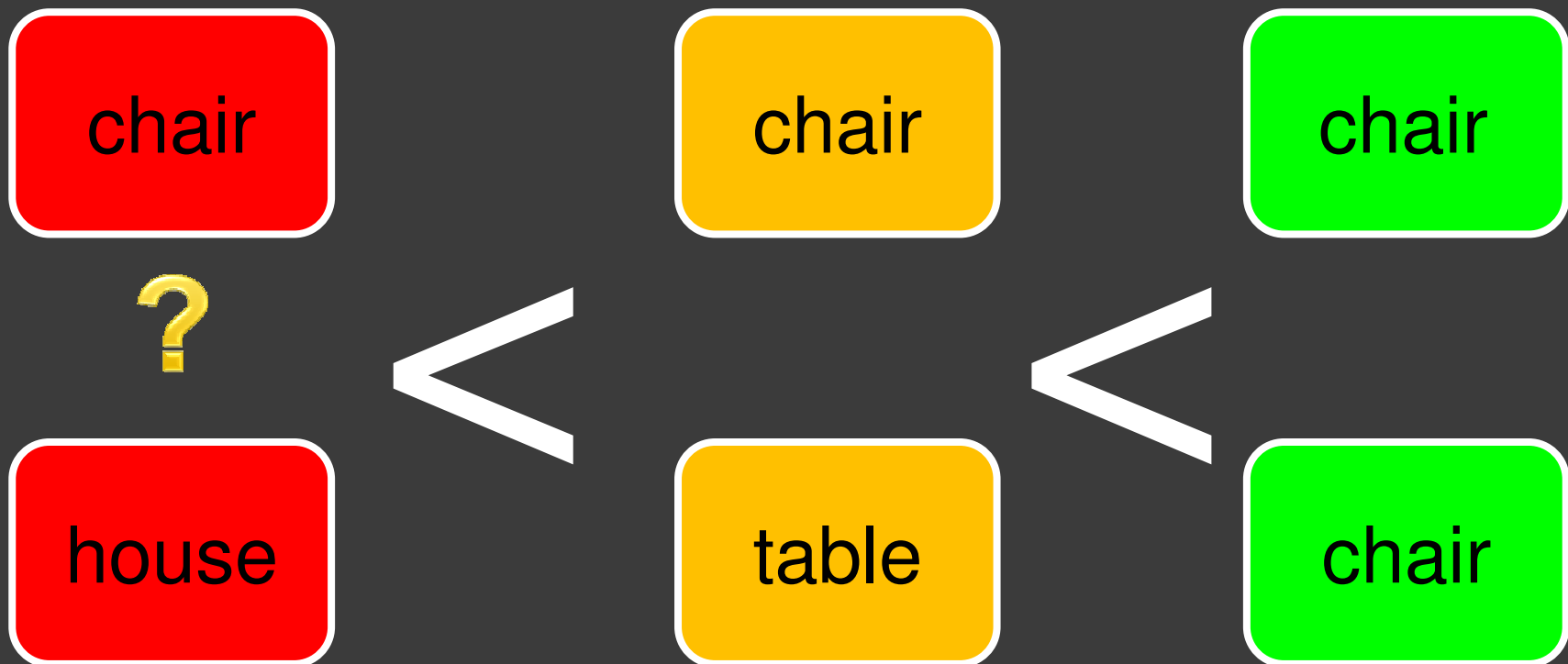
- Rank changes.
- Not conservative.
- Each heuristic has different weight.

$$Rank(p) = \sum_{i=1}^{i=|H|} \alpha_i \cdot HeuristicRank_i(p)$$



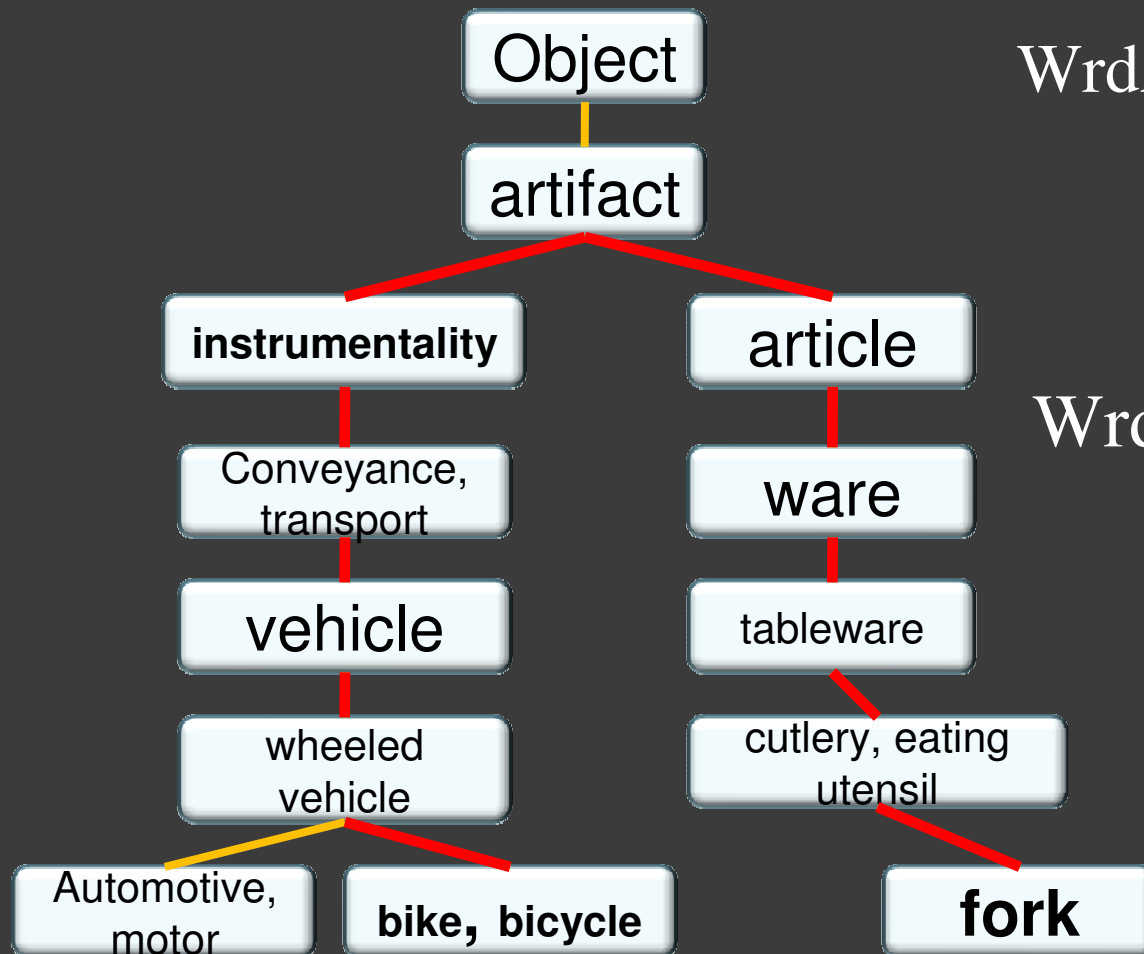
Affinity Ranking

- Affinity – “Close connection marked by similarity in nature or character”
- Measure affinity between words.



Affinity Ranking (Continue)

$$\text{WrdAff}(a,b) = \frac{1}{\text{Distance}(a,b)}$$



$$\text{WrdAff}(bike, fork) = \frac{1}{10}$$

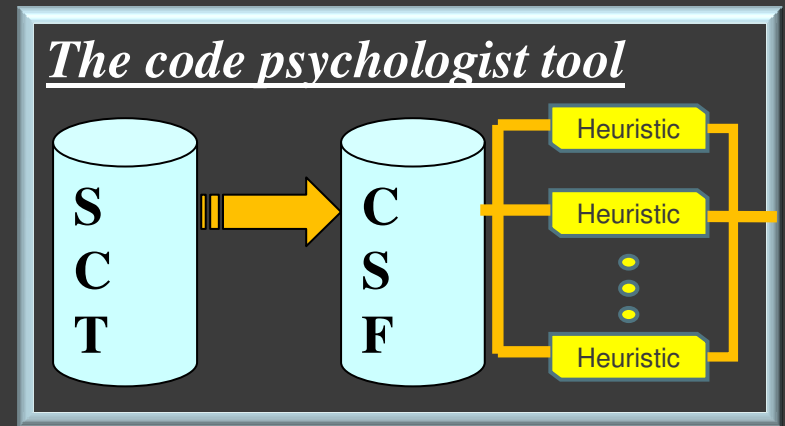
Affinity Ranking (Continue)

$$\text{WrdAff}(a,b) = \frac{1}{\text{Distance}(a,b)}$$

$$\text{AsyGrpAff}(A,B) = \frac{1}{n} \cdot \sum_{i=1}^n \max \{ \text{WrdAff}(a_i, b_j) \mid 1 \leq j \leq m \}$$

$$\text{GrpAff} = (\text{AsyGrpAff}(A,B) + \text{AsyGrpAff}(B,A))/2$$

Heuristics (Second phase)



- Code Lines Affinity. ★
- Check-in comment affinity.
- File Affinity.
- Function Affinity.



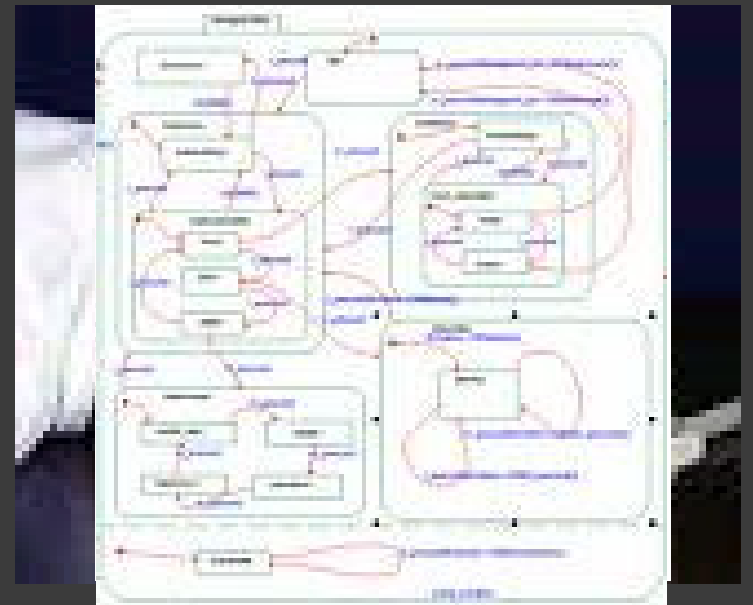
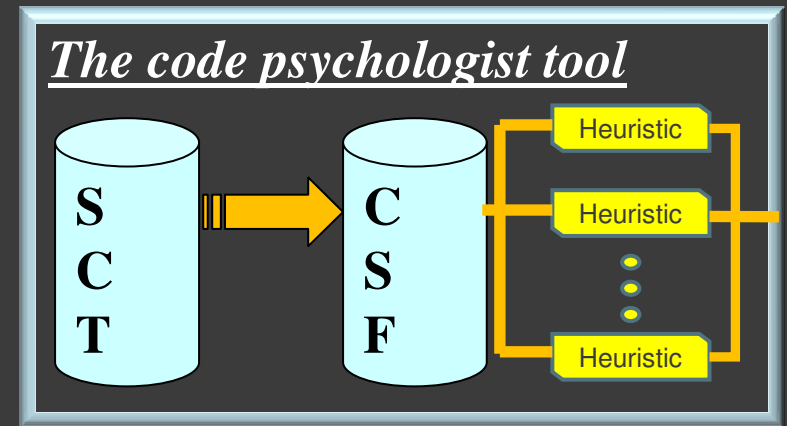
Heuristics (Continue)

● Human factor:

- Programmers history.
- Time of change.
 - Late at night.
 - Close to release deadline.

● Code complexity

- Number of Branches.
- Concurrency.



Synthetic Experiments

- ⦿ C++.
- ⦿ MFC framework.
- ⦿ 891 files in 29 folders.
- ⦿ 3 millions lines of code.
- ⦿ Visual source safe.
- ⦿ 3984 check-ins.

Synthetic Experiments (Continue)

Results:

Bug	Heuristic				Average	
	Code Lines	Check-in	File Affinity	Functions	Simple	Weighted
1	5	3	9	1	1	1
2	-	1	24	-	7	3
3	5	3	3	1	1	1
4	-	-	-	6	6	5
5	2	1	4	1	4	1

Results with file grouping:

Bug	Heuristic				Average	
	Code Lines	Check-in	File Affinity	Functions	Simple	Weighted
1	1	3	9	1	1	1
2	9	1	24	-	3	2
3	3	3	3	2	1	1
4	-	-	-	3	9	4
5	1	1	4	1	1	1

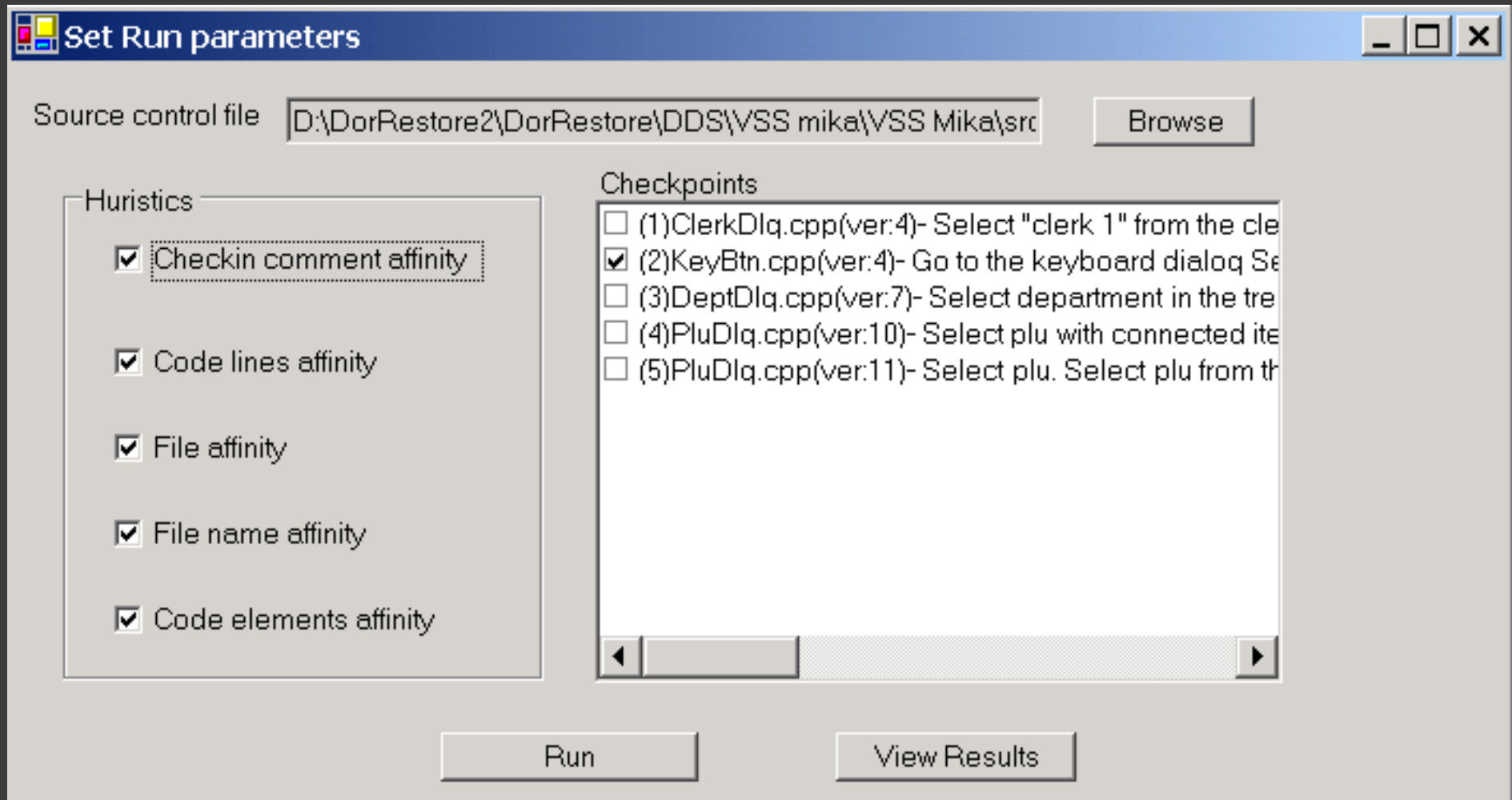
A “Real World” regression bug results

- Locating the bug took **20 hours** of strenuous work of two experienced programmers.
- Fixing the bug took less than an hour.


Heuristic	Rank (group by file)
Code Line Affinity	7
Check-in comment Affinity	-
File Affinity	22
Function Affinity	8
Average	4



The CodePsychologist tool



The CodePsychologist tool (Continue)


Result window

Code lines Affinity(256)			comment(257)			File affinity(256)			File Name affinity(256)			Code items affinity(256)		
File	Ver	Rank	File	Ver	Rank	File	Ver	Rank	File	Ver	Rank	File	Ver	Rank
ClerkDI...	4	0.1666...	ClerkDI...	4	0.0901...	ClerkDI...	4	0.0310...	ClerkDI...	4	0.2000...	ClerkDI...	4	0.2800...
MainFr...	14	0.1145...	String	7	0.0706...	Export...	9	0.0163...	TreeD...	6	0.0333...	MainFr...	12	0.1622...
MainFr...	12	0.1145...	WinButl...	4	0.0706...	Export...	9	0.0163...	TreeD...	6	0.0333...	MainFr...	11	0.1622...
resrc1.h	11	0.0902...	WinButl...	9	0.0706...	Export...	8	0.0163...	DeptDI...	5	0	PluDlg...	9	0.0799...
ChildFr...	7	0.0734...	KeyBtn....	4	0.0460...	Export...	8	0.0163...	KeyBtn....	4	0	Export...	6	0.0799...
resrc1.h	11	0.0722...	config.h	4	0.0314...	Export...	8	0.0163...	DeptDI...	7	0	Export...	6	0.0799...
resourc...	11	0.0649...	config.h	4	0.0314...	Export...	7	0.0163...	DeptDI...	7	0	WinButl...	10	0.0199...
ItemsTr...	15	0.0514...	config.h	4	0.0314...	Export...	7	0.0163...	DeptDI...	7	0	WinButl...	10	0.0199...
MainFr...	13	0.0400...	config.h	4	0.0314...	Export...	7	0.0163...	PluDlg...	11	0	WinButl...	10	0.0199...
KeyBtn....	4	0.0378...	config.h	4	0.0314...	Export...	7	0.0163...	PluDlg...	10	0	WinButl...	10	0.0199...

File	Ver	Rank
ClerkDlg.cpp	4	50
ExportDlg.cpp	8	18
ExportDlg.cpp	9	17
TreeData.h	6	17
MainFrm.cpp	12	17
config.h	4	15
KeyBtn.cpp	4	13
DeptDlg.cpp	7	12

Checkpoint

Select "clerk 1" from the clerk tree (clerk number 2). Go to the next clerk. The next clerk is "clerk 3".

Group ranking

☒ Rank Changes

☐ Group by file

☐ Group by comment

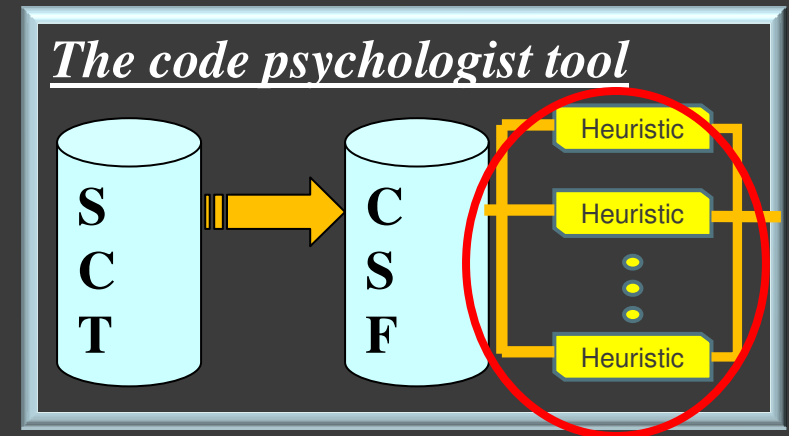
Export tables

Future work

- ⦿ Implementing the human factor and the code complexity heuristics.
- ⦿ Learning mechanism – Automatic tuning of heuristics.
- ⦿ More experiments on “real world” regression bugs.

THANK
YOU

Heuristics (Second phase)



● Code line affinity:

$$\text{Rank}_1(C, P) = \beta \cdot \text{GrpAff}(W(C), W(P)) + \frac{1 - \beta}{L} \cdot \sum_{l=1}^L \text{GrpAff}(W(C), W(P, l))$$

- $W(P, L)$ = Group of words in the source code located L lines from the change P.
- β – coefficient that gives different weight for lines inside the change.

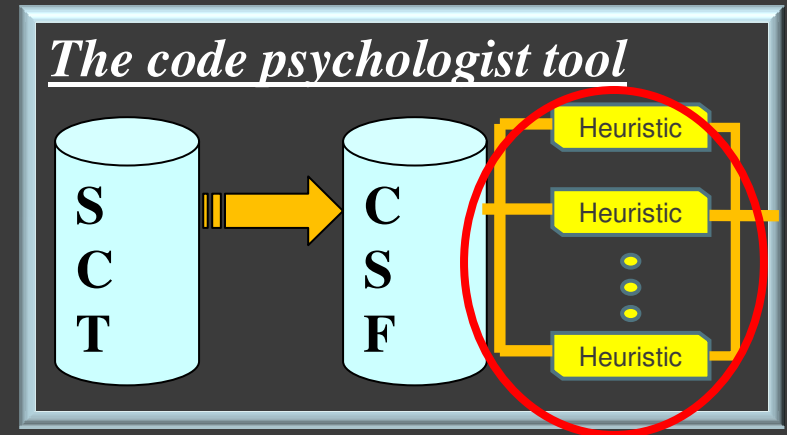
● Check-in comment affinity:

$$\text{Rank}_2(C, P) = \text{GrpAff}(W(C), W(\text{Checkin}(P)))$$



Heuristics (Continue)

● File affinity:



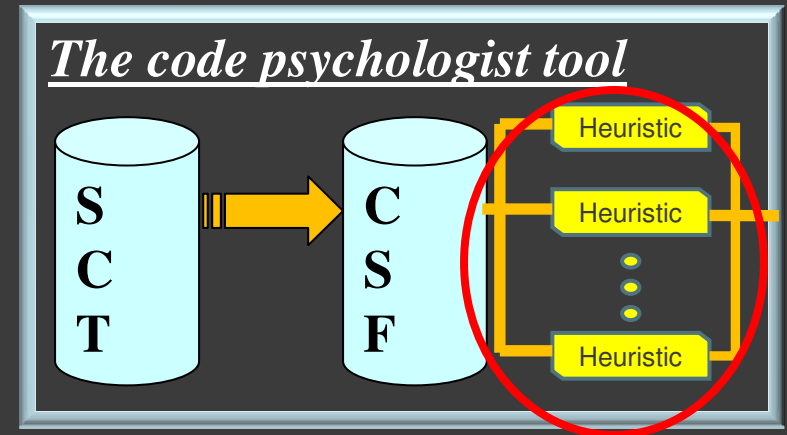
$$\text{maxAff}(a, B, \text{map}) = \max \{ \text{WrdAff}(a, b_i) \cdot \text{map}[b_i] \mid 1 \leq i \leq m \}$$

$$\text{HstAff}(A, B, \text{map}) = \frac{\sum_{i=1}^n \text{MaxAff}(a_i, B)}{n \cdot \max \{ \text{map}[b_j] \mid 1 \leq j \leq m \}}$$

$$\text{Rank}_3(C, P) = \text{HstAff}(W(C), W(F), \text{Hstg}(F))$$

Heuristics (Continue)

● Function affinity:



$$\begin{aligned} \text{FuncAff}(C, f) = & \frac{\alpha}{\alpha + \beta + k} \cdot \text{GrpAff}(W(C), W(f)) + \\ & \frac{\beta}{\alpha + \beta + k} \cdot \text{GrpAff}(W(C), \text{Bdy}(f)) + \\ & \sum_{i=1}^k \frac{1}{\alpha + \beta + k} \cdot \text{FncAff}(C, \text{FncCall}(f, i)) \end{aligned}$$

$$\text{Rank}_4(C, P) = \text{FuncAff}(C, \text{func}(P))$$

Code lines affinity

D:\temp\History\Ver_4(314)\ClerkDlg.cpp

```
305
306 BOOL CClerkDlg::GetNextItem(TreeData& cTreeData) const
307 {
308     WAITER_t* waiter=NULL ,*nextWaiter=NULL ;
309     INT number = cTreeData.number;
310     // find the prev waiter number. it's number is the smallest
311     for(waiter = (WAITER_t*)DB_find_first(DATA->waiters,NULL);
312         waiter;
313         waiter=(WAITER_t*)DB_find_next(DATA->waiters))
314     {
315         if((waiter->number > number) ) //Go to the next waiter
316             nextWaiter = waiter;
317     }
318     if(!nextWaiter)
```

D:\temp\History\Ver_3\ClerkDlg.cpp

```
305
306 BOOL CClerkDlg::GetNextItem(TreeData& cTreeData) const
307 {
308     WAITER_t* waiter=NULL ,*nextWaiter=NULL ;
309     INT number = cTreeData.number;
310     // find the prev waiter number. it's number is the smallest
311     for(waiter = (WAITER_t*)DB_find_first(DATA->waiters,NULL);
312         waiter;
313         waiter=(WAITER_t*)DB_find_next(DATA->waiters))
314     {
315         if((waiter->number > number) &&
316             (!nextWaiter || nextWaiter->number > waiter->number) )
317             nextWaiter = waiter;
318     }
319     if(!nextWaiter)
```

Check point

Select 'clerk 1' from the clerk tree (clerk number 2).
Go to the next clerk.
The next clerk is 'clerk 3'

