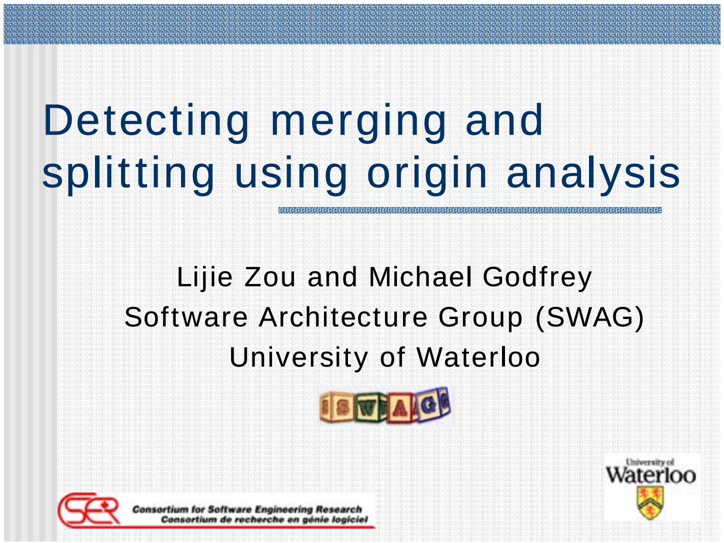
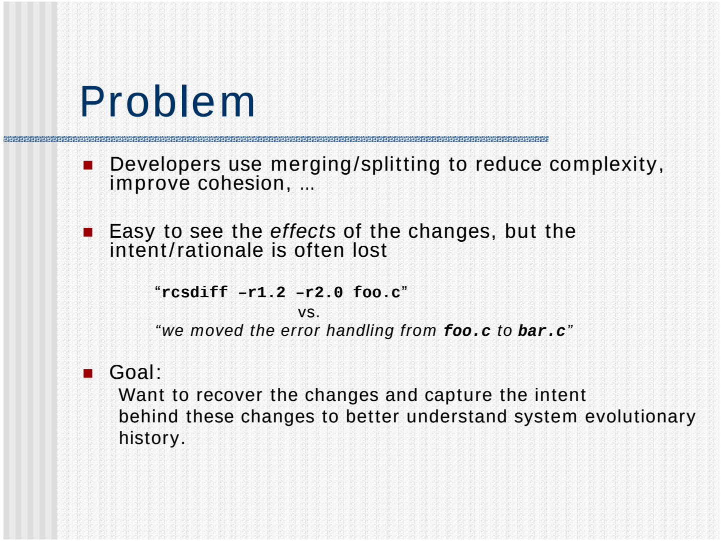


Detecting merging and splitting using origin analysis

Lijie Zou and Michael Godfrey
Software Architecture Group (SWAG)
University of Waterloo

Consortium for Software Engineering Research
Consortium de recherche en génie logiciel

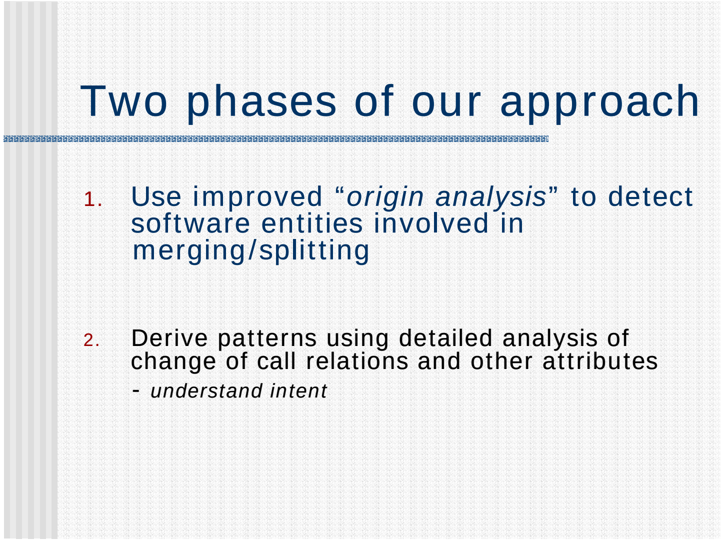


Problem

- Developers use merging/splitting to reduce complexity, improve cohesion, …
- Easy to see the *effects* of the changes, but the intent/rationale is often lost

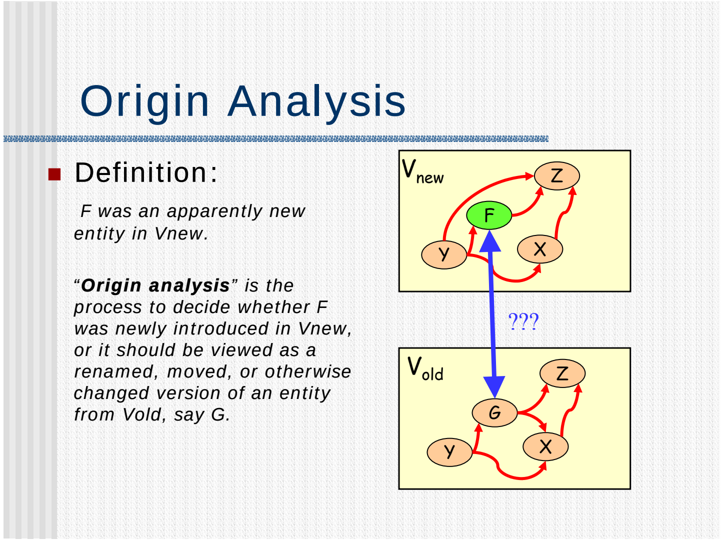
  “**rcsdiff –r1.2 –r2.0 foo.c**”
  vs.
  *“we moved the error handling from **foo.c** to **bar.c**”*

- Goal:
  Want to recover the changes and capture the intent behind these changes to better understand system evolutionary history.



Two phases of our approach

1. Use improved “*origin analysis*” to detect software entities involved in merging/splitting
2. Derive patterns using detailed analysis of change of call relations and other attributes
   - *understand intent*



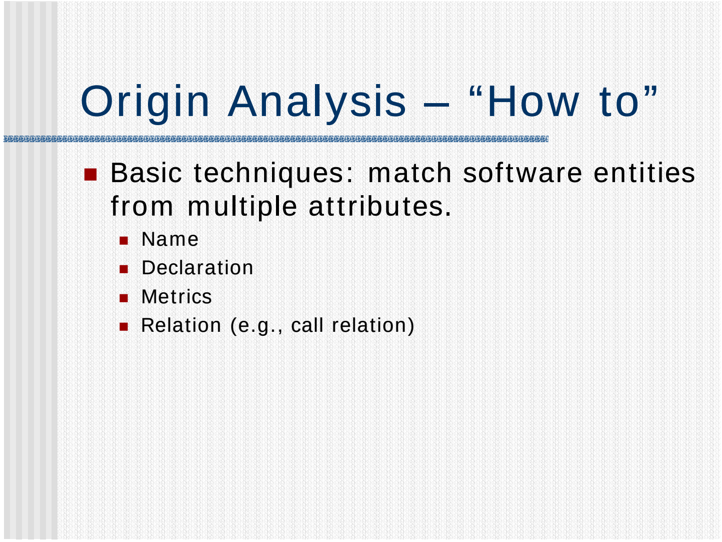
Origin Analysis

- Definition:

  F was an apparently new entity in Vnew.

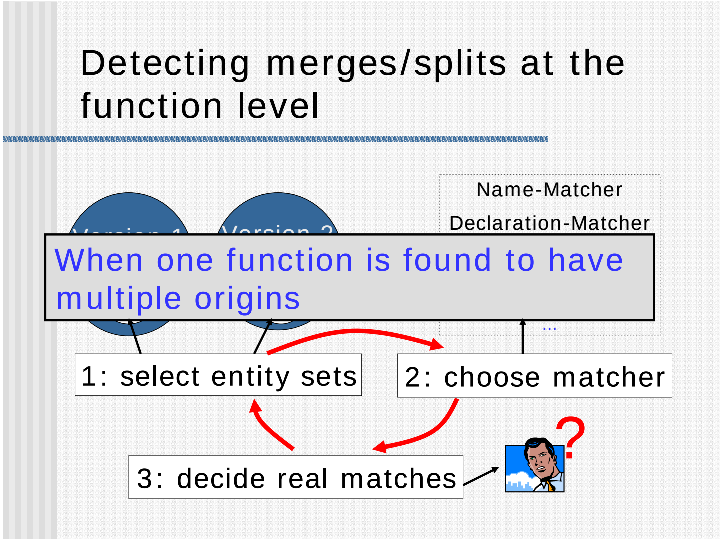
  *“**Origin analysis**” is the process to decide whether F was newly introduced in Vnew, or it should be viewed as a renamed, moved, or otherwise changed version of an entity from Vold, say G.*

V_{new}
Z
F
Y
X
???
V_{old}
Z
G
Y
X



Origin Analysis – “How to”

- Basic techniques: match software entities from multiple attributes.
  - Name
  - Declaration
  - Metrics
  - Relation (e.g., call relation)



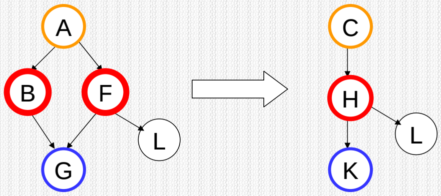
Detecting merges/splits at the function level

When one function is found to have multiple origins

Version 1
Version 2
Name-Matcher
Declaration-Matcher
…
1: select entity sets
2: choose matcher
3: decide real matches
?

Detecting **chained** merges/splits at the function level

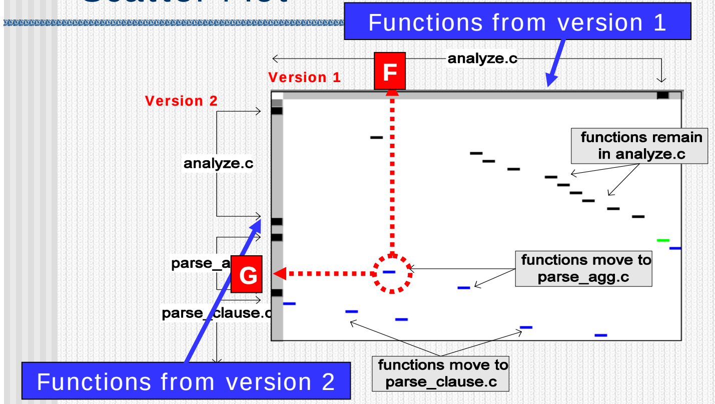
- Functions involved are interdependent
- Multiple iterations



Detecting merges/splits at the file level

- Manually
- File merge:
  - A new file G is found to be composed of most functions from two old files F1 and F2
- File split

Scatter Plot



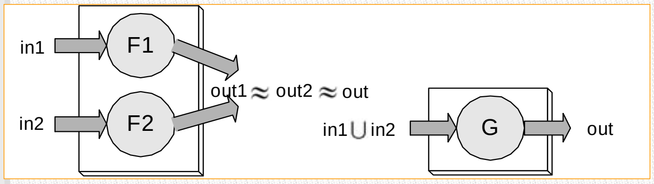
Two phases of our approach

1. Use improved “*origin analysis*” to detect software entities involved in merging/splitting
2. Derive patterns using detailed analysis of change of call relations and other attributes
   - *understand intent*

Patterns

- Clone elimination
- Pipeline extraction
- Service consolidation
- Parameterization
- Partial clone elimination

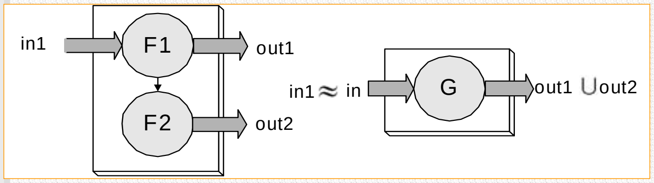
Pattern 1: clone elimination



F1, F2, G	: functions
in1,in2, in	: callers of F1, F2 or G
out1, out2, out	: callees of F1, F2 or G

1. out1 ≈ out2 ≈ out
2. in1∩in2 ≈ Ø , in1 U in2 ≈ in

Pattern 2: pipeline contraction



F1, F2, G	: functions
in1,in2, in	: callers of F1, F2 or G
out1, out2, out	: callees of F1, F2 or G

1. out1 U out2 ≈ out
2. in1 ≈ in

Pattern 3~5

- Service consolidation

  Two functions that perform different services, but are called at the same time by the same clients, are merged into a new, larger function
- Parameterization

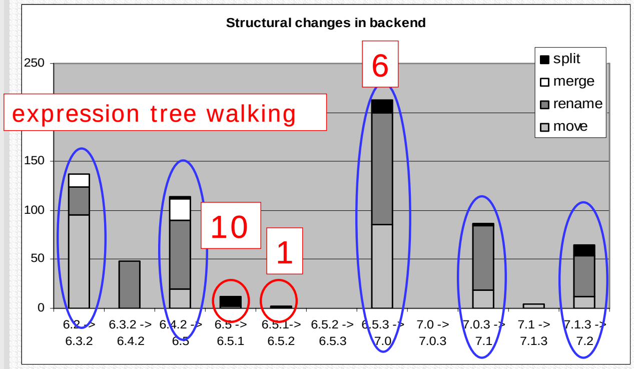
  Two similar functions F1 and F2 are combined into a new function G by adding a parameter to distinguish different functionalities
- Partial clone elimination

  A chunk of code found in F1 and F2 are clones. These clones are extracted out to form a new function G.

Case study - PostgreSQL

- OSS, ORDBMS, widespread used
  - 12 releases from v6.2 (Oct. 1997) to v7.2 (Feb. 2002)
- We looked at the backend subsystem
  - 70% of the codebase
  - KLOC: 186 -> 279, 10% / year
  - Functions: 3262 -> 4531

Overview of Structural changes in PostgreSQL



expression tree walking

6

10

1

Patterns

Pattern	# of instances	Examples
Clone elimination	7	getAttrName, get_attname -> get_attname
Service consolidation	1	Gettypelem, typtoout -> getTypeOutAndElem
Pipeline extraction	29	...
Parameterization	3	...
Partial clone elimination	27	...

Group of merges/splits

- Function level
  - 17 splits in 10 files (expression walker)
  - 6 splits in 4 files from 2 subsystems (to modify expression tree)
  - 4 splits in 4 files in subsystem access (callback mechanism)

  Scattered in different files and subsystems
- File Level
  - parser restructuring (6.2 -> 6.3.2)
  - cleaning up of optimizer subsystem (6.4.2 -> 6.5)

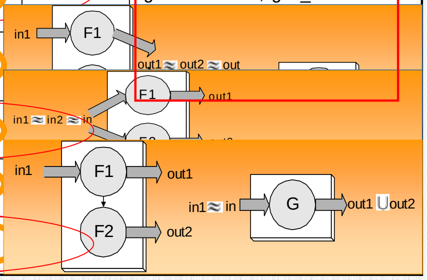
Summary & Future work

- Summary
  - Techniques and tool for detecting instances of merging and splitting
  - Merge/split patterns
- Future work
  - CVS log
  - Catalog of patterns

Questions?

Patterns

Pattern	# of instances	Examples
Clone elimination	7	getAttrName, get_attname
Service consolidation		
Pipeline extraction		
Parameterization		
Partial clone elimination		


in1
F1
out1 ≈ out2 ≈ out
in1 ≈ in2 ≈ in
F1
out1
in1
F1
out1
F2
out2
in1 ≈ in
G
out1 ∪ out2