# Supporting Requirements Traceability through Refactoring

Anas Mahmoud and Nan Niu,

Department of Computer Science and Engineering
Mississippi State University

E-mail: niu@cse.msstate.edu

July 17, 2013 @ RE

## Software Traceability

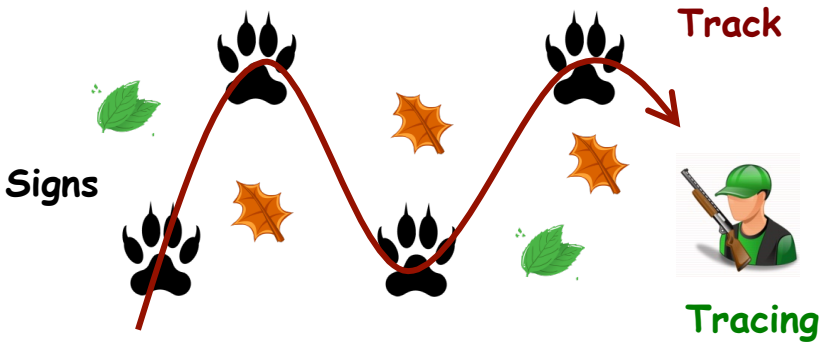
- Definition
  - the ability to describe and follow a stakeholder's concern throughout the software lifecycle [Gotel and Finkelstein, RE 1994]

*RE 2011*

- Importance
  - Recommended by IEEE Standard & SEI's CMM
  - Mandated by NASA, FDA, & FAA
- Value: "connecting the dots"
  - Does the code satisfy the design?
  - What is the change impact of a certain requirement?
  - …
  - Indispensable to many other software engineering tasks

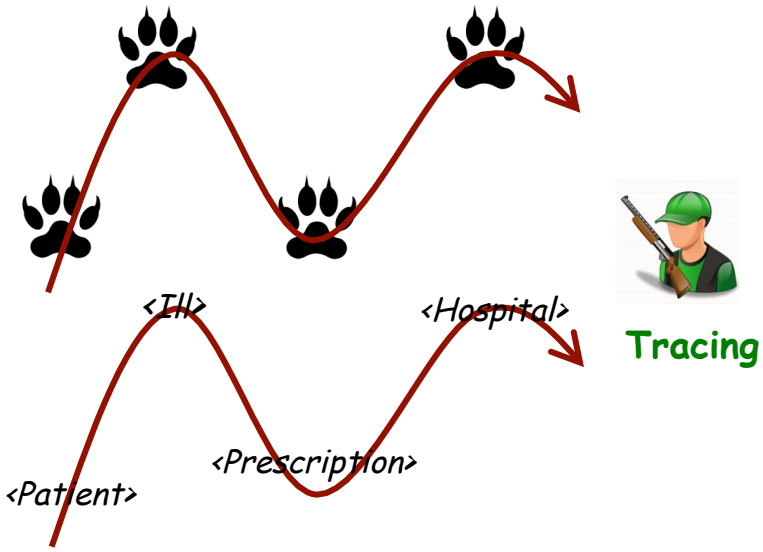
## Out of the Labyrinth [Gotel and Morris, RE 2011]

- How do other (mature) fields tackle tracing?
  - Animal tracking, art provenance, epidemiology, food traceability, luggage handling, metrology



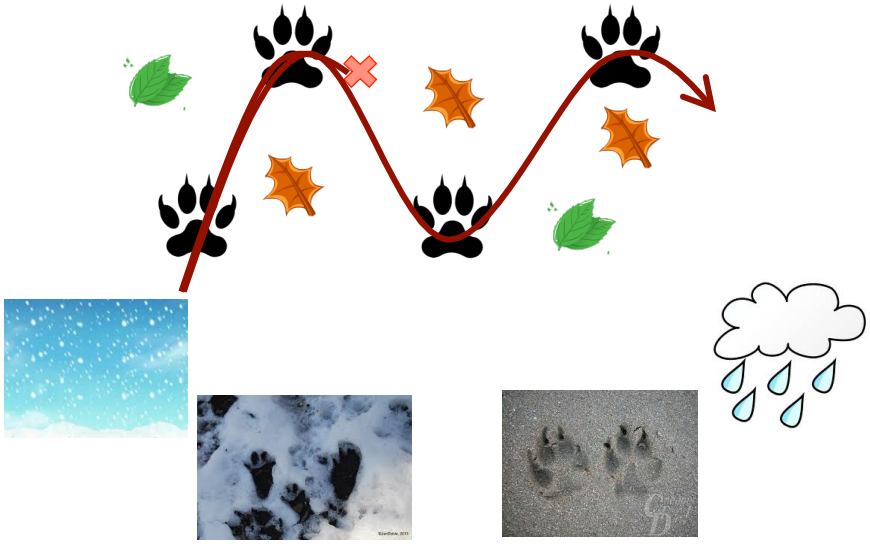

3

## IR-Based Automated Traceability



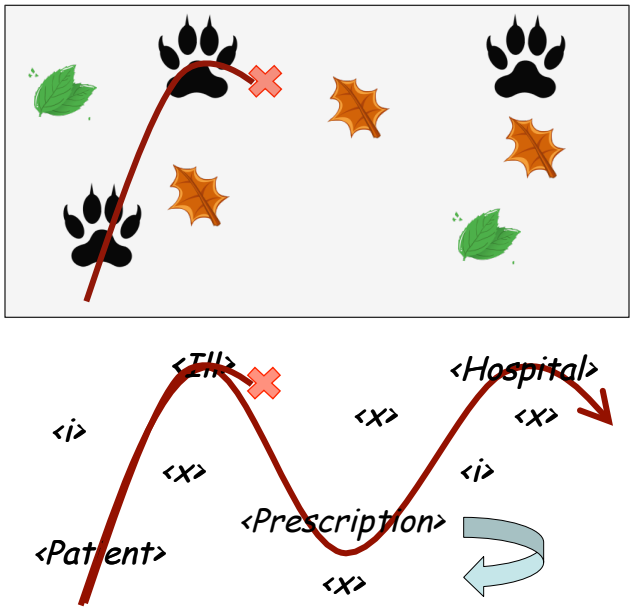

4

# Problem #1: Missing Signs



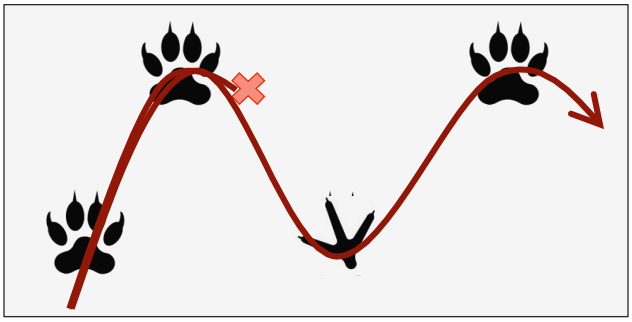
5

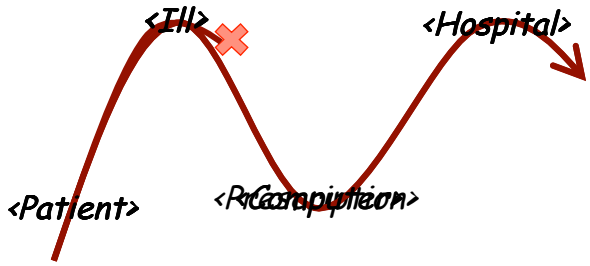
# Missing Signs in Code Base




6

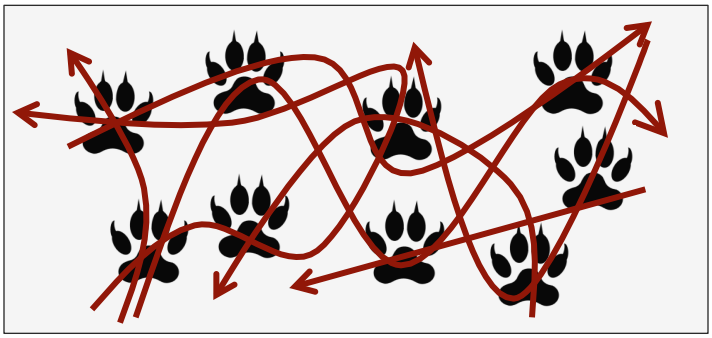
## Problem #2: Misplaced Signs

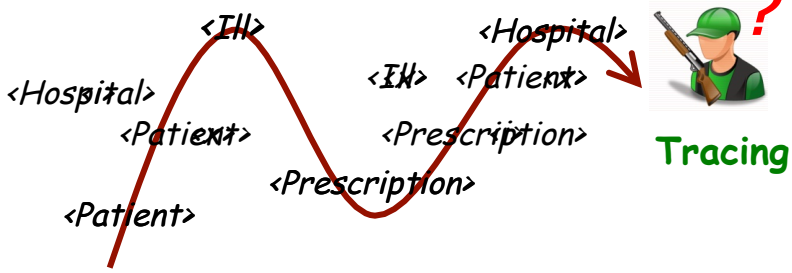





7

## Problem #3: Duplicated Signs






8

# Outline

- ✓ Introduction
  - ✓ Fundamentals: sign ➔ track ➔ trace
  - ✓ Challenges: (1) Missing signs; (2) Misplaced signs; and (3) Duplicated signs
  - ✓ One root cause: software evolution
- ➔ **Central hypothesis**
  - ⇨ Refactoring can help reverse the effect of discontinued and distorted signs, and thus can systematically re-establish track in the software system
- ⇨ Experimental evaluation
- ⇨ Concluding remarks

9

# Refactoring

- ⇨ What?
  - ⇨ Behavior-preserving transformations that improve the internal structure of the code
    - ⇨ Improving maintainability, reusability, understandability, etc.
- ⇨ Why can refactoring help?
  - ⇨ Refactoring works on the informal aspects of the code base (as opposed to formal runtime behaviors)
  - ⇨ IR-based requirements tracing also works on the informal aspects (as opposed to formal semantics)
- ⇨ How can refactoring help?

| **Problem** | Missing signs | Misplaced signs | Duplicated signs |
|---|---|---|---|
| **Refactoring** | Restore information | Move information | Remove information |

10

## Refactoring Classification: A Traceability Perspective

| **Problem** | Missing signs | Misplaced signs | Duplicated signs |
|---|---|---|---|
| **Refactoring** | Restore information | Move information | Remove information |
| **Sample Refactoring Techniques** | 1) Rename Identifier<br>2) Add Parameter<br>3) Split Temporary Variable<br>… | 1) Move Method<br>2) Move Parameter<br>3) Push Down Field<br>4) Push Down Method<br>… | 1) Extract Method<br>2) Decompose Conditional<br>3) Parameterize Method<br>… |

RI (Rename Identifier), MM (Move Method), and EM (Extract Method) as *representative* techniques to fulfill refactoring's potentials in each category

Key criteria to be "representative": (i) coverage, (ii) granularity, and (iii) automation

*A useful source: http://refactoring.com*

11

## RI to Restore Missing Signs

- Rename Identifier
  - Renaming an identifier to give it a more relevant name
- Our operationalizations
  - Manually identify the following "bad smells":
    - identifier with less than 4-character length, e.g., HCP ➔ HealthCarePresonnel
    - identifier including a special word, e.g., PnString ➔ PatientNameString
    - identifier with generic names, e.g., import ➔ importPatientRecords
  - Semi-automatically define name expansions and replacements
  - Automatically apply refactoring in Eclipse 4.2.1 to ensure correctness and consistency

12

## MM to Correct Misplaced Signs

- Move Method
  - To reduce coupling and increase cohesion
- Our operationalizations
  - Semi-automatically identify the "feature envy" bad smells:
    - If method M1 accesses way more fields and other methods in class C2 than its own class C1, then method M1 should probably be placed in C2 rather than C1 [Tsantalis and Chatzigeorgiou, TSE'09]
  - Automatically apply refactoring in Eclipse 4.2.1 to ensure correctness and consistency

13

## EM to Remove Duplicated Signs

- Extract Method
  - To reduce code clones (duplicates) and make them more modular
- Our operationalizations
  - Automatically detect "code clones" by employing the SDD tool (wiki.eclipse.org/Duplicated_code_detection_tool_(SDD))
  - Semi-automatically define the name of the "extracted method" and the class that the "extracted method" belongs to
  - Automatically apply refactoring in Eclipse 4.2.1 to ensure correctness and consistency

14

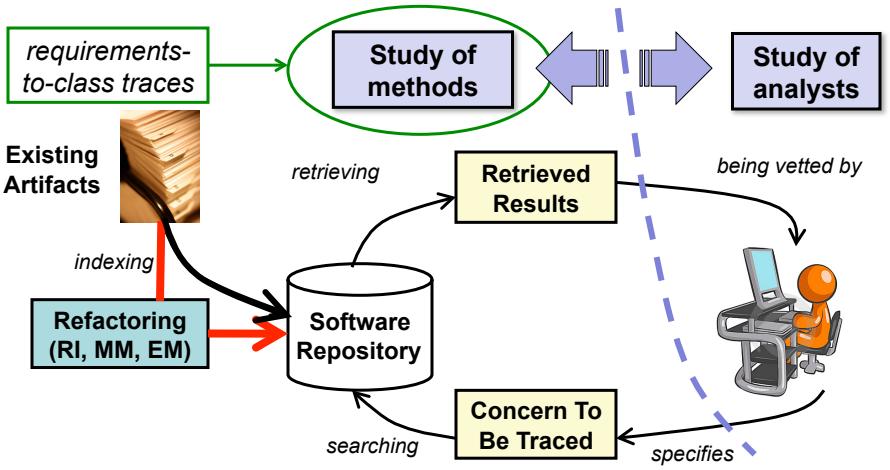
# Outline

- ✓ Introduction
  - ✓ Fundamentals: sign ➔ track ➔ trace
  - ✓ Challenges: (1) Missing signs; (2) Misplaced signs; and (3) Duplicated signs
  - ✓ One root cause: software evolution
- ✓ Central hypothesis
  - ✓ Refactoring can help reverse the effect of discontinued and distorted signs, and thus can systematically re-establish track in the software system
- ⇨ **Experimental evaluation**
- ⇨ Concluding remarks

15

# Experimental Design

| Dataset | LOC | COM | No. Req. | No. SC | Links |
|---|---|---|---|---|---|
| *iTrust* | 20.7K | 9.6K | 50 | 299 | 314 |
| *eTour* | 17.5K | 7.5K | 58 | 116 | 394 |
| *WDS* | 44.6K | 10.7K | 26 | 521 | 229 |




16

## Result: How broad are refactorings' impacts?

| | iTrust | | | eTour | | | WDS | | |
|---|---|---|---|---|---|---|---|---|---|
| **Refactoring** | E | C | C' | E | C | C' | E | C | C' |
| *RI* | 175 | 113 | 110 | 85 | 63 | 57 | 203 | 174 | 166 |
| *MM* | 22 | 44 | 44 | 17 | 31 | 29 | 24 | 62 | 61 |
| *EM* | 132 | 201 | 193 | 45 | 92 | 88 | 62 | 102 | 98 |

| | iTrust | eTour | WDS |
|---|---|---|---|
| E (total) | 299 | 116 | 521 |

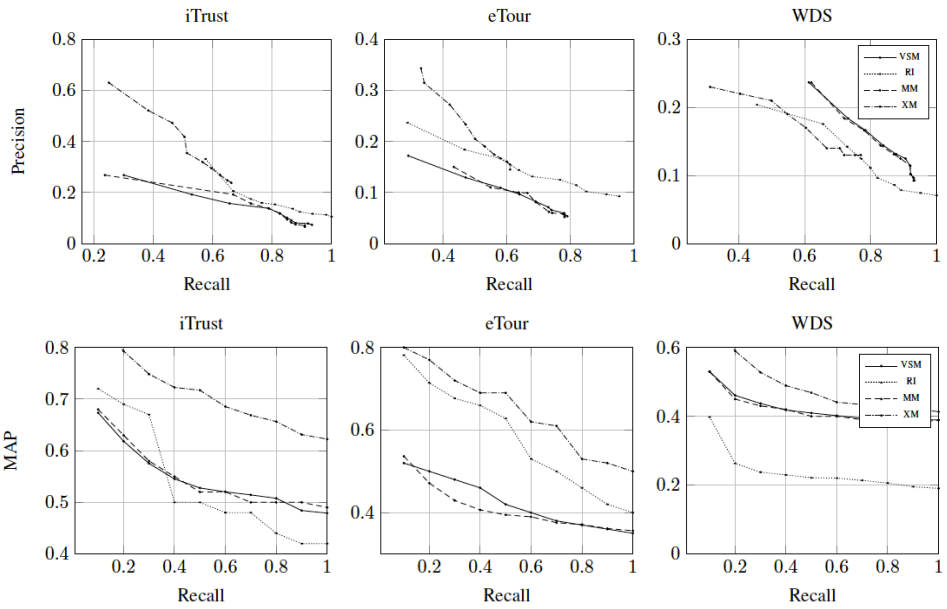
E: # of affected entities

C: # of affected classes in each system

C': # of affected classes in the gold standard

17

## Results: Retrieval Effectiveness and Browsability




18

## Summary and Limitations

- ⇨ Refactorings
  - ⇨ RI (rename identifier) had the most positive effects, though the WDS's effect was not statistically significant
    - ⇨ Restoring information essentially ameliorates the vocabulary mismatch problem & refactoring represents an internal way of handling the problem (as opposed to external thesaurus or query expansion)
  - ⇨ MM (move method) had the least influence
    - ⇨ MM's effect is local and limited
  - ⇨ EM (extract method) had an overall negative impact on the performance, e.g., recall was significantly reduced
    - ⇨ Duplicated signs play a positive role in tracing, as redundancy implies reliability
  - ⇨ Outdated req.s based on code changes [Ben Charrada *et al.*, RE'12]
- ⇨ Major threats to validity
  - ⇨ Only 3 refactorings were experimented (1 in each category) and were tested independently
  - ⇨ Granularity level (requirements-to-class) was fixed

19

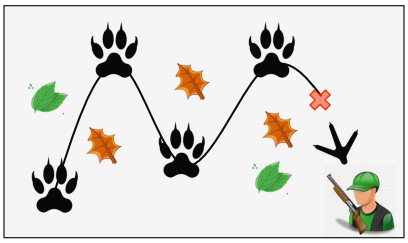
## Outline

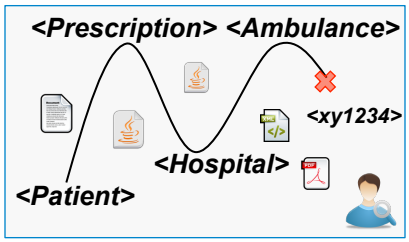
- ✓ Introduction
  - ✓ Fundamentals: sign ➔ track ➔ trace
  - ✓ Challenges: (1) Missing signs; (2) Misplaced signs; and (3) Duplicated signs
  - ✓ One root cause: software evolution
- ✓ Central hypothesis
  - ✓ Refactoring can help reverse the effect of discontinued and distorted signs, and thus can systematically re-establish track in the software system
- ✓ Experimental evaluation
- ⇨ **Concluding remarks**

20

## *Back to the Nature*

## Restoring Lost Traceability Tracks through Refactoring






**"An analysis of the requirements traceability problem" [Gotel and Finkelstein, RE'94]**

**"Out of the labyrinth: leveraging other disciplines for requirements traceability" [Gotel and Morris, RE'11]**

**"The quest for ubiquity: a roadmap for software and systems traceability" [Gotel *et al.*, RE'12]**

21