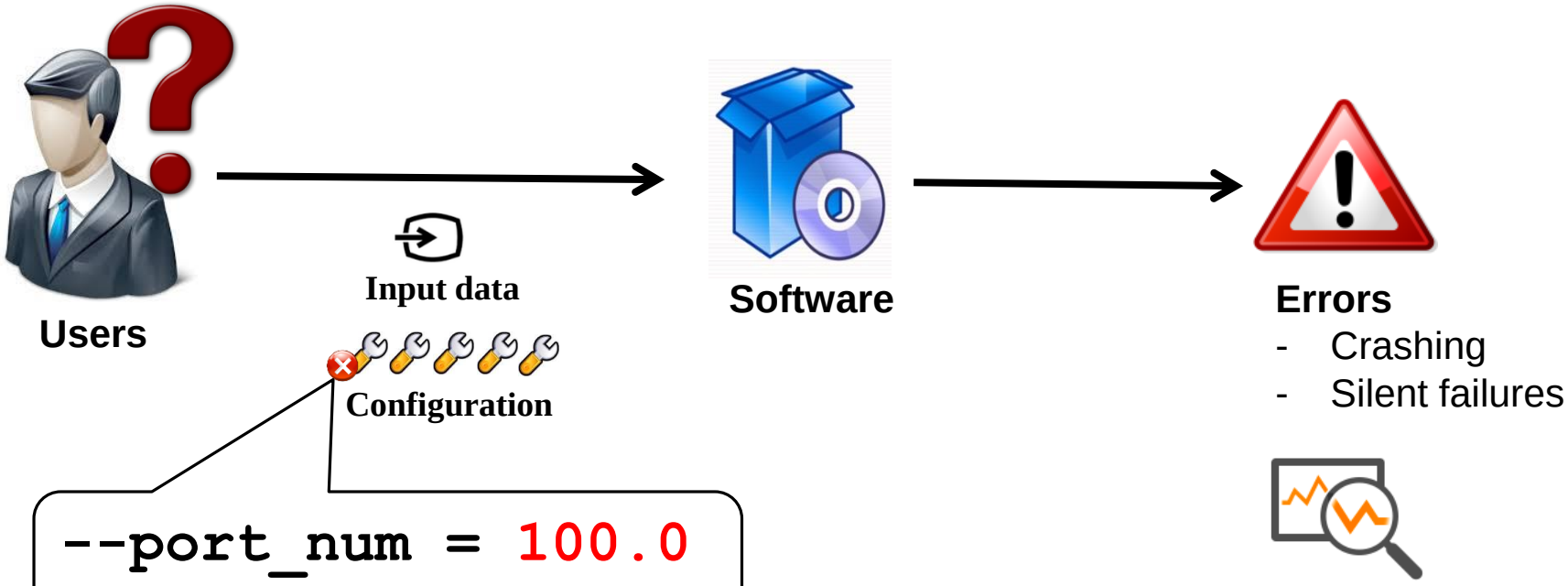
Proactive Detection of Inadequate Diagnostic Messages for Software Configuration Errors

Sai Zhang
Google Research

Michael D. Ernst
University of Washington

***Goal**: helping developers improve software error diagnostic messages*



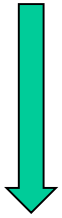
A bad diagnostic message:

"… unexpected system failure …"

***Our technique**: detecting such inadequate diagnostic messages caused by configuration errors*

***Goal**: helping developers improve software error diagnostic messages*

Software

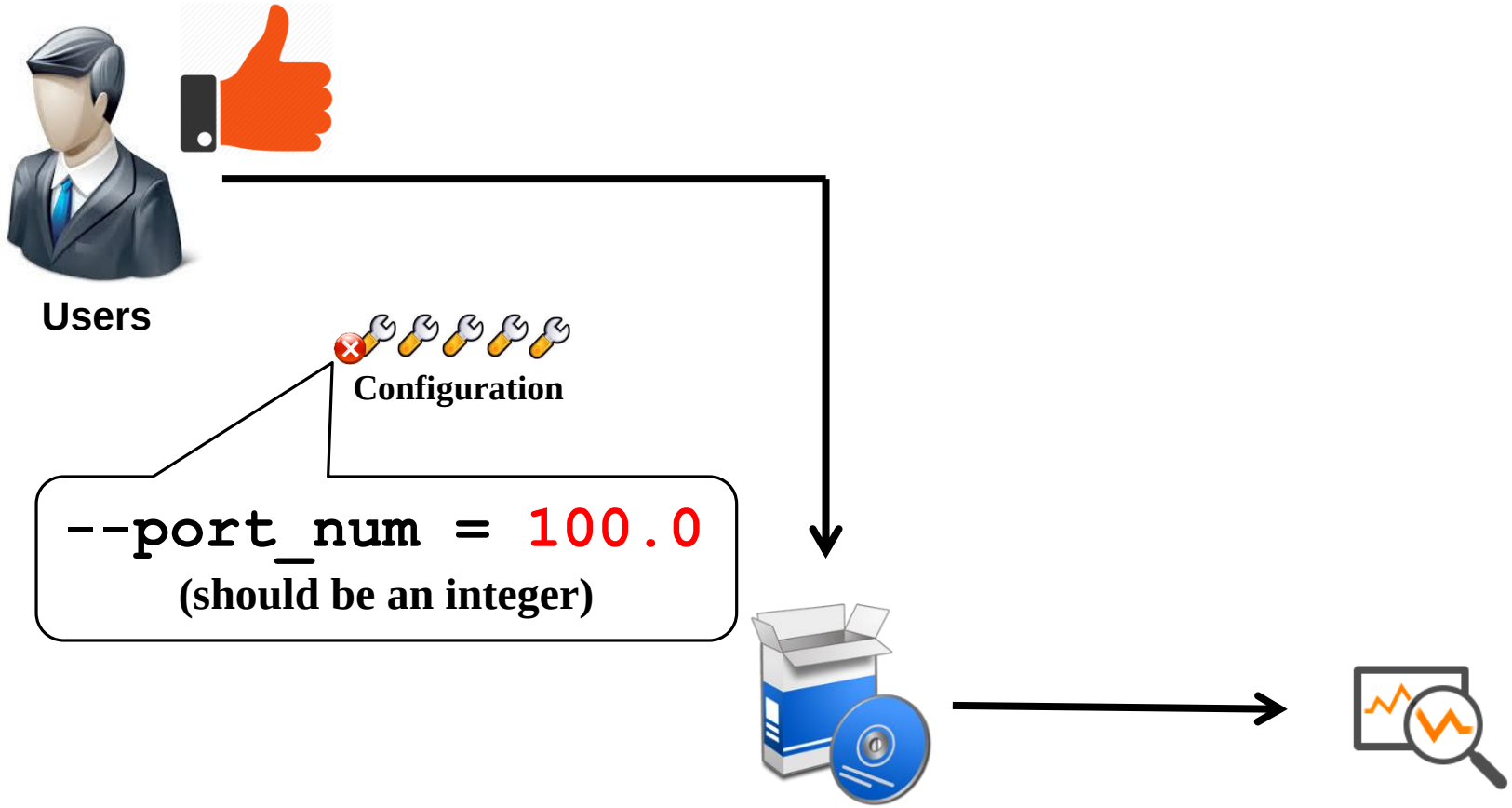


Our technique:
ConfDiagDetector

Developers

Software
(with improved diagnostic message)

Goal: helping developers improve software error diagnostic messages

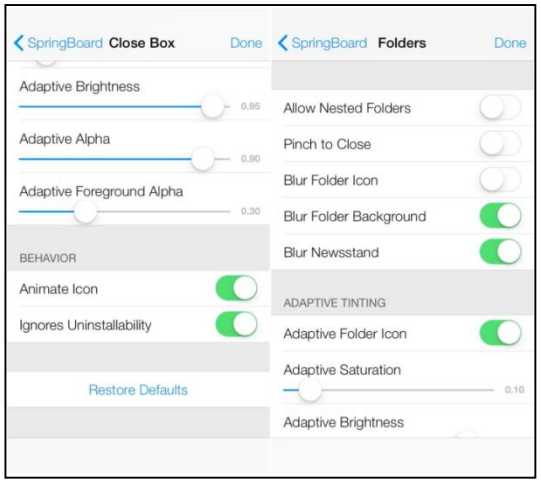


Why configuration errors?

- Software systems often require configuration

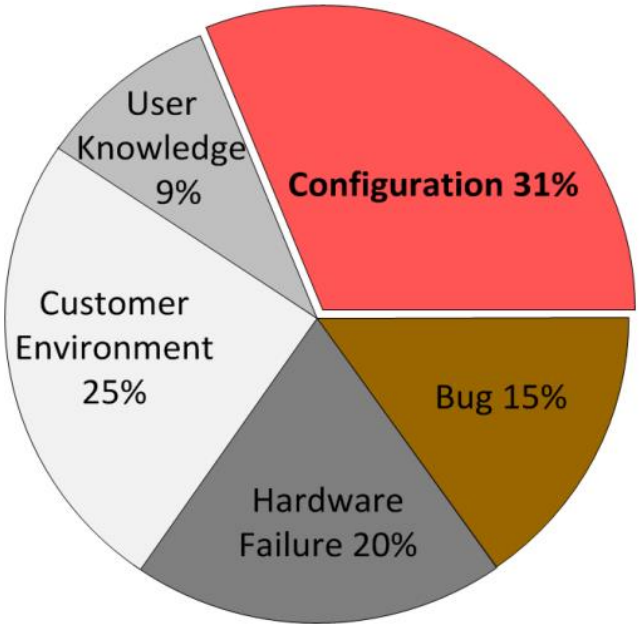
```
#
# * Fine Tuning
#
key_buffer      = 16M
max_allowed_packet  = 16M
thread_stack      = 192K
thread_cache_size        = 8
#
# * Query Cache Configuration
#
query_cache_limit = 1M
query_cache_size         = 16M
#
```

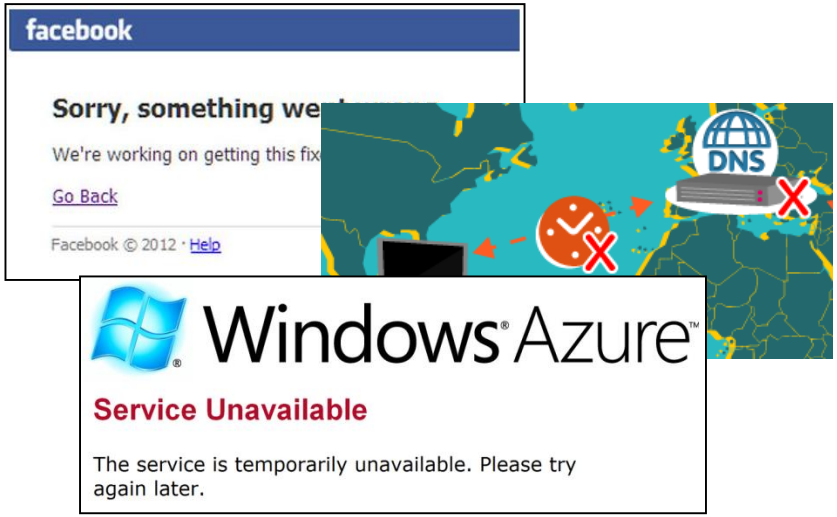
```
Options include:
   --help        Display this help text
   -v or --verbose       Print messages about ISendfile actions
   -s            Silent, opposite of verbose
   -h            No banner for this job
   -F=format     Format is one of the following:
        f - formatted, l - leave control characters, o - Postscript
        p - use 'pr' format, r - FORTRAN, c - CIF, d - dvi, g - plot
        n - ditroff, t - troff, v - raster
   -C=class      Class is used on banner page; up to 31 characters
   -T=title      Job title
```



Why configuration errors?

- Software systems often require configuration
- Software configuration errors are common and severe





Root causes of **high-severity** issues in a major storage company [Yin et al, SOSP'11]

Configuration errors can have **disastrous** impacts (downtime costs **3.6%** of revenue)

Why diagnostic messages?

- Often the sole data source available to understand an error
- Many diagnostic messages in practice are inadequate
  - **Missing**
  - **Ambiguous**

Why diagnostic messages?

- Often the sole data source available to understand an error
- Many diagnostic messages in practice are inadequate
  - **Missing**
  - **Ambiguous**

A misconfiguration in Apache JMeter

`output_format = XYZ` (an unsupported format)

No diagnostic message, but JMeter saves output in the default "XML" format

Why diagnostic messages?

- Often the sole data source available to understand an error
- Many diagnostic messages in practice are inadequate
  - **Missing**
  - **Ambiguous**

A misconfiguration in Apache Derby

```
derby.stream.error.method = hello
```

Diagnostic message:

IJ ERROR: Unable to establish connection

Why diagnostic messages?

- Often the sole data source available to understand an error
- Many diagnostic messages in practice are inadequate
  - **Missing**
  - **Ambiguous**

***Our technique**: detecting those inadequate messages before they arise in the field.*

Outline

- Motivation
- The ConfDiagDetector technique
- Evaluation
- Related work
- Contributions

Challenges of proactive detection of inadequate diagnostic messages

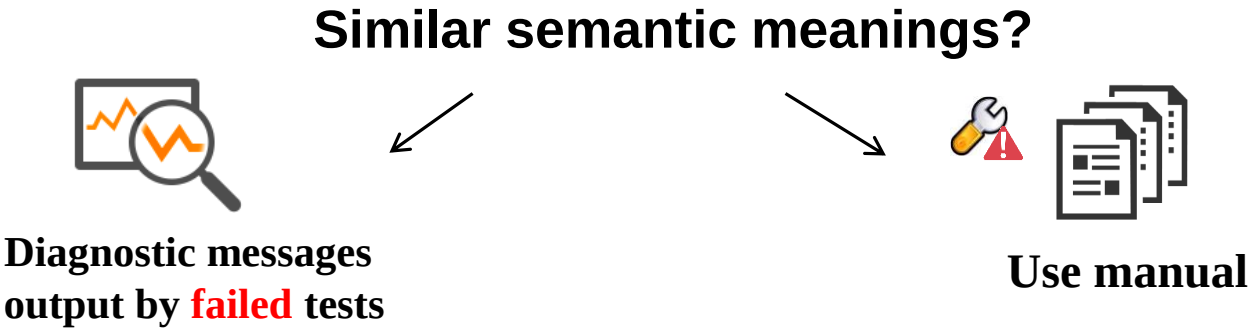
- How to *trigger a configuration error*?

- How to *determine the inadequacy* of a diagnostic message?

ConfDiagDetector's solutions

- How to *trigger a configuration error*?
  - Configuration mutation + checking system tests' results

- How to *determine the inadequacy* of a diagnostic message?
  - Use a NLP technique to check its semantic meaning



ConfDiagDetector workflow

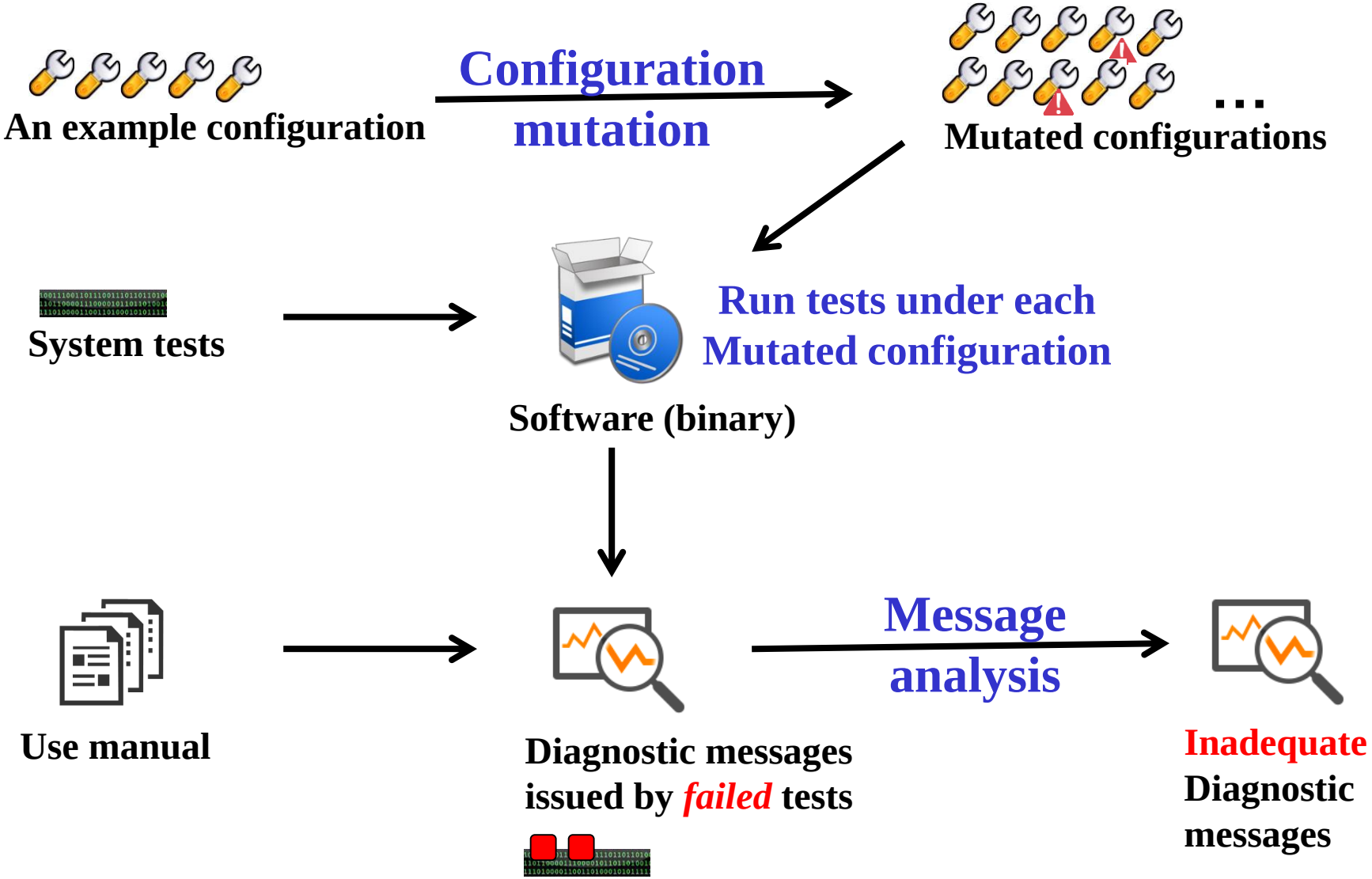
An example configuration

System tests

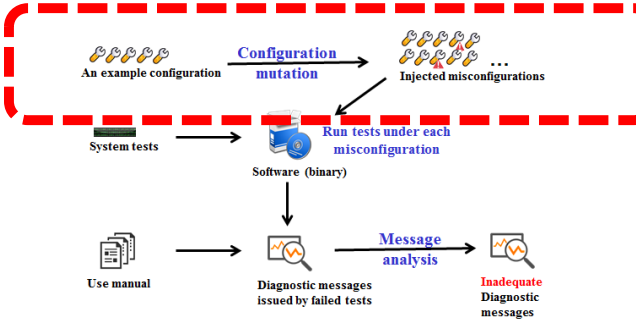
All tests pass!

Software (binary)

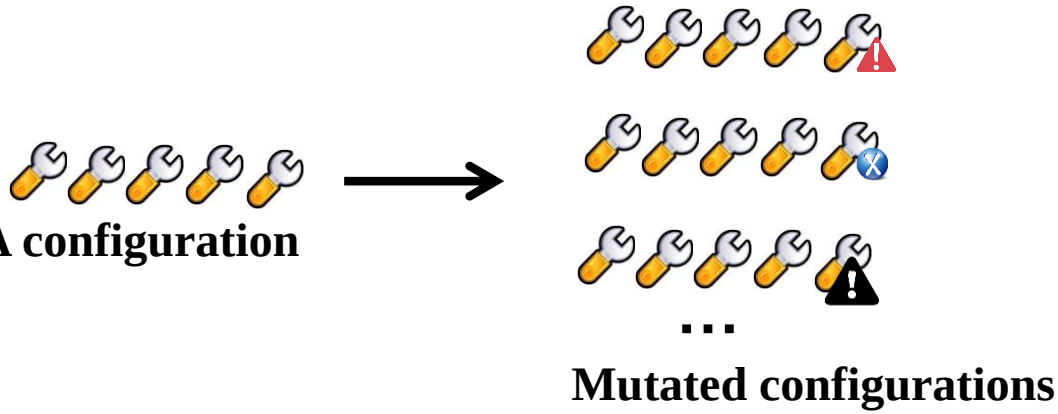
ConfDiagDetector workflow



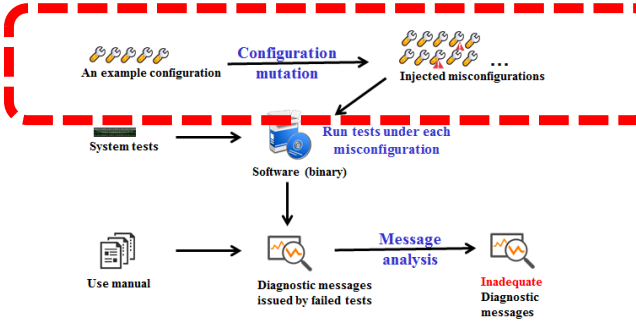
Configuration mutation



- Randomly mutates option values
  - One mutated option in each mutated configuration



Configuration mutation



- Randomly mutates option values
  - One mutated option in each mutated configuration
- Mutation rules for one configuration option
  - Delete existing value

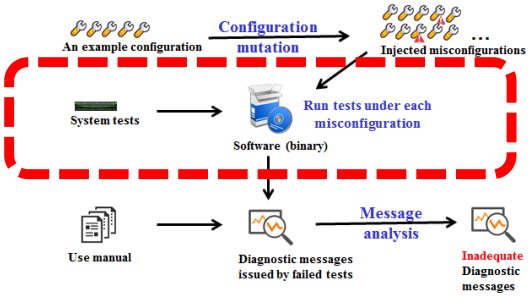
    `format=xml → format=`
  - Using a random value

    `format=xml → format= xyz`
  - Injecting spelling mistakes

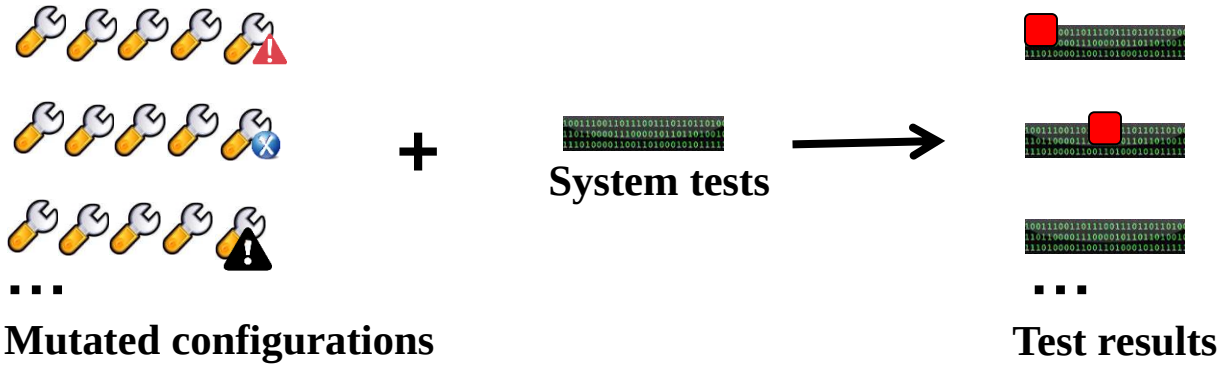
    `format=xml → format= xmk`
  - Change the case of text

    `format=xml → format= XML`

Running tests

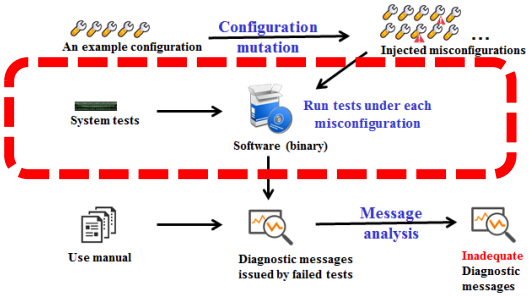


- Run the *all* tests under *each* mutated configuration

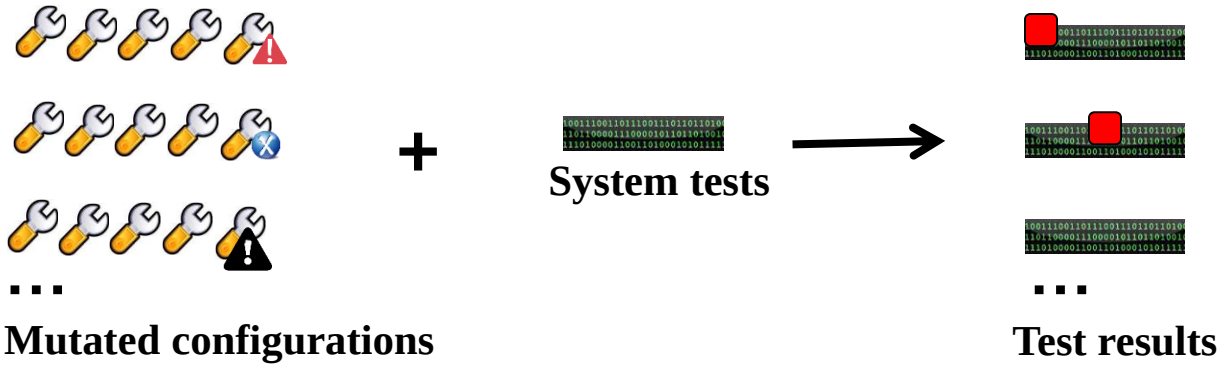


- Parse *each failed test*'s log file or console to get the diagnostic message

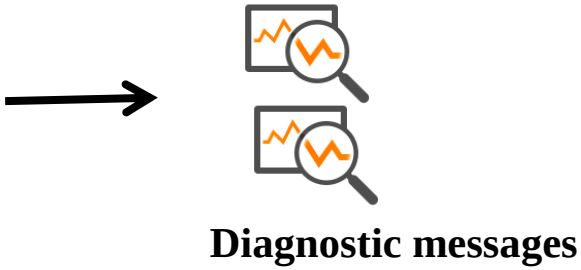
Running tests



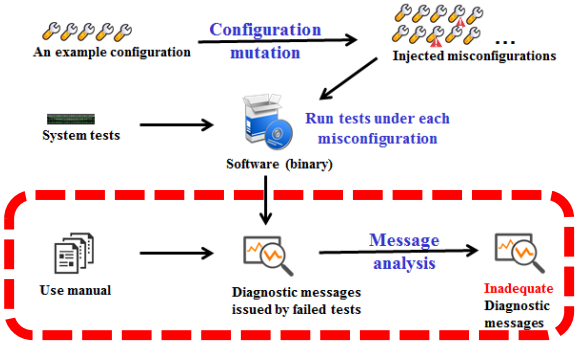
- Run the *all* tests under *each* mutated configuration



- Parse *each failed test*'s log file or console to get the diagnostic message



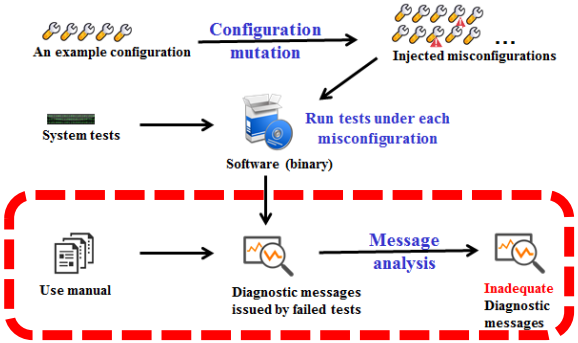
Message analysis



- A message is adequate, if it
  - contains the mutated option name or value

  OR
  - has a similar semantic meaning with the manual description

Message analysis



- A message is adequate, if it
  - contains the mutated option name or value

  OR
  - has a similar semantic meaning with the manual description

Example:

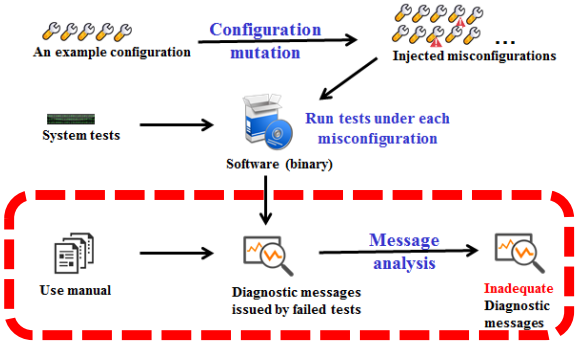
Mutated option:

`--percentage-split`

Diagnostic message:

"`the value of percentage-split should be > 0`"

Message analysis



- A message is adequate, if it
  - contains the mutated option name or value

  OR
  - has a similar semantic meaning with the manual description

Example:

Mutated option:

`--fnum`

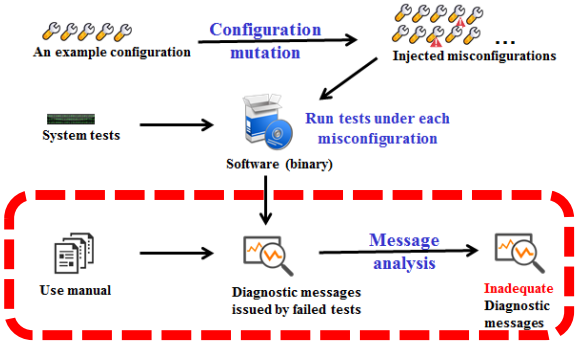
Diagnostic message:

`"Number of folds must be greater than 1"`

User manual description of **`--fnum`**:

`"Sets number of folds for cross-validation"`

Message analysis



- A message is adequate, if it
  - contains the mutated option name or value

  OR
  - has a similar semantic meaning with the manual description

A NLP technique [Mihalcea'06]

Key idea of the employed NLP technique

A message

Manual description

Has similar semantic meanings, if many words in them have similar meanings

<u>Example:</u>

~~The~~ program goes wrong

~~The~~ software fails

- *Remove all stop words*
- *For each word in the diagnostic message, tries to find the similar words in the manual*
- *Two sentences are similar, if "many" words are similar between them.*

Outline

- Motivation
- The ConfDiagDetector technique
- Evaluation
- Related work
- Contributions

Research questions

- ConfDiagDetector's effectiveness
  - The detected inadequate messages
  - Time cost in inadequate message detection
  - Comparison with two existing techniques

4 mature configurable software systems

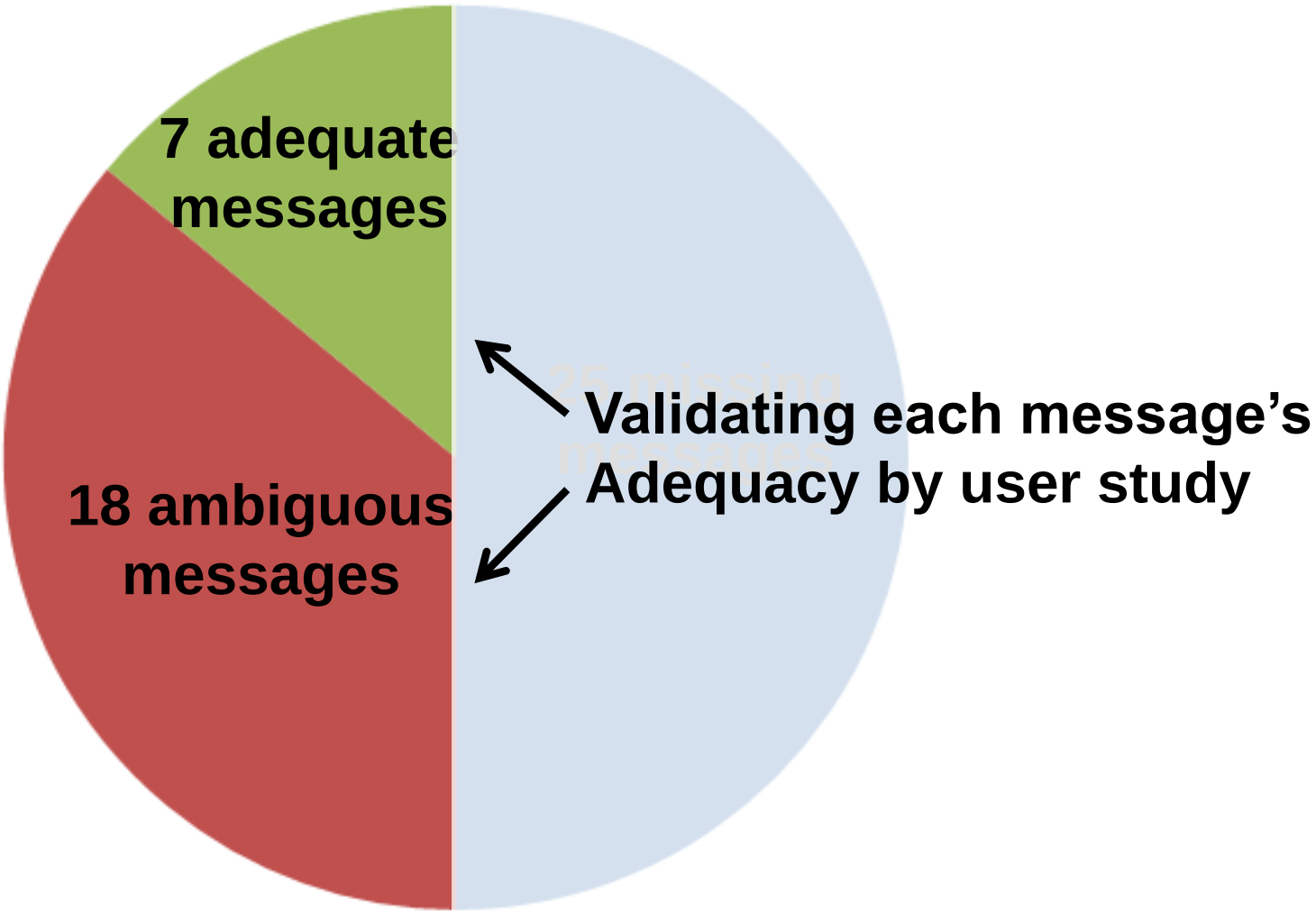
Subject	LOC	#Options	#System Tests
Weka	274,448	125	16
JMeter	91,979	212	5
Jetty	123,028	23	7
Derby	645,017	56	7

Converted from usage examples in the user manual.

Detected inadequate diagnostic messages

Detected inadequate diagnostic messages

Detected inadequate diagnostic messages



User study

User manual

Diagnostic message

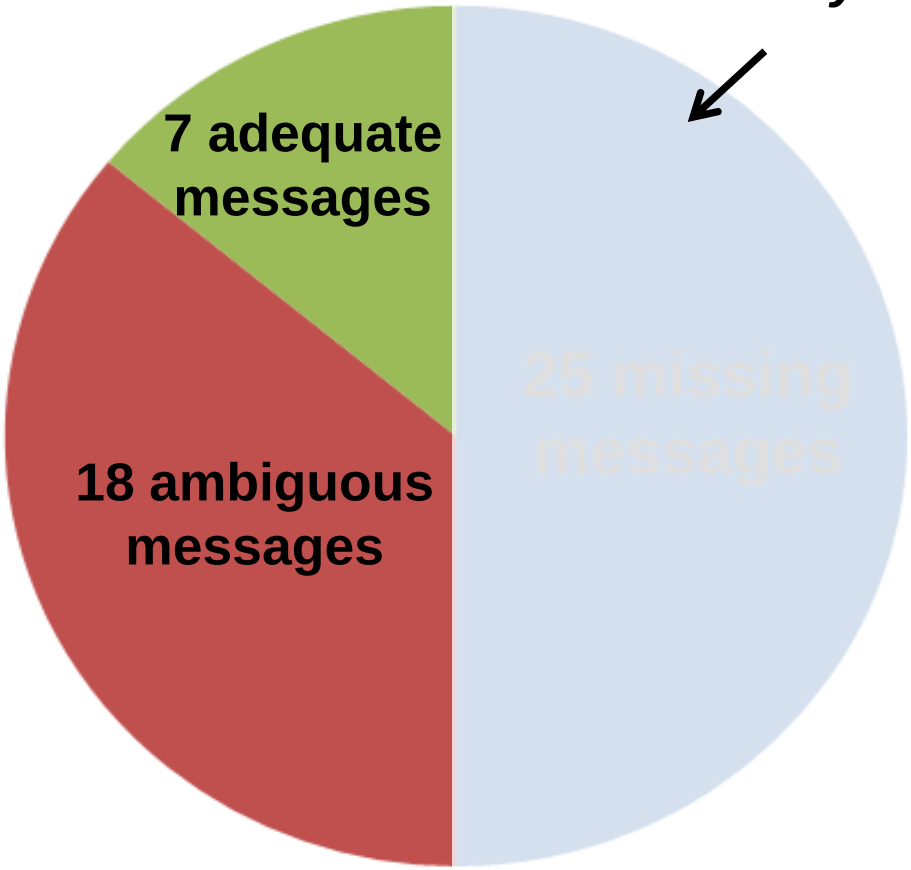
3 grad students

Each with 10 years coding experience

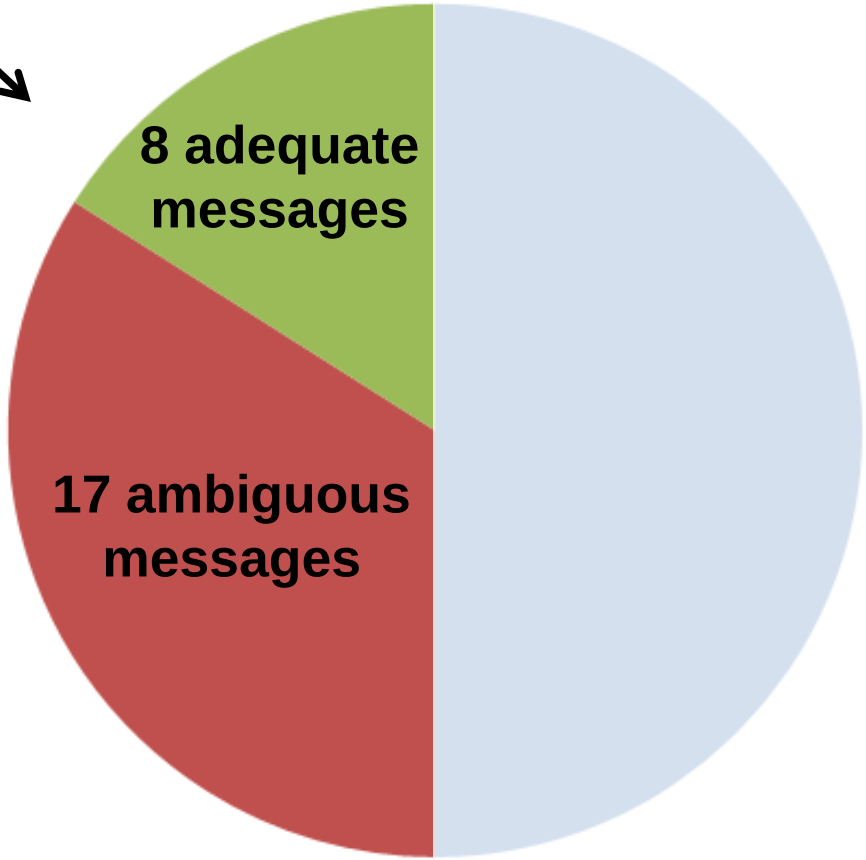
Adequate or not?

User study results

Differs only in 1 message



ConfDiagDetector's results



User's judgment

Zero false negative, and 2% false positive rate

Time cost

- **Manual effort**

  – 3.5 hours in total (4.2 minutes per message)
    - Converting usage examples into tests
    - Extract configuration option description from the user manual

- **ConfDiagDetector's efficiency**

  – 3 minutes per message, on average

Comparison with two existing techniques

- No Text Analysis
  - Implemented in ConfErr [Keller'08] and Spex-INJ [Yin'11]
  - A message is adequate if the misconfiguration option name or value appears in it
  - False positive rate: **16%** (ConfDiagDetector' rate: 2%)
- Internet search
  - Search the diagnostic message in Google
  - A message is adequate if the misconfiguration option appears in the top 10 entries
  - False positive rate: **12%** (ConfDiagDetector' rate: 2%)

Outline

- Motivation
- The ConfDiagDetector technique
- Evaluation
- Related work
- Contributions

Related work

- Configuration error diagnosis techniques
  - Dynamic tainting [Attariyan'08], static tainting [Rabkin'11], Chronus [Whitaker'04]

  Troubleshooting an exhibited error rather than detecting inadequate diagnostic messages

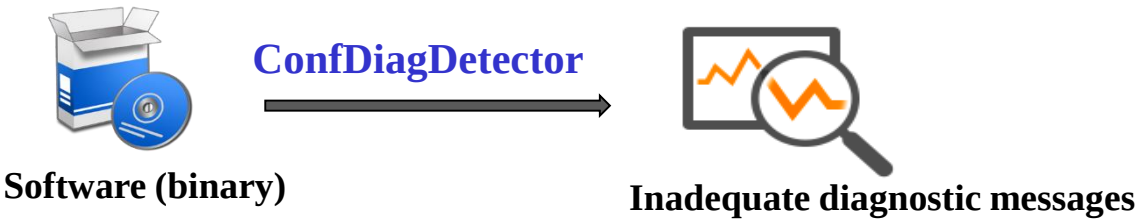
- Software diagnosability improvement techniques
  - PeerPressure [Wang'04], RangeFixer [Xiong'12], ConfErr [Keller'08] and Spex-INJ [Yin'11], EnCore [Zhang'14]

  Requires source code, usage history, or OS-level support

Outline

- Motivation
- The ConfDiagDetector technique
- Evaluation
- Related work
- Contributions

Contributions



- A technique to detect inadequate diagnostic messages

  Combine ***configuration mutation*** *and* ***NLP techniques***

  – Requires no source code and prior knowledge
  – Analyzes diagnostic messages in natural language
  – Requires no OS-level support
  – Accurate and fast

- An evaluation on 4 mature, configurable systems

  – Identify 25 missing and 18 inadequate messages
  – No false negative, 2% false positive rate