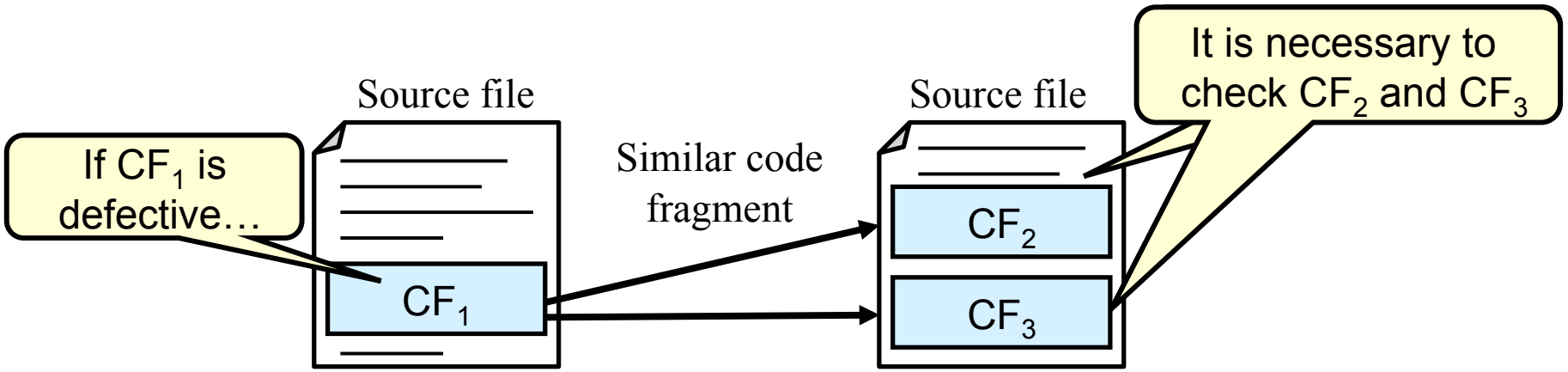
Similar code fragment

- A code fragment that has similar part to it in source code
  - introduced in source code because of various reasons.
    - e.g. “copy-and-paste”
  - makes software maintenance difficult.



Similar defects in Linux 2.6.6

```
for(iter=0; iter<num_regs; iter++) {
        prom_prom_taken[iter].start_adr =
                prom_reg_memlist[iter].phys_addr;
        prom_prom_taken[iter].num_bytes =
                prom_reg_memlist[iter].reg_size;
        prom_prom_taken[iter].theres_more =
                &prom_phys_total[iter+1];
// should be:&prom_prom_taken[iter+1];
}
```

```
for(iter=0; iter<num_regs; iter++) {
        prom_prom_taken[iter].start_adr =
                (char *) prom_reg_memlist[iter].phys_addr;
        prom_prom_taken[iter].num_bytes =
                (unsigned long) prom_reg_memlist[iter].reg_size;
        prom_prom_taken[iter].theres_more =
                &prom_phys_total[iter+1];
// should be:&prom_prom_taken[iter+1];
}
```

Similar defects in Linux 2.6.6

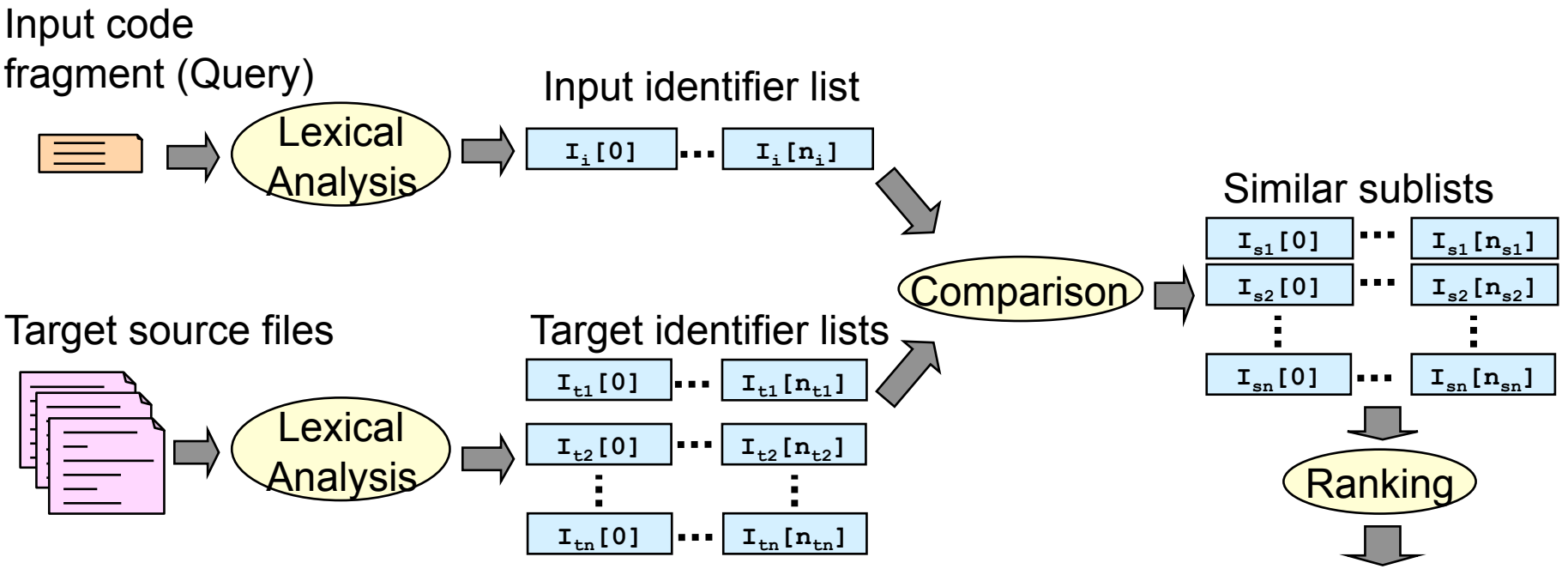
```
for(iter=0; iter<num_regs; iter++) {
        prom_prom_taken[iter].start_adr =
              prom_reg_memlist[iter].phys_addr;
        prom_prom_taken[iter].num_bytes =
              prom_reg_memlist[iter].reg_size;
        prom_prom_taken[iter].theres_more =
              &prom_phys_total[iter+1];
// should be:&prom_prom_taken[iter+1];
}
```

```
for(iter=0; iter<num_regs; iter++) {
        prom_prom_taken[iter].start_adr =
              (char *) prom_reg_memlist[iter].phys_addr;
        prom_prom_taken[iter].num_bytes =
              (unsigned long) prom_reg_memlist[iter].reg_size;
        prom_prom_taken[iter].theres_more =
              &prom_phys_total[iter+1];
// should be:&
}
```

Type cast operations are inserted.

Clone detection tools cannot treat the code fragments as a clone pair.

An overview of proposed method



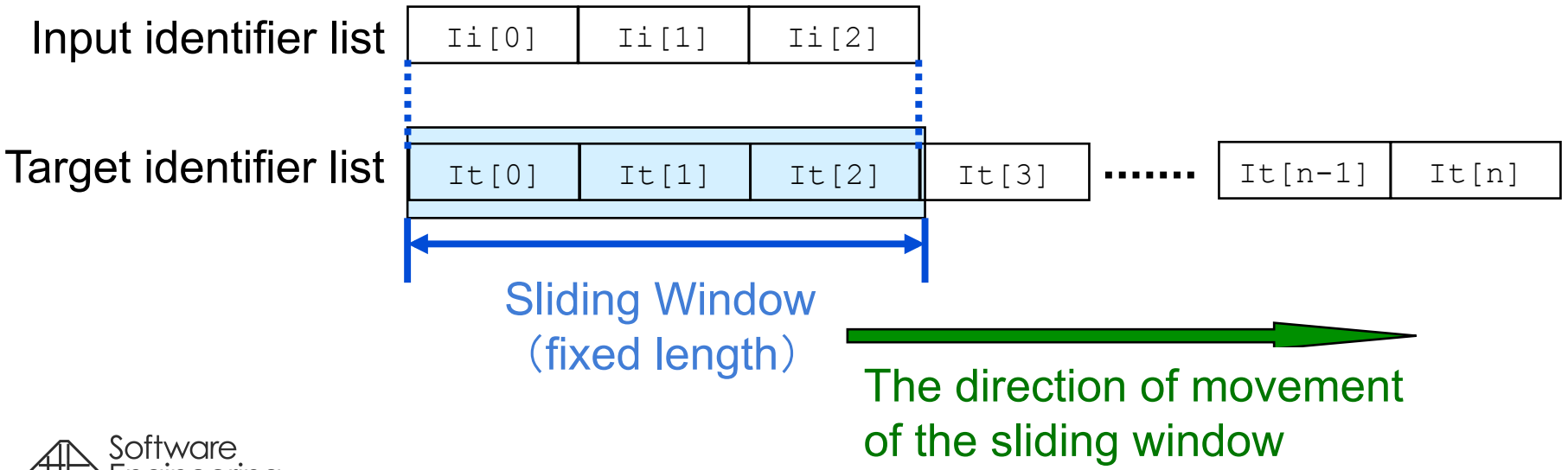
The method retrieves code fragments similar to an input code fragment.

Similarity Ranking

Rank	Start line #	End line #	Similarity
1	$Line_{s1}$	$Line_{e1}$	Sim_1
2	$Line_{s2}$	$Line_{e2}$	Sim_2
⋮	⋮	⋮	⋮

Comparison

- Scan a target identifier list with a sliding window
  - We compare identifiers in the sliding window with the input identifier list.
- Extract a code fragment corresponding to the sliding window if the window involves one or more identifiers in the input list



Similarity-based ranking

- The extracted code fragments are sorted according to the following similarity.
  - $\mathbf{S_i}$: a set of elements in an **input identifier list**
  - $\mathbf{S_w}$: a set of elements in a **sliding window**

$$\text{Similarity}(Si, Sw) = \frac{2 \times |Si \cap Sw|}{|Si| + |Sw|}$$

- Developers investigate the resultant ***similarity-based ranking***.

Case Study

- Target open source software systems
  - *arch/* directory in *Linux 2.6.6*
    - Architecture-specific implementations in OS
    - 2 incorrect pointer accesses
  - *server/* directory in *Canna 3.6*
    - Japanese input system
    - 19 buffer overflow errors
- Procedure
  1. extract code fragments sharing similar defects
  2. enter each code fragment into the tool implementing our method
  3. inspect if the similarity ranking ranks highly code fragments involving defects

Result

- *Linux 2.6.6*
  - We used 2 code fragments as queries.
    - Each code fragment involves an incorrect pointer access.
  - In both of those queries, **the 2 code fragments are the top 2**.
- *Canna 3.6*
  - We used 19 code fragments as queries.
    - Each code fragment involves a buffer overflow error.
  - In all of those queries, **18 or 19 code fragments are the top 30**.

In our case studies, we could detect most of similar defects.

Summary & Future work

- We proposed a method to retrieve similar code fragments based on identifier similarity.
  - Sliding window comparison
  - Similarity-based ranking
- We need further case studies.
  - Application to similar defects in other software systems
  - Effects from changing “similarity” definition