

# A Framework for Experimental Evaluation of Clustering Techniques

International Workshop on Program Comprehension, June 2000

Rainer Koschke and  
Thomas Eisenbarth

*University of Stuttgart*



# Clustering techniques for re-modularization

---

## connection-based

- *Global Object Reference*, Yeh et al. (1995)
- *Same Module*, Girard, Koschke (1997)
- *Part Type*, Liu, Wilde (1990)
- *Same Expression*, Koschke (2000)
- *Internal Access*, Yeh et al. (1995)

## metric-based

- *Delta-IC*, Canfora et al. (1993, 1996)
- *Arch*, Schwanke (1991)
- *Type-based Cohesion*, Patel et al. (1991)
- *Data Bindings*, Belady, Evangelisti (1984)
- *Data Bindings*, Hutchens, Basili (1984)

## graph-based

- *Strongly Connected Components*, Cimitile, Visaggio (1995)
- *Dominance Analysis*, Cimitile and Visaggio (1995)
- *Functional Cohesion*, Girard, Würthner (1999)



# Clustering techniques for re-modularization

---

## **concept-based**

- Lindig, Snelting (1997)
- Siff, Reps (1997)
- Sahraoui et al. (1997)
- Canfora et al. (1999)

## **dataflow-based**

- Valasareddi, Carver (1998)
- Gall, Klösch (1995)

## **domain-based**

- Gall, Klösch (1995)

## **semi-automatic bottom-up**

- Müller et al. (1993)
- Kazman, Carrière (1997)

## **semi-automatic combined (bottom-up and top-down)**

- Gall, Klösch, Weidl (1998)
  
- ... more to come...



# Commonalities and variabilities among techniques

---

## techniques are functions

- **input**
  - system entities to be grouped
    - subprograms
    - variables
    - user-defined types
  - a wide range of different information on these entities
- **output**
  - a set of candidate modules
    - module = set of related entities

$E \rightarrow \wp(E)$  where  $E$  is the set of entities



# Quantitative evaluation

---

## Goal

- have an oracle to compare the results of automatic techniques.

## Steps

- let software engineers detect modules
  - ⇒ **references**
- let automatic techniques propose modules
  - ⇒ **candidates**
- compare candidates to references
  - identify *immediate* corresponding candidates and references
    - ⇒ **good match**
  - identify corresponding *submodules*; i.e., a module corresponds only to a part of another module
    - ⇒ **O.k. match**
  - measure accuracy of correspondences
    - ⇒ **detection quality**

## But...

... when do modules actually correspond to each other?



# Good matches

---

## Considerations

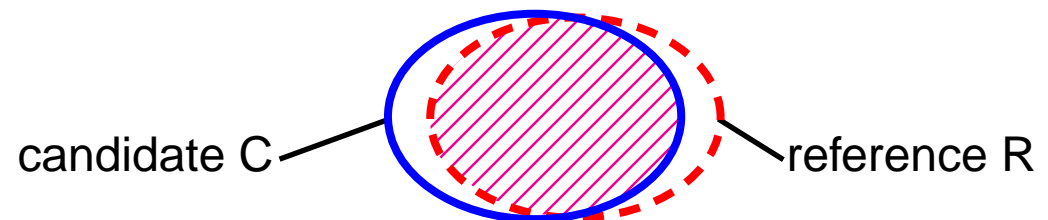
- modules are sets of entities
- software engineers do not always agree to 100%
- pragmatically, we cannot expect techniques to yield 100% accurate matches

## Good match $C \approx_p R$

$$\text{iff } \frac{|\text{elements}(C) \cap \text{elements}(R)|}{|\text{elements}(C) \cup \text{elements}(R)|} \geq p$$

## Tolerance parameter $p$

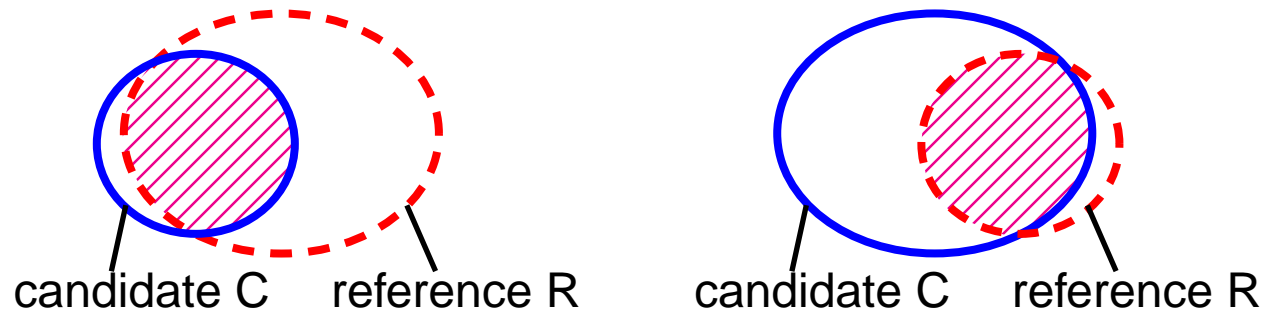
- if  $p = 1$ ,  $C$  and  $R$  must be the same
- more pragmatically,  $p = 0.7$ 
  - $C$  and  $R$  overlap at 70%
  - $\{a, b, c, d\} \approx_{0.7} \{b, c, d\}$ , overlap is  $3/4 = 0.75$
  - $\{a, b, c, d\} \not\approx_{0.7} \{b, c, d, e\}$ , overlap is  $3/5 = 0.6$



# Part-of matches

## Consideration

- also accept too fine-grained candidates
  - candidate matches only with a part of reference
- also accept too coarse-grained candidates
  - only a part of a candidate matches with a reference



- pragmatically, do not expect 100% containment

## Matching relation $S \subseteq_p T$

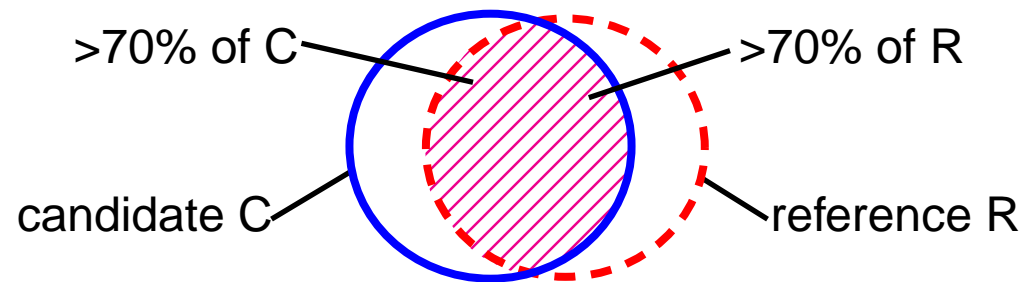
$$\text{iff } \frac{|\text{elements}(S) \cap \text{elements}(T)|}{|\text{elements}(S)|} \geq p$$

- $p$  is tolerance parameter as above
- $S$  is **part of**  $T$

# Mutually part-of matches

## Considerations

- $C \approx_p R \Rightarrow C \subseteq_p R \wedge R \subseteq_p C$
- but not:  $C \subseteq_p R \wedge R \subseteq_p C \Rightarrow C \approx_p R$
- yet, there is a distinct correspondence between C and R if  $C \subseteq_p R \wedge R \subseteq_p C$



## C and R are a mutually part-of match iff

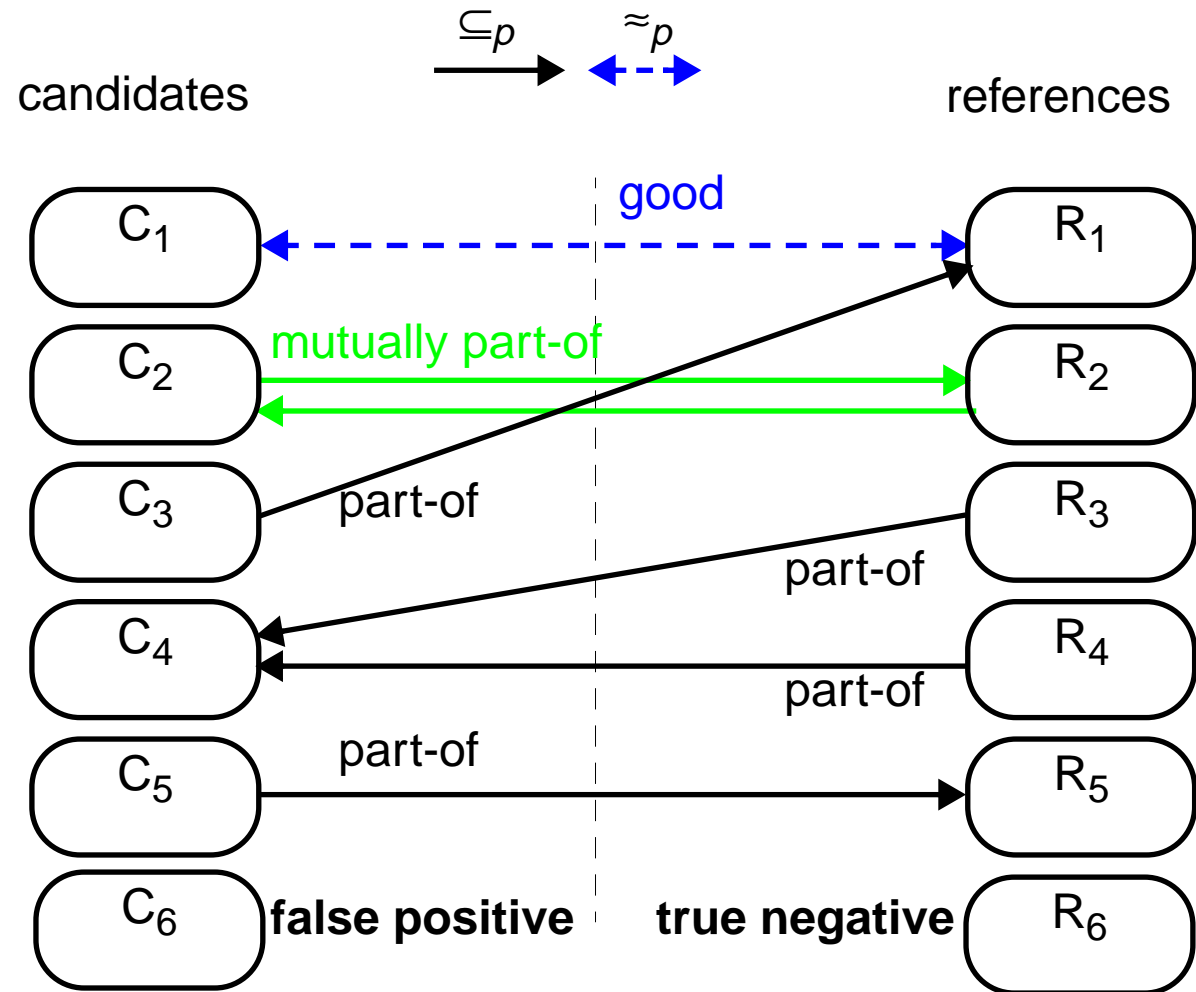
- $C \subseteq_p R \wedge R \subseteq_p C$
- ⇒ mutually part-of matches are part-of matches
- ⇒ good matches are mutually part-of matches

## O.k. matches

- part-of and mutually part-of are O.k. matches

# Summary of correspondences

## Example correspondence map



# Accuracy of good matches

## Matches can be non-exact

- use degree of overlap to assess accuracy of a match
- $\text{overlap}(C, R) = \frac{|C \cap R|}{|C \cup R|}$

## Pitfall:

let **C** and **R** be a good match

- $\text{accuracy}(C, R) = \text{overlap}(\text{elements}(C), \text{elements}(R))$  is not sufficient due to additional part-of matches
- $\text{matchings}(X) = \{Y \mid Y \subseteq_p X\}$

**accuracy (C, R) where C and R are a good match**

$$\text{overlap}\left(\bigcup_{r_i \in \text{matchings}(C)} \text{elements}(r_i), \bigcup_{c_i \in \text{matchings}(R)} \text{elements}(c_i)\right)$$

## Example



## Consequence

- ⇒ additional part-of matches are subsumed under good matches

# Accuracy of mutually part-of matches

---

**Pitfall:**

**let C and R be a mutually part-of match**

- C or R can be involved in a good match
  - only possible if there are overlapping candidates or references
  - even then very unlikely
- C or R can have additional part-of matches (more likely)

**if C or R is involved in a good match**

- ignore the match (C, R)
  - it is already subsumed by the dominant good match

**if neither C nor R is involved in a good match**

$$\text{overlap} \left( \bigcup_{r_i \in \text{matchings}(C)} \text{elements}(r_i), \bigcup_{c_i \in \text{matchings}(R)} \text{elements}(c_i) \right)$$

- ⇒ additional part-of matches are subsumed by mutually part-of matches



# Accuracy of plain part-of matches

**Pitfall:**  
let C be a part-of R

if C or R is involved in a good or mutually part-of match

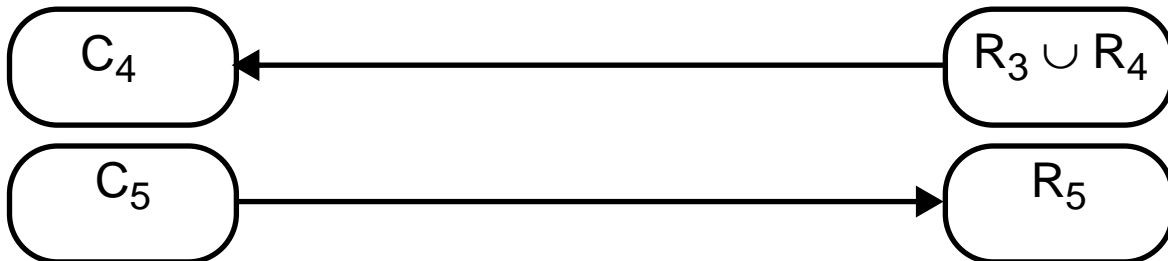
if neither C nor R is involved in a good or mutually part-of match

- C and R can be involved in a good or mutually part-of match
- R can have additional parts
- ignore the plain part-of match (C, R)
  - it is already subsumed under a dominant match

$$\text{overlap}\left(\bigcup_{c_i \in \text{matchings}(R)} \text{elements}(c_i), R\right)$$

**Example**

Analogous for references that are part of a candidate.



# Detection quality

---

## Multiple aspects

- **precision**
  - # false positives
  - average size of false positives
- **recall**
  - # true negatives
  - granularity of matches
    - # good matches
    - # mutually part-of matches
    - # part-of matches
- **precision and recall**
  - accuracy of each match

## Overall recall for quick overview

$$\frac{\sum_{(a, b) \in \text{GOOD}} \text{accuracy}(a, b) + \sum_{(a, b) \in \text{OK}} \text{accuracy}(a, b)}{|\text{GOOD}| + |\text{OK}| + |\text{true negatives}|}$$



# Evaluation of semi-automatic methods

---

## Considerations

- humans are involved
  - ⇒ controlled experiments are needed
- controlled experiments with many experimental subjects are expensive

## Experimental design

- completely randomized assignment of subjects to  $n+1$  disjoint groups:
  - 1 group that searches for modules without automatic support beyond *grep* and simple cross-reference tools
  - $n$  groups using  $n$  different semi-automatic methods

## What to measure:

- absolute number of clustered elements of each subject
  - ⇒ does not require agreement among subjects
  - ⇒ measures “*speed*” of search
- recall of each subject with respect to reference corpus
  - reference corpus may be the set of merged and validated references of all subjects



# Evaluation of semi-automatic methods

---

## Considerations

- large-scale experiments are expensive if not even unfeasible
  - ⇒ use fewer people
  - ⇒ **but:** use the appropriate statistics
    - ANOVA is not appropriate for experiments with few participants
    - use non-parameterized statistics

## How to measure:

### 1. Exact U-test (Mann, Withney)

- assumes data at ordinal scale (only lesser/greater relationship is leveraged)
  - ⇒ ignores actual differences in recall

### 2. Exact Fisher-Pitman randomization test

- assumes data at interval scale
  - ⇒ actual differences are taken into account
  - ⇒ assumes samples to be an exact image of the whole population



# Gathering benchmarks

---

## Aspects of benchmarks (reference corpora)

- **system characteristics**
    - size, programming language, application domain, age, #subprograms, #user-defined types, #global variables
  - **people**
    - #people who analyzed the system, characteristics of these people (experience, familiarity with the system, etc.) analyzed jointly or separately?
  - **validity**
    - were the references reviewed by others?
  - **available time**
    - time spent on the analysis
- Ideally:**
- teams should exhaustively and jointly investigate a realistic system in sufficient time and the results should be reviewed by independent reviewers
- In reality:**
- gather the above characteristics to allow researchers to judge a benchmark



# Conclusions

---

## Today's situation

- overwhelming number of techniques
- but no comparable evaluation
  - Lakothia and Gravely pointed out this deplorable state of affairs in 1995
  - little has been achieved since then

## Our proposal

- easy to use and precise measurement of recall and precision
- suitable for module recovery techniques
- not suited for subsystem recovery techniques
  - where hierarchical artifacts are derived

## How to put the framework into action

- tool for quantitative evaluation is made available
- our benchmark systems (altogether 100 KLOC of 4 C programs) will be made available
  - we expect feedback and contributions, however

**<http://www.informatik.uni-stuttgart.de/ifi/ps/clustering>**

