Architectural Design Recovery Using Data Mining Techniques

Kamran Sartipi [1]
Kostas Kontogiannis [2]
Farhad Mavaddat [1]

Dept. of Computer Science [1]
Dept. of Electrical & Computer Engineering [2]

University of Waterloo
Canada

1

Outline

- Introduction.
- Application of data mining techniques in architectural recovery.
- Architectural Query Language (AQL).
- Branch and bound search.
- Features of the recovery tool.
- Result of the recovery process.
- Related work and contributions.

2

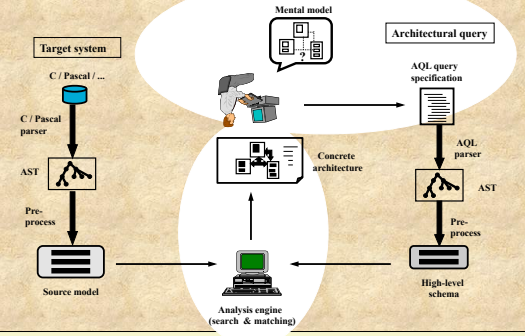
Software Architecture Recovery

Extracting high-level information from low-level software representation (e.g., source code).

- Constitutes a major part of software maintenance.
- Should relate with specific re-engineering requirements.
- Major approaches:
  - Clustering techniques based on static and dynamic properties.
  - Constraint satisfaction to satisfy user-defined constraints.
  - Query-based techniques based on specialized queries and high level architectural styles.

3

Structured Query-based Software Architecture Recovery



4

Data Mining Technique (Apriori)

- Discovery of interesting and non-trivial relations among data in large databases
- Apriori algorithm [Agrawal]
  - Frequent itemsets

Itemset	Baskets
	1, 3, 5, 6
	1, 2, 3, 5, 8
	1, 3, 4, 5
	1, 3, 5

  - Discovery of association rules (X ⟶ Y)
    e.g., 60% of the transactions that contain set also contain that is:
    ⟶ 3 / 5 = 60% (confidence)

5

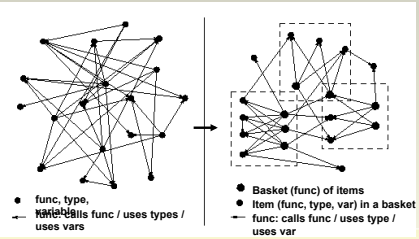
Extending the relations in Data Mining to Reverse Engineering

	Entity	Relation	Entity
Data Mining	transaction	consists-of	item
Reverse Engineering	container	consists-of	item-operation
	file / function file / function file / function file / function file	consists-of	call-to — file / function use (read / write) — file / type / var write-to / read-from — pipe send-to / receive-from — socket import / export — function / types/ var
	containment	consists-of	item-operation
	function file / type / var …	consists-of	called-by — file / function used (read / written) by — file / function …

6

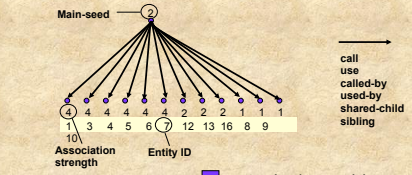
Application o f Data Mining in Recovery Process

Revealing cohesive groups of entities in the form of bi-partite sub-graphs in the target system' s graph

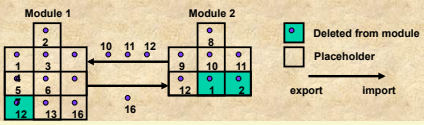


7

Relationships and association strength between a main-seed and its domain entities



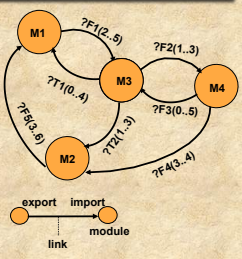
Assigning entities to modules



8

Architecture Query Language (AQL)

Conceptual Architectural Pattern

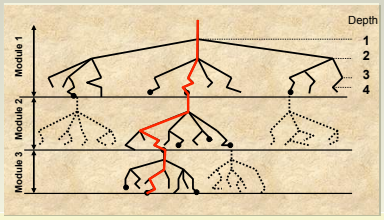



```
MODULE: M1
  MAIN-SEED:      func  search_class ()

  IMPORTS:
    FUNCTIONS:    func  ?IF,
                  func  ?F5(3..6)  M2
    TYPES:        type  ?IT,
                  type  ?T1(0..4)  M3
    VARIABLES:    var   ?IV

  EXPORTS:
    FUNCTIONS:    func  ?EF,
                  func  ?F1(2..5)  M3
    TYPES:        type  ?ET
    VARIABLES:    var   ?EV

  CONTAINS:
    FUNCTIONS:    func  $CF(15 .. 18),
                  func   search_class (),
                  func   inherit_facts (),

    TYPES:        type  $CT(0 .. 2)
    VARIABLES:    var   $CV(3 .. 5)

END-ENTITY
```

9

Branch and bound search

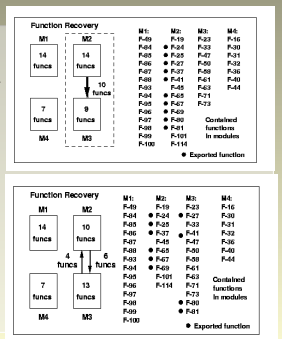
- Separate searches for each module.
- Overall backtracking for different modules.
- Satisfying similarity and link constraints.



10

Architectural Recovery & Restructuring

- Recovery:
  - Similarity constraints only.
  - Decomposing three files of CLIPS system into four modules.
- Restructuring:
  - Both similarity and link constraints.
  - Rearranging the entities of the recovered modules M2 and M3.

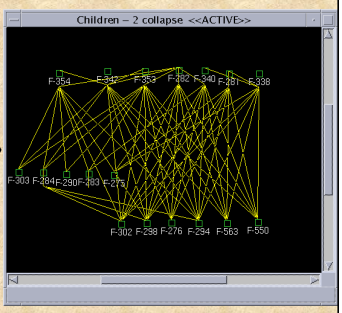


11

User interface:

Gaph visualizater (Rigi)

- Results as hierarchy of subsystems and modules
- Properties of entities in recovered modules (bi-partite sub-graphs).
- Import/export links between modules.
- Assistance in decomposing the system using domain coupling analysis.

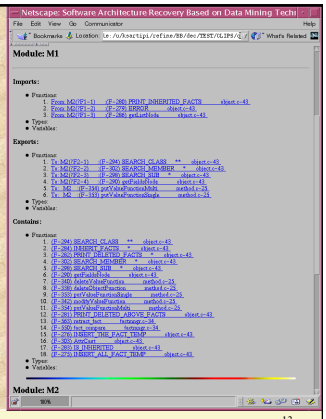


12

User interface:

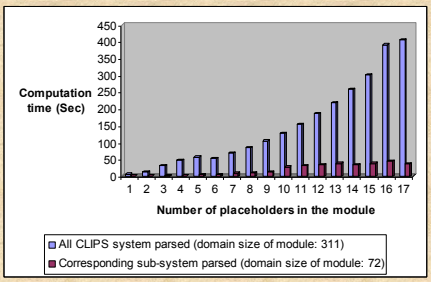
Web browser(NetScape)

- Hypertext links to actual entities in the source file.
- Various information:
  - Distribution of recovered entities into files.
  - Browsing the query.
  - Statistical information for link-constraint violations.
  - Links between modules.



13

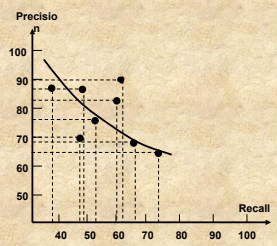
Run-time Complexity of the Recovery Process



14

Matching Process Accuracy

- Recovery of CLIPS files as modules.
- Precision level of %90 for a Recall level of %60.



15

Conclusion

- Goal driven analysis and decomposition
- Design of a query language to compose structural queries.
- Semi-Automatic analysis at subsystem and module levels.
- Data-mining as scoring mechanism.
- AI search technique to provide efficiency for approximate matching analysis.

16

Architectural Design Recovery Using Data Mining Techniques

Kamran Sartipi [1]
Kostas Kontogiannis [2]
Farhad Mavaddat [1]

Dept. of Computer Science [1]
Dept. of Electrical & Computer Engineering [2]

University of Waterloo
Canada

17

Source model: Collection of Domains

- Frequent itemsets (4-itemsets)

1) <[V-3, T-42, T-44, T-58] [**F-83**, F-176, F-464, F-647] 4>
2) <[V-3, T-43, T-44, T-58] [F-83, F-647] 2>
3) <[V-3, F-478, F-649, F-719] [F-647, F-648] 2>
4) <[V-4, T-41, T-42, T-44] [F-83, F-647, F-648] 3>
5) <V-30, F-552, F-553, F-567] [F-547, F-548] 2>

- Domain of an entity in the system

$$D(s) = \{d \mid \forall k \in [1..|F|], \{d, s\} \subset (F_k.I \cup F_k.T)\}$$

- Example of a domain

Domain-of (**F-83**) =
{<V-3, 4> <T-42, 4> <T-44, 4> <T-58, 4> <F-176, 4> <F-646, 4> <F-647, 4> <V-4, 3> <T-41, 3> <F-648, 3> <T-43, 2>}

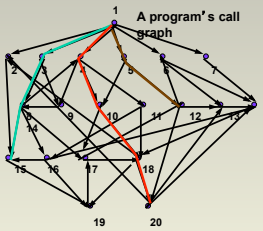
19

Features of the Architectural Recovery tool

- Hierarchically decomposing the target system into subsystems and modules.
- Architectural recovery and restructuring.
- Automatic and incremental recovery modes.
- Automatic creation of template queries by analyzing the source model.
- Decomposing the target system into subsystems based on domain-coupling notion.
- Interface to Netscape and Rigi tool.

20

Future Work: Sequential patterns in behavioral recovery

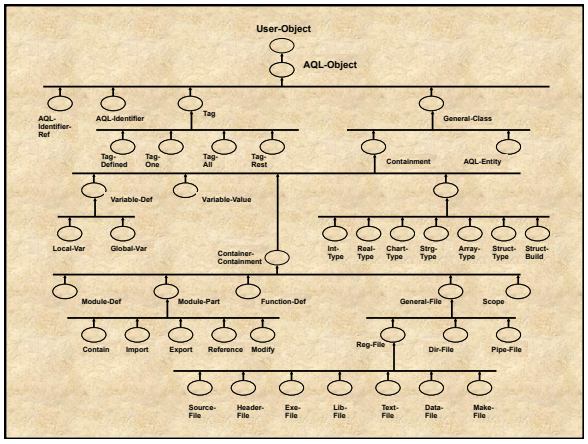


Frequent sequential pattern | Support
1, 4, 10, 18, 20 | 3
1, 3, 8, 15 | 3
1, 5, 12 | 3

Event trace

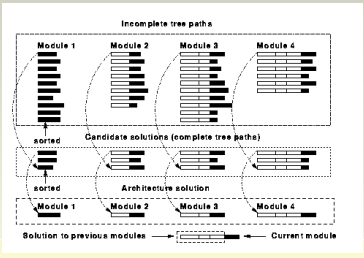
1) 1, 3, 8, 4, 9
2) 1, 2, 15, 19, 17, 12, 13, 14
3) 1, 3, 8,16, 15
4) 1, 4, 10, 18, 20
5) 1, 5, 10, 18, 20, 13, 14
6) 1, 6, 14
7) 1, 5, 12, 13
8) 1, 5, 12, 20, 13, 14, 18, 20, 13
9) 1, 3, 9, 8, 15
10) 1, 7, 14
11) 1, 5, 11, 6
12) 1, 4, 10, 17, 18, 20
13) 1, 5, 10, 17
14) 1, 4, 11, 16, 15
15) 1, 4, 10, 18, 19, 17, 18, 20
16) 1, 5, 12
17) 1, 3, 9, 8, 15

21



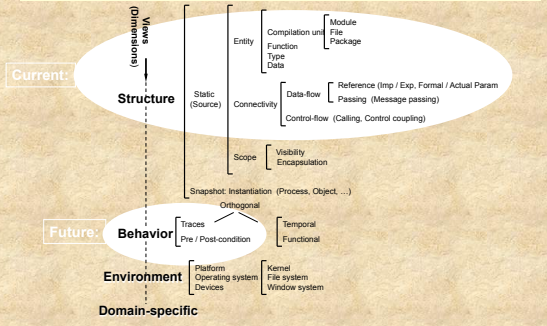
Implementation of branch & bound

- Candidate solutions
- Backtracking



23

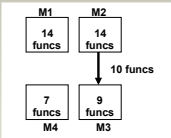
Feature Classification of Software Architecture Views



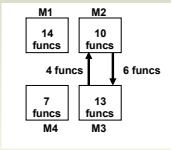
24

Architectural Recovery & Restructuring

- Recovery:
  - Similarity constraints only.
  - Decomposing three files of CLIPS system into four modules.



- Restructuring:
  - Both similarity and link constraints.
  - Rearranging the entities of the recovered modules M2 and M3.



25

Related Work

- Software reflexion model [Murphy & Notkin]
- Approximate matching [Kontogiannis]
- Recognizing architectural aspects
  - styles [Harris] [Girard]
  - views [Fiutem]
- Portable bookshelf [Holt]
- Clustering files into subsystems [Mancordis]
- ADLs [Unicon, ACME, Rapide]

26