b
UNIVERSITÄT
BERN

Software Architecture Recovery

Mircea Lungu

Selected material courtesy Oscar Nierstrasz

Roadmap

- Introduction to SAR
- Top-down SAR
- Bottom-up SAR
- Tool Demo

Roadmap

- **Introduction to SAR**
  - Architecture
  - Viewpoints, Styles, ADL's
  - Recovery
- Top-down SAR
- Bottom-up SAR
- Tool Demo

Roadmap

- Introduction to SAR
  - **Architecture**
  - Viewpoints, Styles, ADL's
  - Recovery
- Top-down SAR
- Bottom-up SAR
- Tool Demo

衆聲

探象之圖

Architecture is...

...an abstraction level

Architecture

Design

Code

[Lungu09]

Architecture is...

...an abstraction level

Architecture

Design

Code

[Lungu09]

"[...] the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution."

[IEEE 1421]

Roadmap

- Introduction to SAR
  - Architecture
  - **Viewpoints, Styles, ADL's**
  - Recovery
- Top-down SAR
- Bottom-up SAR
- Tool Demo

Related Concepts

Related Concepts

- “A system stakeholder is an individual, team, or organization with interests in, or concerns relative to, a system.”

Related Concepts

- "A system stakeholder is an individual, team, or organization with interests in, or concerns relative to, a system."
- "A concern is an interest which pertains to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations as performance, reliability, security, distribution, and evolvability."

Related Concepts

- "A system stakeholder is an individual, team, or organization with interests in, or concerns relative to, a system."
- "A concern is an interest which pertains to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations as performance, reliability, security, distribution, and evolvability."
- "A view is a representation of a whole system from the perspective of a related set of concerns."

Related Concepts

- "A system stakeholder is an individual, team, or organization with interests in, or concerns relative to, a system."
- "A concern is an interest which pertains to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations as performance, reliability, security, distribution, and evolvability."
- "A view is a representation of a whole system from the perspective of a related set of concerns."
- "A viewpoint is a specification of the conventions for constructing and using a views. A pattern or a template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis."

Architectural Viewpoints...

- Consensus in SE community
- Viewpoints catalogues
  - Kruchten'95
  - Hofmeister'99

Architectural Viewpoints

Run-time	How are responsibilities distributed amongst run-time entities?
Process	How many concurrent threads/processes exist; how do they they communicate and synchronize?
Dataflow	How do data and tasks flow through the system?
Deployment	How are components physically distributed?
Module	How is the software partitioned into modules?
Build	What dependencies exist between modules?
File	How is the software physically distributed in the file system?

Architectural Viewpoints

Run-time	How are responsibilities distributed amongst run-time entities?
Process	How many concurrent threads/processes exist; how do they they communicate and synchronize?
Dataflow	How do data and tasks flow through the system?
Deployment	How are components physically distributed?
Module	How is the software partitioned into modules?
Build	What dependencies exist bet
File	How is the software

Most of the architecture recovery processes focus on recovering module viewpoints.

Architectural Styles

An architectural style defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style defines a vocabulary of components and connector types, and a set of constraints on how they can be combined.

— Shaw and Garlan

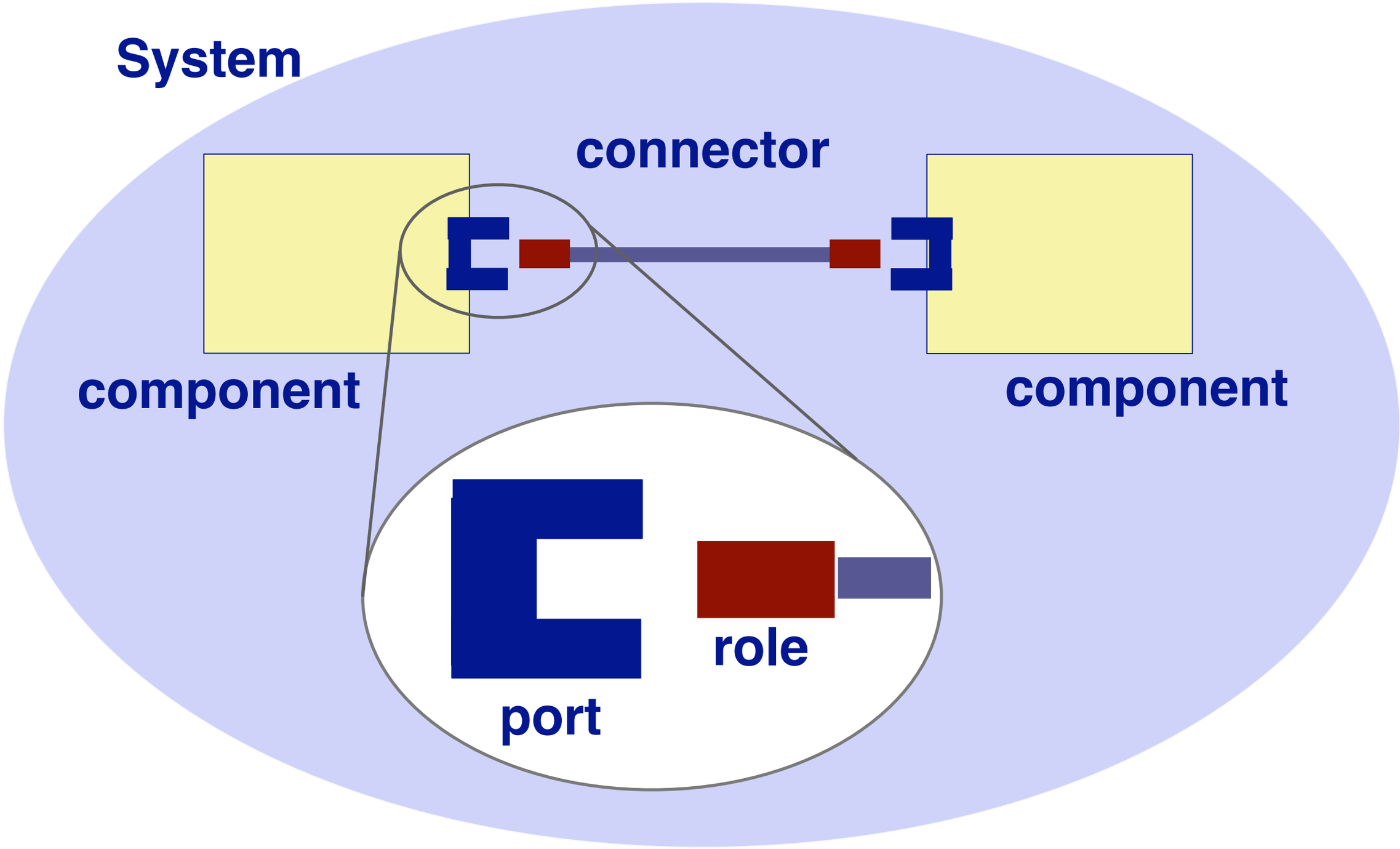
Classical Architectural Styles

Layered	Elements in a given layer can only see the layer below. Callbacks used to communicate upwards
Client-Server	Separate application logic from interaction logic. Clients may be “fat” or “thin”
4-Tier	Server is further divided into generic part, business logic and legacy adaptor.
Dataflow	Data or tasks strictly flow “downstream”.
Blackboard	Tools or applications coordinate through shared repository.

Architectural Description Languages

- ADLs
  - Formal languages for representing and reasoning about software architecture.
  - Provide a conceptual framework and a concrete syntax for characterizing architectures.
  - Some are executable, or implemented in a general-purpose programming language.

Common ADL Concepts



Common ADL Concepts

- **Component**: unit of computation or data store. Typically contains interface (ports) and formal behavioral description.

- **Connector**: architectural building block used to model interactions among components. Typically contains interface (roles) and formal behavioral description.

- **Configuration**: connected graphs of components and connectors that describe architectural structure.

Some ADLs

- **Darwin**: focuses on supporting distributed applications. Components are single-threaded active objects.

- **Wright**: underlying model is CSP, focuses on connectivity of concurrent components.

- **C2**: component- and message-based architectural style with concurrent components linked together by connectors in accordance with a set of style rules.

- **Rapide**: focuses on developing a new technology for building large-scale, distributed multi-language systems.

http://en.wikipedia.org/wiki/Architecture_Description_Language

Roadmap

- Introduction to SAR
  - Architecture
  - Viewpoints, Styles, ADL's
  - **Recovery**
- Top-down SAR
- Bottom-up SAR
- Tool Demo

Architecture Recovery

- A process important for
  - consulting
  - re-architecting
  - refactoring

- Two types of process
  - Top-Down
  - Bottom-Up

Roadmap

- Introduction to SAR
- **Top-down SAR**
  - **Overview**
  - **Reflexion Models**
- Bottom-up SAR
- Tool Demo

Top-Down SAR: Overview

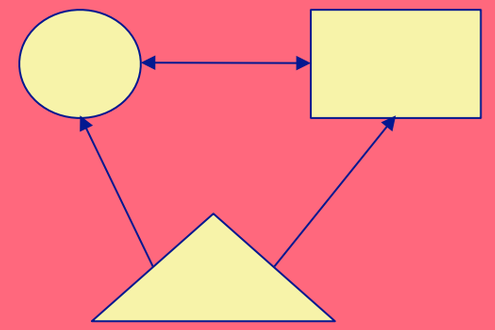
- Verifies whether the system conforms to the model the stakeholders have in mind
- Different approaches
  - Reflexion Models
  - Save/Pulse

Reflexion models

- Semi-automated approach
- Repeat
  - —Define/Update *high-level model* of interest
  - —Extract a *source model*
  - —Define/Update declarative mapping between high-level model and source model
  - —System computes a *software reflexion model*
  - —Interpret the software reflexion model.
- Until “happy”

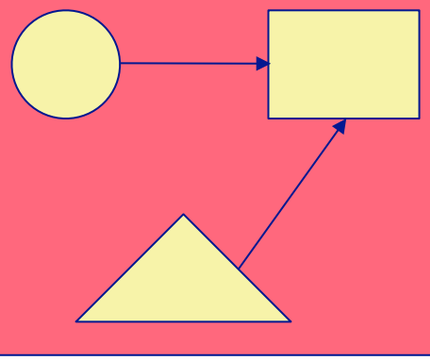
Reflexion Approach

Reflexion Model



Mapping

High-Level Model



Extraction Specification

Source Code

Extraction Tool

Reflexion Model Tools

Source Model

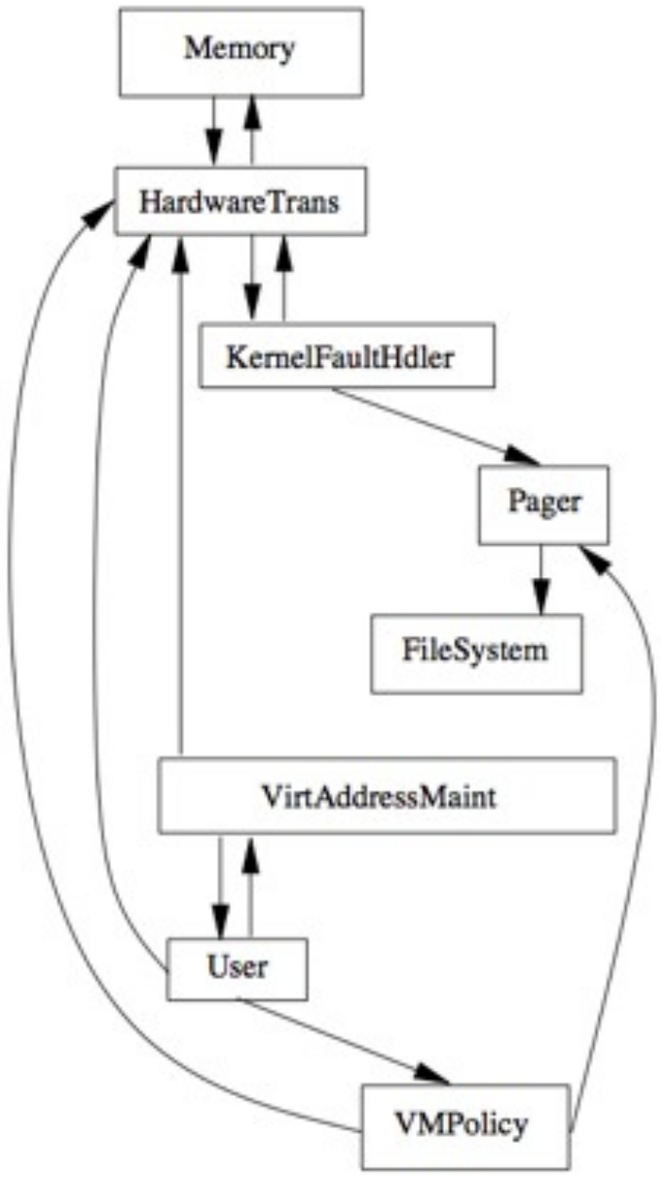
Software Reflexion Model

- Indicates where the source model and high-level model differ:
  - Convergences
  - Divergences
  - Absences
- Has to be interpreted by developer.

Case Study: The VMS of the Linux Kernel

VMS = Virtual Memory System

The High-level Model



Mapping

- Relates source model entities to high-level model entities.

- Example:

```
[ file= .*pager.*          mapTo=Pager ]
[ file= vm_map.*           mapTo=VirtAddressMaint ]
[ file=vm_fault\.c         mapTo=KernelFaultHandler ]
[ dir=[un]fs               mapTo=FileSystem ]
[ dir=sparc/mem.*]         mapTo=Memory ]
[ file=pmap.*              mapTo=HardwareTrans ]
[ file=vm_pageout\.c       mapTo=VMPolicy ]
```

Source Model

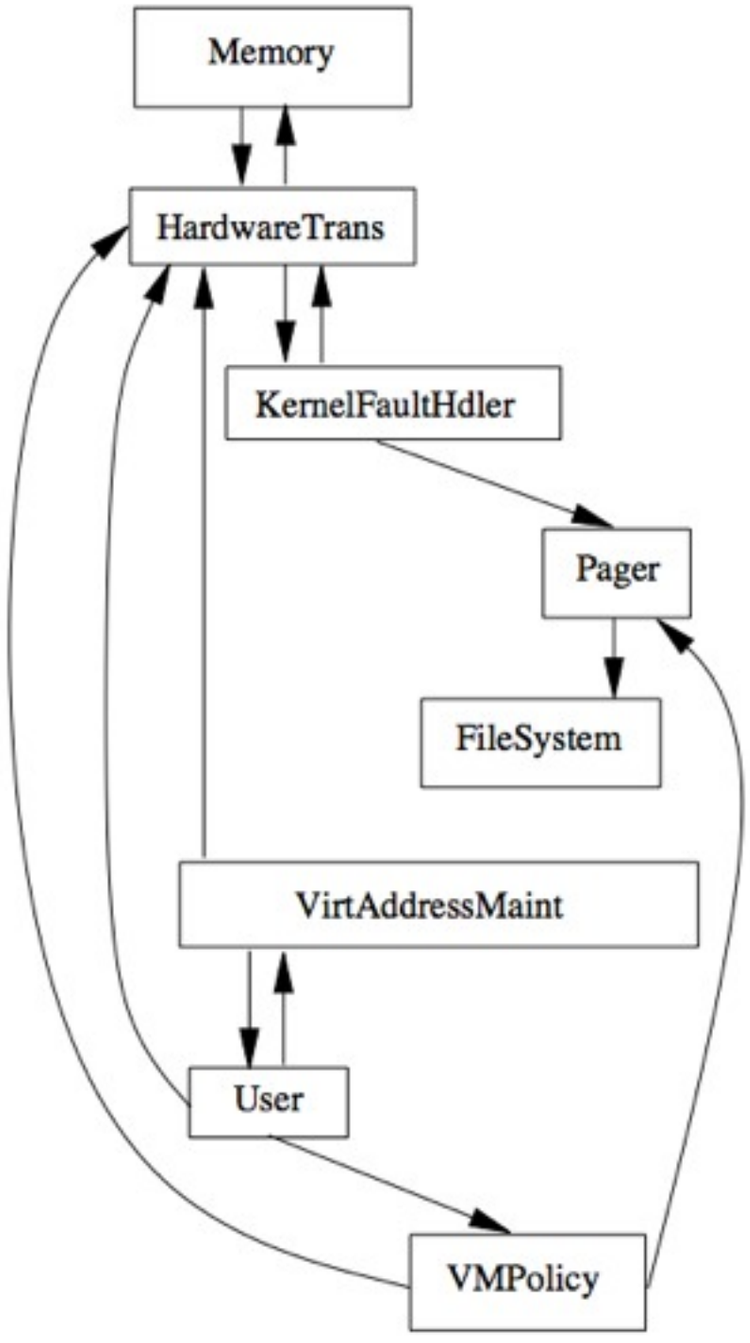
- Particular information extracted from source code
- Calculated with Lightweight Source extraction:
  - Flexible: few constraints on source
  - Tolerant: source code can be incomplete, not compilable, …
- Lexical Approach
- Intrinsically Approximate
- For every Source Model, a new scanner is generated

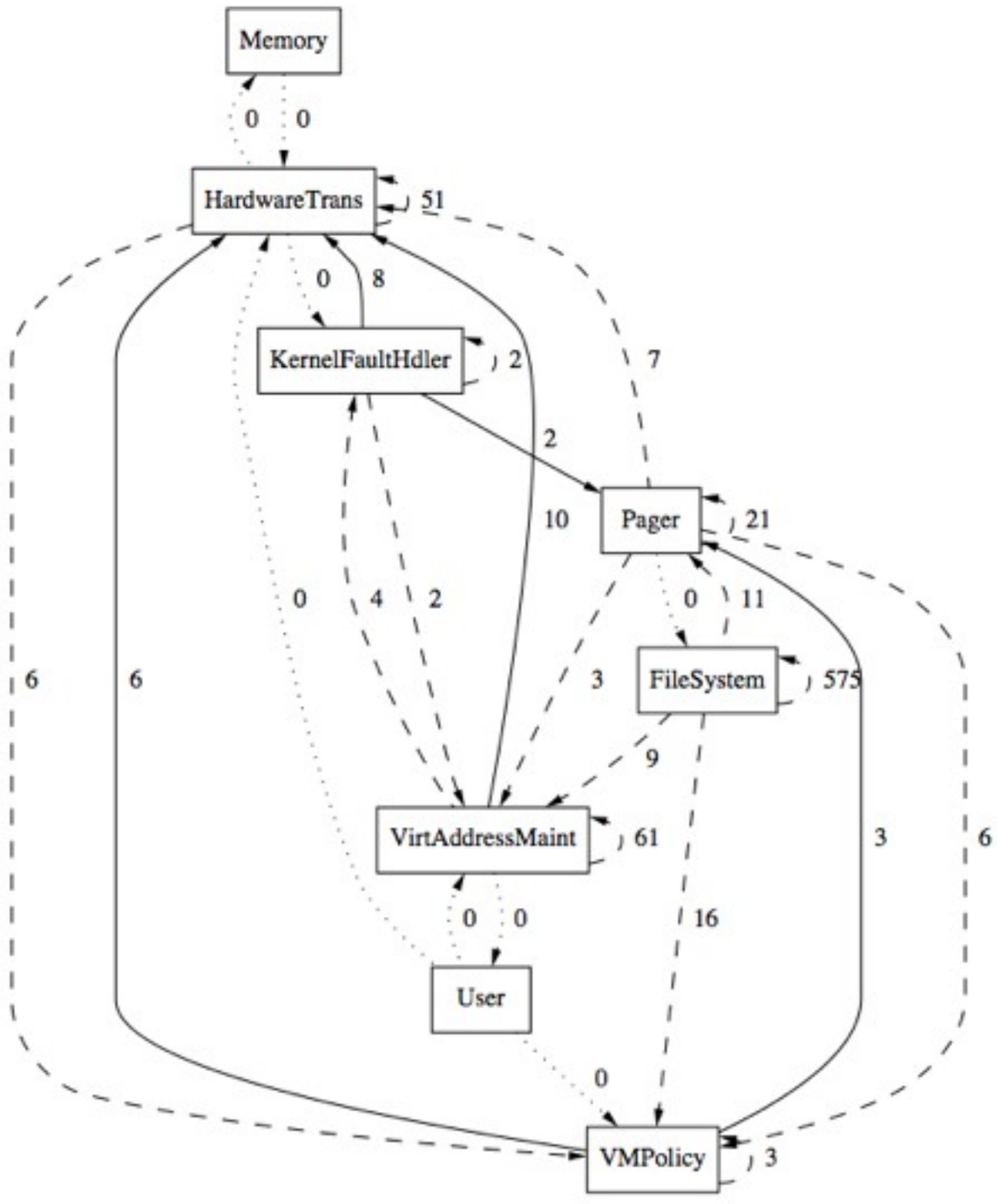
Source Model Specification

- Writing out C calls

```
[ <type> ] <functionName> \( [ { <formalArg> }+ ] \)
            [ { <type> <argDecl> ; }+ ] \{

    <calledFunction>
    @ write ( functionName, " calls ", calledFunction ) @
    \( [ { <parm> }+ ] \) ( \) | ; )
```

A Reflexion Model





Roadmap

- Introduction to SAR
- Top-down SAR
- **Bottom-up SAR**
  - The Architecture of Architecture Recovery Tools
  - Data Extraction
  - Knowledge Organization
  - Exploration
  - Examples
- Tool Demo

Top-Down SAR: Overview

- Tries to verify whether the system conforms to the model the stakeholders have in mind
- Different approaches
  - Reflexion Models
  - Save/Pulse

Roadmap

- Introduction to SAR
- Top-down SAR
- **Bottom-up SAR**
  - **The Architecture of Architecture Recovery Tools**
  - Data Extraction
  - Knowledge Organization
  - Exploration
  - Examples
- Tool Demo

1. Data Extraction

Alborz [110]
ArchView [99]
ArchVis [45]
ARES [26]
ARM [40]
ARMIN [58]
ART [32]
Bauhaus [13, 25, 62]
Bunch [79, 90]
Cacophony [28]
Dali [56, 57]
DiscoTect [146]
Focus [18, 84]
Gupro [24]
Intensive [87, 145]
ManSART [4, 43]
MAP [117]
PBS/SBS [8, 31, 49, 113]
PuLSE/SAVE [61, 103]
QADSAR [118, 119]
Revealer [100, 101]
RMTool [92, 93]
SARTool [30, 64]
SAVE [89, 94]
Softwarenaut [77]
Symphony,Nimeta [106, 135]
URCA
W4 [44]
X-Ray [86]

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

1. Data Extraction

	src
Alborz [110]	x
ArchView [99]	x
ArchVis [45]	x
ARES [26]	x
ARM [40]	x
ARMIN [58]	x
ART [32]	x
Bauhaus [13, 25, 62]	x
Bunch [79, 90]	x
Cacophony [28]	
Dali [56, 57]	x
DiscoTect [146]	x
Focus [18, 84]	x
Gupro [24]	x
Intensive [87, 145]	x
ManSART [4, 43]	x
MAP [117]	x
PBS/SBS [8, 31, 49, 113]	x
PuLSE/SAVE [61, 103]	x
QADSAR [118, 119]	x
Revealer [100, 101]	x
RMTool [92, 93]	x
SARTool [30, 64]	x
SAVE [89, 94]	x
Softwarenaut [77]	x
Symphony,Nimeta [106, 135]	
URCA	
W4 [44]	x
X-Ray [86]	x

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

1. Data Extraction

	src	text
Alborz [110]	x	
ArchView [99]	x	
ArchVis [45]	x	x
ARES [26]	x	
ARM [40]	x	
ARMIN [58]	x	
ART [32]	x	
Bauhaus [13, 25, 62]	x	
Bunch [79, 90]	x	
Cacophony [28]		
Dali [56, 57]	x	
DiscoTect [146]	x	
Focus [18, 84]	x	
Gupro [24]	x	
Intensive [87, 145]	x	
ManSART [4, 43]	x	
MAP [117]	x	
PBS/SBS [8, 31, 49, 113]	x	
PuLSE/SAVE [61, 103]	x	
QADSAR [118, 119]	x	
Revealer [100, 101]	x	x
RMTool [92, 93]	x	
SARTool [30, 64]	x	
SAVE [89, 94]	x	
Softwarenaut [77]	x	x
Symphony,Nimeta [106, 135]		
URCA		
W4 [44]	x	
X-Ray [86]	x	

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

1. Data Extraction

	src	text	dyn
Alborz [110]	x		x
ArchView [99]	x		x
ArchVis [45]	x	x	x
ARES [26]	x		
ARM [40]	x		
ARMIN [58]	x		
ART [32]	x		
Bauhaus [13, 25, 62]	x		x
Bunch [79, 90]	x		
Cacophony [28]			
Dali [56, 57]	x		
DiscoTect [146]	x		x
Focus [18, 84]	x		
Gupro [24]	x		
Intensive [87, 145]	x		
ManSART [4, 43]	x		
MAP [117]	x		
PBS/SBS [8, 31, 49, 113]	x		
PuLSE/SAVE [61, 103]	x		
QADSAR [118, 119]	x		
Revealer [100, 101]	x	x	
RMTool [92, 93]	x		
SARTool [30, 64]	x		
SAVE [89, 94]	x		
Softwarenaut [77]	x	x	
Symphony,Nimeta [106, 135]			x
URCA			x
W4 [44]	x		
X-Ray [86]	x		

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

1. Data Extraction

	src	text	dyn	phys
Alborz [110]	x		x	
ArchView [99]	x		x	
ArchVis [45]	x	x	x	x
ARES [26]	x			
ARM [40]	x			
ARMIN [58]	x			
ART [32]	x			
Bauhaus [13, 25, 62]	x		x	
Bunch [79, 90]	x			
Cacophony [28]				
Dali [56, 57]	x			
DiscoTect [146]	x		x	
Focus [18, 84]	x			
Gupro [24]	x			
Intensive [87, 145]	x			
ManSART [4, 43]	x			x
MAP [117]	x			
PBS/SBS [8, 31, 49, 113]	x			x
PuLSE/SAVE [61, 103]	x			
QADSAR [118, 119]	x			
Revealer [100, 101]	x	x		
RMTool [92, 93]	x			
SARTool [30, 64]	x			
SAVE [89, 94]	x			
Softwarenaut [77]	x	x		x
Symphony,Nimeta [106, 135]			x	
URCA			x	
W4 [44]	x			
X-Ray [86]	x			

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

1. Data Extraction

	src	text	dyn	phys	hist
Alborz [110]	x		x		
ArchView [99]	x		x		x
ArchVis [45]	x	x	x	x	
ARES [26]	x				
ARM [40]	x				
ARMIN [58]	x				
ART [32]	x				
Bauhaus [13, 25, 62]	x		x		
Bunch [79, 90]	x				
Cacophony [28]					
Dali [56, 57]	x				
DiscoTect [146]	x		x		
Focus [18, 84]	x				
Gupro [24]	x				
Intensive [87, 145]	x				
ManSART [4, 43]	x			x	
MAP [117]	x				
PBS/SBS [8, 31, 49, 113]	x			x	
PuLSE/SAVE [61, 103]	x				
QADSAR [118, 119]	x				
Revealer [100, 101]	x	x			
RMTool [92, 93]	x				
SARTool [30, 64]	x				
SAVE [89, 94]	x				
Softwarenaut [77]	x	x		x	x
Symphony,Nimeta [106, 135]			x		
URCA			x		
W4 [44]	x				x
X-Ray [86]	x				x

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

1. Data Extraction

	src	text	dyn	phys	hist	exp
Alborz [110]	x		x			x
ArchView [99]	x		x		x	x
ArchVis [45]	x	x	x	x		
ARES [26]	x					x
ARM [40]	x					x
ARMIN [58]	x					x
ART [32]	x					x
Bauhaus [13, 25, 62]	x		x			x
Bunch [79, 90]	x					x
Cacophony [28]						x
Dali [56, 57]	x					x
DiscoTect [146]	x		x			x
Focus [18, 84]	x					x
Gupro [24]	x					x
Intensive [87, 145]	x					x
ManSART [4, 43]	x			x		x
MAP [117]	x					x
PBS/SBS [8, 31, 49, 113]	x			x		x
PuLSE/SAVE [61, 103]	x					x
QADSAR [118, 119]	x					x
Revealer [100, 101]	x	x				x
RMTool [92, 93]	x					x
SARTool [30, 64]	x					x
SAVE [89, 94]	x					x
Softwarenaut [77]	x	x		x	x	x
Symphony,Nimeta [106, 135]			x			x
URCA			x			x
W4 [44]	x				x	x
X-Ray [86]	x				x	x

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

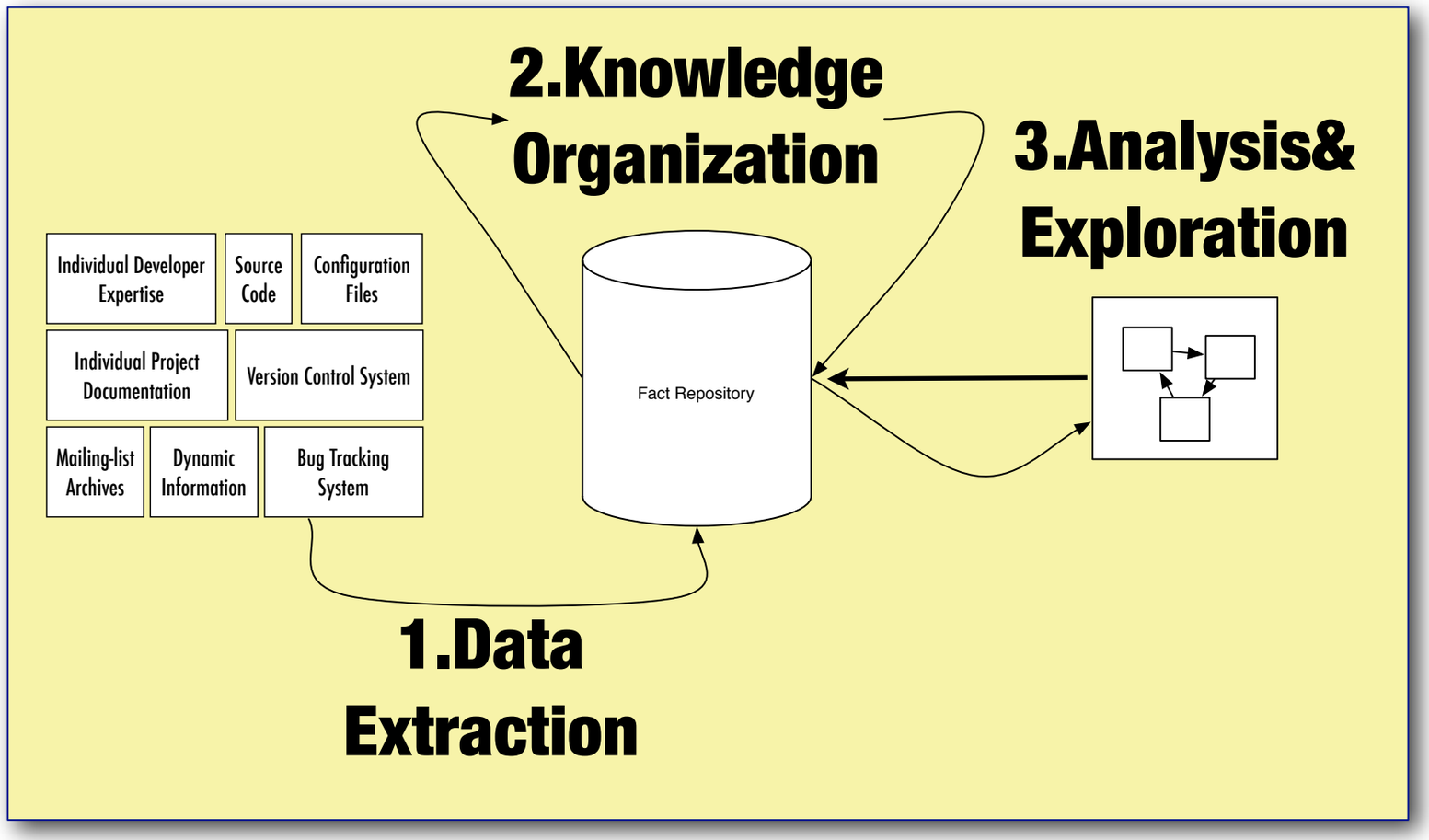
1. Data Extraction

	src	text	dyn	phys	hist	exp	style
Alborz [110]	x		x			x	
ArchView [99]	x		x		x	x	
ArchVis [45]	x	x	x	x			x
ARES [26]	x					x	
ARM [40]	x					x	
ARMIN [58]	x					x	
ART [32]	x					x	x
Bauhaus [13, 25, 62]	x		x			x	
Bunch [79, 90]	x					x	
Cacophony [28]						x	
Dali [56, 57]	x					x	
DiscoTect [146]	x		x			x	x
Focus [18, 84]	x					x	x
Gupro [24]	x					x	
Intensive [87, 145]	x					x	
ManSART [4, 43]	x			x		x	x
MAP [117]	x					x	x
PBS/SBS [8, 31, 49, 113]	x			x		x	
PuLSE/SAVE [61, 103]	x					x	
QADSAR [118, 119]	x					x	
Revealer [100, 101]	x	x				x	
RMTool [92, 93]	x					x	
SARTool [30, 64]	x					x	
SAVE [89, 94]	x					x	
Softwarenaut [77]	x	x		x	x	x	
Symphony,Nimeta [106, 135]			x			x	
URCA			x			x	
W4 [44]	x				x	x	
X-Ray [86]	x				x	x	x

src - source code
text - textual information
dyn - dynamic analysis
phys - physical organiation
exp - human expertise
style - architectural style

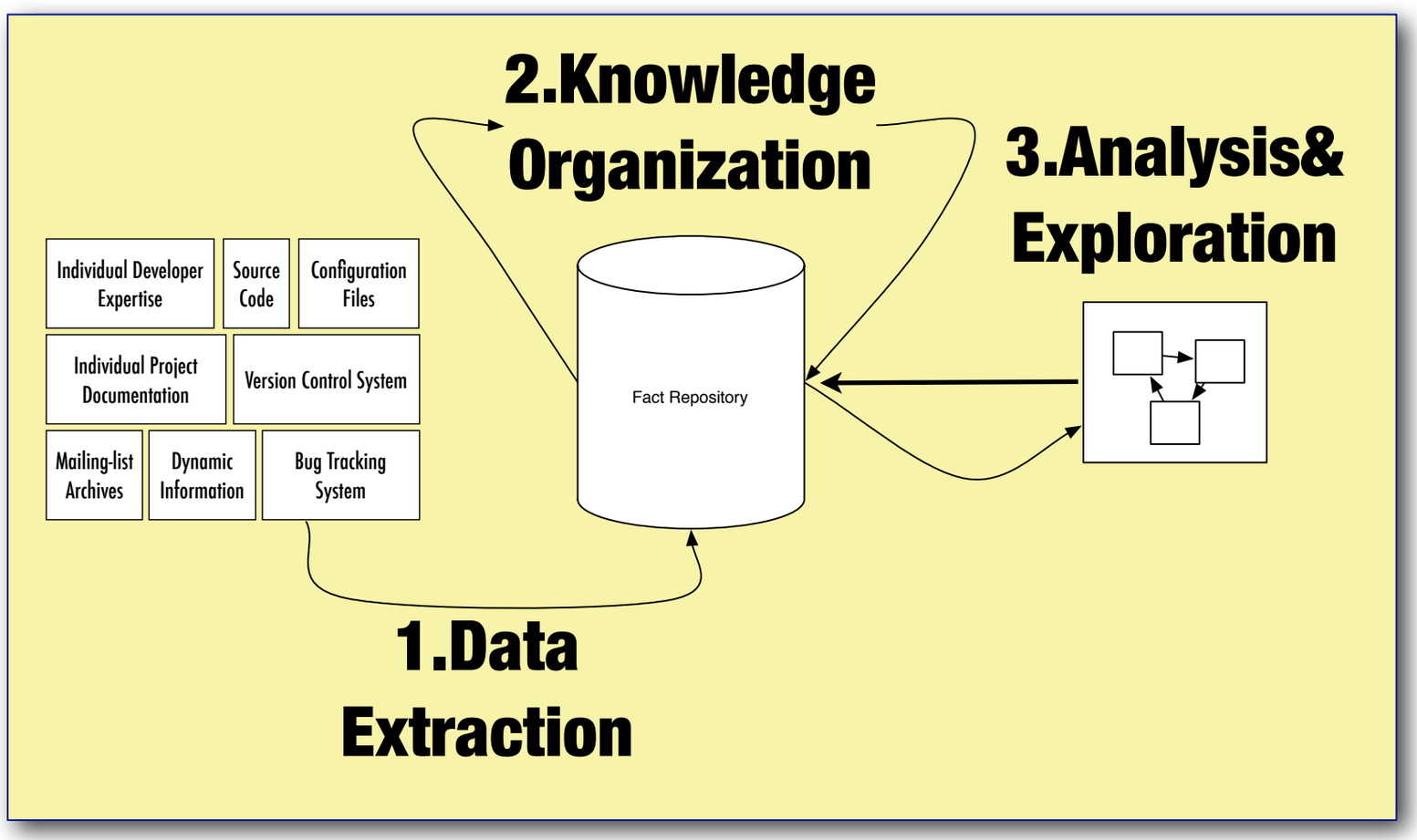
TheArchitecture of Architecture Recovery Tools

“extract-abstract-present” [Tilley]
“extract-abstract-explore” [Lungu]

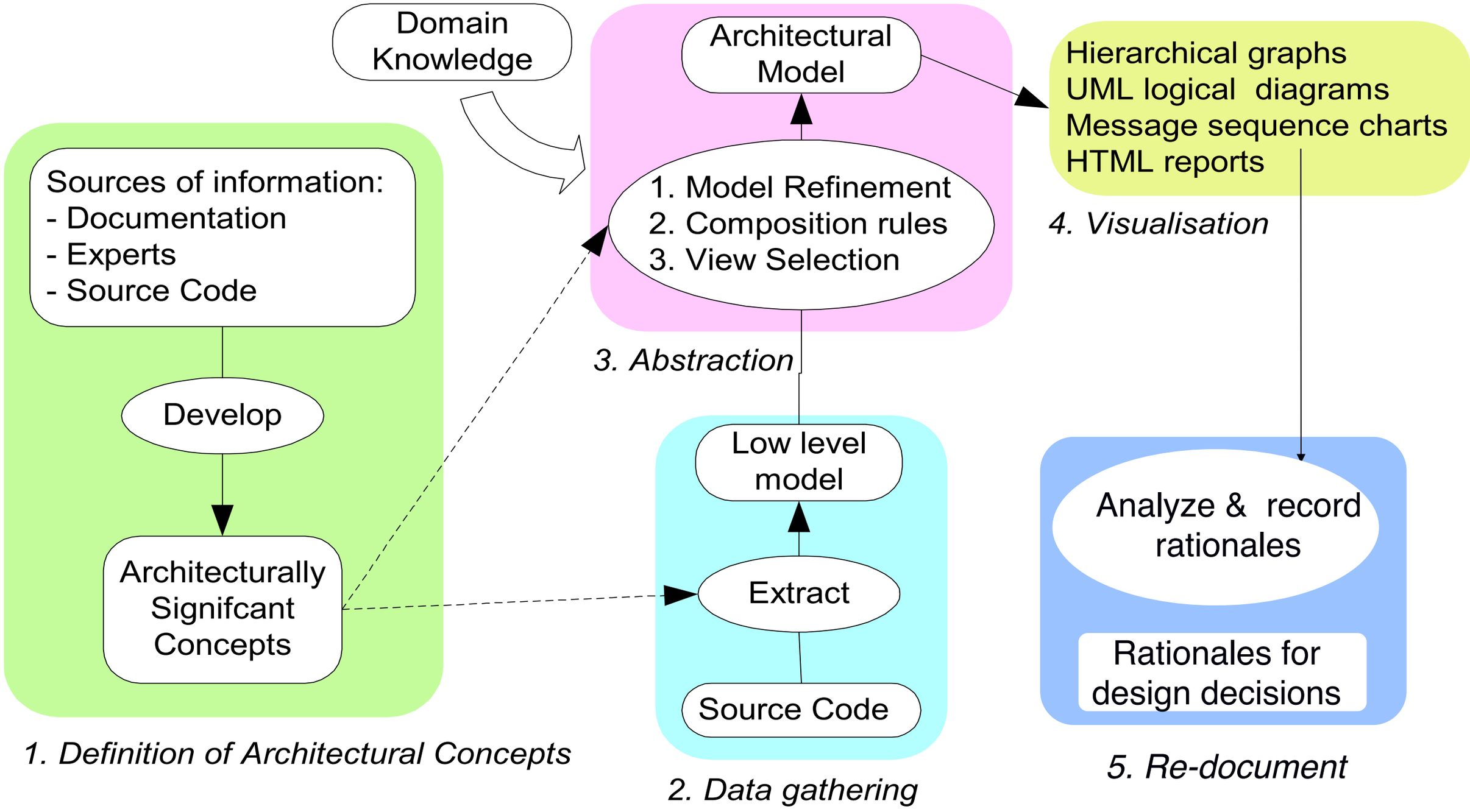


TheArchitecture of Architecture Recovery Tools

“extract-abstract-present” [Tilley]
“extract-abstract-explore” [Lungu] ← For Good Grade!!!



Architecture Reconstruction



Roadmap

- Introduction to SAR
- Top-down SAR
- **Bottom-up SAR**
  - The Architecture of Architecture Recovery Tools
  - **Data Extraction**
  - Knowledge Organization
  - Exploration
  - Examples
- Tool Demo

Roadmap

- Introduction to SAR
- Top-down SAR
- **Bottom-up SAR**
  - The Architecture of Architecture Recovery Tools
  - Data Extraction
  - **Knowledge Organization**
  - Exploration
  - Examples
- Tool Demo

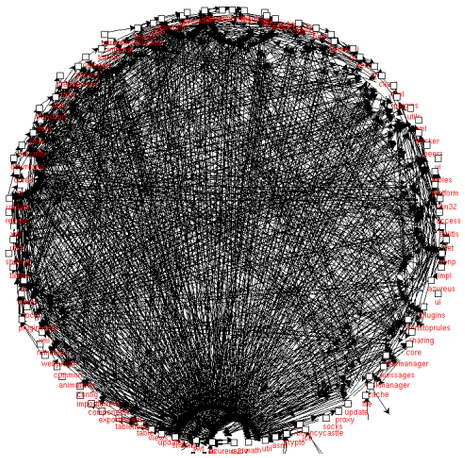
Knowledge Organization

- Different techniques
  - Automated Aggregation
  - Clustering
  - Concept Analysis

a. Aggregation

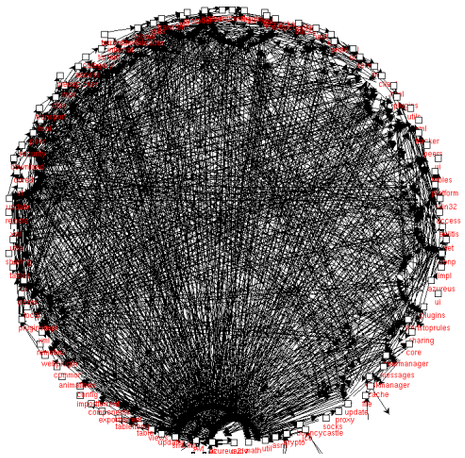
a. Aggregation

Package
Dependencies



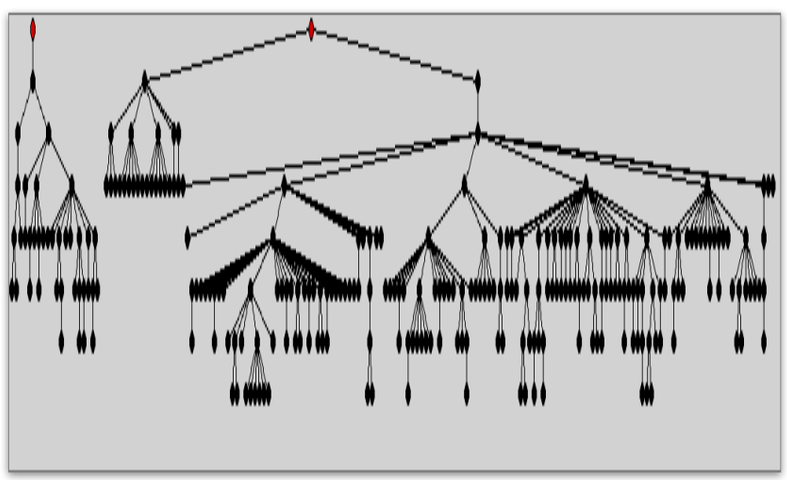
a. Aggregation

Package
Dependencies



+

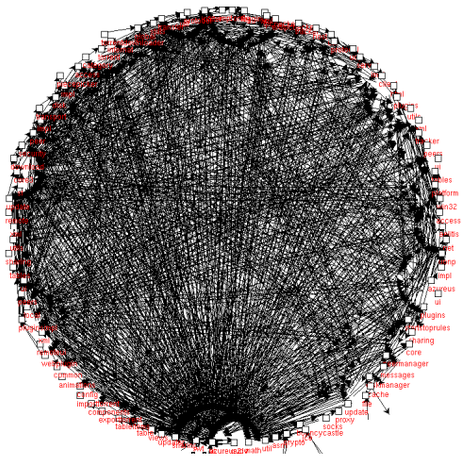
In-place
Hierarchical
Decomposition



=

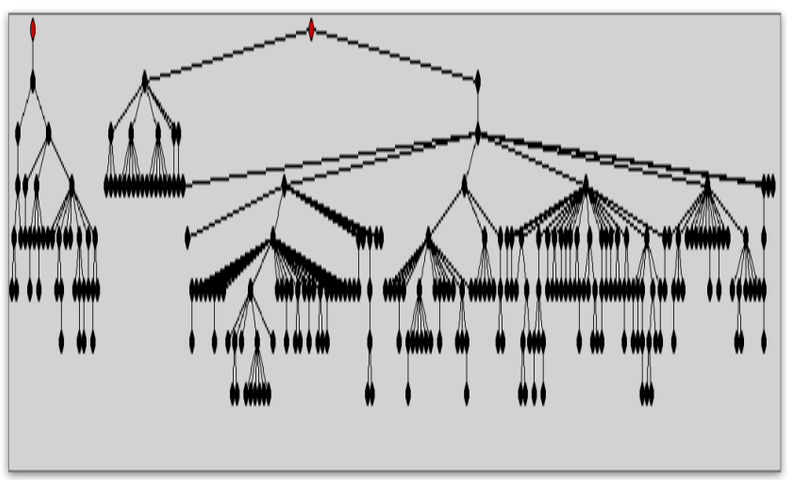
a. Aggregation

Package Dependencies



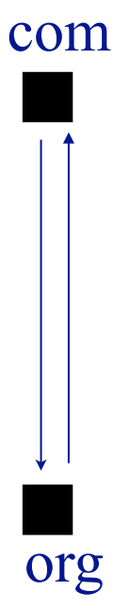
+

In-place Hierarchical Decomposition



=

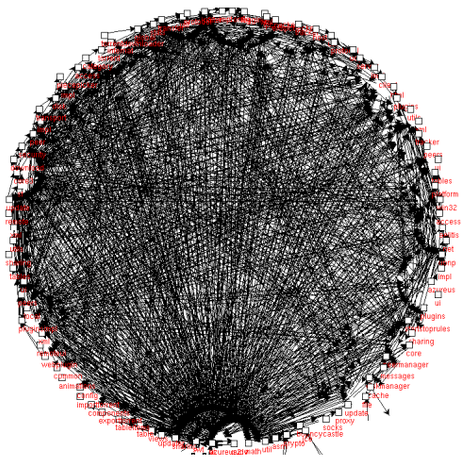
Highest-Level Dependency View



a. Aggregation

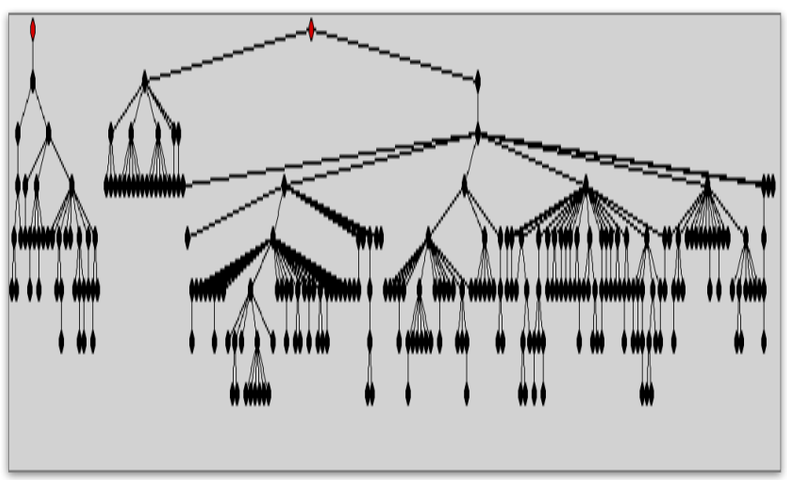
Data Structure: The Hierarchical Graph

Package Dependencies



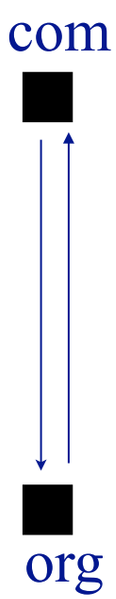
\+

In-place Hierarchical Decomposition



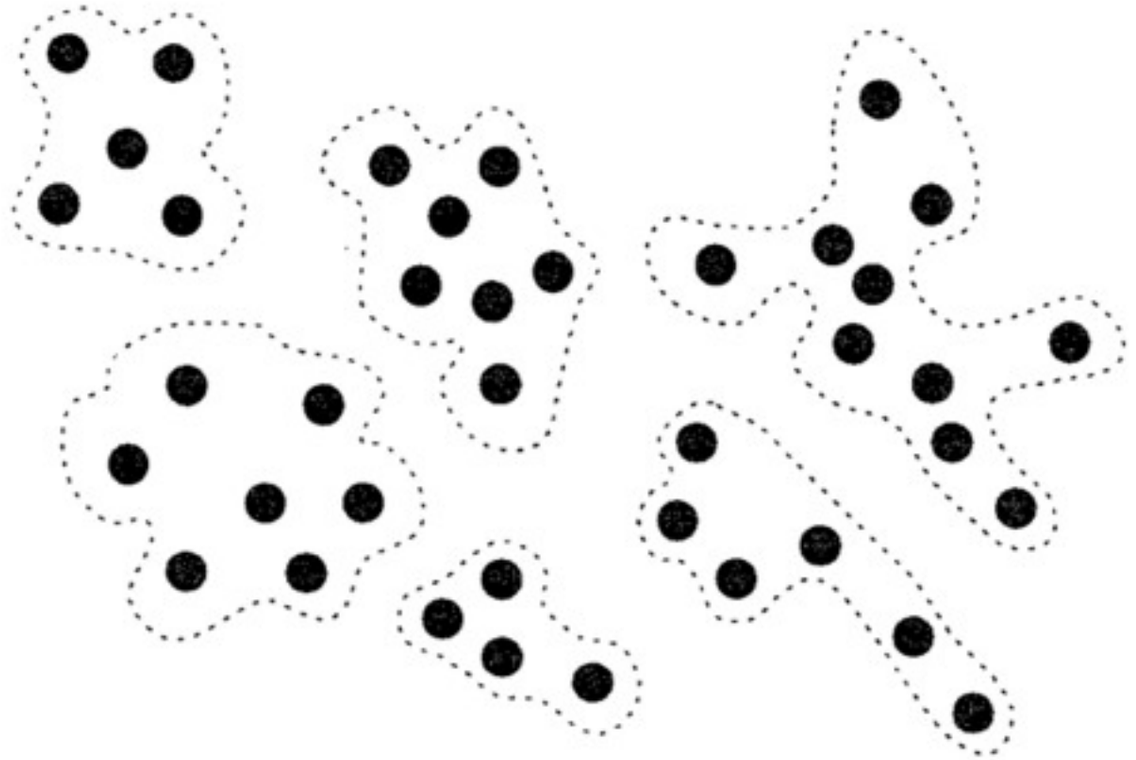
=

Highest-Level Dependency View



b. Clustering

- Ingredients
  - Entities
  - Similarity
  - Algorithms

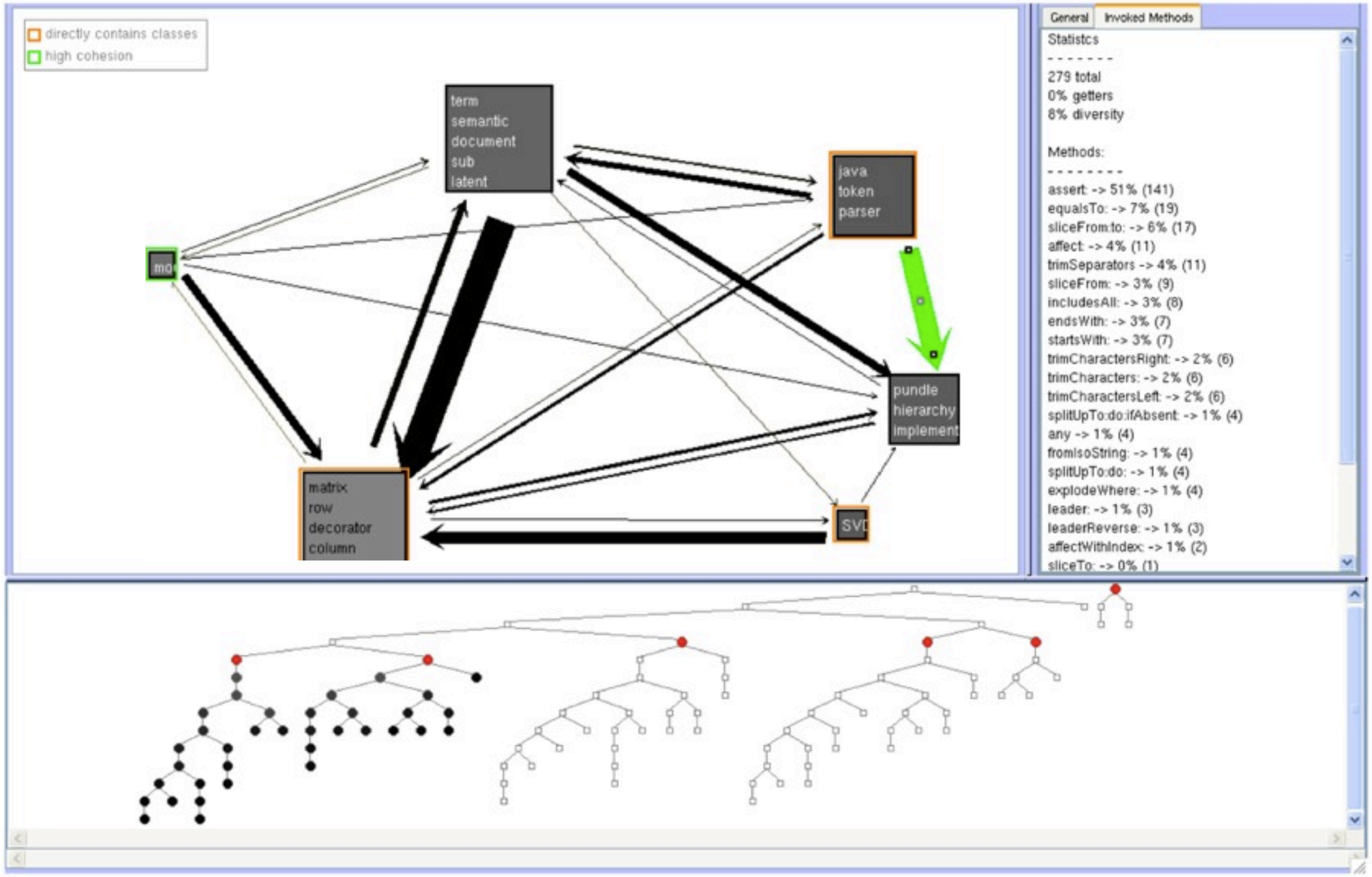


- Tools: Hapax, Bunch

Ingredient 1: Similarity

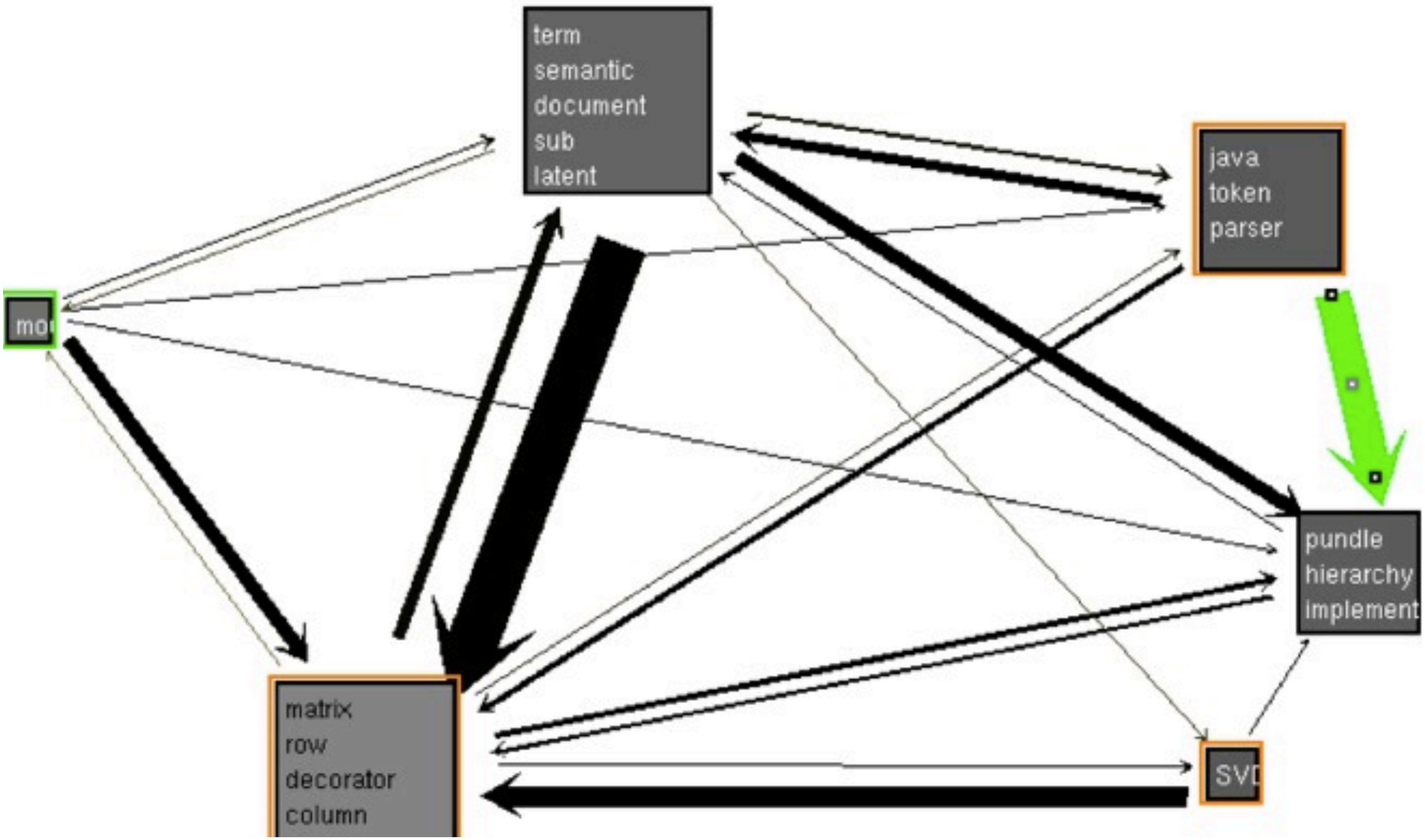
- Metric
  - Based on relationships between the elements or common properties
  - Updating similarities
  - Special types for software
    - *communication*
    - *natural language similarity*

Natural language similarity



[Lungu et al.'05]

Natural language similarity



[Lungu et al.'05]

Ingredient 2: Algorithms

Flat

Ingredient 2: Algorithms

Flat

place each entity in a group by itself
repeat
  identify the *two most similar groups*
  combine them
until the existing groups are satisfactory

Ingredient 2: Algorithms

Flat

place each entity in a group by itself
repeat
  identify the *two most similar groups*
  combine them
until the existing groups are satisfactory

Hierarchical

place each entity in a group by itself
repeat
  identify *the most similar groups* Si and Sj
  combine Si and Sj add a subtree with children Si and Sj to the clustering tree
until the existing groups are satisfactory or only one group is left

A Dendrogram

How do you select the cutoff factor?

Clustering based recommenders

- Arch: Developed at Siemens Research
- MARS: Master project developed at USI

Clustering based recommenders

- Arch: Developed at Siemens Research
- MARS: Master project developed at USI

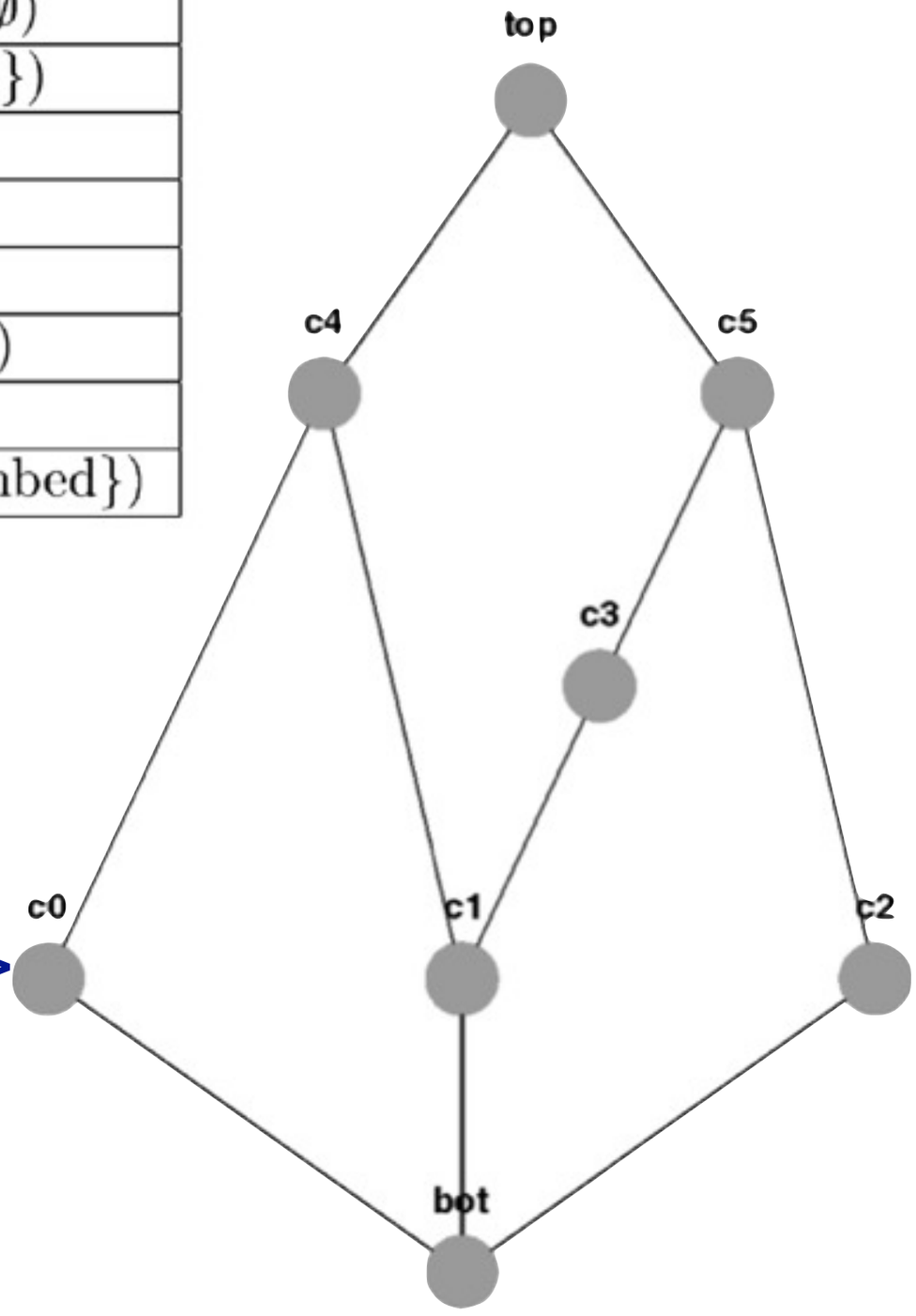
c. Formal Concept Analysis

- Derives an ontology from a set of objects and their properties

Concept Analysis (Biology Example)

top	({cats, gibbons, dogs, dolphins, humans, whales}, ∅)
c_5	({gibbons, dolphins, humans, whales}, {intelligent})
c_4	({cats, gibbons, dogs}, {hair-covered})
c_3	({gibbons, humans}, {intelligent, thumbed})
c_2	({dolphins, whales}, {intelligent, marine})
c_1	({gibbons}, {hair-covered, intelligent, thumbed})
c_0	({cats, dogs}, {hair-covered, four-legged})
bot	(∅, {four-legged, hair-covered, intelligent, marine, thumbed})

A Concept Lattice ->



Concept Analysis (Biology Example)

		attributes				
		four-legged	hair-covered	intelligent	marine	thumbed
objects	cats	√	√			
	dogs	√	√			
	dolphins			√	√	
	gibbons		√	√		√
	humans			√		√
	whales			√	√	

Concept Analysis (Biology Example)

		attributes				
		four-legged	hair-covered	intelligent	marine	thumbed
objects	cats	√	√			
	dogs	√	√			
	dolphins			√	√	
	gibbons		√	√		√
	humans			√		√
	whales			√	√	

top	({cats, gibbons, dogs, dolphins, humans, whales}, ∅)
c_5	({gibbons, dolphins, humans, whales}, {intelligent})
c_4	({cats, gibbons, dogs}, {hair-covered})
c_3	({gibbons, humans}, {intelligent, thumbed})
c_2	({dolphins, whales}, {intelligent, marine})
c_1	({gibbons}, {hair-covered, intelligent, thumbed})
c_0	({cats, dogs}, {hair-covered, four-legged})
bot	(∅, {four-legged, hair-covered, intelligent, marine, thumbed})

Concept Analysis: A Problem

```
#define QUEUE_SIZE 10
struct stack { int *base, *sp, size; };
struct queue { struct stack *front, *back; };

struct stack* initStack(int sz) {
 struct stack* s =
   (struct stack*) malloc(sizeof(struct stack));
 s->sp = (int*)malloc(sz * (sizeof(int)));
 s->base = s->sp;
 s->size = sz;
 return s; }

struct queue* initQ() {
 struct queue* q =
   (struct queue*) malloc(sizeof(struct queue));
 q->front = initStack(QUEUE_SIZE);
 q->back = initStack(QUEUE_SIZE);
 return q; }

int isEmptyS(struct stack* s) {
 return (s->sp == s->base); }

int isEmptyQ(struct queue* q) {
 return (isEmptyS(q->front)
          && isEmptyS(q->back)); }
```

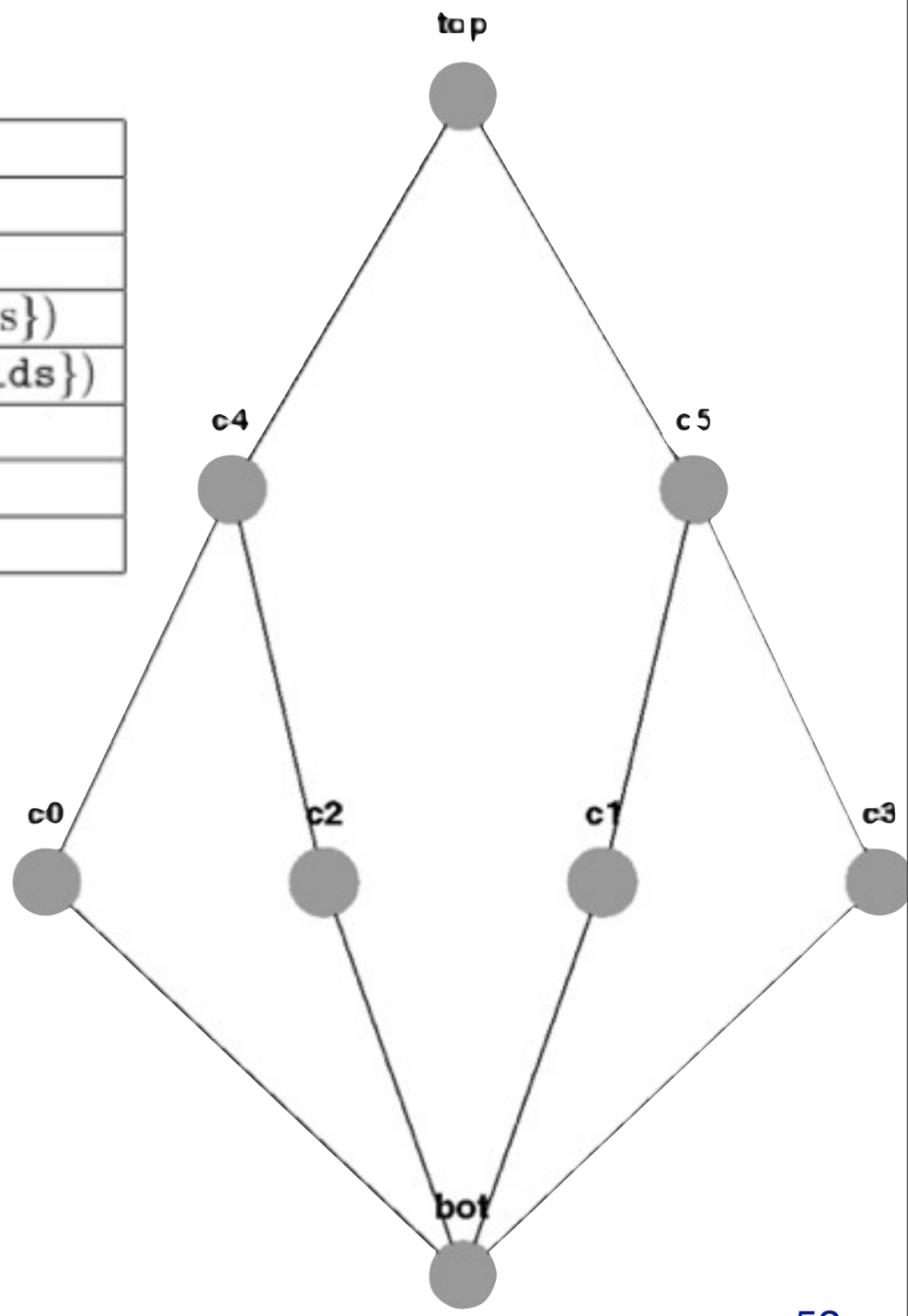
```
void push(struct stack* s, int i) {
 /* no overflow check */
 *(s->sp) = i; s->sp++; }

void enq(struct queue* q, int i) {
 push(q->front, i); }

int pop(struct stack* s) {
 if (isEmptyS(s)) return -1;
 s->sp--;
 return (*(s->sp)); }

int deq(struct queue* q) {
 if (isEmptyQ(q)) return -1;
 if (isEmptyS(q->back))
  while(!isEmptyS(q->front))
   push(q->back, pop(q->front));
 return pop(q->back); }
```

The Context

	returns `stack`	*returns* `queue`	*has* `stack` *arg.*	*has* `queue` *arg.*	*uses* `stack` *fields*	*uses* `queue` *fields*
`initStack`	√				√	
`initQ`		√				√
`isEmptyS`			√		√	
`isEmptyQ`				√		√
`push`			√		√	
`enq`				√		√
`pop`			√		√	
`deq`				√		√

Concepts

top	(*all objects*, $\emptyset$)
c_5	({`initQ`, `isEmptyQ`, `enq`, `deq`}, {uses `queue` fields})
c_4	({`initStack`, `isEmptyS`, `push`, `pop`}, {uses `stack` fields})
c_3	({`isEmptyQ`, `enq`, `deq`}, {has `queue` argument, uses `queue` fields})
c_2	({`isEmptyS`, `push`, `pop`}, {has `stack` argument, uses `stack fields`})
c_1	({`initQ`}, {returns `queue`})
c_0	({`initStack`}, {returns `stack`})
bot	($\emptyset$, *all attributes*)



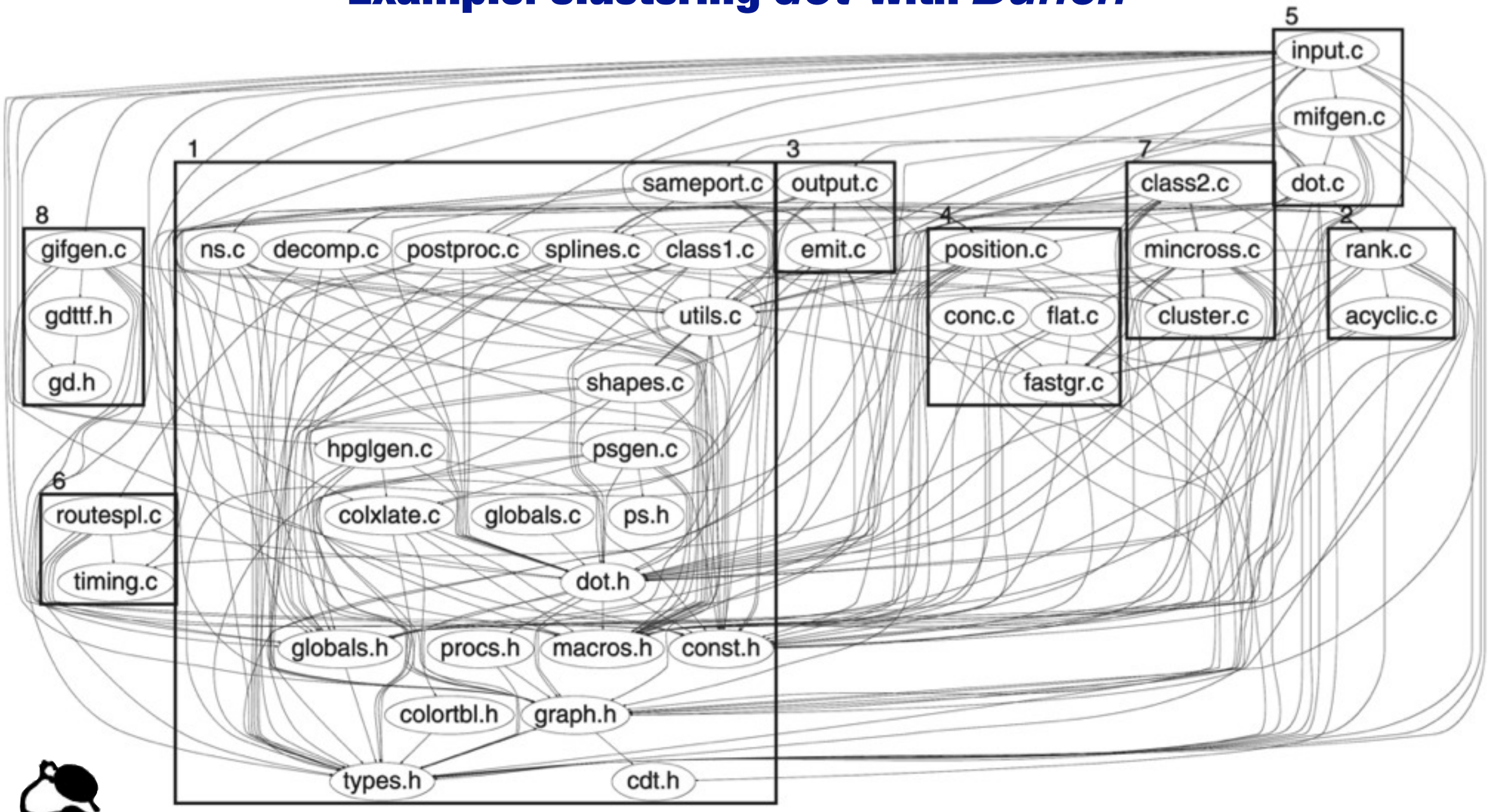
Roadmap

- Introduction to SAR
- Top-down SAR
- **Bottom-up SAR**
  - The Architecture of Architecture Recovery Tools
  - Data Extraction
  - Knowledge Organization
  - **Exploration**
  - Examples
- Tool Demo

Exploration

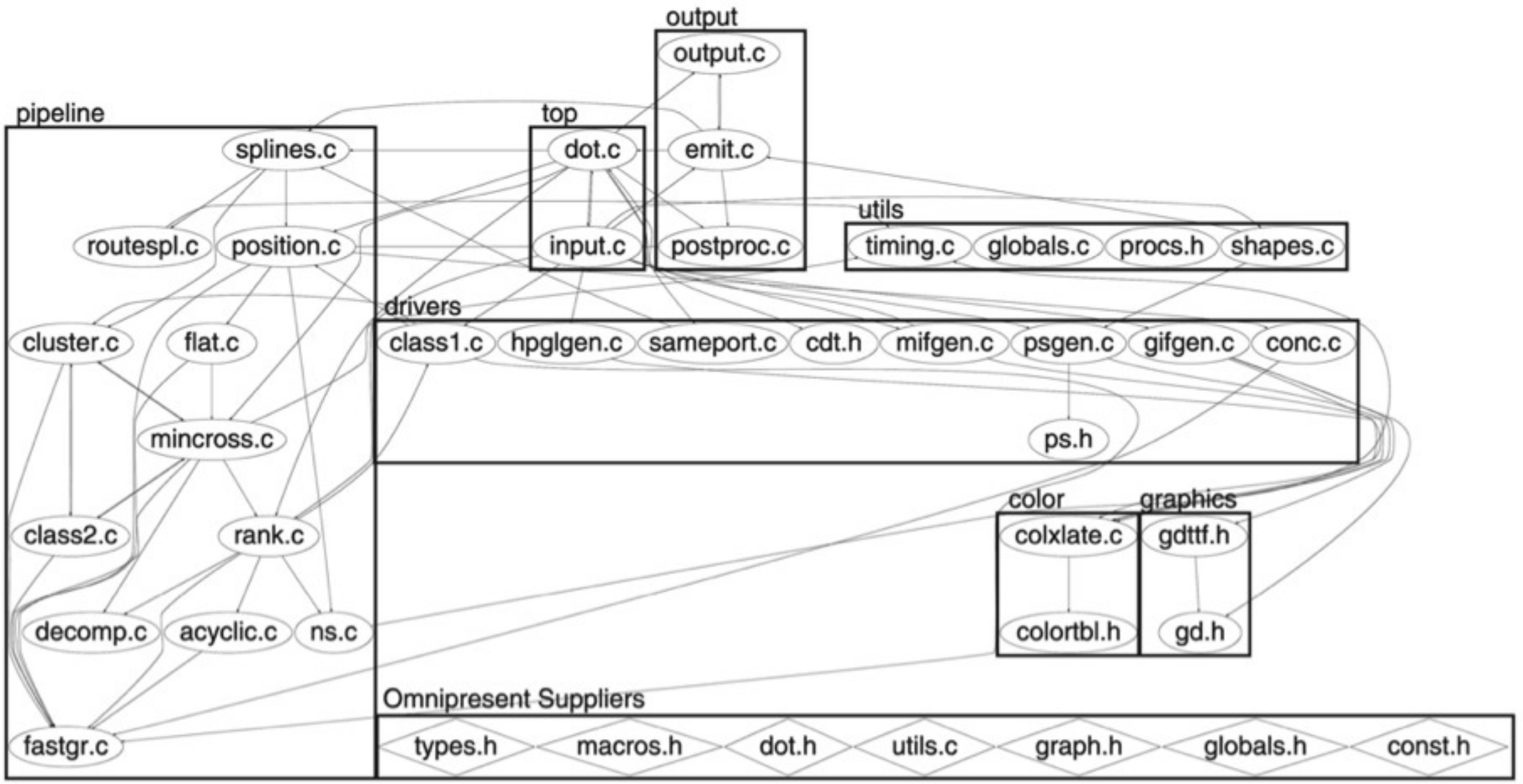
- Interactive Analysis of the Data
- Feeds back into the analysis
- Information Visualization Mantra: “Overview, Zoom, Filter, and Details on demand” (Schneiderman)

Example: Clustering *dot* with *Bunch*



How would you improve this?

Clustering *dot* with *Bunch*



Filtering!

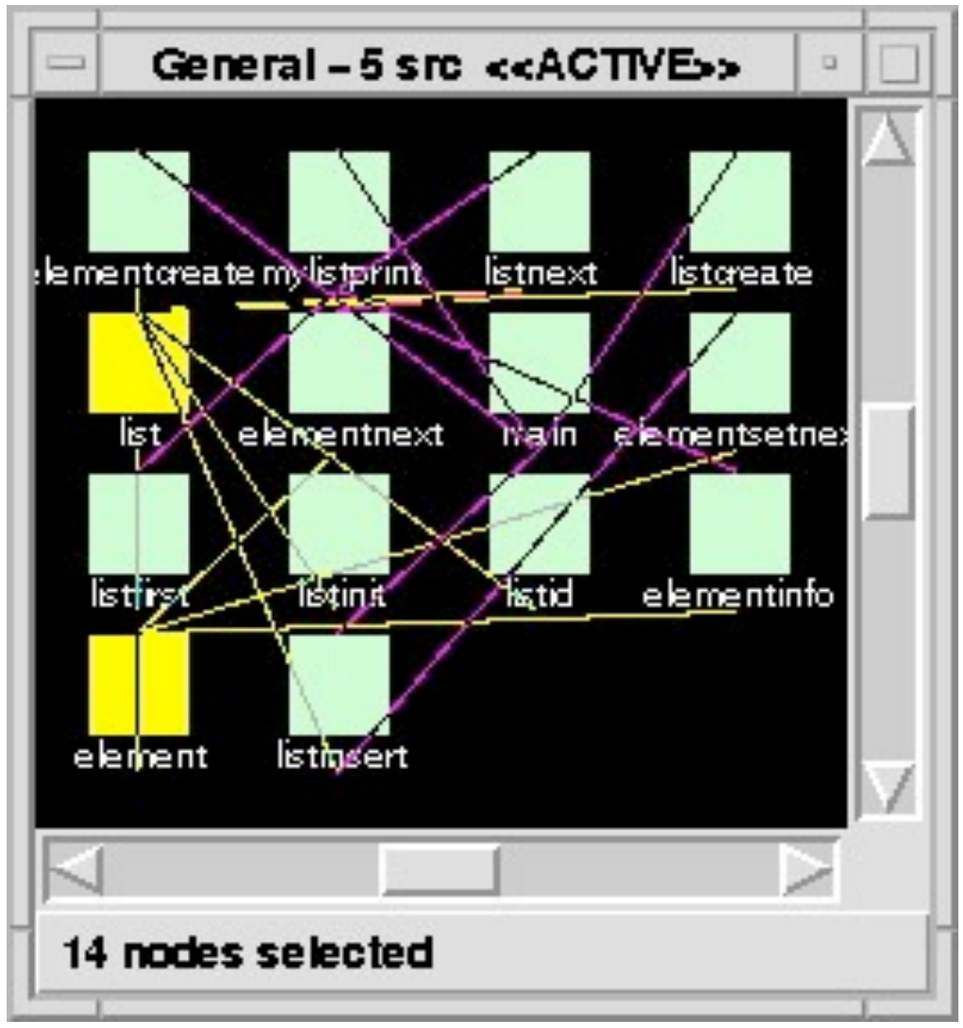
Roadmap

- Introduction to SAR
- Top-down SAR
- **Bottom-up SAR**
  - The Architecture of Architecture Recovery Tools
  - Data Extraction
  - Knowledge Organization
  - Exploration
  - **Examples**
- Tool Demo

Approaches: Rigi

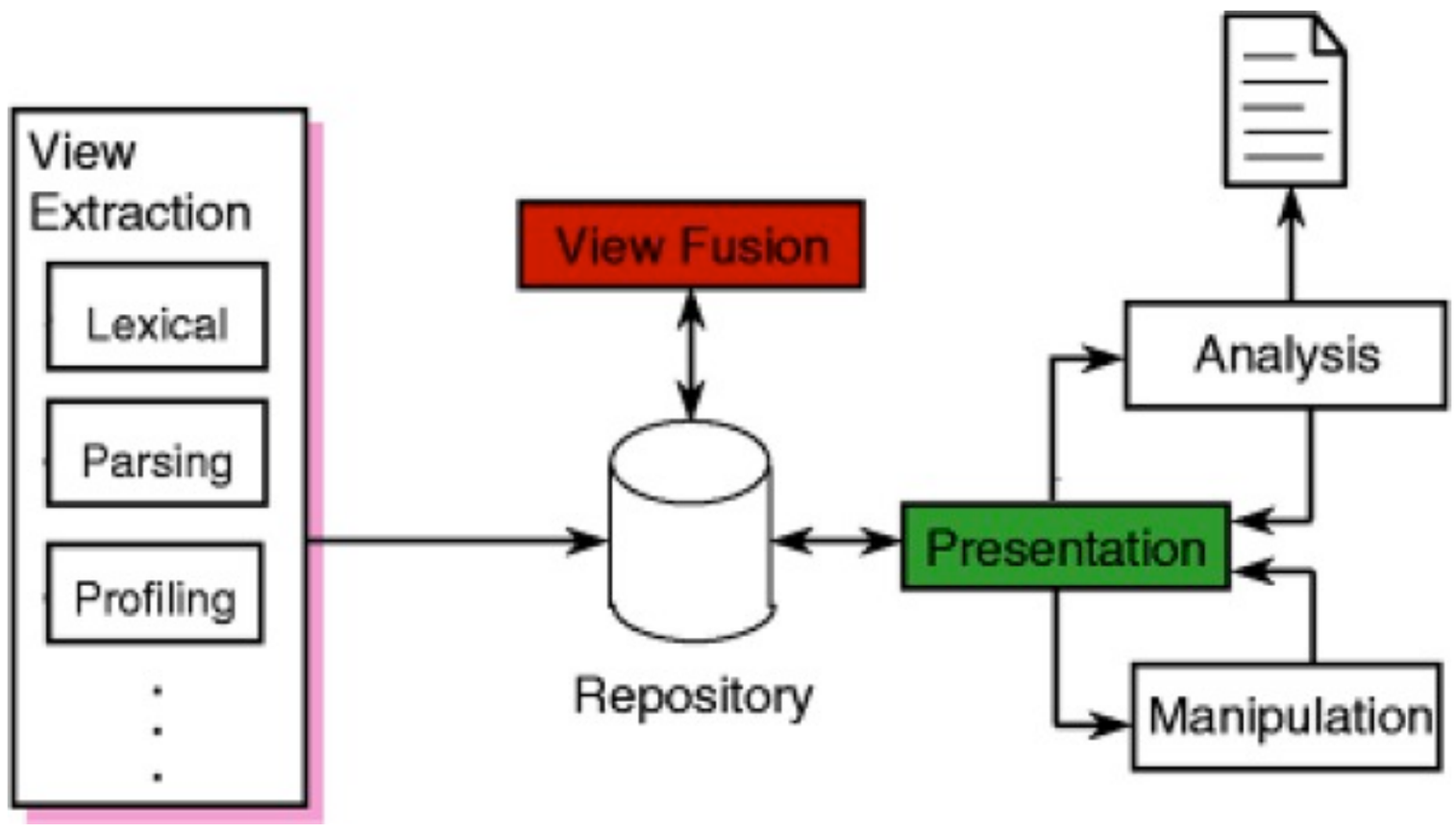
Programmable reverse engineering environment

- —C parser; relational data import
- —Visualization of hierarchical typed graphs
- —Graph manipulation, filtering, layout
- —Tcl-programmable
- —www.rigi.csc.uvic.ca/



Approaches: Dali Workbench

- Workbench built on Rigi, PostgreSQL, perl scripts …
- Three techniques used:
  - Architectural extraction from source artifacts
  - User-defined patterns
  - Visualization

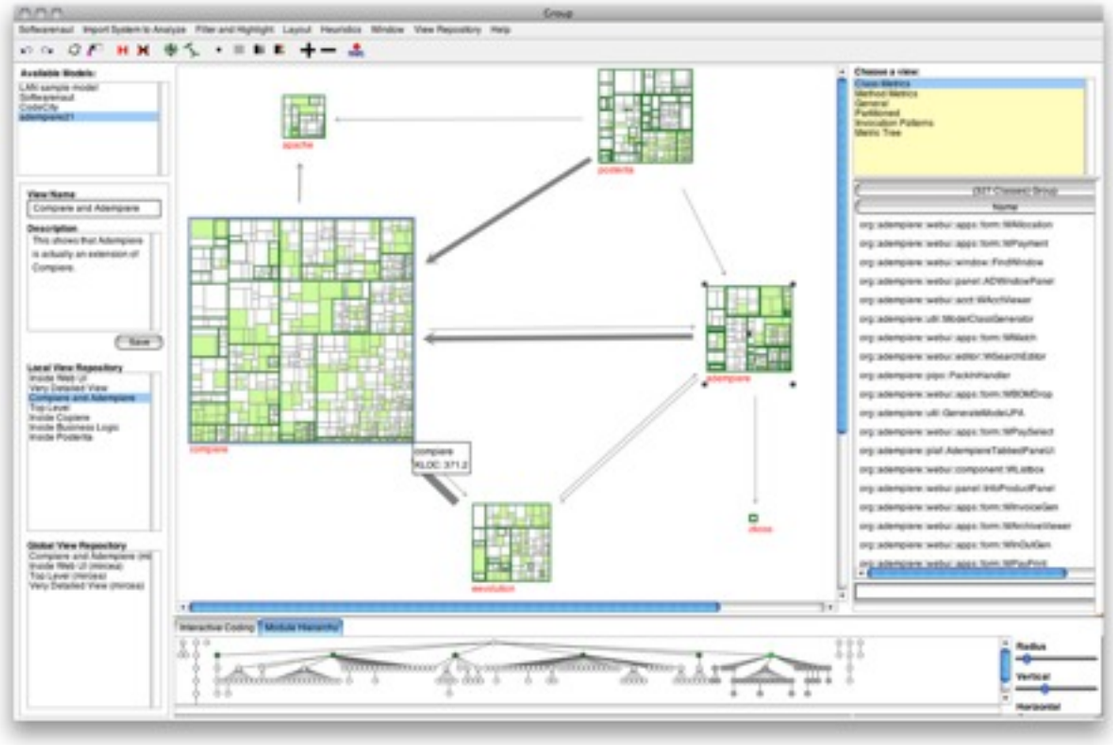


Roadmap

- Introduction to SAR
- Top-down SAR
- Bottom-up SAR
- **Tool Demo**

Tool Demo

- Based on FAMIX
- Extract-Abstract-Present(/Explore)



http://scg.unibe.ch/softwarenaut

Softwarenaut: Discussion

- Hierarchical Graphs
- Collaboration

What you should know!

- Concepts: Architecture, architectural styles, viewpoints
- The extract-abstract-explore meta-architecture of SAR
- SAR can not be fully automated
- The recipe for clustering software artefacts

Can you answer these questions?

- Which are several types of architectural viewpoints?
- How does concept analysis work?
- What types of SAR processes do you know?
- What is formal concept analysis?