BITS Pilani

Software Architecture
Module 5
Architecture Reconstruction

Harvinder S Jabbal

SEZG651/SSZG653 Software Architectures

Contents

- Purpose of architecture reconstruction
- Architecture reconstruction technique
- Reconstruction tools.

Purpose of architecture reconstruction

- To understand the architecture of a system for which no documentation exists
- To migrate a system from old technology to new. Ex. Mainframe to Web
- To identify reusable components in a system, such as logging component, security component, etc.

Phases of architecture reconstruction

- Identify components and their relationship (using a tool)
- Aggregate components into abstract components (specify grouping to tool)
- Analyse the architecture (tool displays architecture)

Identify components & their relationship

- Extract information from
  - Source code
  - Execution traces
  - Build scripts
  - Etc.

- This gets info such as
  - classes
  - file they use
  - ‘Caller – callee’ relationship
  - global data accessed by different objects

Examples of component extraction

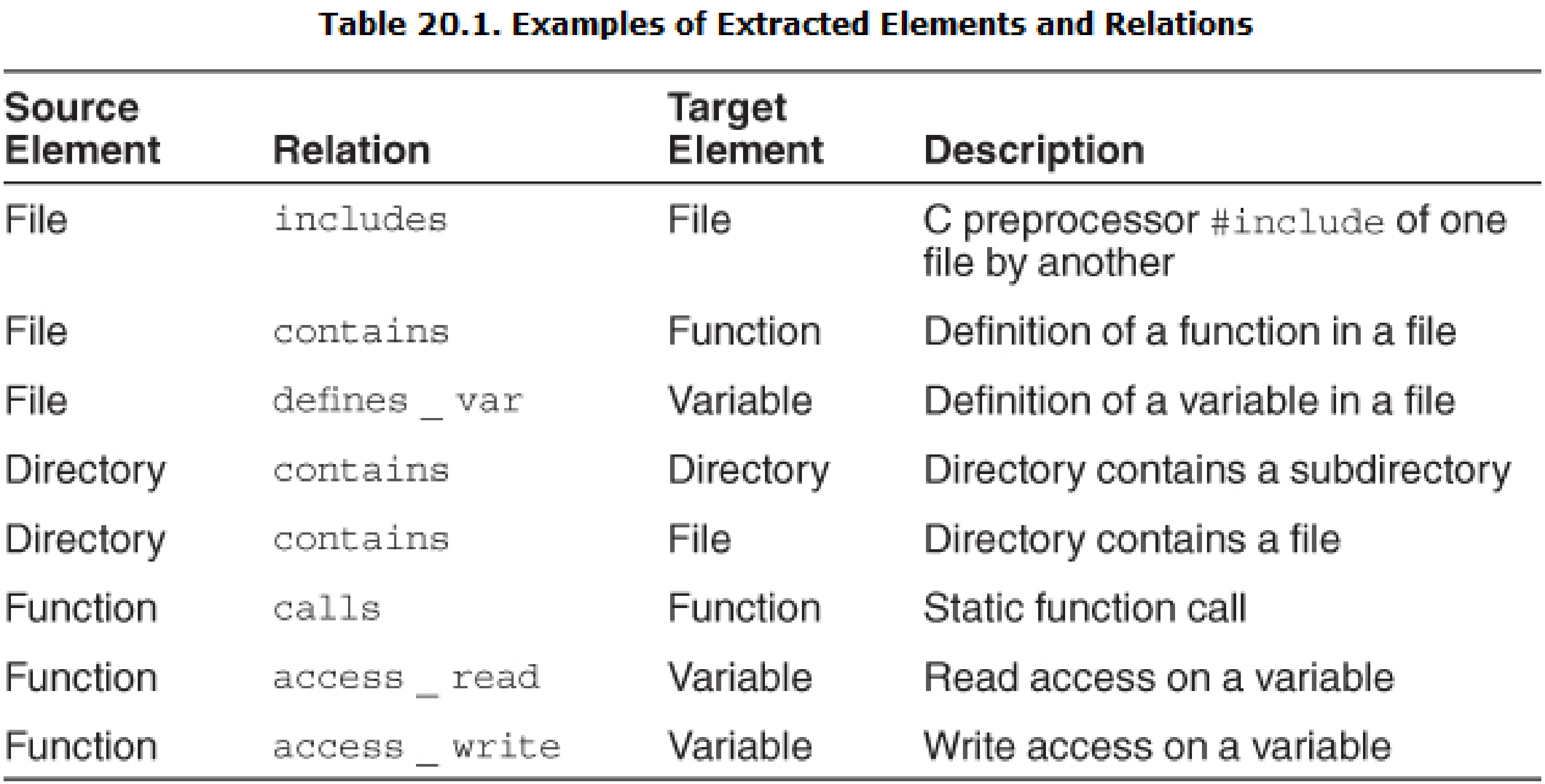


Table 20.1. Examples of Extracted Elements and Relations

Source Element	Relation	Target Element	Description
File	`includes`	File	C preprocessor `#include` of one file by another
File	`contains`	Function	Definition of a function in a file
File	`defines_var`	Variable	Definition of a variable in a file
Directory	`contains`	Directory	Directory contains a subdirectory
Directory	`contains`	File	Directory contains a file
Function	`calls`	Function	Static function call
Function	`access_read`	Variable	Read access on a variable
Function	`access_write`	Variable	Write access on a variable

Ref: Text book: Software Arch in practice by Len Bass and others

Execution trace of method calls

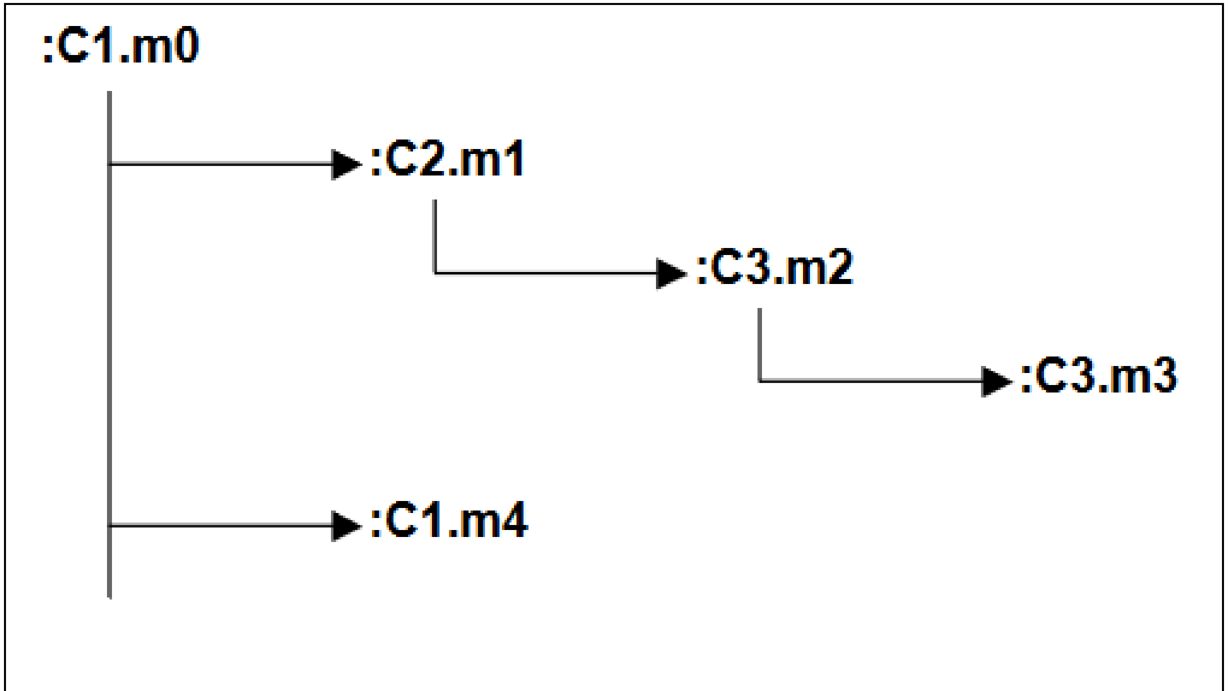
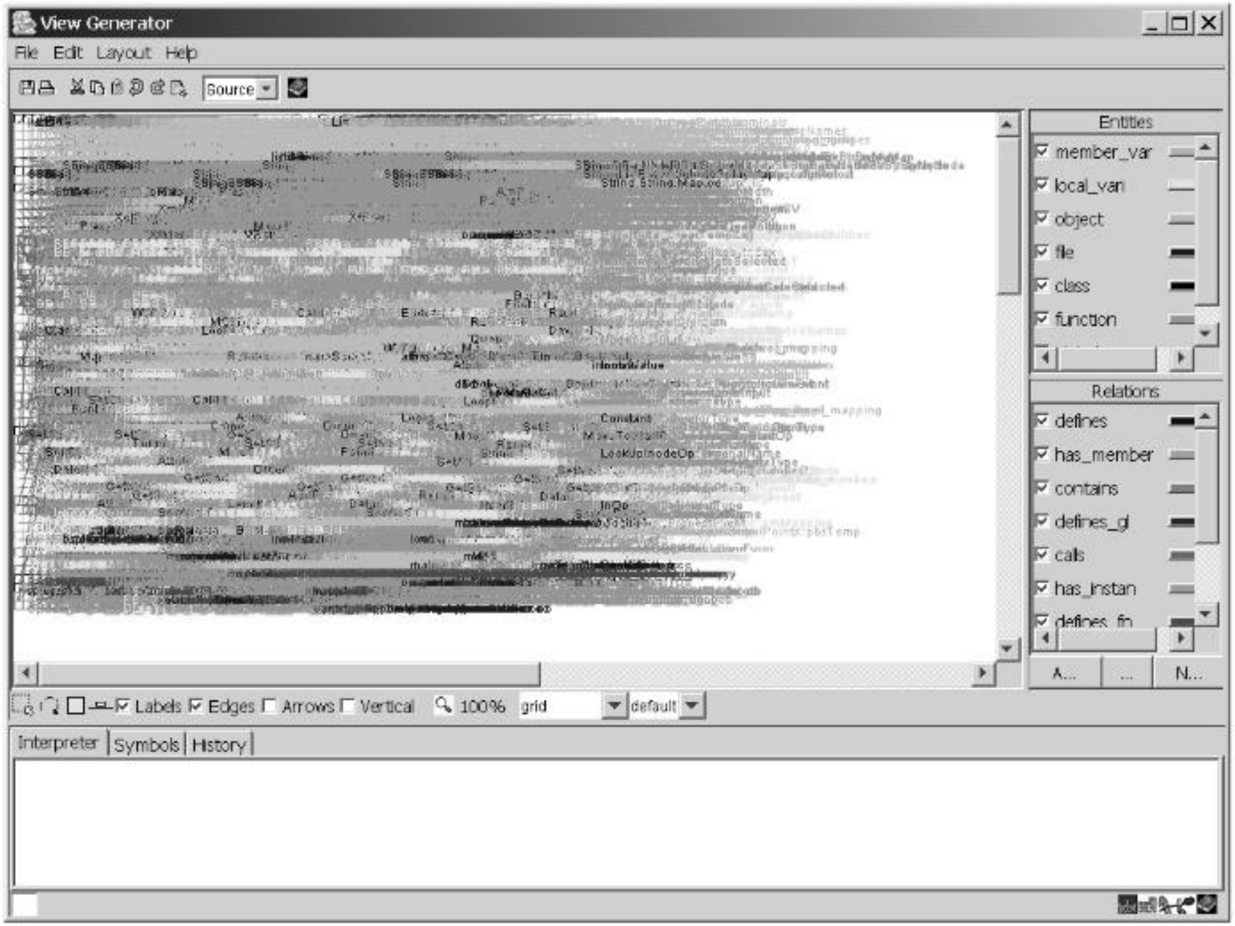


Figure 1.1. Tree representation of a trace of routine calls

Ref: Techniques to Simplify the Analysis of Execution Traces for Program Comprehension by Abdelwahab Hamou-Lhadj

https://pdfs.semanticscholar.org/3db0/dd198058c0a9d489e4b94c2996f117df2d5.pdf

Case study: ‘Vanish’ System



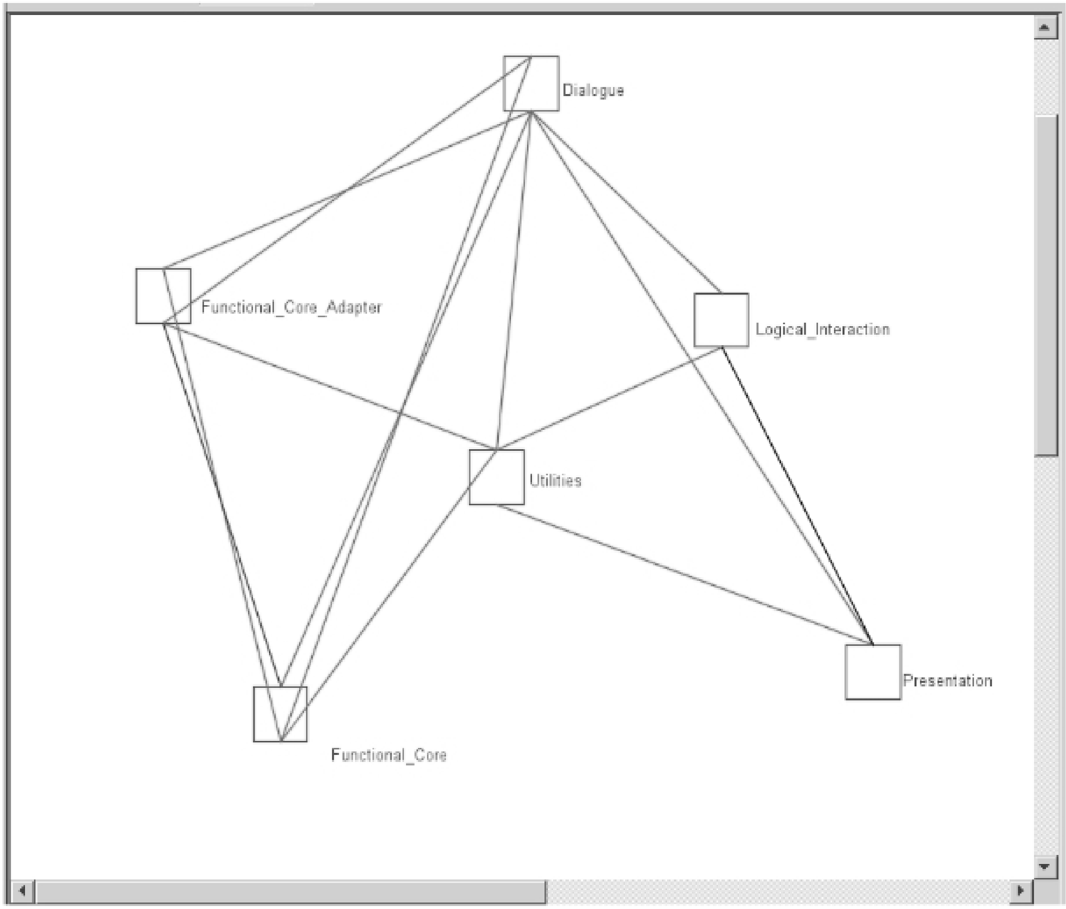
Tool used for reconstruction: **ARMIN**

(ARchitecture Reconstruction and MINing)

White-Noise” View Showing All of the Elements and Relations

Ref: https://resources.sei.cmu.edu/asset_files/TechnicalNote/2003_004_001_14141.pdf

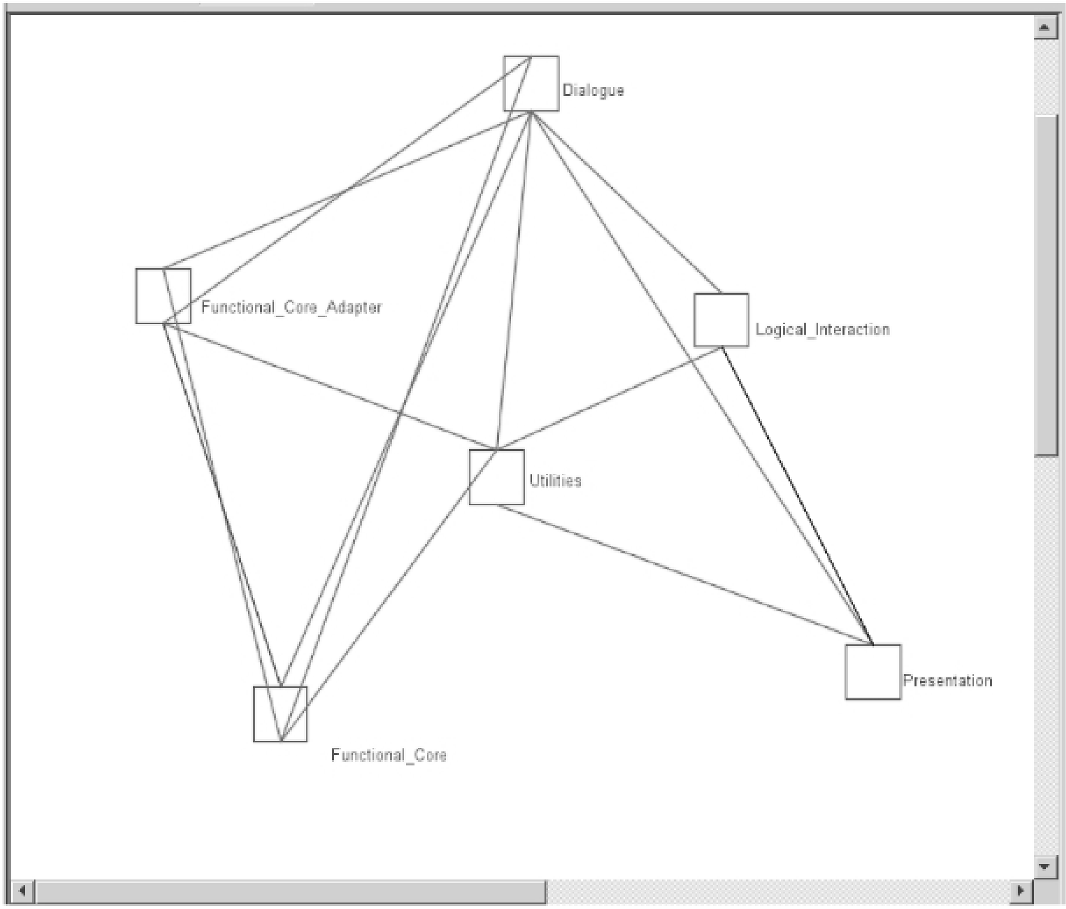
Case study: ‘Vanish’ System



Architecture view after aggregation of components

One can take help of the technical team to get a broad understanding of the system, before starting the aggregation exercise

Case study: ‘Vanish’ System



Analyse architecture

One can notice that the architecture of ‘Vanish’ is not strictly layered

Tools

Dali, ARMIN, Lattix, Sonar J, Structure 101

References:

https://resources.sei.cmu.edu/asset_files/TechnicalReport/2003_005_001_14081.pdf

Experience sharing

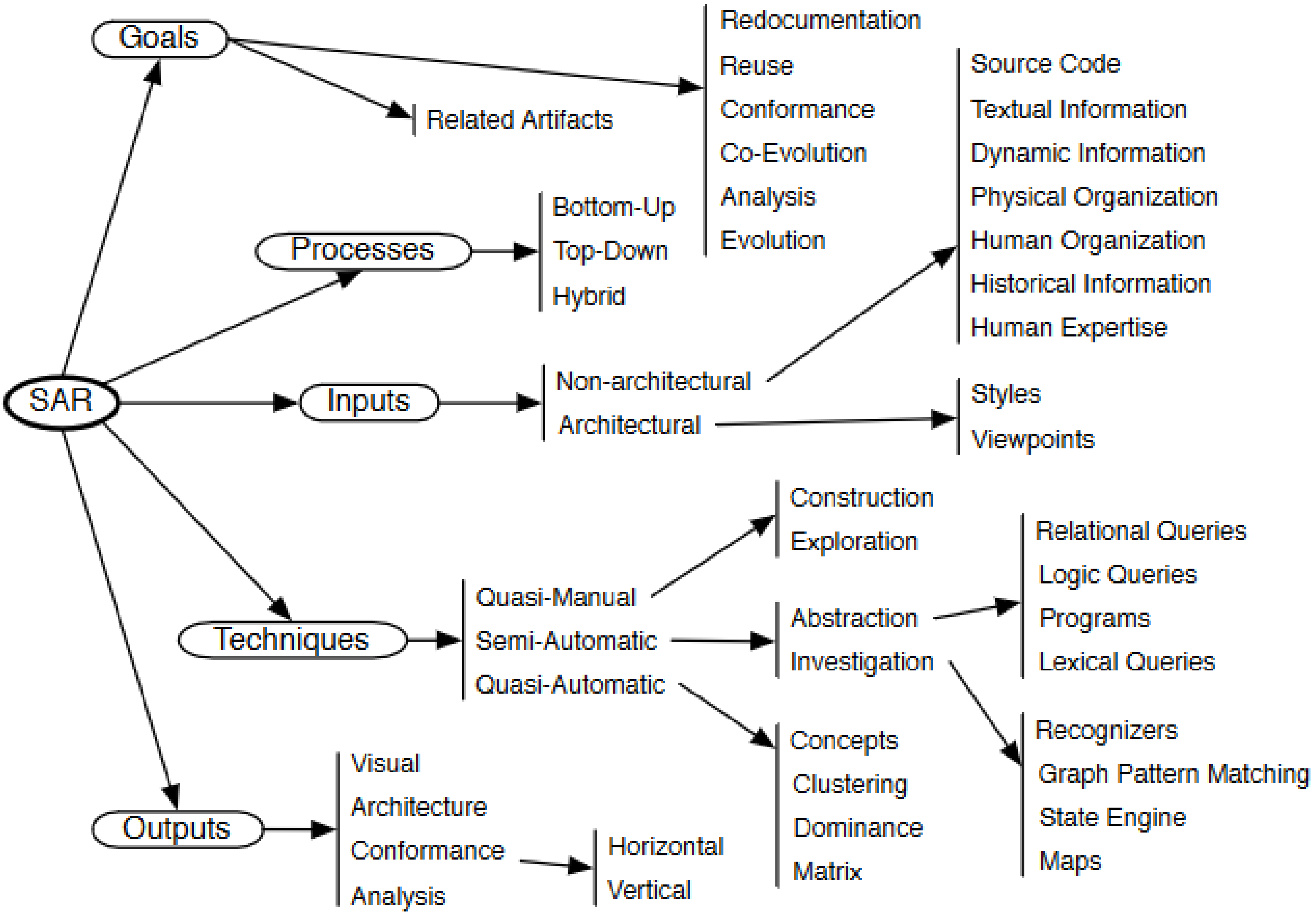
- Were you involved in architecture reconstruction of an existing system?

- How did you go about it?

- What tools did you use?

Appendix

SAR: Sw Arch Reconstruction



Ref: https://rmod.inria.fr/archives/papers/Duca09c-TSE-SOAArchitectureExtraction.pdf

Phases of architecture reconstruction

- **Raw view extraction** gets information from source code, execution traces and build scripts. It gets info such as classes, file / data they use, caller – callee relationship, global data accessed by different objects
- **DB construction**: Putting extracted data into a common format
- **View fusion**: Combines views of info stored in DB. Source code analysis gives a static view. If some objects are dynamically bound at run time then execution trace will provide this information. Then an expert may group the elements into a layer
- **Arch analysis**: Validate the correctness of architecture elements obtained from view fusion phase. Ex. There could be restriction that a layer calls objects in adjacent layers only. Or all db access should be via an entity bean only
- **Iterate**

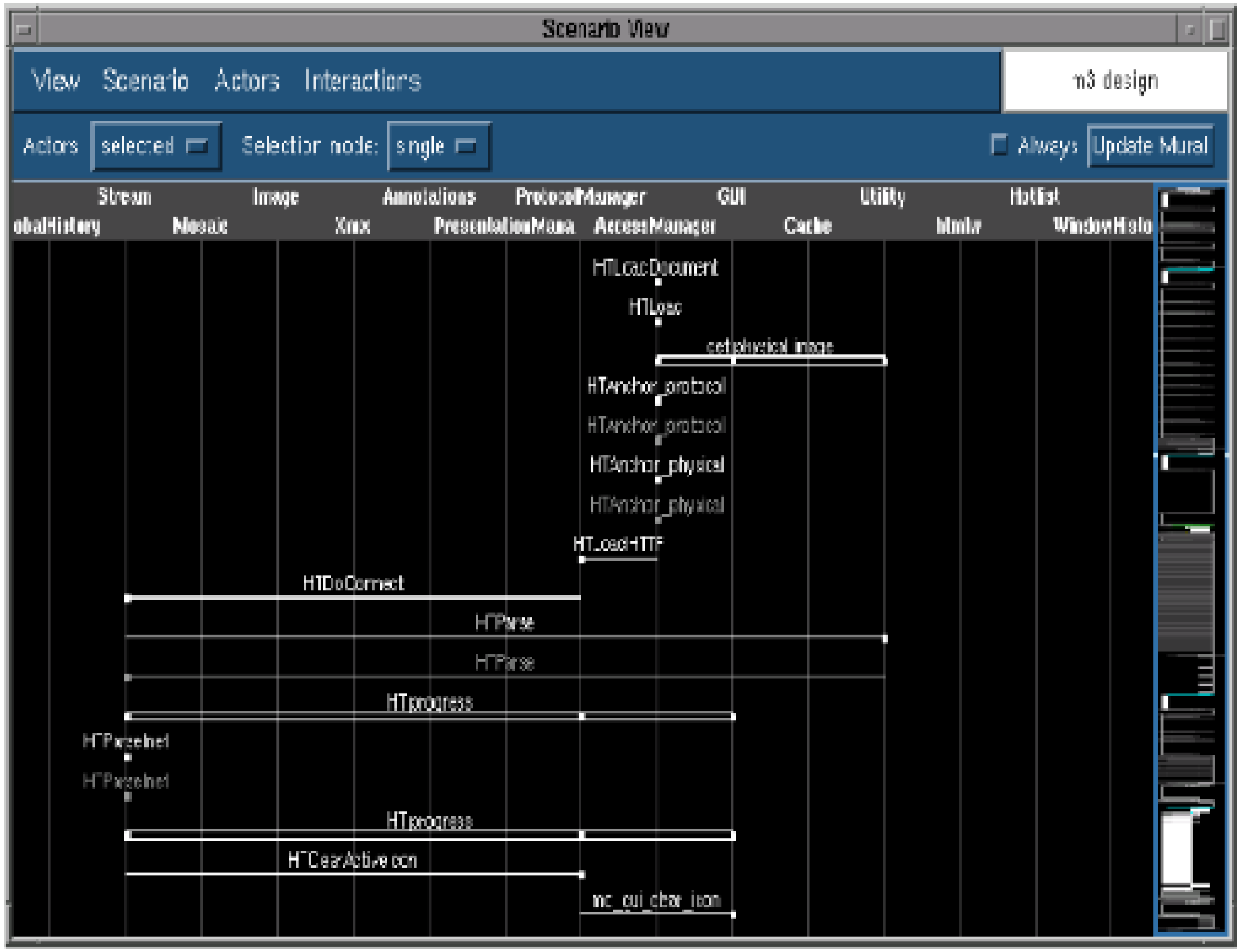


Figure 2.1. ISVis scenario view which consists of the information mural view (on the right) and the temporal message-flow diagram (center).

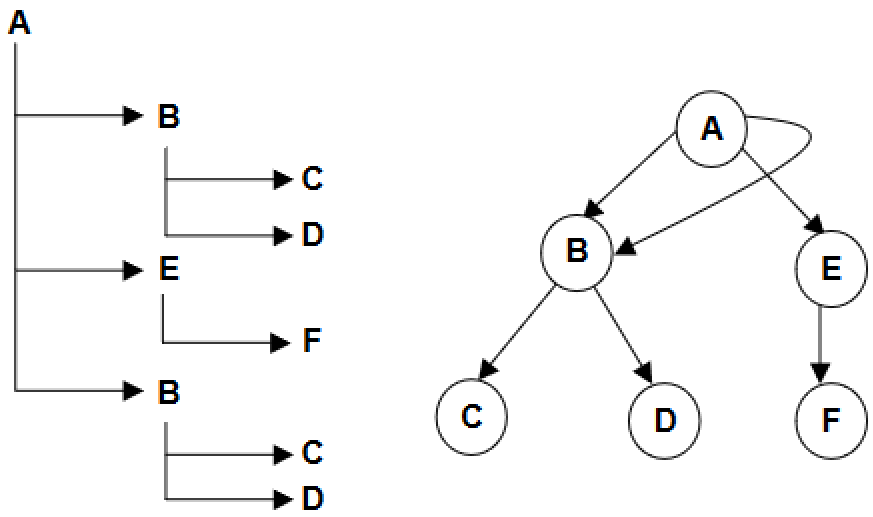
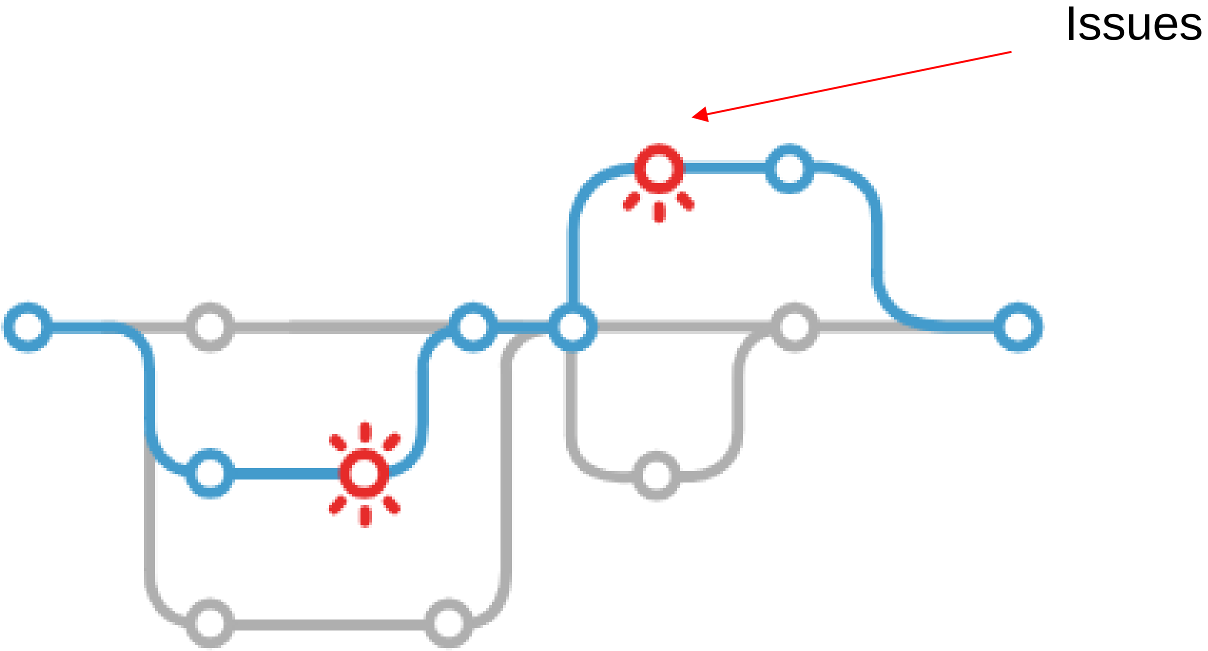


Figure 3.1. The graph representation of a trace is a better way to spot the number of distinct subtrees it contains

Ref: Techniques to Simplify the Analysis of Execution Traces for Program Comprehension by Abdelwahab Hamou-Lhadj

https://pdfs.semanticscholar.org/3db0/dd1980586c0a9d489e4b94c2996f117df2d5.pdf

SonarCube Explores All Execution Paths



https://www.sonarqube.org/features/issues-tracking/

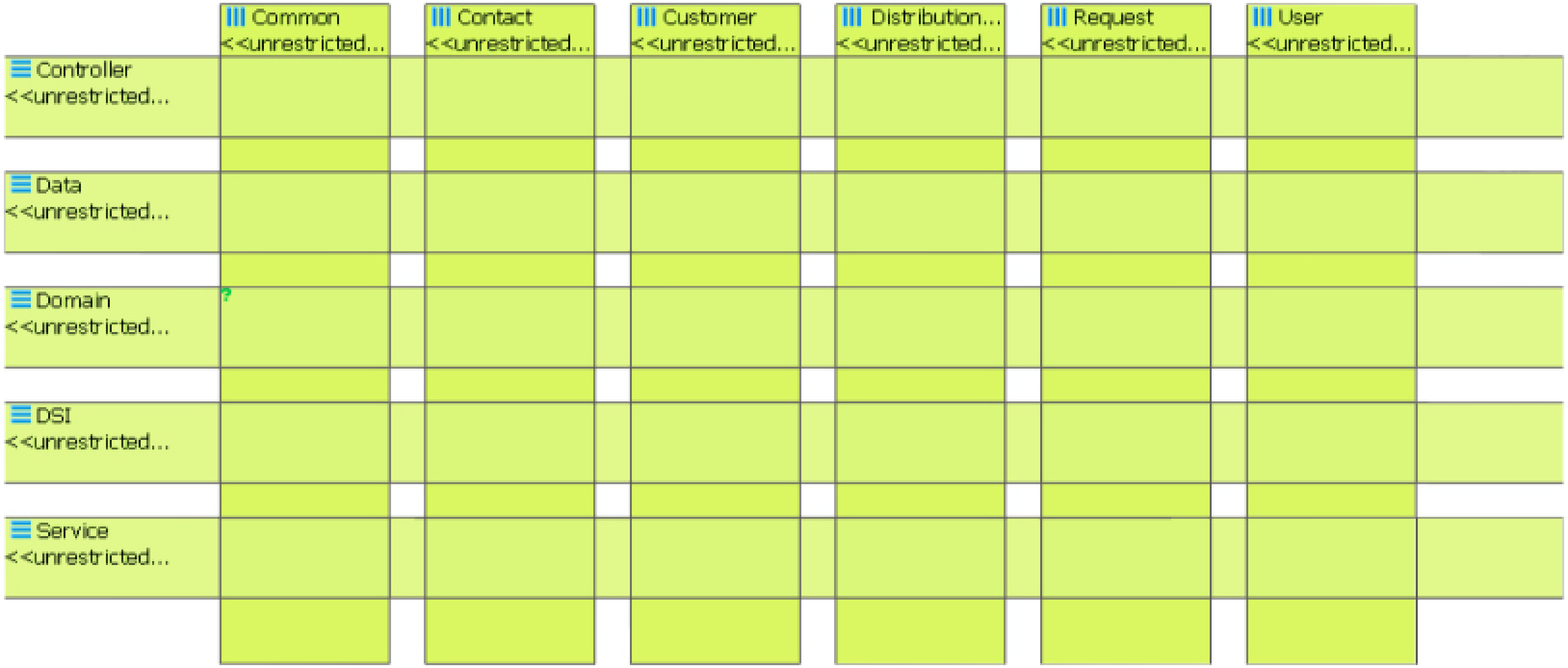
Group extracted components

- Combines views of info stored in DB.

- View 1: Source code analysis gives a static view.
- View 2: If some objects are dynamically bound at run time then execution trace will provide this information.
- View 3: Then an expert may group the elements into a layer

- Combine all these views to form a consolidated view

Sample View fusion using Sonar tool



	Common <<unrestricted...	Contact <<unrestricted...	Customer <<unrestricted...	Distribution... <<unrestricted...	Request <<unrestricted...	User <<unrestricted...
Controller <<unrestricted...						
Data <<unrestricted...						
Domain <<unrestricted...	?					
DSI <<unrestricted...						
Service <<unrestricted...						

Figure 20.3. Hypothesized layers and vertical slices

Sonar tool allows definition of layers and vertical slices through the layers

The tool will populate the layers & slices with components / elements

Architecture analysis

- **Check conformance to architecture**

- Ex. There could be restriction that a layer calls objects in adjacent layers only. Or
- Ex. All db access should be via an entity bean only

Example of violation of architecture (detected by Sonar)

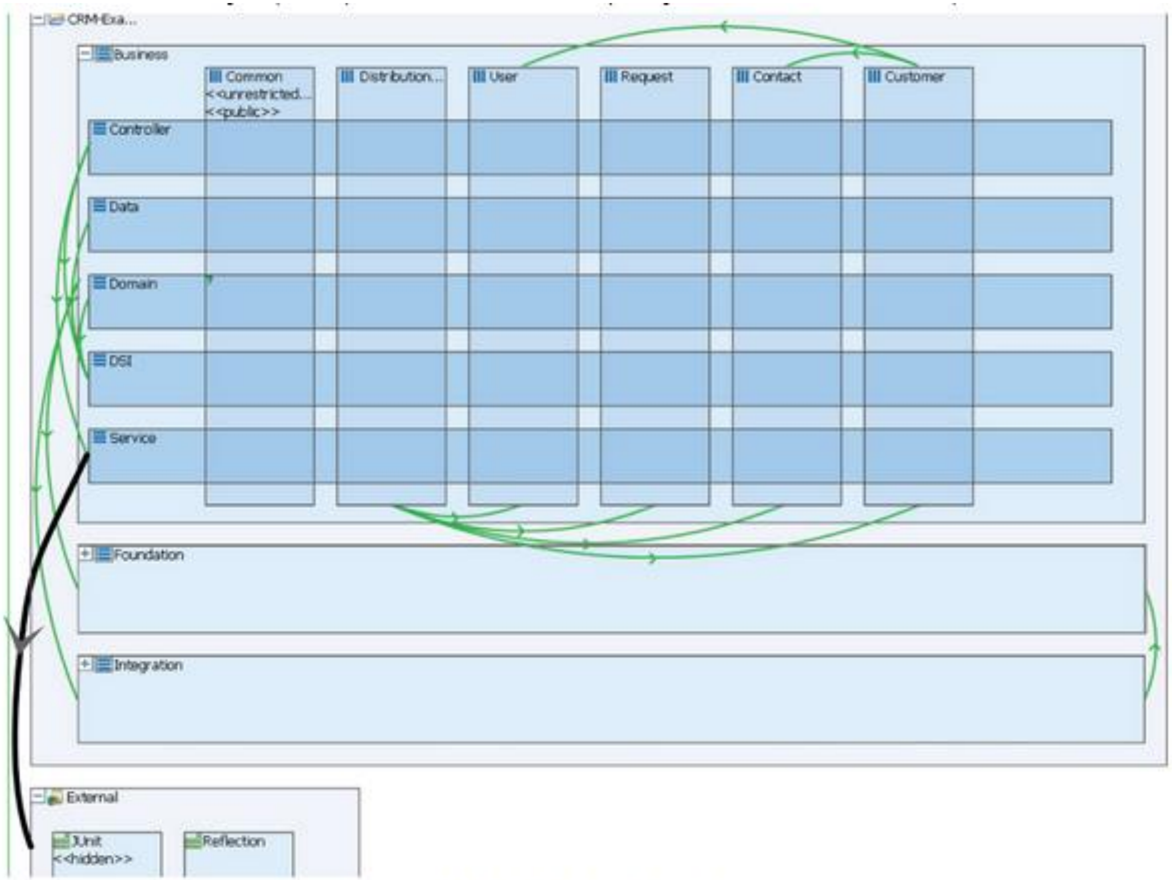


Figure 20.5. Highlighting an architecture violation

No portion of the application should depend upon Junit. Based on this specification, Sonar detects the rule violation

Example of architecture violation (detected by Sonar)

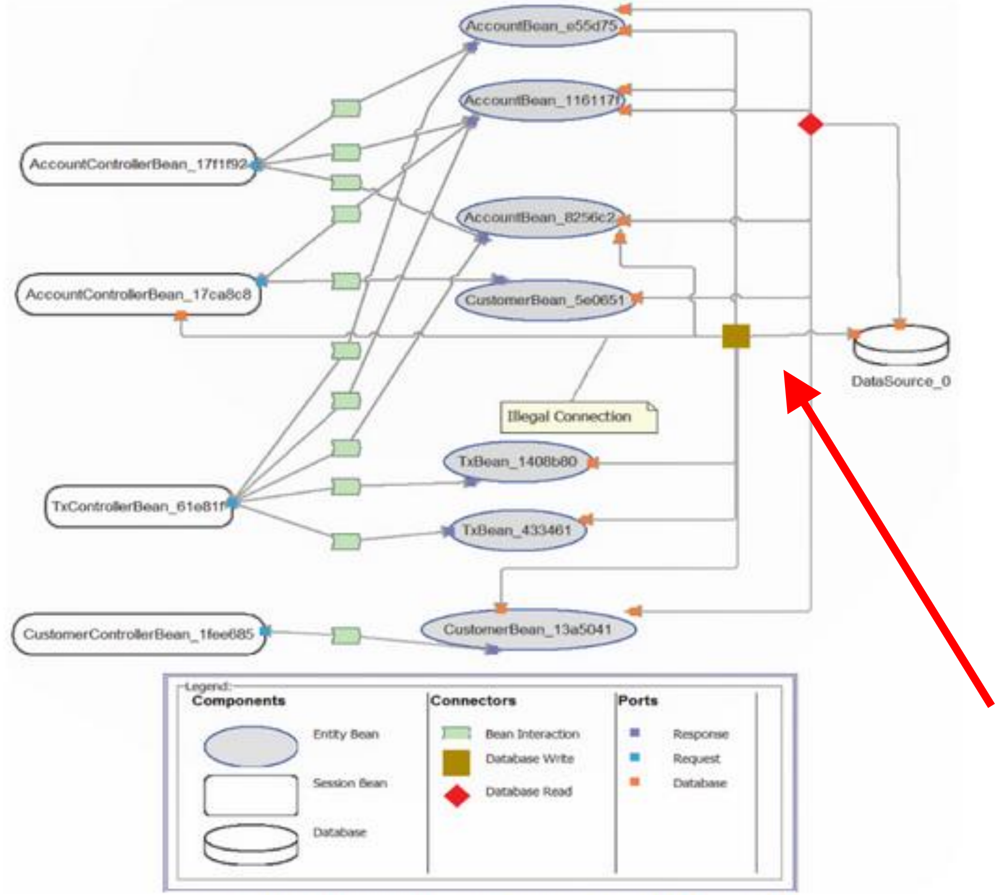


Figure 20.6. An architecture violation discovered by dynamic analysis

All database access is supposed to be managed by entity beans. Discovered by tool Disco Tect