

Reverse Engineering in practice

Joost Gabriels, Serguei Roubtsov, *Alexander Serebrenik*
LaQuSo & SET



TU / **e**

Technische Universiteit
Eindhoven
University of Technology

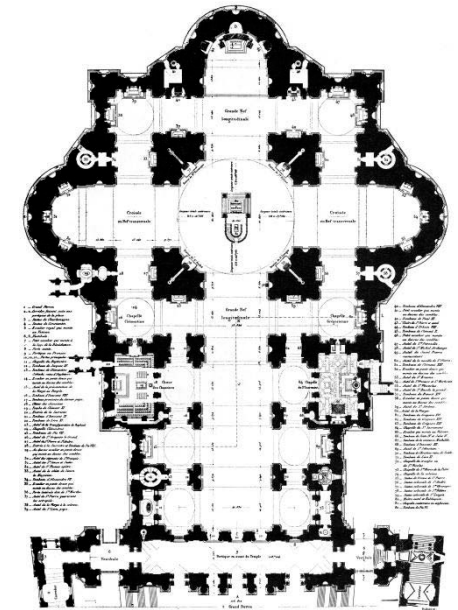
Where innovation starts

Engineering principle

- Construct
- Improve
- Maintain
- Assess
- (Re)use

objects

We need models!



Models?

- **Model** is an object **similar** to an original object in important features:
 - What is **important**?
- **Modelling** aims at **new knowledge** about the original by analysing the model.
- **Conclusion by analogy**: If two objects are similar in some features, then they are similar in other (yet unknown) features.

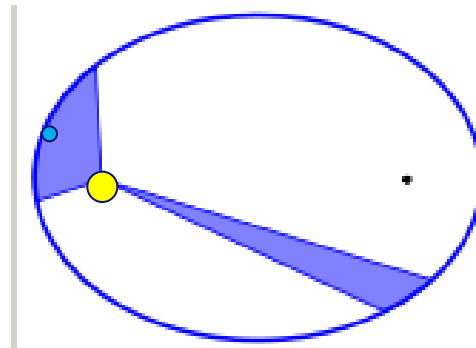
Types of models

- **Physical** (have the same physical nature as an original):
 - e.g. a model of a plane tested for aerodynamics
- **Abstract** (may have different physical nature):
 - Textual descriptions

"A line joining a planet and the sun sweeps out equal areas during equal intervals of time ."
 - Mathematical

$$\int_{t_1}^{t_2} \frac{1}{2} \cdot \text{base} \cdot d(\text{height}) = \int_{t_1}^{t_2} \frac{1}{2} \cdot r \cdot r \dot{\nu} dt = \frac{1}{2} \cdot \ell \cdot (t_2 - t_1)$$

- Graphical



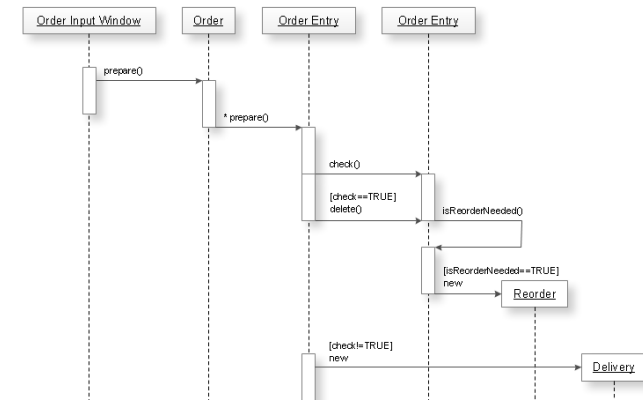
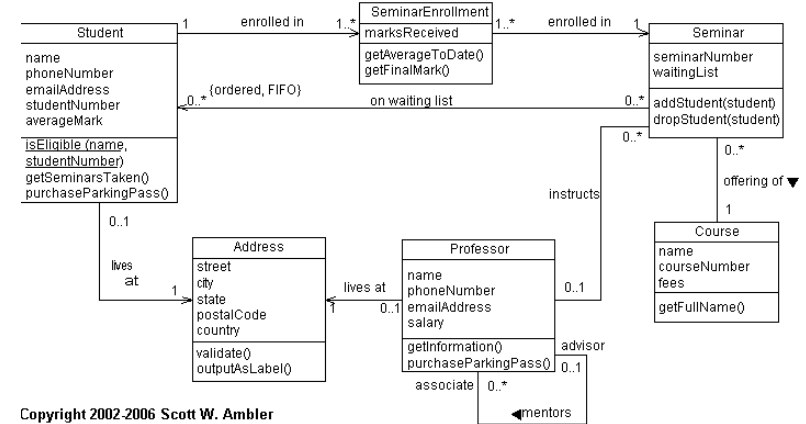
Software: Models, models and more models

- **Structure**

- UML deployment diagrams
- UML class diagrams
- Dependency graphs

- **Behaviour**

- UML sequence diagrams
- UML activity diagrams
- UML state diagrams
- (Coloured/timed) Petri nets
- Process algebras
- Flow charts
- Performance model
- Communication model



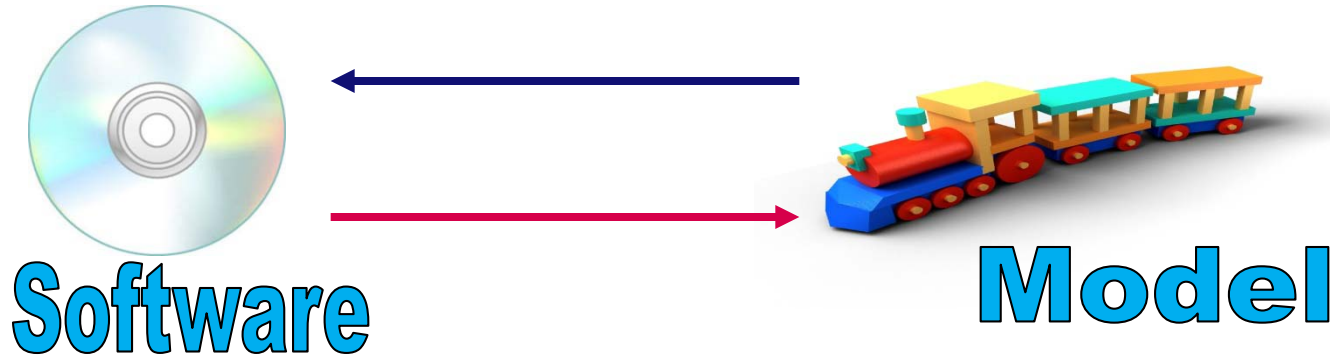
Sounds familiar? Kruchten's 4+1 views

	Static (structure)	Dynamic (behaviour)
Abstract	Logical	Process
Concrete	Development (code in files)	Deployment (processors)

+ Use case scenarios traced through the architecture

Reverse Engineering?

Deriving model from the code.



Ingolf Krüger, U. of California:

“We have been successful in moving from models to code, the challenge is **round-trip engineering**”

That’s what **reverse engineering** is about!

From Code to Models – Why?

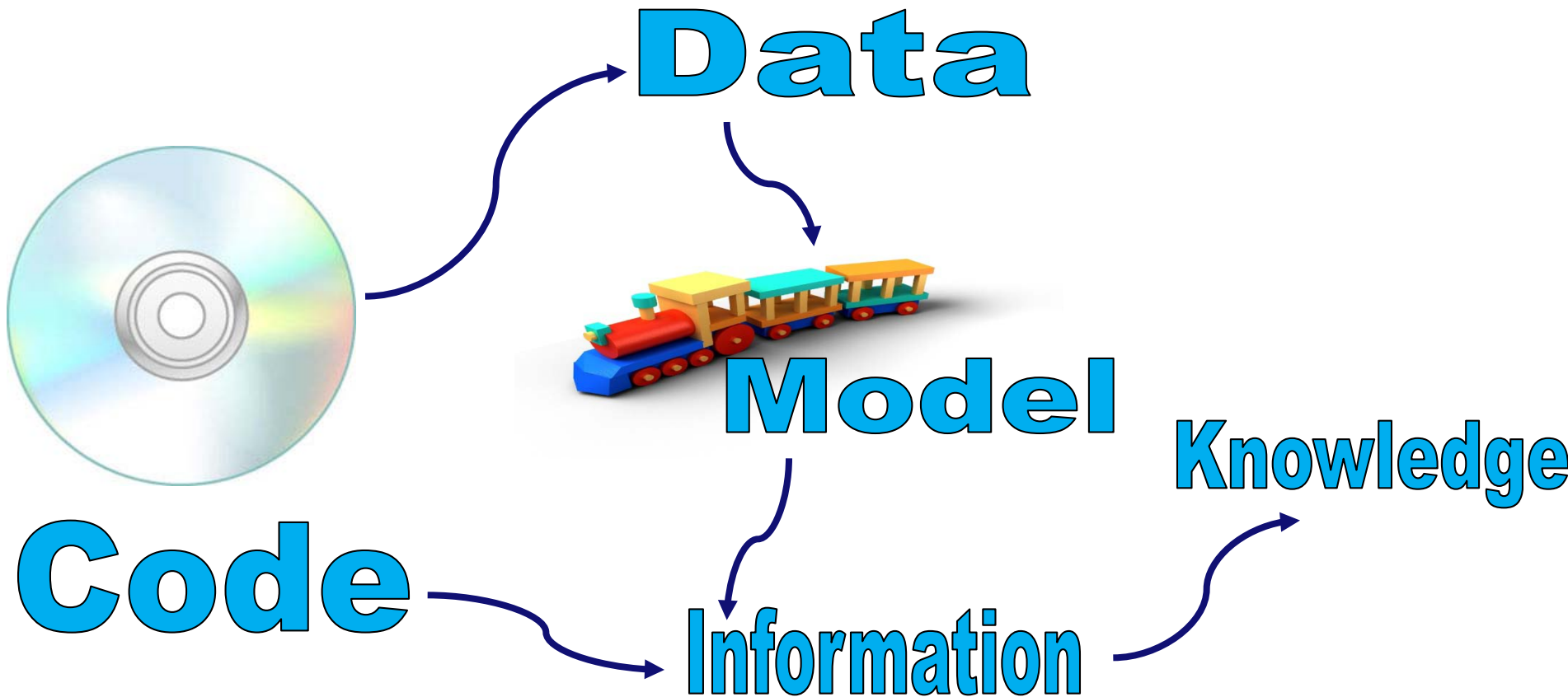
- **Consistency check**
 - Implementation vs. documentation
- **Understanding software**
 - In absence of architectural documentation
- **Software quality assessment**
 - Models may be easier to analyze
 - Software models are often **graphical**
- **Preliminary step for**
 - Re-engineering
 - Migration
 - High-level documentation generation

Reverse Engineering Approach

- Depends on
 - What kind of model would we like?
 - structure / behaviour
 - precise / approximate
 - What kind of code do we have?
 - complete / incomplete
 - compilable / executable / neither
 - programming languages:
heterogeneous / homogeneous
 - “special cases”:
 - process models
 - business rules

No silver bullets!

From code to model



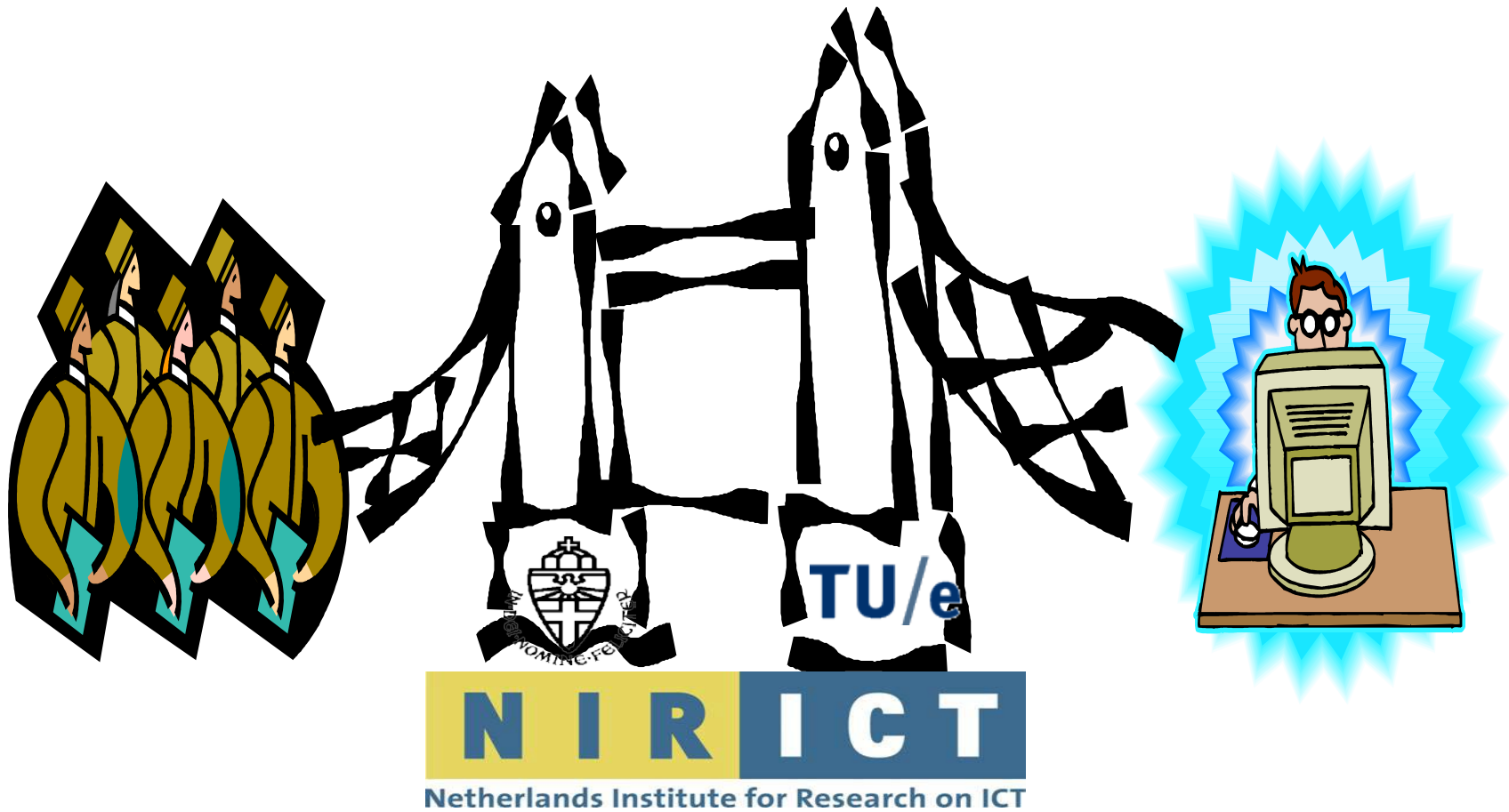
Reverse Engineering Approach

- **Code \Rightarrow Data**
 - Parsing
 - Scripting
 - Focused search (grep, ...)
- **Data \Rightarrow Model**
 - Fact extraction
- **Model \Rightarrow Information**
 - Measurement
 - Visualisation
- **Code \Rightarrow Information**
 - Inspection
 - Walkthrough
- **Information \Rightarrow Knowledge:**
 - Review



1000 facets of reverse engineering based on LaQuSo case studies

LaQuSo?

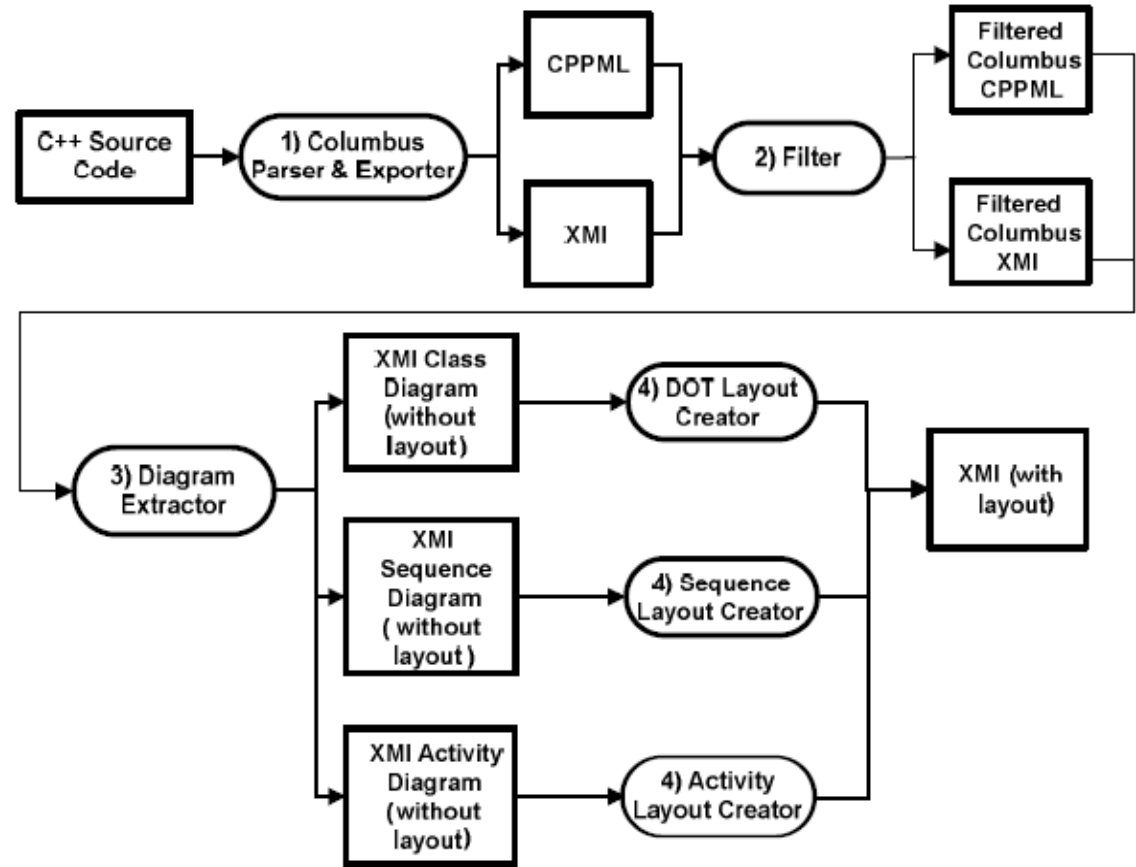


Toy example: Bellflower

- What kind of **code** do we have?
 - complete
 - compilable
 - programming languages: homogeneous: MS VS C++
 - originates from Rational Rose models
 - Original models vs. inferred models!
- What kind of **model** would we like?
 - **structure**: UML class diagrams
 - **behaviour**: UML sequence diagrams
 - **precise**

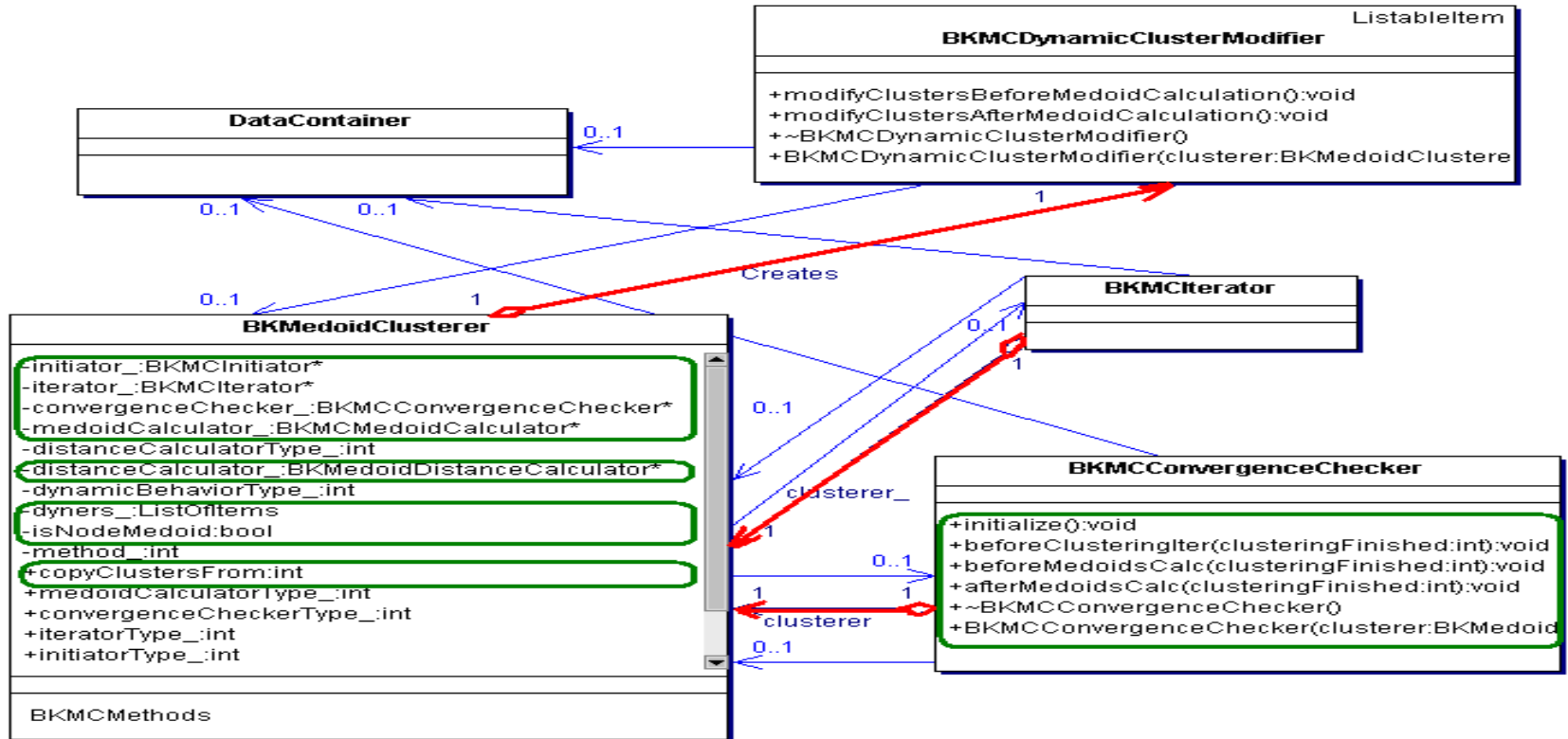
What did we do? Approach

- **Code \Rightarrow Data**
 - Parsing
- **Data \Rightarrow Model**
 - **Fact extraction:**
 - Filtering
 - Diagram extraction
- **Model \Rightarrow Information**
 - Visualisation
- **Information \Rightarrow Knowledge**
 - Review



CPP2XMI

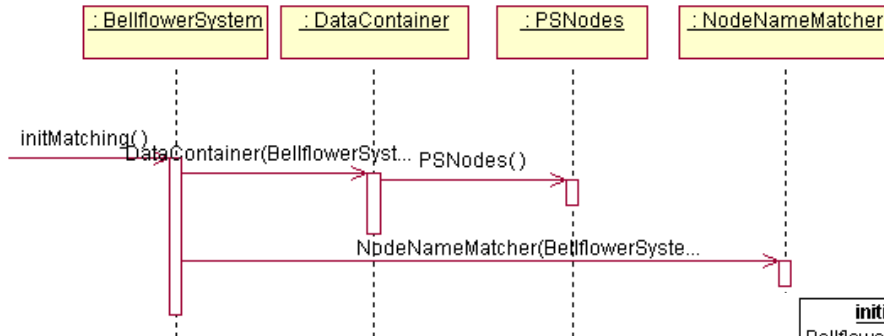
Information (structure)



Inferred class diagram contains more details than the original one:

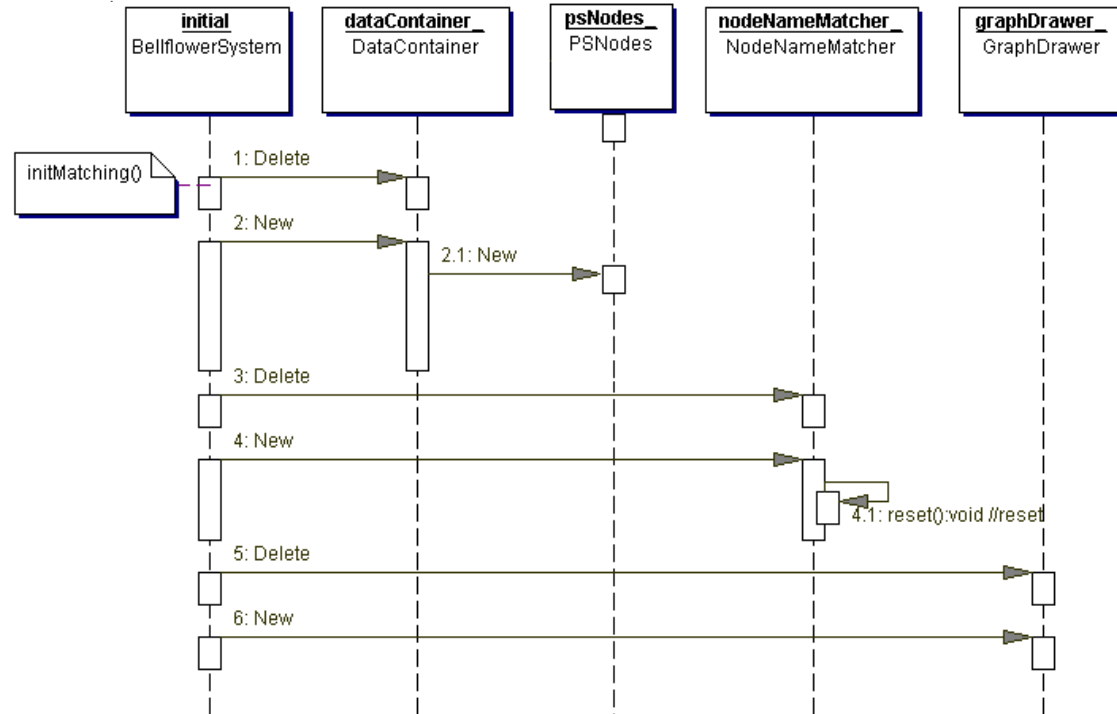
- Additional fields and methods in certain classes
- Additional relationship: aggregation

Information (behaviour)



Sequence diagrams:

- The inferred one contains **more detailed behaviour**: `new()` and `delete()` methods and
- One **more object** derived from implementation



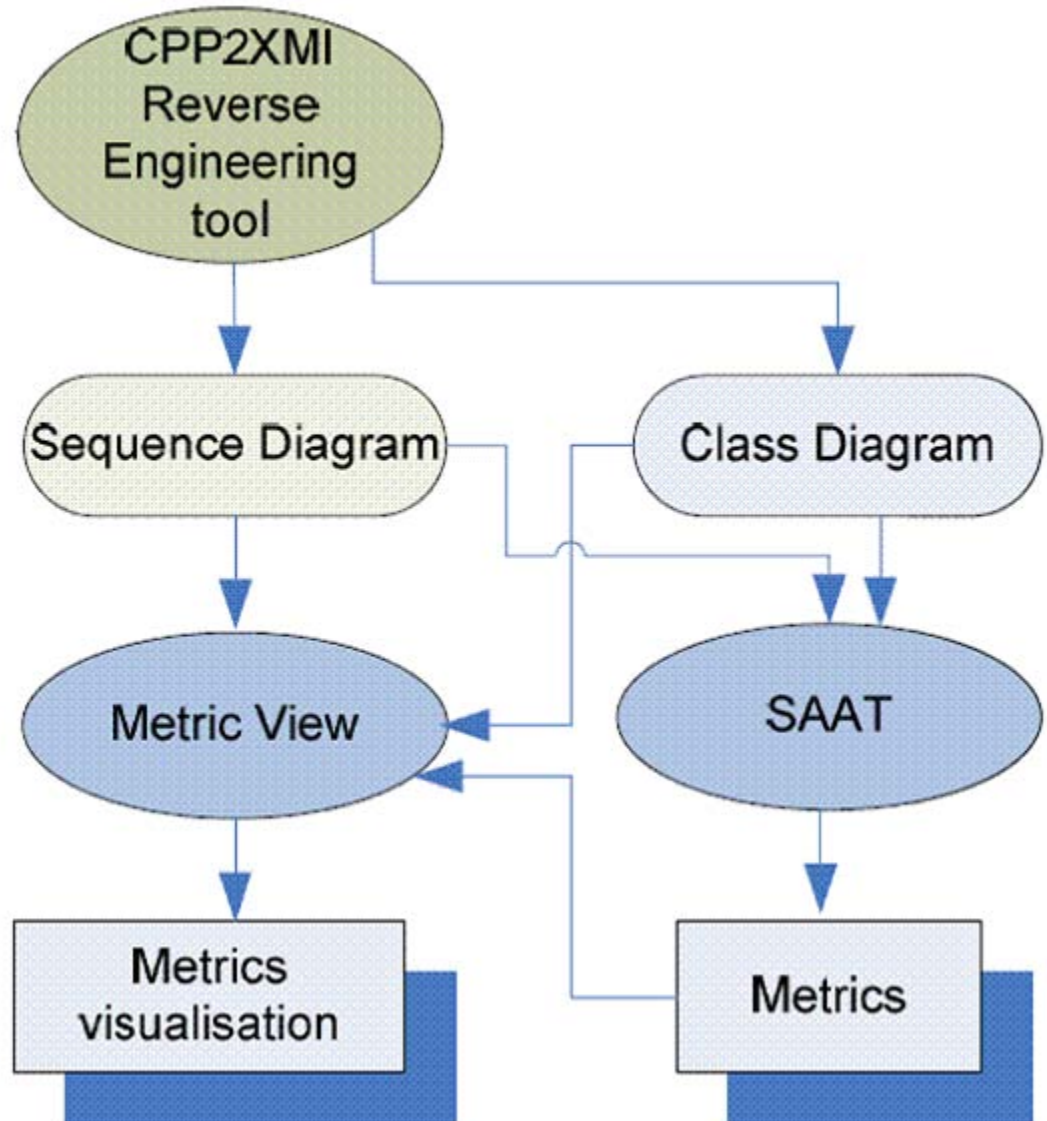
Inferred model is **consistent** wrt design.

Industrial case with CPP2XML: Printer-producing company's software


- What kind of **code** do we have?
 - complete
 - compilable
 - programming languages: homogeneous: C++
 - 60 KLOC
 - No documentation
- What kind of **model** would we like?
 - **structure**: UML class diagrams
 - **behaviour**: UML sequence diagrams
 - **precise**

What did we do? Approach (continued)

- **Model** ⇒ **Information**
 - Measurement
 - Visualisation
- **Information** ⇒ **Knowledge: Review**



1 picture = 1000 words?



unused
classes

We need something better...

Metrics (1)

Metrics	Subsystems	
	A	B
Number of classes	176	70
Number of methods	1106	383
Avg. methods per class	6.28	5.45
Classes with > 30 methods	4	2
Max fan-in / Max fan-out	27 / 27	23 / 21

- Subsystem A is quite **big**.
- **Big parts** of functionality are implemented in a **few files**.
- Many files **depend** on these few.

Metrics (2)

- **Models derived: (illegible) sequence diagrams**

Metrics		Subsystems	
		A	B
Incoming and outgoing messages per class	Maximum	112	271
	Classes with > 30 mess.	5	6
Max. depth of scenario		41	55

- A number of **heavily used** classes
- Scenarios' depth: too high → functionality should be differently distributed.

So far...

- **Case studies:**
 - **Code**
 - Complete, compilable, homogeneous (OO)
 - **Model**
 - UML class/sequence/activity
 - precise
- **Approach**
 - parsing, fact extraction, visualisation, measurement
- **Results**
 - Precision \Rightarrow Illegibility (too many details)
 - Metrics can be of great help!

Real life industrial systems

They are often:

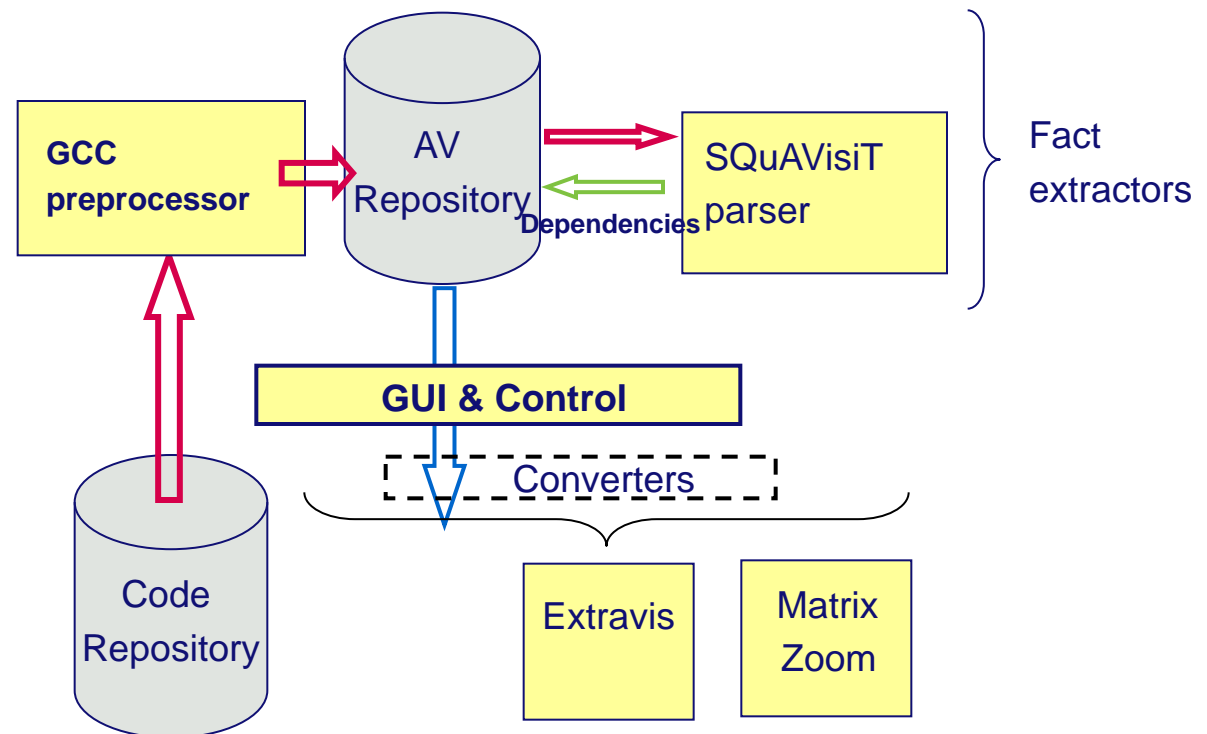
- **Not only OO (legacy systems)**
- **Heterogeneous (C/Assembler, Cobol/PL SQL,...)**
- **Incomplete (some code is in libraries and third-party components)**
- **Not compilable and executable within analysis environment ('weird' OS, proprietary development environment, ...)**

Industrial case with SQuAVisiT: Embedded System

- **What kind of code do we have?**
 - **Not OO**
 - **Almost homogeneous: mostly C with embedded Assembler**
 - **Complete**
 - **Modules of interest are compilable (at least can be parsed)**
 - **Medium size: 150 KLOC**
 - **No documentation**
- **What kind of model would we like?**
 - **Structure: dependencies and layering**
 - **Approximate (function pointer calls and Assembler code ignored)**

What did we do?

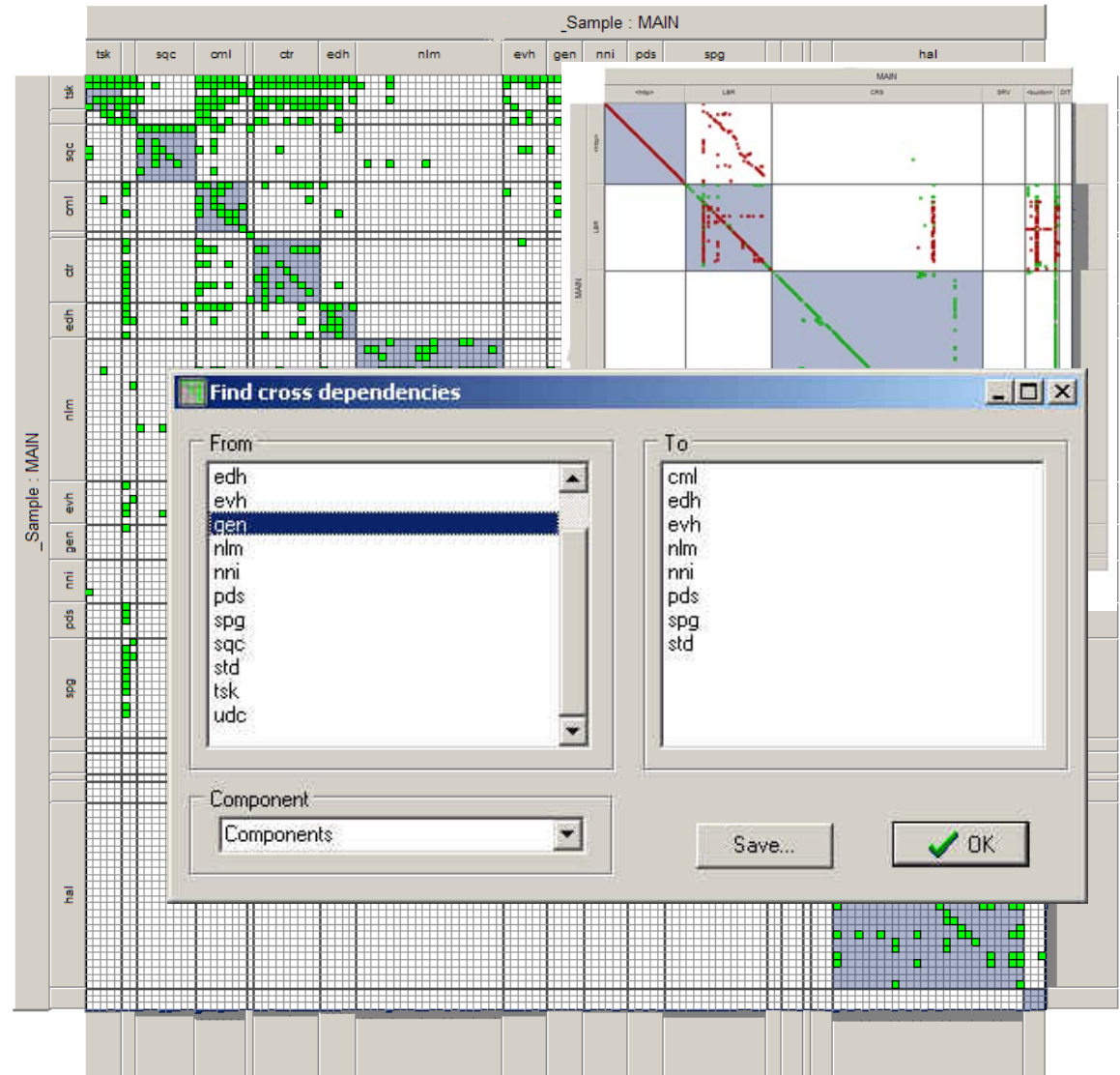
- **Code** \Rightarrow **Data**
 - parsing
- **Data** \Rightarrow **Model**
 - **Fact extraction:**
 - Dependencies extraction
- **Model** \Rightarrow **Information**
 - Visualisation
- **Information** \Rightarrow **Knowledge**
 - Review



SQuAVisiT

Structure (Matrix View, 2)

- system is poorly layered
- unexpected cross-dependencies exist between components



Case conclusions

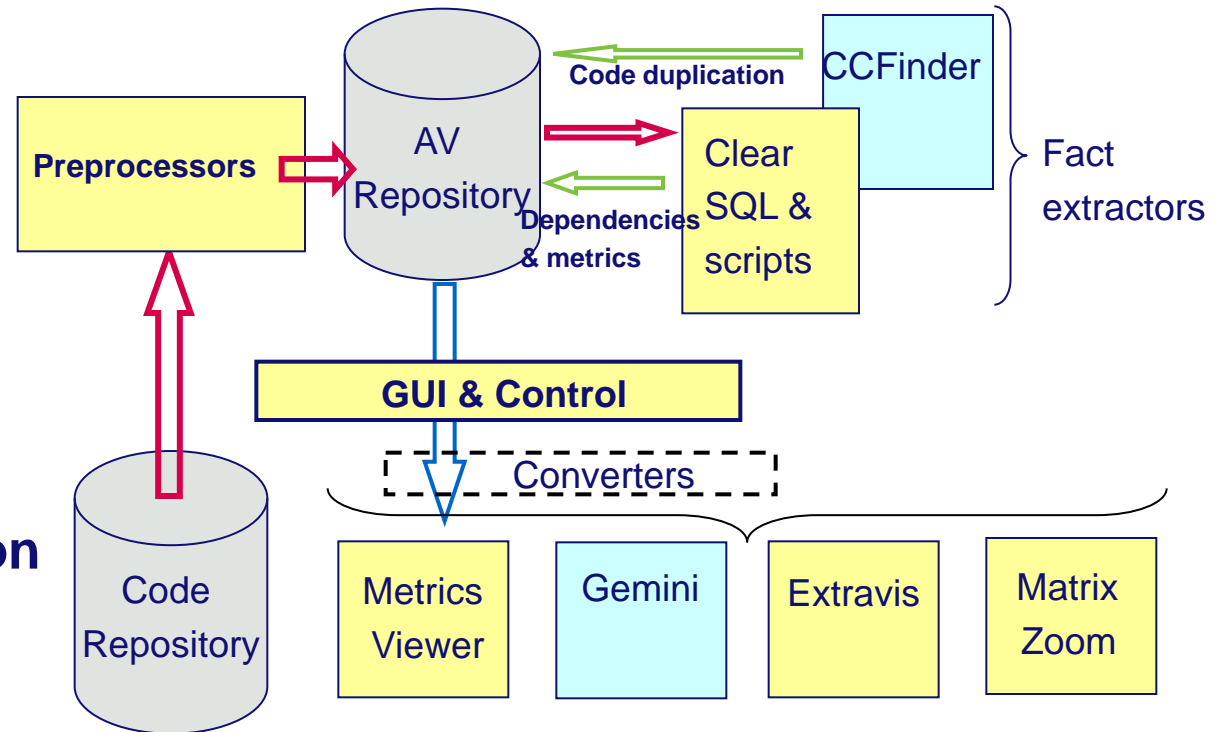
- **Non OO systems demand different models:**
 - **Dependencies**
- **Visualisation – key to understanding**

Industrial case: Insurance company's expert system

- **What kind of code do we have?**
 - Not OO
 - Heterogeneous: Javascript, PL SQL, C++, Java, Cobol
 - Complete
 - Not compilable (less relevant here)
 - Medium size: 300 KLOC
 - Scarce documentation
- **What kind of model would we like?**
 - **Structure:** dependencies and layering with implications for maintenance

What did we do? Alternative approach

- **Code** \Rightarrow **Data**
 - Ad-hoc scripting
 - ClearSQL tool
- **Data** \Rightarrow **ModelS**
 - **Fact extraction:**
 - Filtering
 - Dependency extraction
 - Duplication extraction
- **ModelS** \Rightarrow **Information**
 - Visualisation
 - Measurement
- **Information** \Rightarrow **Knowledge**
 - Review



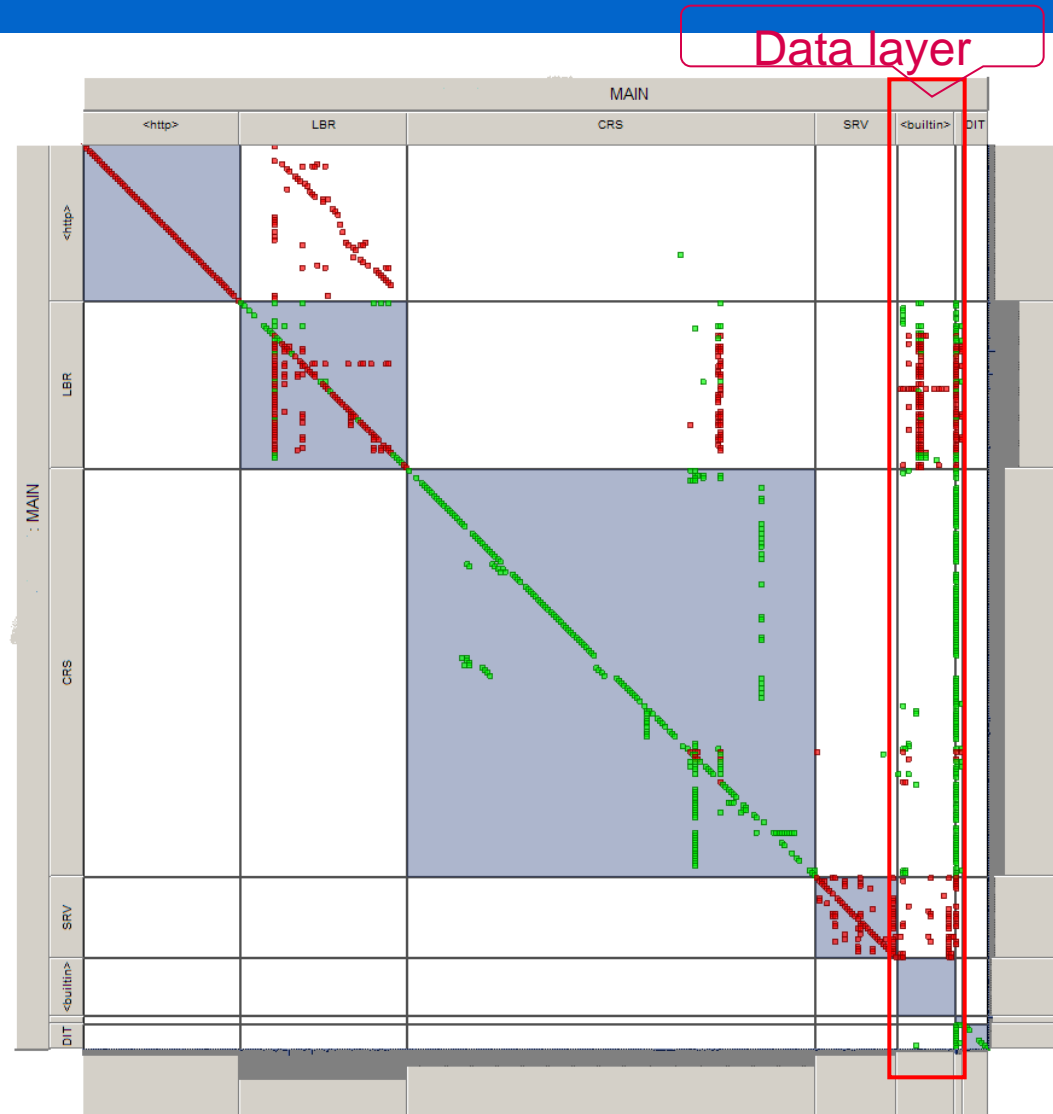
SQUAVISIT

Dependencies Model: Matrix View (1)

Model \Rightarrow Information

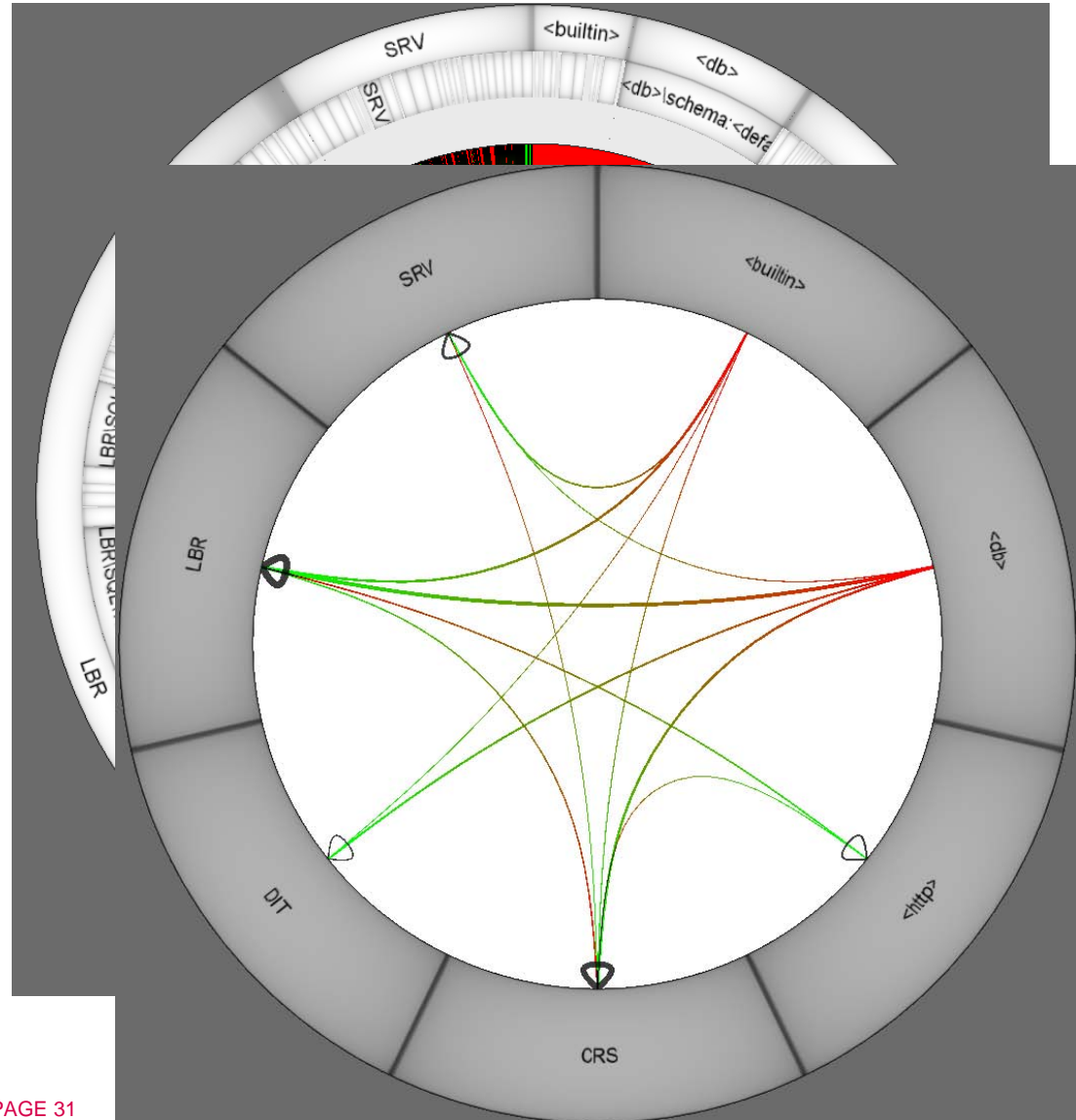
- (Almost) layered:
good design
 - BUT data layer is
accessed from
several layers
- Layers affected by
calls from top layer are
visible (red squares)

What are the
maintenance
implications of this
figure?



Dependencies Model: Extravis

- Huge green 'bubbles' reflect some controversial coding approach: getting rid of parameters by means of naming like $f(1,3) \rightarrow f_1_3$
- Absence of dedicated data access layer is confirmed



Dependencies Model: Metrics

- ‘Change propagators’ - modules with big Fan-in & Fan-out –bottlenecks
- Modules with zero fan-in – dead code?

Fan-in and Fan-out Counter

of components: 51

Component	Fan-in	Fan-out
LBR\SQL\XML.SQL	0	0
LBR\SQL\TLP11000.SQL	1	4
LBR\SQL\TLP12310.SQL	1	4
LBR\SQL\TLP12330.SQL	1	4
LBR\SQL\TLP12500.SQL	1	9
LBR\SQL\TLP12610.SQL		
LBR\SQL\TLP12630.SQL		
LBR\SQL\TLP31100.SQL		
LBR\SQL\TLP31200.SQL		
LBR\SQL\TLP31300.SQL		
LBR\SQL\TLP43100.SQL		
LBR\SQL\TLP45100.SQL		
LBR\SQL\TLP51000.SQL		
LBR\SQL\TLP52200.SQL		
LBR\SQL\TLP56000.SQL		
LBR\SQL\TLP61000.SQL		
LBR\SQL\TLP61210.SQL		
LBR\SQL\TLP62100.SQL		
LBR\SQL\TLP62200.SQL		
LBR\SQL\TLP62630.SQL		
LBR\SQL\TLP90010.SQL		
LBR\SQL\TLP12100.SQL		
LBR\SQL\TLP12200.SQL		
LBR\SQL\TLP16000.SQL		
LBR\SQL\TLP21200.SQL		
LBR\SQL\TLP21400.SQL		
LBR\SQL\TLP51210.SQL		
LBR\SQL\TLP13100.SQL		
LBR\SQL\TLP52100.SQL		
LBR\SQL\TLP53100.SQL		
LBR\SQL\TLP54000.SQL		
LBR\SQL\TLP63100.SQL		
LBR\SQL\TLP12320.SQL		
LBR\SQL\TLP14000.SQL		
LBR\SQL\TLP21210.SQL		
LBR\SQL\TLP52000.SQL		
LBR\SQL\TLP62320.SQL		
LBR\SQL\TLP62000.SQL		
LBR\SQL\TLP64000.SQL		
LBR\SQL\TLP12000.SQL		
LBR\SQL\TLP12400.SQL		
LBR\SQL\TLP00000.SQL		

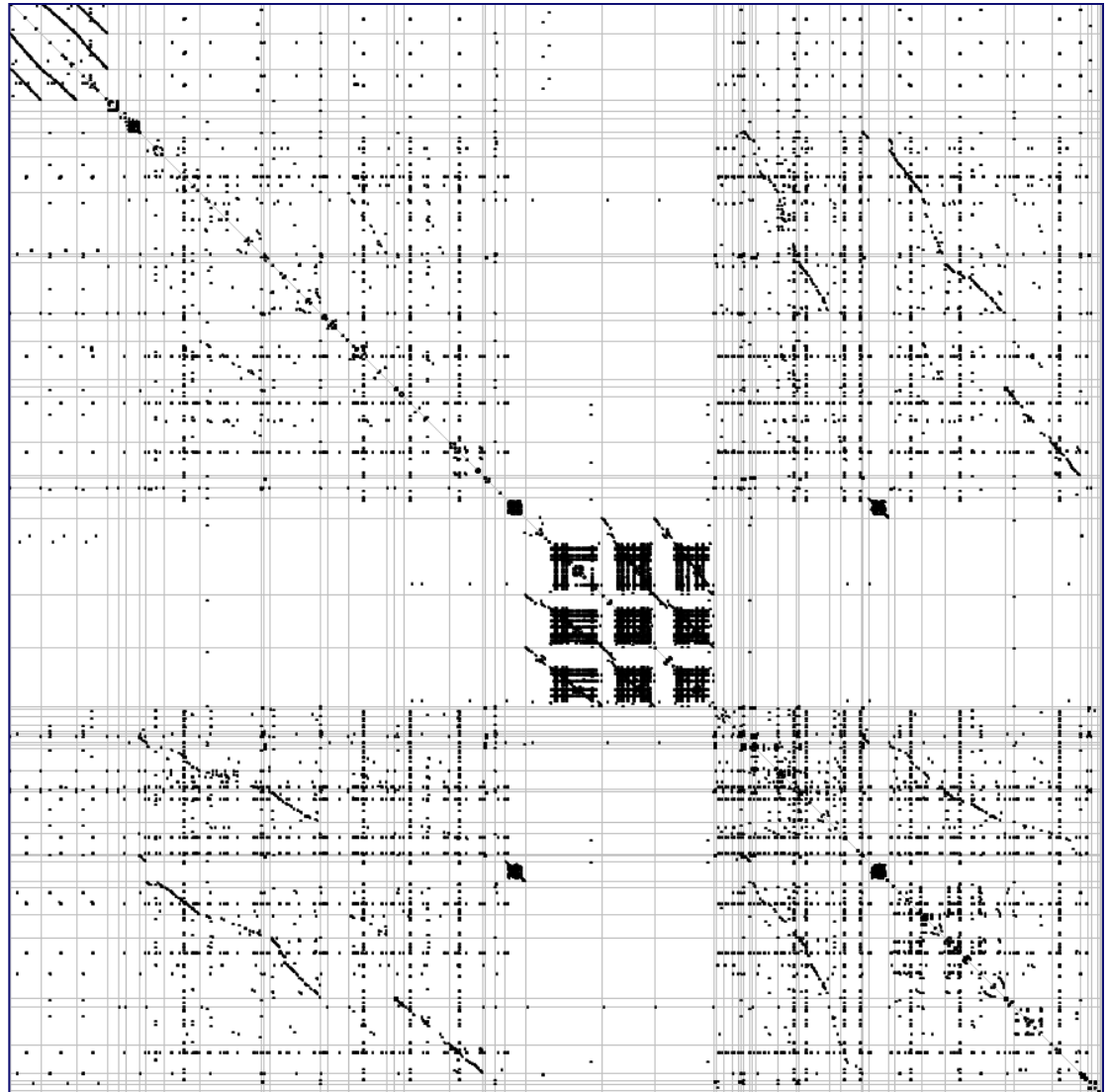
Fan-in and Fan-out Counter

of components: 35

Component	Fan-in	Fan-out
<http>\lexbr_test_mod	0	1
CRS\SQL\CC_PROC.SQL	0	2
CRS\SQL\CRS11000.SQL	0	4
CRS\SQL\CRS12000.SQL	0	3
CRS\SQL\F_FL5_SOM_OBLIGO_INV.SQL	0	2
CRS\SQL\F_FL5_SOM_OBLIGO_INV_EUR.SQL	0	2
CRS\SQL\F_INV_BEDFRAG.SQL	0	1
CRS\SQL\F_SOM_OBLIGO_INV.SQL	0	2
CRS\SQL\F_SOM_OBLIGO_INV_1.SQL	0	2
CRS\SQL\F_SOM_OBLIGO_INV_1_EUR.SQL	0	2
CRS\SQL\F_SOM_OBLIGO_INV_EUR.SQL	0	2
CRS\SQL\NSTEMP3.SQL	0	2
CRS\SQL\TGS0040.SQL	0	1
CRS\SQL\TGS0045.SQL	0	1
CRS\SQL\TGS0090.SQL	0	1
CRS\SQL\TRD1100.SQL	0	3
CRS\SQL\TRP0040.SQL	0	1
CRS\SQL\TRX1005.SQL	0	2
CRS\SQL\TRX1009.SQL	0	3
CRS\SQL\TRX1010.SQL	0	4
CRS\SQL\TRX1021.SQL	0	1
CRS\SQL\TRX1035.SQL	0	1
CRS\SQL\TRX1036.SQL	0	1
CRS\SQL\TRX2000.SQL	0	11
CRS\SQL\TRX3001.SQL	0	2
CRS\SQL\TRX3002.SQL	0	1
CRS\SQL\TRX4000.SQL	0	1
DIT\SQL\DIT_REDUNDANT.SQL	0	1
DIT\SQL\DIT_REDUNDANT_1.SQL	0	1
DIT\SQL\DIT_REDUNDANT_2.SQL	0	1
LBR\ONT\DYNAMISCHE_PAGINAS.SQL	0	1
LBR\ONT\NSTEMP.SQL	0	1
LBR\ONT\TEST2.SQL	0	1
LBR\ONT\TEST_TO_ZEGGE.SQL	0	2
LBR\ONT\TEST_XML.SQL	0	1

Code duplication model: CCFinder/Gemini

- **Code is polluted with duplication: restructuring would improve maintainability but may change the architecture**



Quality model: Metrics

Module	LOC	Comments	Blanks	Vg	Comments_p	Difficulty	Effort	MI
LBRIS QL4TLP13100.\$QL:calculeer_aanvraag	772	423	1	68	35	16	126274	-12
LBRIS QL4TLP12400.\$QL:valideer_verpl_velden	412	550	1	60	57	14	65200	63
LBRIS QL4TLP14000.\$QL:verwerk_in_crs	800	416	1	53	34	16	131184	-2
LBRIS QL4TLP12200.\$QL:schrijf_object	1252	529	1	52	30	21	20000	-26
LBRIS QL4TLP21400.\$QL:genereer_xml_bericht	653	279	1	50	30	12	4229	-12
LBRIS QL4TLP62200.\$QL:schrijf_object	548	340	1	41	35	17	104781	17
LBRIS QL4TLP12100.\$QL:schrijf_aanvraag	603	257	2	39	34	17	39116	18
LBRIS QL4TLP21200.\$QL:lees_documenten	607	183	2	37	28	17	43761	-7
LBRIS QL4TLP21210.\$QL:bepaal_medeondertek	630	140	2	36	30	22	216213	-19
LBRIS QL4TLP21210.\$QL:bepaal_tenaamondertek	600	141	2	36	22	22	105517	-15
LBRIS QL4TLP61210.\$QL:bepaal_medeondertek	657	129	2	36	31	21	242519	-17
LBRIS QL4TLP61210.\$QL:bepaal_tenaamondertek	617	134	2	36	21	21	105478	-16
LBRDISV4TLP63100.\$QL:schrijf_calculatie	713	223	2	35	24	21	212155	-17
LBRIS QL4TLP6100.\$QL:schrijf_aanvraag	462	196	1	32	30	19	163214	-11
LBRIS QL4TLP11000.\$QL:zoek_aanvragen	388	125	2	31	24	15	64121	29
LBRIS QL4TLP61000.\$QL:zoek_aanvragen	405	134	2	31	25	14	212155	-17
LBRIS QL4TLP12500.\$QL:schrijf_object	710	280	1	30	28	12	276235	-24
LBRIS QL4TLP64000.\$QL:verwerk_in_crs	549	333	1	28	38	12	72097	-2
LBRIS QL4TLP12400.\$QL:valideer_verpl_velden_html	226	116	1	27	34	10	64425	11
LBRIS QL4TLP53100.\$QL:calculeer_mantel	291	101	1	21	26	11	75628	-2
LBRIS QL4TLP12630.\$QL:toon_his	194	50	2	20	20	10	74359	-1
LBRIS QL4TLP13100.\$QL:schrijf_calculatie	312	162	2	20	34	12	39306	0
							65519	43
							19998	43
							29787	16
							20231	16
							32440	46

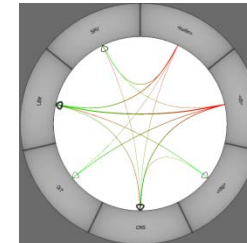
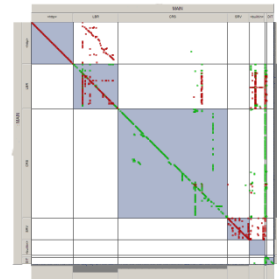
Long and (or) complex functions should be refined or better commented

$$MI = 171 - 5.2 * \ln(\text{aveV}) - 0.23 * \text{aveV}(g') - 16.2 * \ln(\text{aveLOC}) + 50 * \sin(\sqrt{2.4 * \text{perCM}})$$

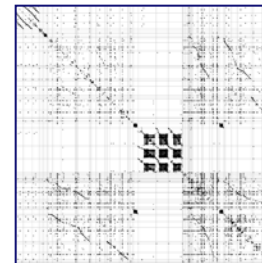
Case conclusions

- Analysis required **multiple models:**

- **Dependency model**



- **Code duplication model**



- **Quality model**

Module	LLOC	Comments	Errors	Va	Comments_M	Difficulty	Effort	MI
ERBISQLITP13100SQLcoltbler_samraag	472	43	1	65	10	10	10	10
ERBISQLITP12400SQLcoltbler_voyl_wildon	412	43	1	65	10	10	10	10
ERBISQLITP10800SQLcoltbler_samraag	402	43	1	65	10	10	10	10
ERBISQLITP12000SQLcoltbler_samraag	382	43	1	65	10	10	10	10
ERBISQLITP10400SQLcoltbler_samraag	372	43	1	65	10	10	10	10
ERBISQLITP12200SQLcoltbler_samraag	362	43	1	65	10	10	10	10
ERBISQLITP12600SQLcoltbler_samraag	352	43	1	65	10	10	10	10
ERBISQLITP12800SQLcoltbler_samraag	342	43	1	65	10	10	10	10
ERBISQLITP13000SQLcoltbler_samraag	332	43	1	65	10	10	10	10
ERBISQLITP13200SQLcoltbler_samraag	322	43	1	65	10	10	10	10
ERBISQLITP13400SQLcoltbler_samraag	312	43	1	65	10	10	10	10
ERBISQLITP13600SQLcoltbler_samraag	302	43	1	65	10	10	10	10
ERBISQLITP13800SQLcoltbler_samraag	292	43	1	65	10	10	10	10
ERBISQLITP14000SQLcoltbler_samraag	282	43	1	65	10	10	10	10
ERBISQLITP14200SQLcoltbler_samraag	272	43	1	65	10	10	10	10
ERBISQLITP14400SQLcoltbler_samraag	262	43	1	65	10	10	10	10
ERBISQLITP14600SQLcoltbler_samraag	252	43	1	65	10	10	10	10
ERBISQLITP14800SQLcoltbler_samraag	242	43	1	65	10	10	10	10
ERBISQLITP15000SQLcoltbler_samraag	232	43	1	65	10	10	10	10
ERBISQLITP15200SQLcoltbler_samraag	222	43	1	65	10	10	10	10
ERBISQLITP15400SQLcoltbler_samraag	212	43	1	65	10	10	10	10
ERBISQLITP15600SQLcoltbler_samraag	202	43	1	65	10	10	10	10
ERBISQLITP15800SQLcoltbler_samraag	192	43	1	65	10	10	10	10
ERBISQLITP16000SQLcoltbler_samraag	182	43	1	65	10	10	10	10
ERBISQLITP16200SQLcoltbler_samraag	172	43	1	65	10	10	10	10
ERBISQLITP16400SQLcoltbler_samraag	162	43	1	65	10	10	10	10
ERBISQLITP16600SQLcoltbler_samraag	152	43	1	65	10	10	10	10
ERBISQLITP16800SQLcoltbler_samraag	142	43	1	65	10	10	10	10
ERBISQLITP17000SQLcoltbler_samraag	132	43	1	65	10	10	10	10
ERBISQLITP17200SQLcoltbler_samraag	122	43	1	65	10	10	10	10
ERBISQLITP17400SQLcoltbler_samraag	112	43	1	65	10	10	10	10
ERBISQLITP17600SQLcoltbler_samraag	102	43	1	65	10	10	10	10
ERBISQLITP17800SQLcoltbler_samraag	92	43	1	65	10	10	10	10
ERBISQLITP18000SQLcoltbler_samraag	82	43	1	65	10	10	10	10
ERBISQLITP18200SQLcoltbler_samraag	72	43	1	65	10	10	10	10
ERBISQLITP18400SQLcoltbler_samraag	62	43	1	65	10	10	10	10
ERBISQLITP18600SQLcoltbler_samraag	52	43	1	65	10	10	10	10
ERBISQLITP18800SQLcoltbler_samraag	42	43	1	65	10	10	10	10
ERBISQLITP19000SQLcoltbler_samraag	32	43	1	65	10	10	10	10
ERBISQLITP19200SQLcoltbler_samraag	22	43	1	65	10	10	10	10
ERBISQLITP19400SQLcoltbler_samraag	12	43	1	65	10	10	10	10
ERBISQLITP19600SQLcoltbler_samraag	2	43	1	65	10	10	10	10

What about behaviour? Performance issues with pension fund's 'Calculation engine'

- **What kind of code do we have?**
 - **complete**
 - **not compilable in analysis environment but executable at the customer's site**
 - **heterogeneous: PL SQL, Cobol with SQL*Plus inside**
 - **large size: ~3000 KLOC of Cobol code only**
 - **abundant sources: Cobol traces, Oracle logs**
- **What kind of model would we like?**
 - **Behavioural to explain why the system is so slow**

What did we do?

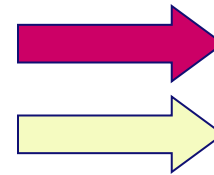
- **Running system \Rightarrow Data**
 - **Scripting & Focused search in Oracle logs**
 - **Cobol code instrumentation to obtain traces**
- **Data \Rightarrow ModelS**
 - **Fact extraction:**
 - **Filtering**
- **Model \Rightarrow Information**
 - **Testing**
 - **Visualisation**
- **Information \Rightarrow Knowledge**
 - **Code review & visualization analysis**

Running system \Rightarrow Data

Environment:

Cobol application and Oracle DBMS run on the same machine under AIX OS

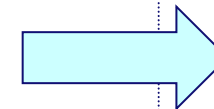
UNIX program 'time' was used to determine execution time for Cobol side



Cobol time

Cobol 'system' time

```
EXEC #5:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568305
FETCH #5:c=0,e=0,p=0,cr=2,cu=0,mis=0,r=1,dep=0,og=3,tim=1112568305
EXEC #153:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568305
FETCH #153:c=0,e=0,p=0,cr=2,cu=0,mis=0,r=1,dep=0,og=3,tim=1112568305
FETCH #153:c=0,e=0,p=0,cr=1,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568305
EXEC #179:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568305
FETCH #179:c=0,e=0,p=0,cr=5,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
EXEC #181:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
FETCH #181:c=0,e=0,p=0,cr=3,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
EXEC #113:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
FETCH #113:c=0,e=0,p=0,cr=4,cu=0,mis=0,r=1,dep=0,og=3,tim=1112568306
EXEC #201:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
FETCH #201:c=0,e=0,p=0,cr=4,cu=0,mis=0,r=1,dep=0,og=3,tim=1112568306
EXEC #5:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
FETCH #5:c=0,e=0,p=0,cr=2,cu=0,mis=0,r=1,dep=0,og=3,tim=1112568306
EXEC #6:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
FETCH #6:c=0,e=0,p=0,cr=4,cu=0,mis=0,r=1,dep=0,og=3,tim=1112568306
EXEC #7:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=3,tim=1112568306
```

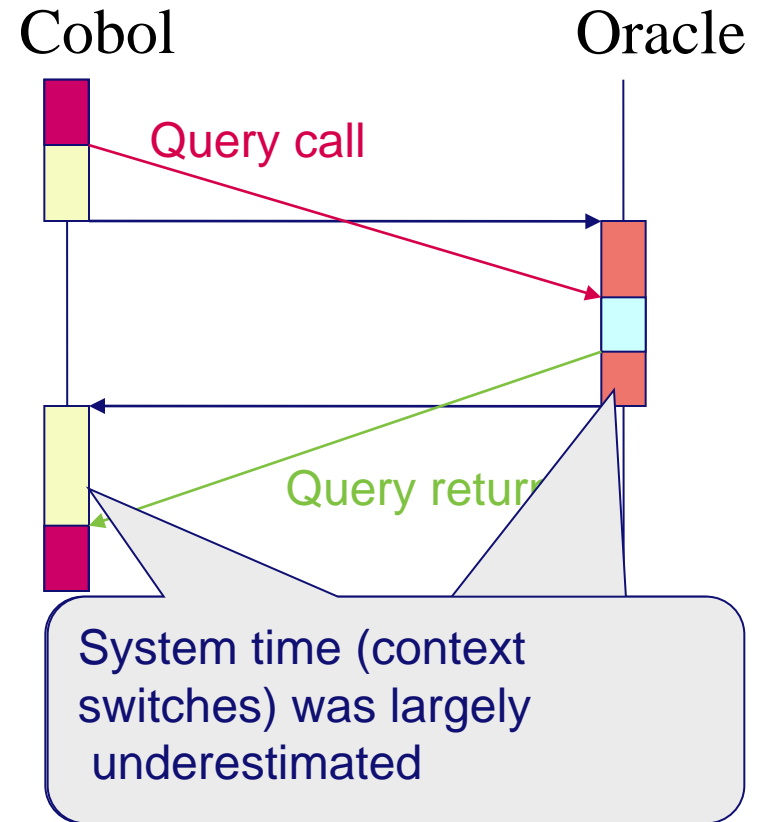


Oracle time

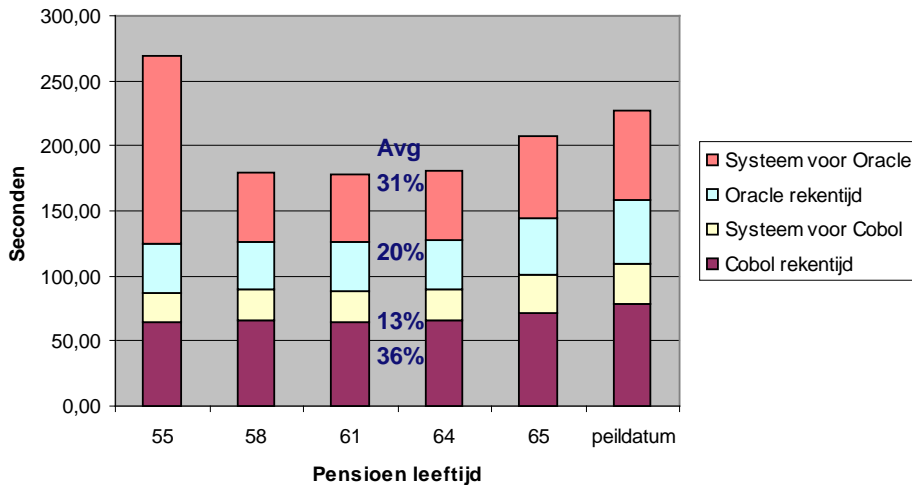
Data ⇒ Communication model

Oracle logs show very moderate execution time. **Where is the rest?**

1. Cobol time: pension calculation, query parameters/return processing ■
2. System time for Cobol: Communication with Oracle, disk usage ■
3. System time for Oracle: communication with Cobol, disk usage ■
4. Oracle time for the Cobol application: search in the database ■

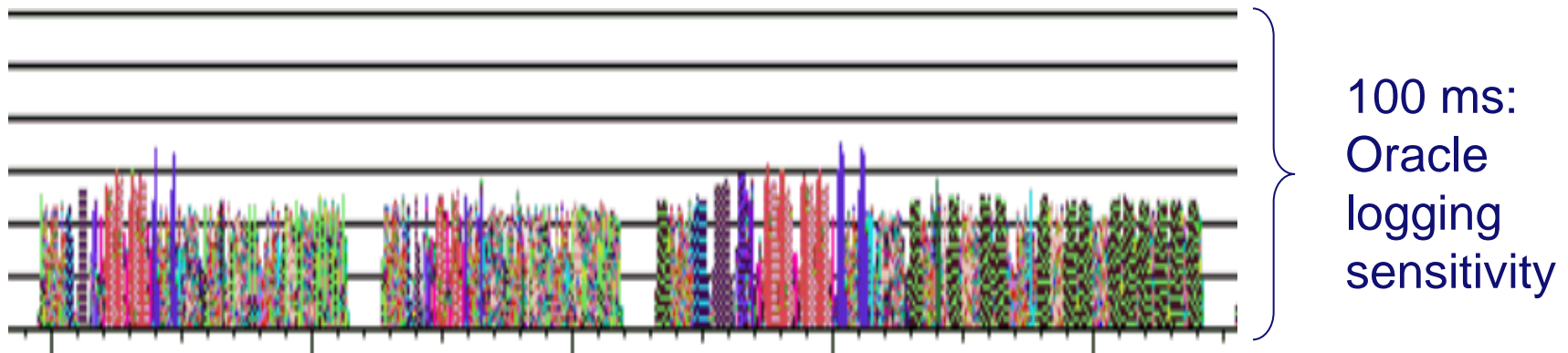


Tijdverdeling tussen Cobol en Oracle



Data \Rightarrow Performance model

Which queries are most time consuming ?



Different colours – different types of queries

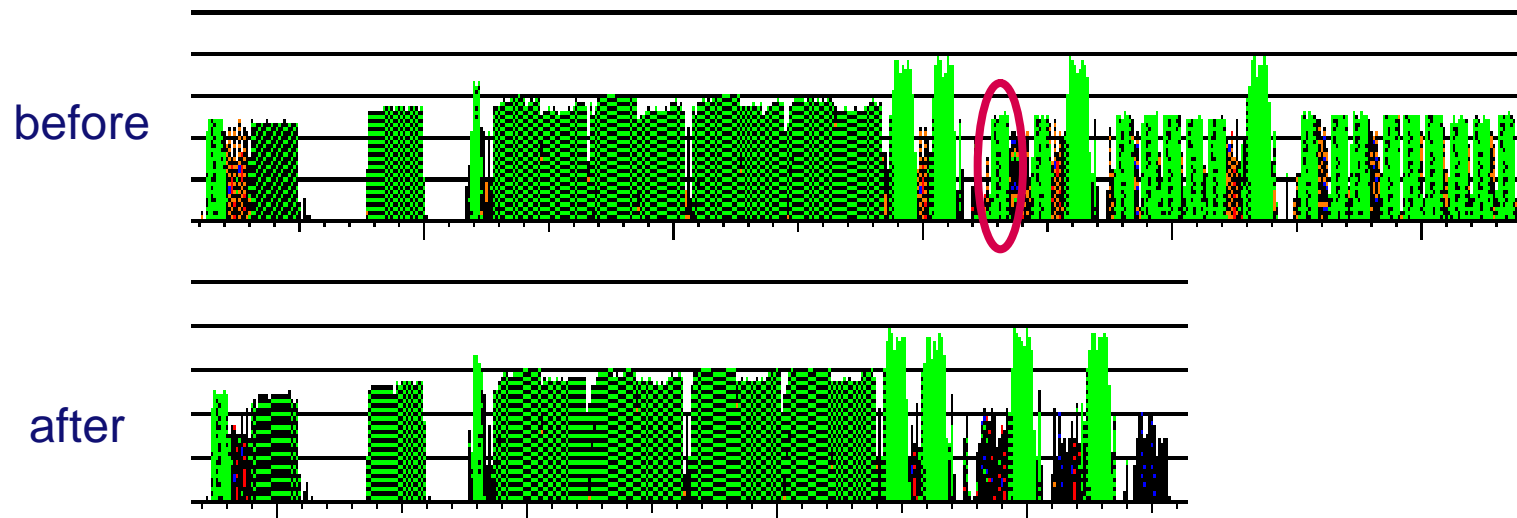
- Types can be chosen on the fly
- **Adaptable** model (or a class of models)

Information \Rightarrow Knowledge

- Get rid of parameter 'up-to-datedness' control

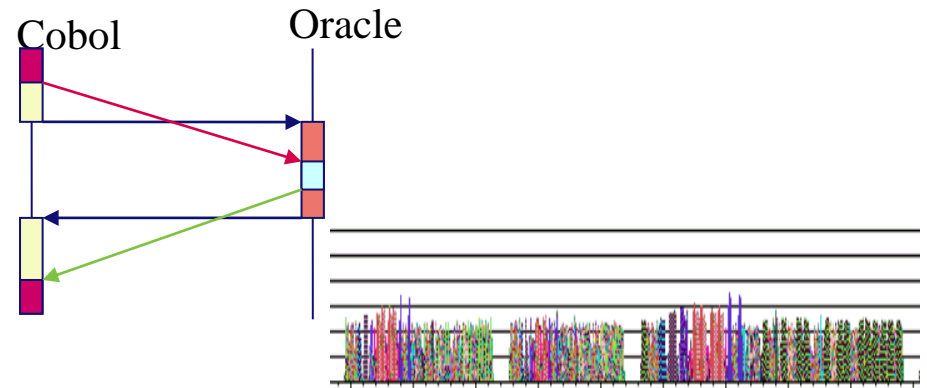


- Make use of Cobol 'static memory'



Case conclusions

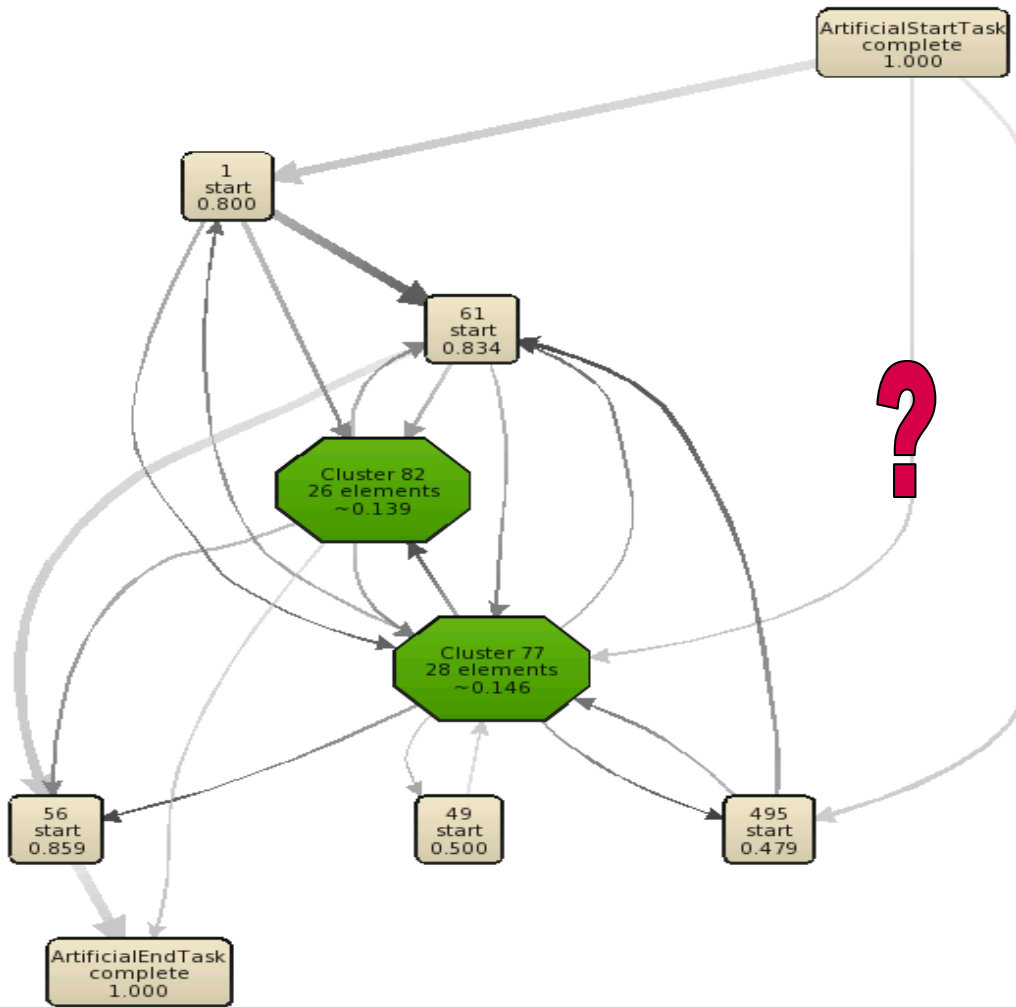
- Behavioural model
- Data is obtained
 - From a running system
 - By **different** means
- **Multiple** models
 - Communication
 - Performance



Industrial case: Certificate issuing

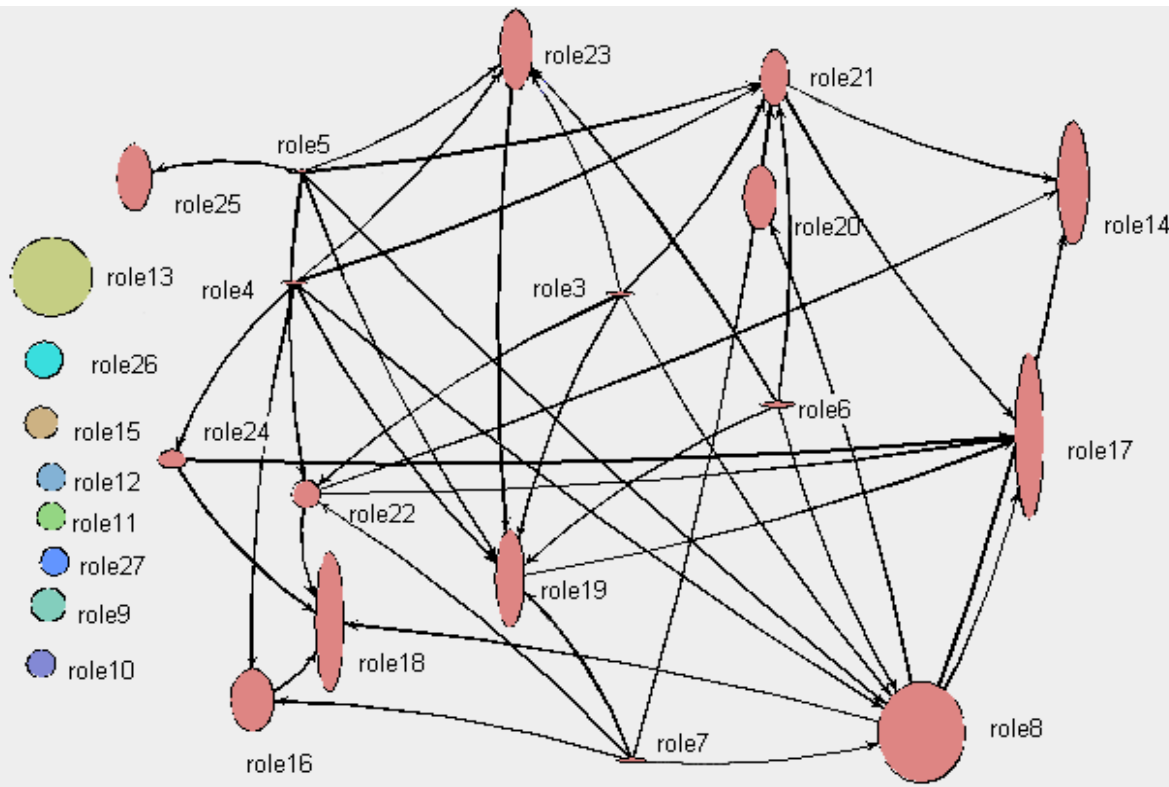
- **What kind of code do we have?**
 - **Workflow system log**
 - **Context:**
 - **Certificates are requested**
 - **Data is analysed**
 - **Certificates are granted (or not)**
- **What kind of models would we like?**
 - **Process models: granted certificates only**
 - **Task transfer model**
 - **Performance model**

Process Model



- Thickness of arrows shows frequency
- Thin lines = anomalies (?)
- Closely interrelated tasks are clustered

Task transfer



- Height: incoming arcs
- Width: outgoing arcs
- 3-7: flat
 - Process initiators
- 14: tall
 - Process finalisers
- 13, 26: disconnected
 - Incidental participants
- 8: many in/out-arcs
 - Process facilitators

Performance Model

Pattern	Occurrences (#, %)		Average (h)	Min (h)	Max (h)	St. dev (h)
1 - 61 - 56	41422	49.5	255.14	8.52	1725.63	104.82
495 - 61 - 56	10250	12.3	22.82	6.24	96.69	12.64
1 - 44 - 61 - 56	4546	5.4	253.29	10.00	879.77	114.91
1 - 61 - 24 - 56	3495	4.2	346.05	49.81	948.34	123.58
1 - 61 - 24 - 45 - 56	3101	3.7	381.79	90.97	1132.73	125.67
1 - 61 - 71 - 56	1086	1.3	397.86	73.00	701.25	146.02
495 - 61 - 24 - 56	701	0.8	119.65	18.92	233.29	46.92
495 - 44 - 61 - 56	429	0.5	197.49	12.12	368.49	60.17
1 - 44 - 61 - 24 - 45 - 56	421	0.5	377.24	126.25	926.75	127.56
1 - 44 - 61 - 24 - 56	343	0.4	350.92	87.36	911.38	127.91
495 - 61 - 71 - 56	264	0.3	80.78	72.20	123.06	8.83
1 - 61 - 71 - 24 - 56	154	0.2	432.58	117.87	646.09	132.53
495 - 61 - 24 - 45 - 56	138	0.2	159.37	31.07	300.36	53.44
1 - 61 - 71 - 24 - 45 - 56	121	0.1	450.98	122.12	677.39	122.97
1 - 44 - 61 - 71 - 56	111	0.1	427.23	75.75	578.13	114.14
Total	66582	79.6				

- Throughput times of traces

Case conclusions

- From **detailed** log files we can extract information
 - Process model
 - Task transfer model
 - Performance model
- Models **beyond** the software: organizational context!
- Answering the question:
 - Does my company actually work the way I thought?

Conclusions

- **Reverse engineering = getting models from existing system**
- **Models are useful if they give additional knowledge about software system**
- **The choice of models depends on the task in hand (the knowledge we want to obtain)**
- **Visualisation is important BUT**
- **Numbers really matter**