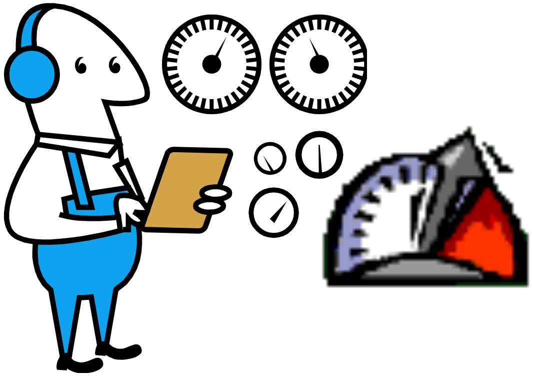
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Reverse Architecting and Design-Code Conformance

Truong Ho-Quang
truongh@chalmers.se



Schedule

Week		Date	Time	Lecture	Note
3	L1	Wed, 20 Jan	10:15 – 12:00	Introduction & Organization	Truong Ho
3	L2	Thu, 21 Jan	13:15 – 15:00	Architecting Process & Views	Truong Ho
4		Tue, 26 Jan	10:15 – 12:00	**Skip**	
4	S1	Wed, 27 Jan	10:15 – 12:00	<< Supervision: Launch Assignment 1>>	TAs
4	L3	Thu, 28 Jan	13:15 - 15:00	Roles/Responsibilities & Functional Decomposition	Truong Ho
5	L4	Mon, 1 Feb	13:15 – 15:00	Architectural Styles P1	Truong Ho
5	S2	Wed, 3 Jan	10:15 – 12:00	<< Supervision/Ass	
5	L5	Thu, 4 Jan	13:15 – 15:00	Architectural Styles P2	ara
6	L6	Mon, 8 Feb	13:15 – 15:00	Architectural Styles P3	Ho
6	S3	Wed, 10 Feb	10:15 – 12:00	<< Supervision/Ass	
6	L7	Thu, 11 Feb	13:15 – 15:00	Design Principles (Maintainability, Modifia	Ho
7	L8	Mon, 15 Feb	13:15 – 15:00	Architectural Tactics & Analysis	Truong Ho
7	S4	Wed, 17 Feb	10:15 – 12:00	<< Supervision/Assignr nt>>	TAs
7	L9	Thu, 18 Feb	13:15 – 15:00	Architecture Evaluation	Truong Ho
8	L10	Mon, 22 Feb	13:15 – 15:00	Reverse Engineering & Correspondence	Truong Ho
8	S5	Wed, 24 Feb	10:15 – 12:00	<< Supervision/Assignment>>	TAs
8	L11	Thu, 25 Feb	13:15 – 15:00	Guest Lecture 1	TBD
9	L12	Mon, 1 Mar	13:15 – 15:00	Guest Lecture 2: Architectural Changes in Volvo AB	Anders M.
9	S6	Wed, 3 Mar	10:15 – 12:00	<< Supervision/Assignment>>	TAs
9	L13	Thu, 4 Mar	13:15 – 15:00	To be determined (exam practice?)	Truong Ho
9		Fri, 5 Mar	Whole day	Group presentation of Assignment (TBD)	Teachers
11	Exam	Thu, 18 Mar	AM		

We are HERE!

Asignment schedule

Week		Date	Lecture	Assignment 1 – Task 1 (A1T1)	Assignment 1 – Task 2 (A1T2)	Assignment 2 (A2)
3	L1	Wed, 20 Jan	10:15 – 12:00			
3	L2	Thu, 21 Jan	13:15 – 15:00			
4		Tue, 26 Jan	10:15 – 12:00			
4	S1	Wed, 27 Jan	10:15 – 12:00	Launch A1T1		
4	L3	Thu, 28 Jan	13:15 - 15:00			
5	L4	Mon, 1 Feb	10:15 – 12:00			
5	S2	Wed, 3 Jan	10:15 – 12:00	Work A1T1		
5	L5	Thu, 4 Jan	13:15 – 15:0			
6	L6	Mon, 8 Feb	10:15 – 12:0			
6	S3	Wed, 10 Feb	13:15 – 15:0	Work A1T1		
6	L7	Thu, 11 Feb	13:15 – 15:0	Hand-in A1T1 Peer Rev A1T1		
7	L8	Mon, 15 Feb	10:15 – 12:0			
7	S4	Wed, 17 Feb	13:15 – 15:0	Hand-in PR A1T1	A1T2 released MQTT intro	~~A2 released~~
7	L9	Thu, 18 Feb	10:15 – 12:0			
8	L10	Mon, 22 Feb	13:15 – 15:00			
8	S5	Wed, 24 Feb	13:15 – 15:00		Work A1T2	A2 released
8	L11	Thu, 25 Feb	10:15 – 12:00			
9	L12	Mon, 1 Mar	13:15 – 15:00			
9	S6	Wed, 3 Mar	10:15 – 12:00		Work A1T2	Hand-in A2
9	L13	Thu, 4 Mar	13:15 – 15:00			
9		Fri, 5 Mar	Whole day		Present A1T2	
10					Hand-in A1T2	Hand-in A2
11	Exam	Thu, 18 Mar				

We are HERE!

Online Written Exam

18th of March in the AM/Morning

Outline

- Reverse Architecting –

  based on slides by prof. Arie van Deursen,
  TU Delft, Netherlands

- Monitoring Implementation-Design conformance

  includes slides by Reinder Bril,
  TU Eindhoven, Netherlands

Reverse Architecting: Motivation

- Architecture description lost or outdated
- Obtain advantages of explicit architecture:
  - Shared representation of system
  - Stakeholder communication
  - Explicit design decisions
- Architecture conformance checking
- Quality attribute analysis

Program Understanding

- the task of building mental models of an underlying software system
- at various abstraction levels, ranging from
  - models of the code itself to
  - ones of the underlying application domain,
- for software maintenance, evolution, and reengineering purposes

~50% of maintenance effort!!

Architecture erosion

[...] the documentation about the internal architecture ***becomes rapidly obsolete****. To make changes, developers* ***need a clear understanding of the underlying architecture*** *of the products.*

C. Riva,
Software Architecture Group, Nokia Research

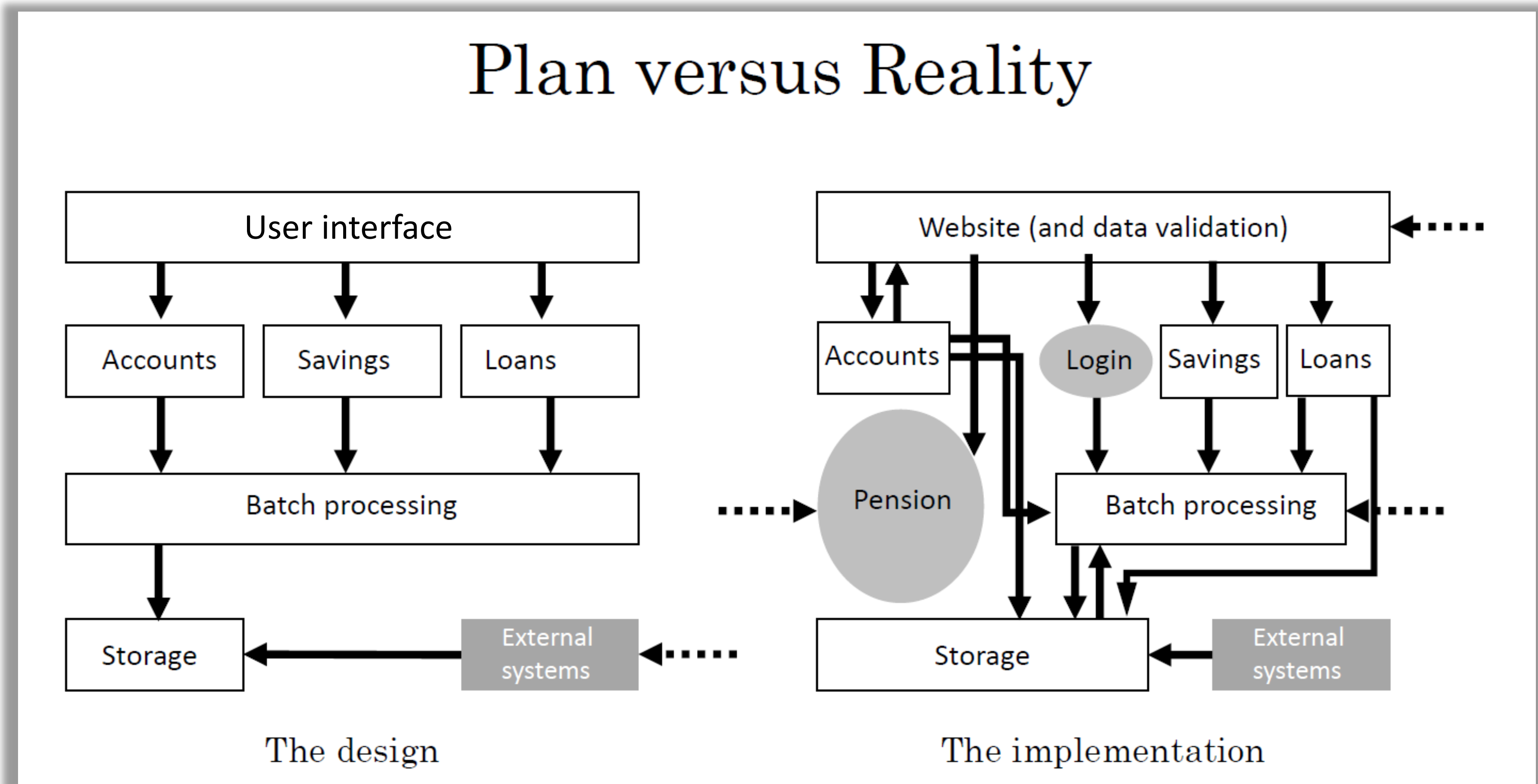
Architecture Evolution

- Architectural drift
- Architectural erosion
- *Architectural upgrade*

One frequently accompanying property of **evolution** is an **increasing brittleness** of the system -- that is, an **increasing resistance to change**, or at least to changing gracefully.

[Perry & Wolf'92]

Architecture
Plan vs Reality ('as-is')



Problems with Engineering Documentation

Difficult to:

- find information
  due to: - large size & complexity
  - scattering of information

- Keep (check) ‘up-to-date’

- To cater for multiple audiences
  tasks, experience, ...

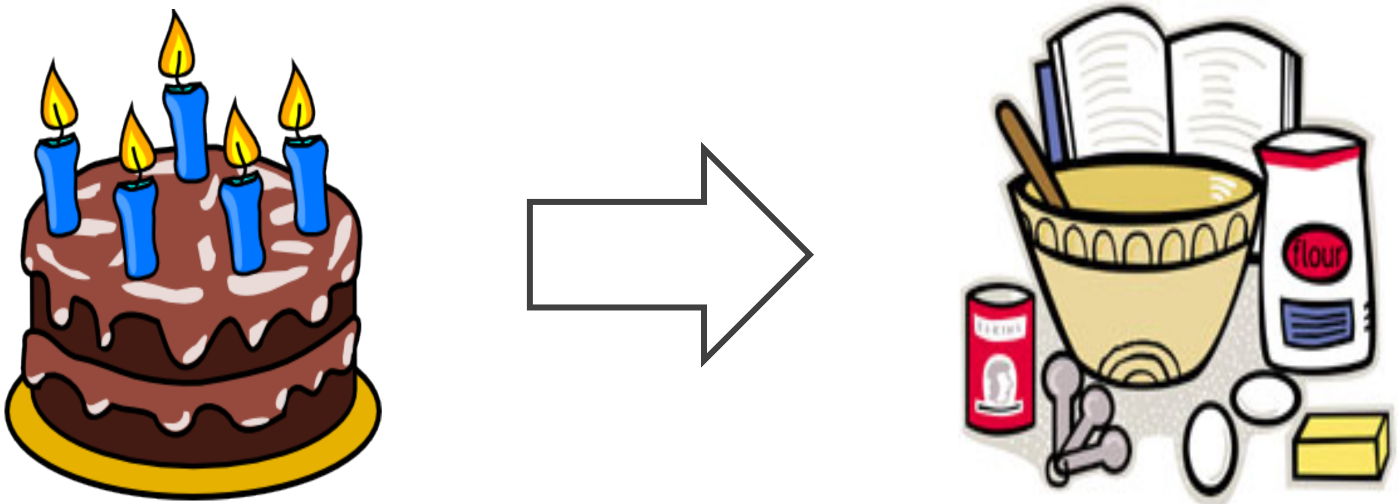
How can a developer get the latest information he needs for his current task?

Reverse Engineering

The process of analyzing a subject system with two goals in mind:

- to identify the system's components and their interrelationships; and,
- to create representations of the system in another form or at a higher level of abstraction.

Reverse Engineering (analogy)

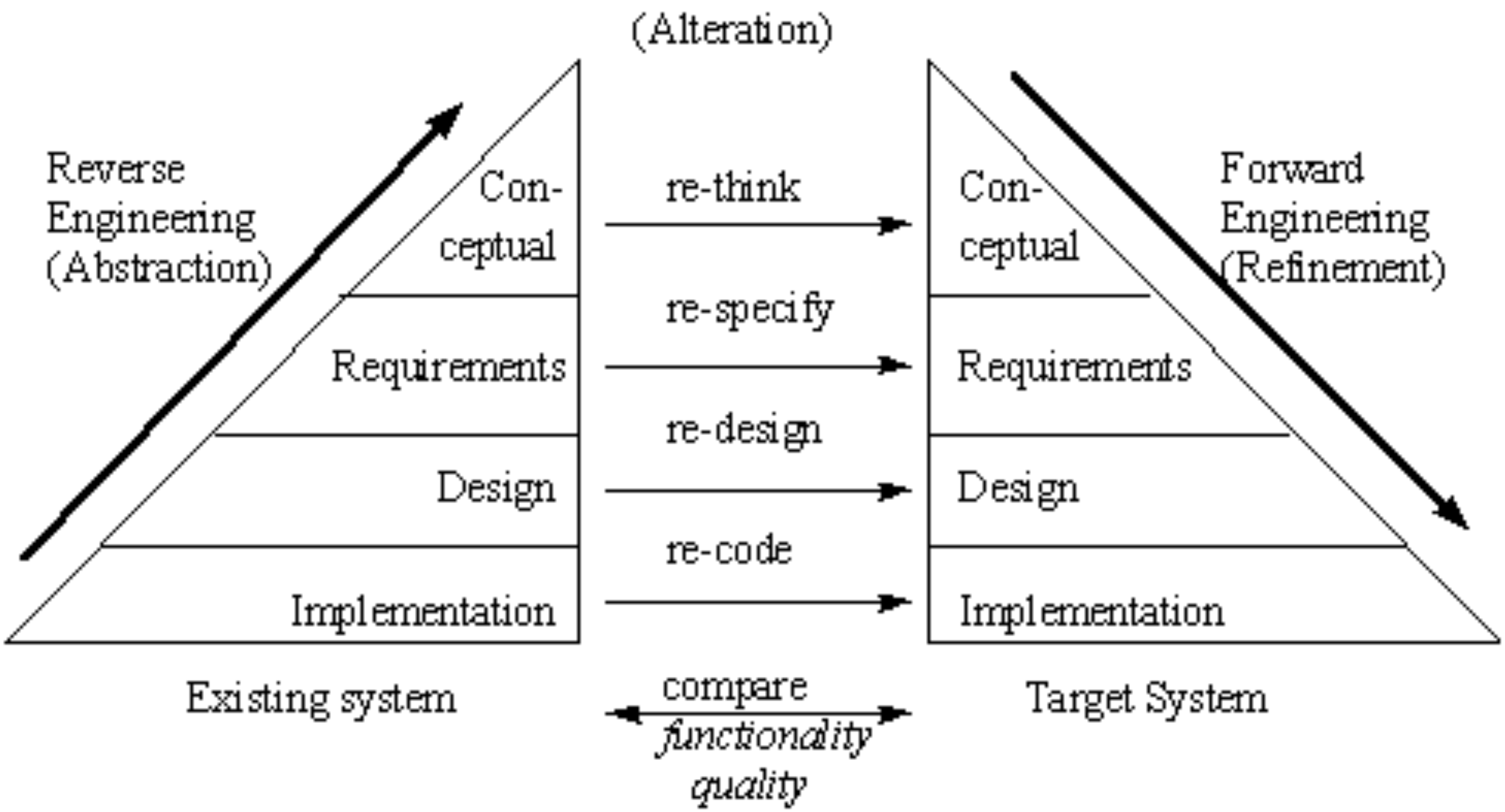


Re-engineering

- The examination and alteration of a subject system
- to reconstitute it in a new form
- and the subsequent implementation of that new form

*Beyond analysis -- actually **improve**.*

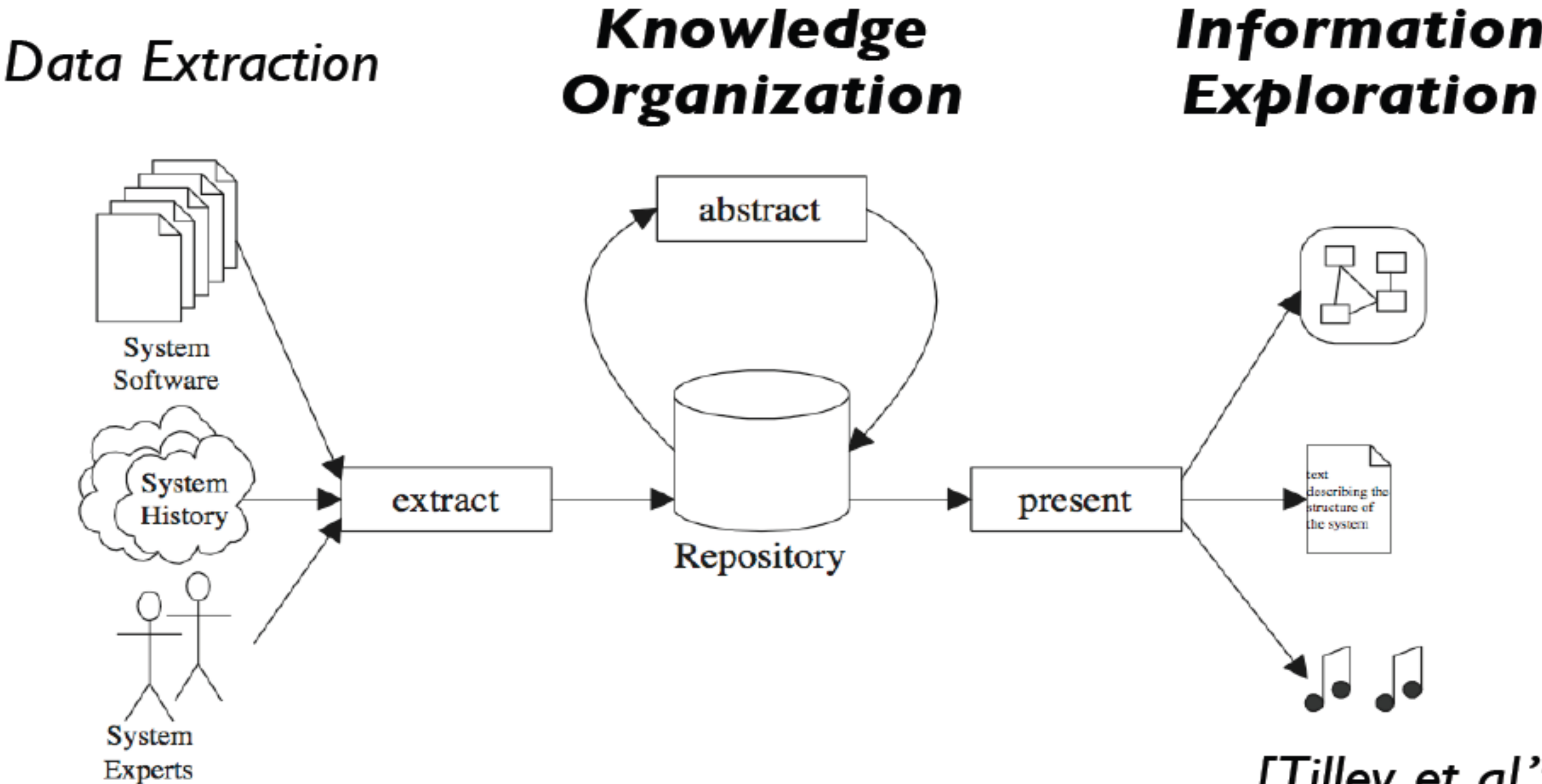
Reengineering



Challenges

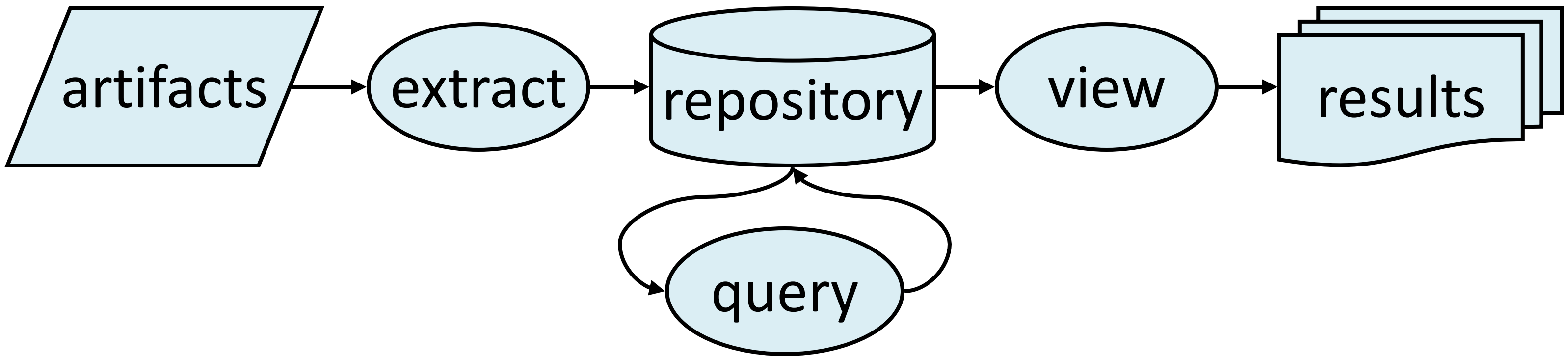
- What process can support uncovering the software architecture within a system?
- How much can you automate in this process?
- What are the limits of architecture recovery? (e.g., Recovering *all design decisions*).

Phases of Reconstruction



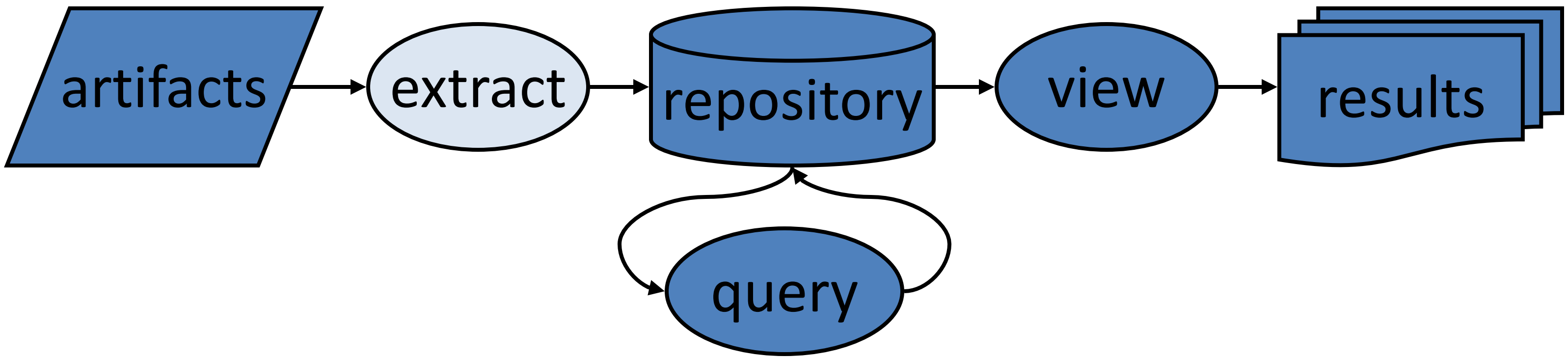
[Tilley et al.'96]

Reverse Engineering: Exploration



- Extract src models from system artifacts
- Query/manipulate to infer new knowledge
- Present different views on results

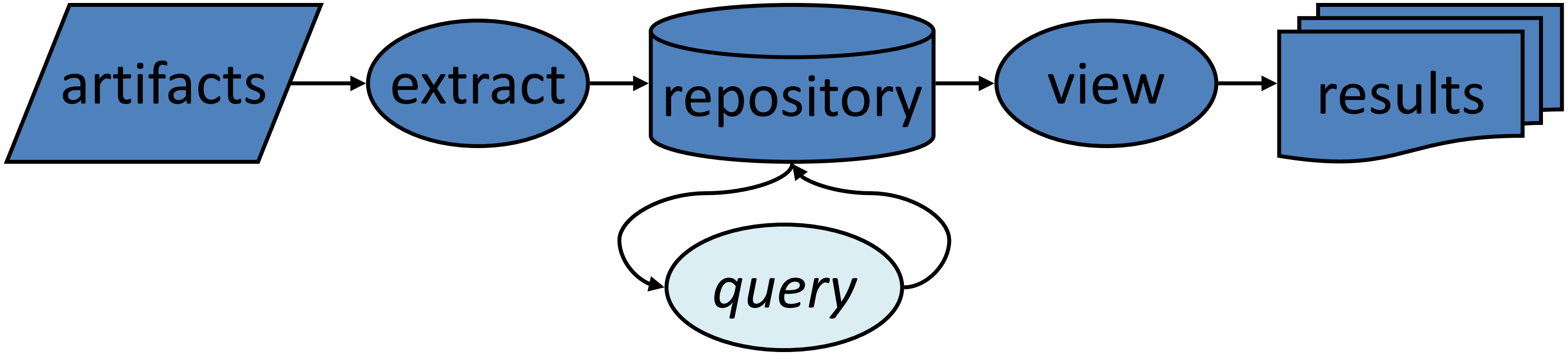
Source Model Extraction



Source Model Extraction

- Derive information from system artifacts
  - variable usage, call graphs, file dependencies, database access, …
- Challenges
  - *Accurate* & *complete* results
  - *Flexible*: easy to write and adapt
  - *Robust*: deal with irregularities in input

Query and Manipulate



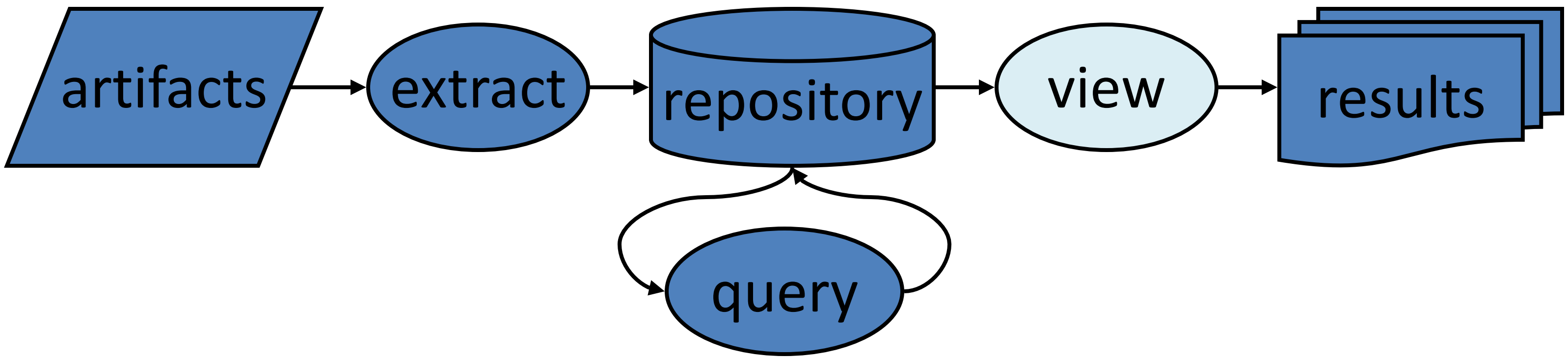
Query and Manipulate

- Goals:
  - *infer* (new) knowledge & abstractions
  - *filter* information

- Example structures:
  - Perform graph
  - Call graph (OI, PVL)
  - Screen flow
  - Batch job
  - Subsystem dbs

In search for more abstraction

Presentation of Results



Presentation Desiderata

- Browsing and searching
- Multiple levels of abstraction
  - Zoom in, zoom out
- Visual as well as textual information
  - Graph visualization
- Show multiple structures
  - E.g. Package hierarchy + control-flow

#3: Results

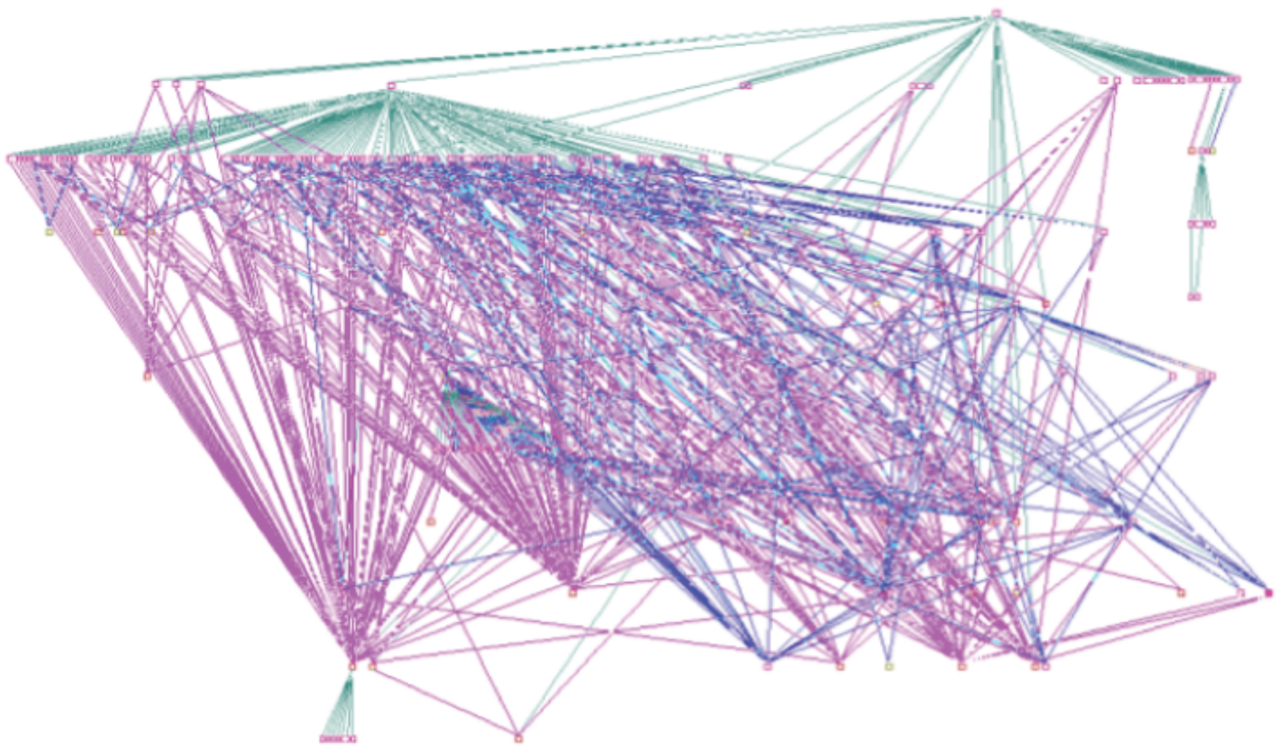
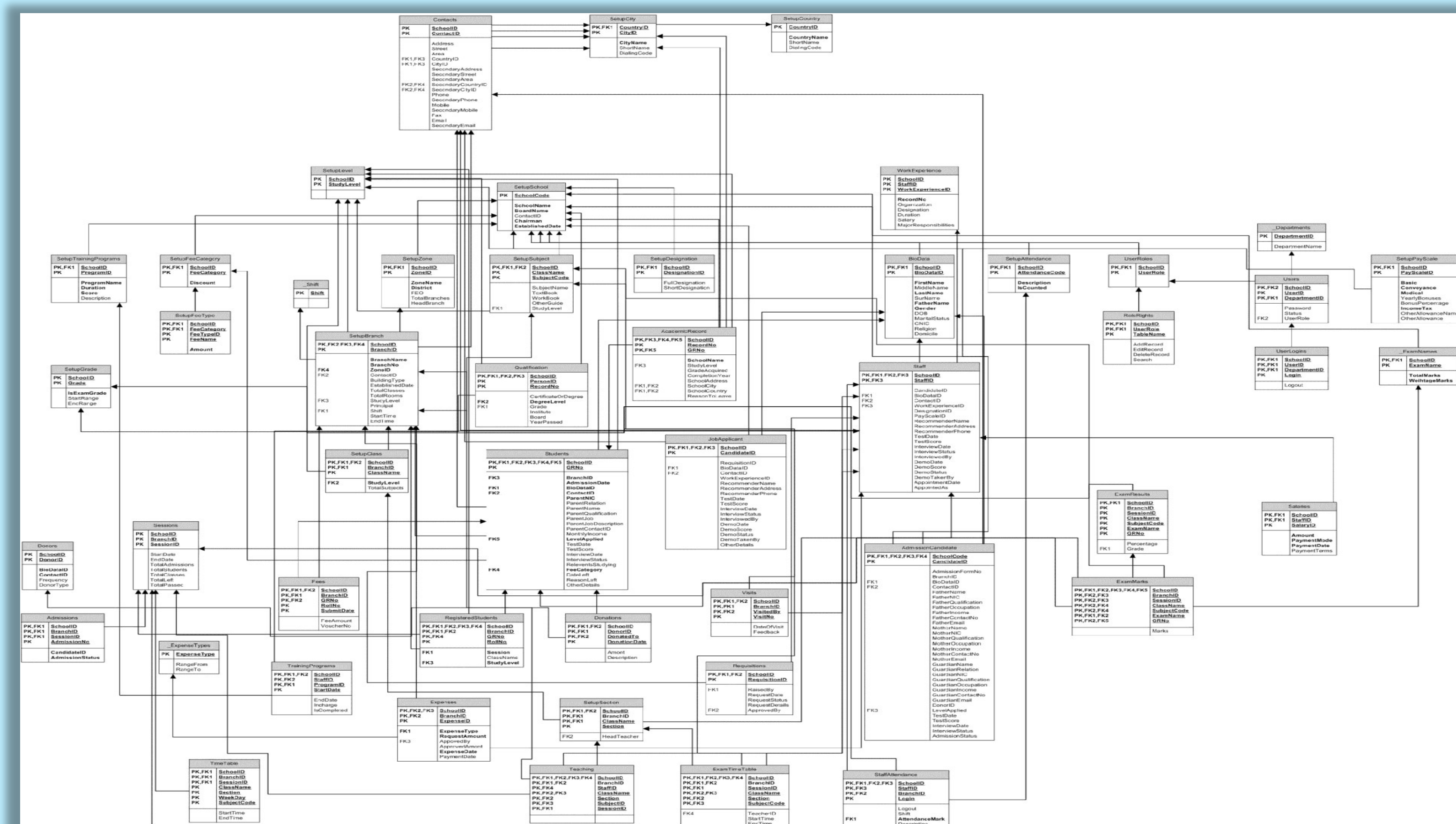


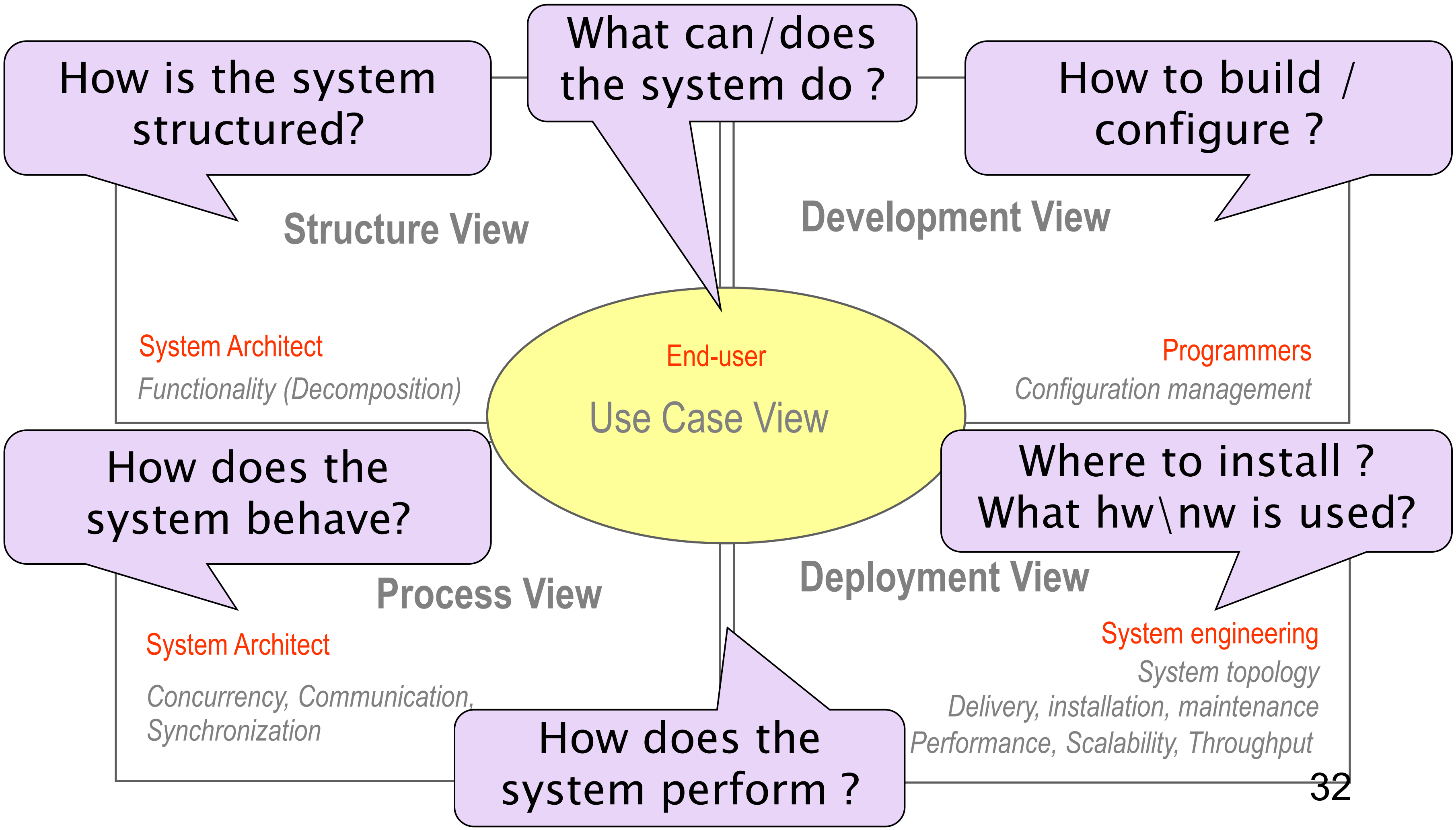
Figure 1. The graph of the source code model.

Reverse Engineering of a small system



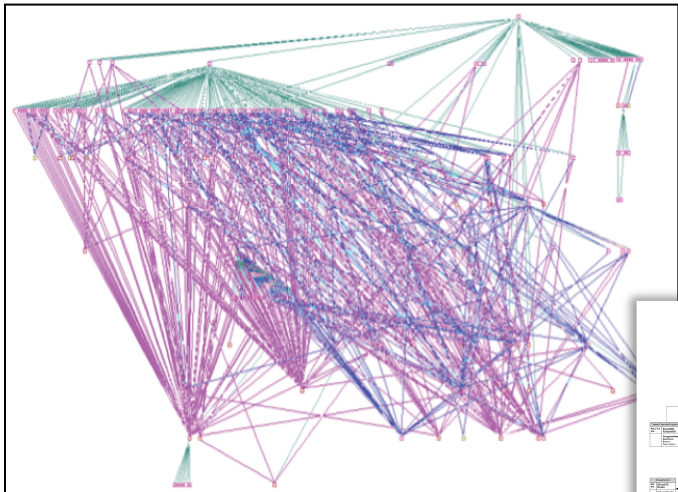
Clearly different from forward designed UML designs
(o.a. in size, layout, detail, naming, ….)

Recall: 4+1 Views Representation of Systems

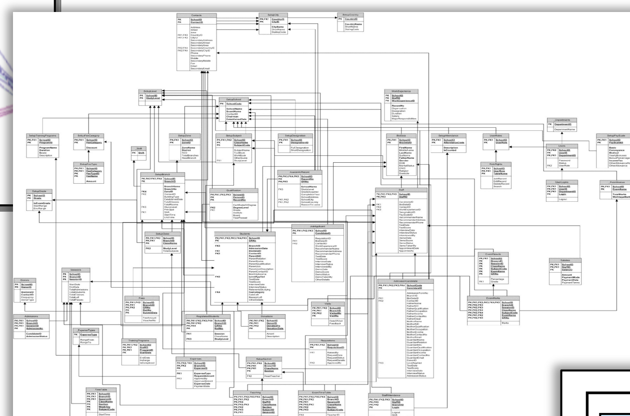


Idea 1

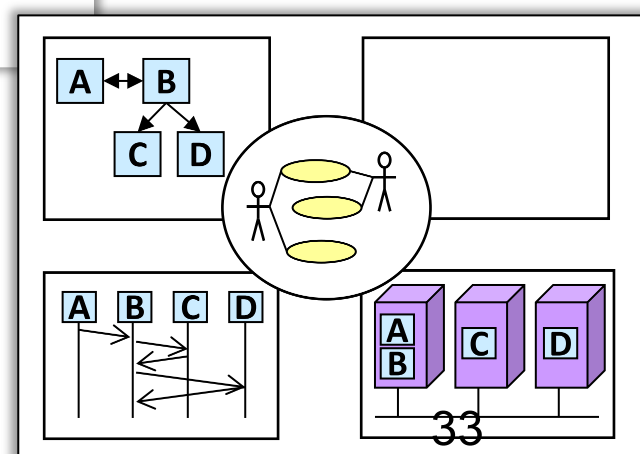
source code



classes



views



Interesting Structures of Software System

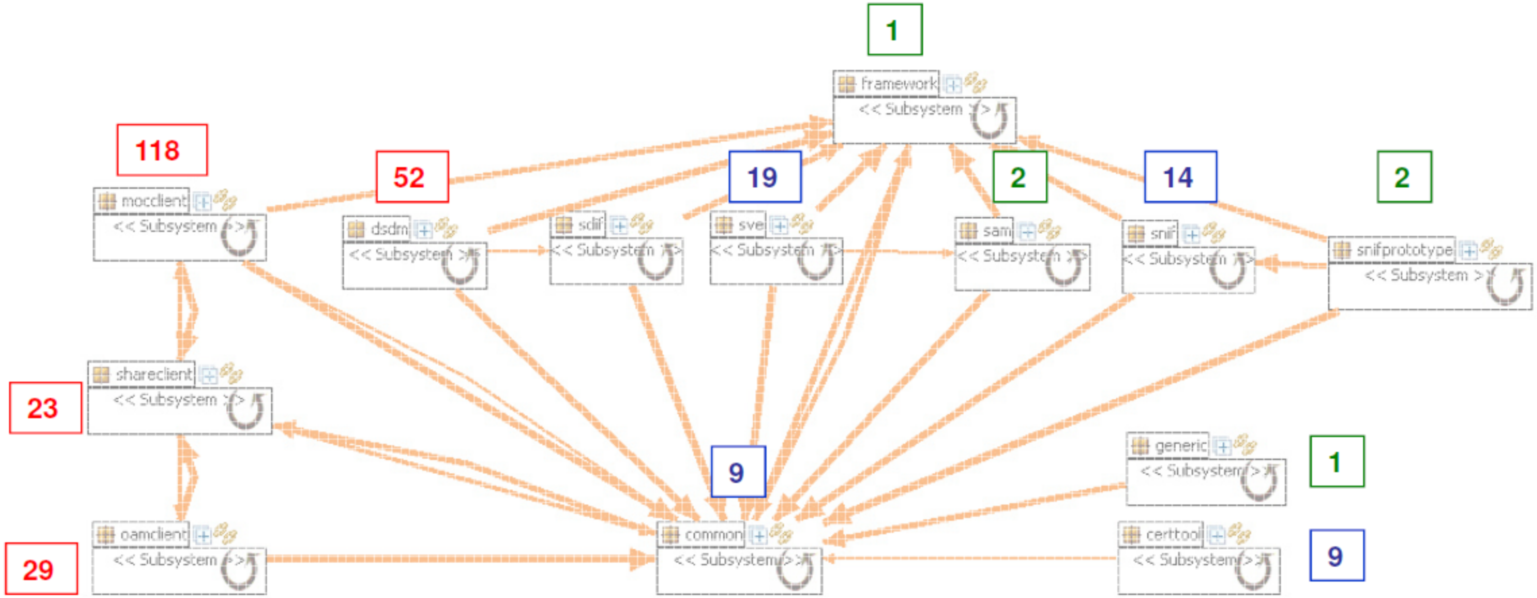
- Module structure
  - Modules & dependencies
  - Layering
  - Hierarchy

- Data model structure
  - Type structure

Behaviour is also Interesting!

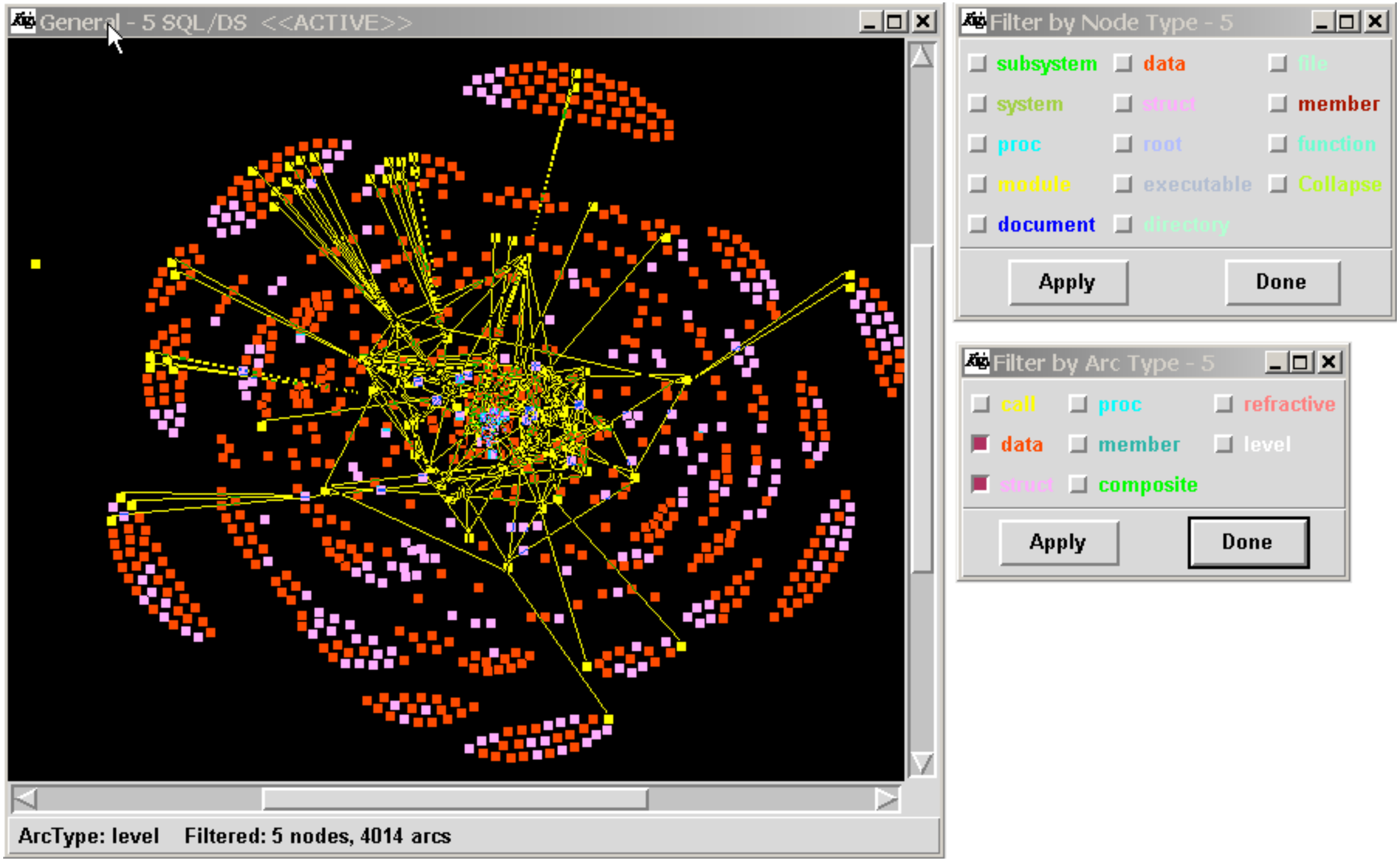
- Call structure
- Process structure
- GUI flow
- ...

Combine Architecture with other Development metrics: High Priority Bugs



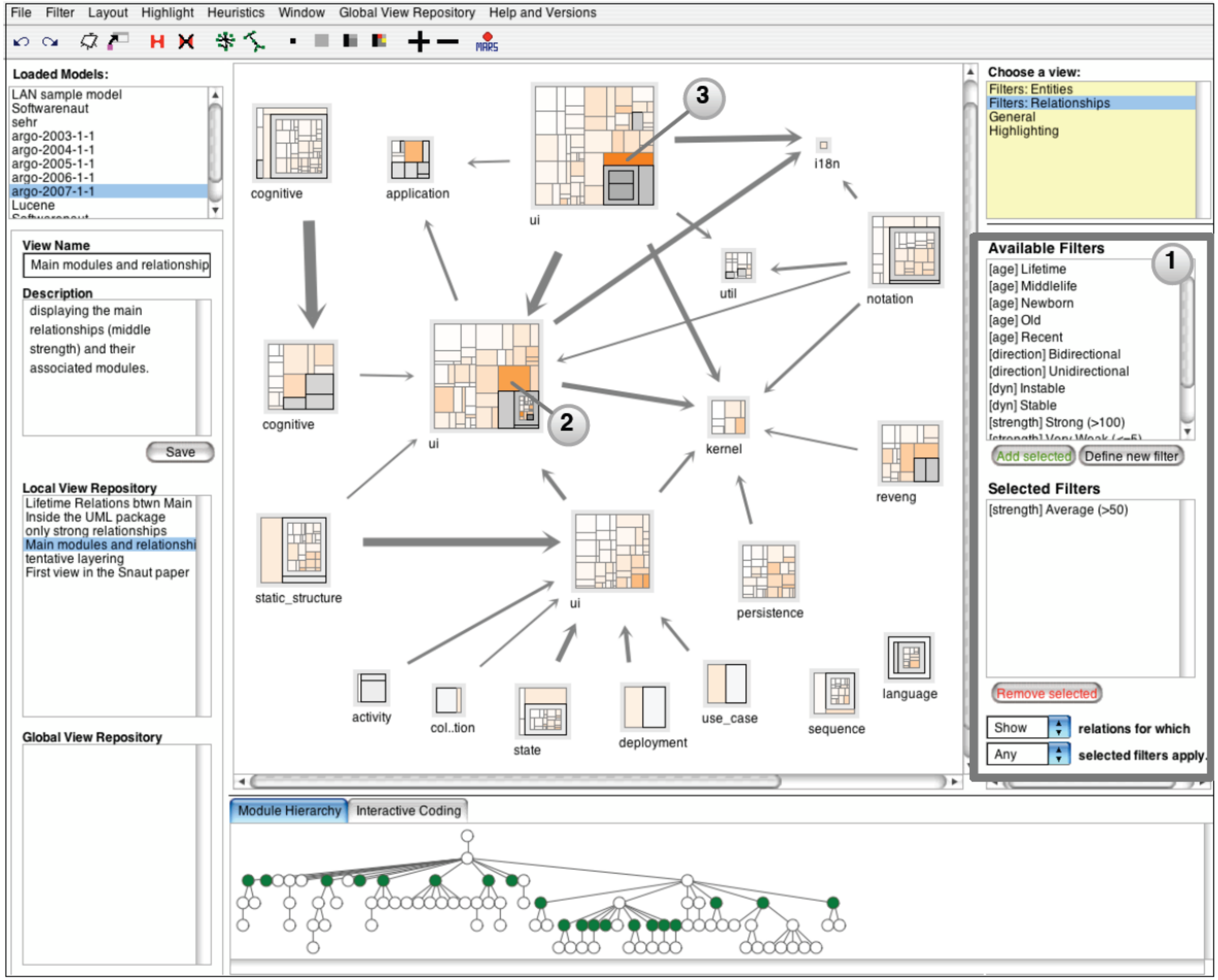
- Number of high priority bugs for each high level component
  - mocclient is the most buggy package with 118 bugs
  - dsdm, shareclient, and oamclient also contain many highly severe bugs

Rigi tool



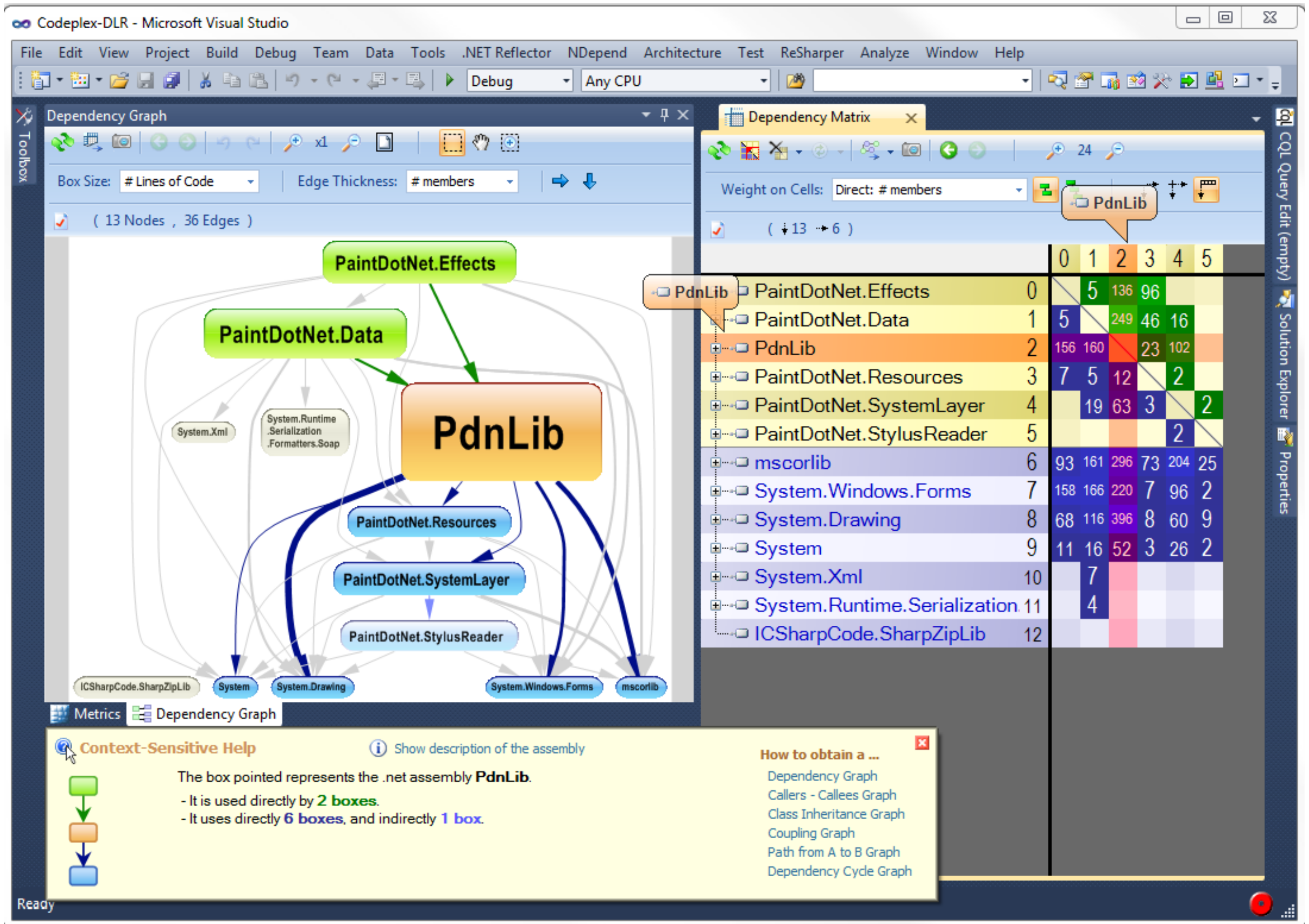
http://www.svgopen.org/2002/papers/kienle_weber_mueller__rigi_reverse_engineering/

SoftwareNaut

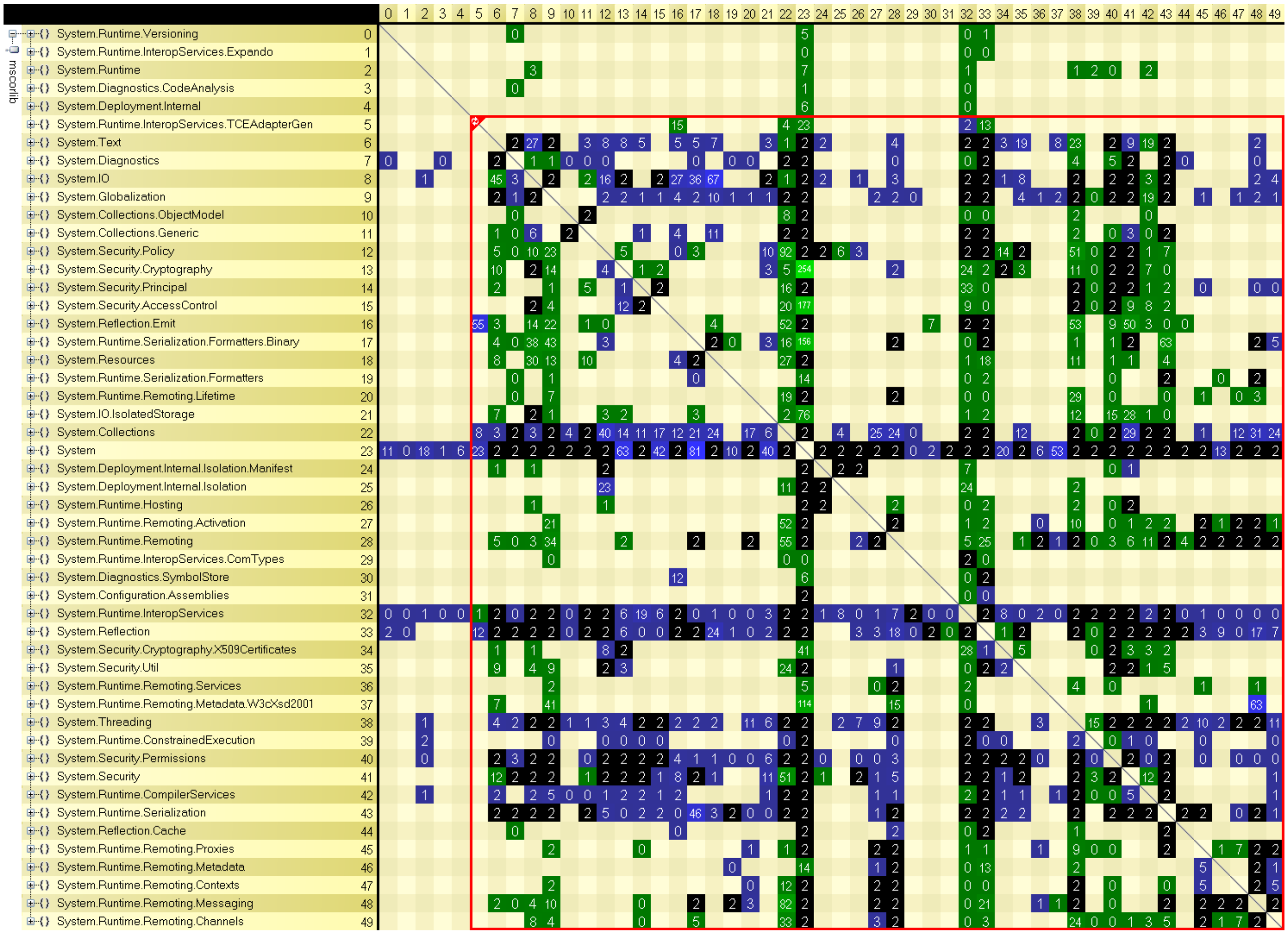


- Mircea Lungu, Michele Lanza, and Oscar Nierstrasz. Evolutionary and Collaborative Software Architecture Recovery with Softwarenaut. In Science of Computer Programming (SCP), 2012. DOI

NDepend


Codeplex-DLR - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools .NET Reflector NDepend Architecture Test ReSharper Analyze Window Help
Debug
Any CPU
Dependency Graph
Box Size: # Lines of Code
Edge Thickness: # members
(13 Nodes , 36 Edges)
PaintDotNet.Effects
PaintDotNet.Data
PdnLib
System.Xml
PaintDotNet.Resources
PaintDotNet.SystemLayer
PaintDotNet.StylusReader
ICSharpCode.SharpZipLib
System
System.Drawing
System.Windows.Forms
mscorlib
Metrics
Dependency Graph
Dependency Matrix
Weight on Cells: Direct: # members
PdnLib
PaintDotNet.Effects
PaintDotNet.Data
PdnLib
PaintDotNet.Resources
PaintDotNet.SystemLayer
PaintDotNet.StylusReader
mscorlib
System.Windows.Forms
System.Drawing
System
System.Xml
System.Runtime.Serialization
ICSharpCode.SharpZipLib
Context-Sensitive Help
Show description of the assembly
The box pointed represents the .net assembly PdnLib.
- It is used directly by 2 boxes.
- It uses directly 6 boxes, and indirectly 1 box.
How to obtain a ...
Dependency Graph
Callers - Callees Graph
Class Inheritance Graph
Coupling Graph
Path from A to B Graph
Dependency Cycle Graph
Ready
CQL Query Edit (empty)
Solution Explorer
Properties
Toolbox

NDepend



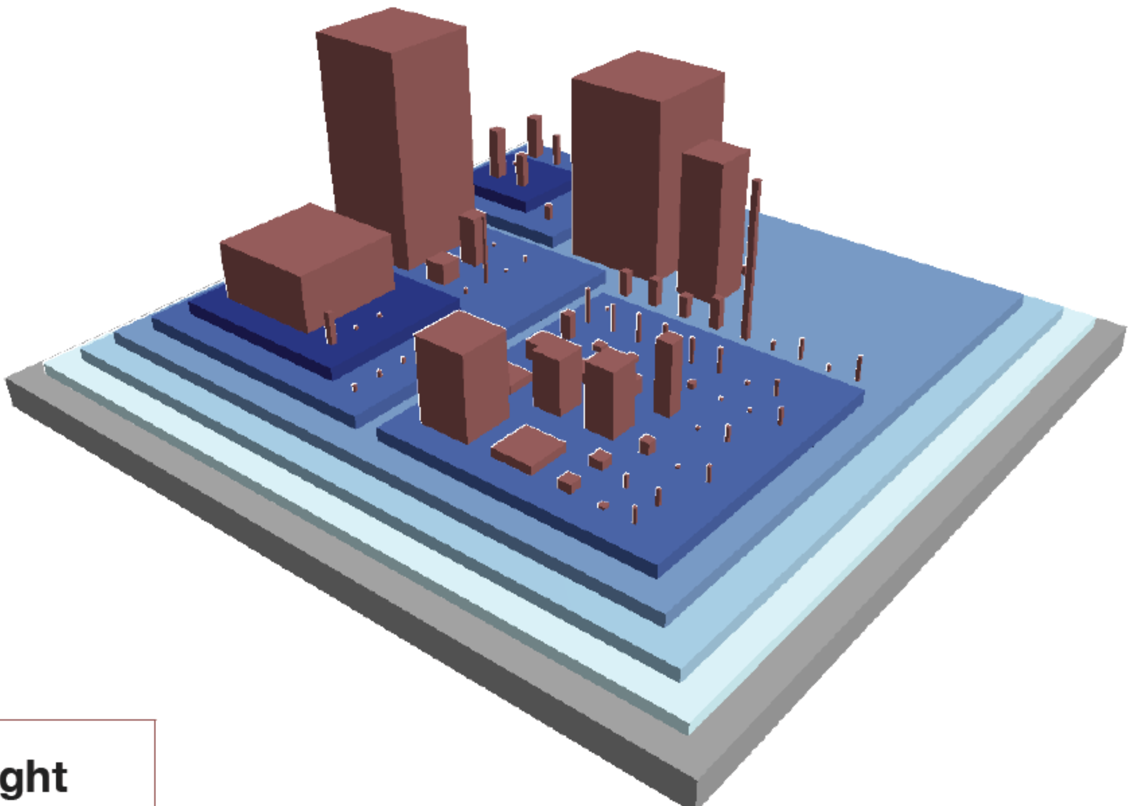
mscorlib
System.Runtime.Versioning 0
System.Runtime.InteropServices.Expando 1
System.Runtime 2
System.Diagnostics.CodeAnalysis 3
System.Deployment.Internal 4
System.Runtime.InteropServices.TCEAdapterGen 5
System.Text 6
System.Diagnostics 7
System.IO 8
System.Globalization 9
System.Collections.ObjectModel 10
System.Collections.Generic 11
System.Security.Policy 12
System.Security.Cryptography 13
System.Security.Principal 14
System.Security.AccessControl 15
System.Reflection.Emit 16
System.Runtime.Serialization.Formatters.Binary 17
System.Resources 18
System.Runtime.Serialization.Formatters 19
System.Runtime.Remoting.Lifetime 20
System.IO.IsolatedStorage 21
System.Collections 22
System 23
System.Deployment.Internal.Isolation.Manifest 24
System.Deployment.Internal.Isolation 25
System.Runtime.Hosting 26
System.Runtime.Remoting.Activation 27
System.Runtime.Remoting 28
System.Runtime.InteropServices.ComTypes 29
System.Diagnostics.SymbolStore 30
System.Configuration.Assemblies 31
System.Runtime.InteropServices 32
System.Reflection 33
System.Security.Cryptography.X509Certificates 34
System.Security.Util 35
System.Runtime.Remoting.Services 36
System.Runtime.Remoting.Metadata.W3cXsd2001 37
System.Threading 38
System.Runtime.ConstrainedExecution 39
System.Security.Permissions 40
System.Security 41
System.Runtime.CompilerServices 42
System.Runtime.Serialization 43
System.Reflection.Cache 44
System.Runtime.Remoting.Proxies 45
System.Runtime.Remoting.Metadata 46
System.Runtime.Remoting.Contexts 47
System.Runtime.Remoting.Messaging 48
System.Runtime.Remoting.Channels 49

The city metaphor

domain mapping	
class	building
package	district
system	city

nesting level	color

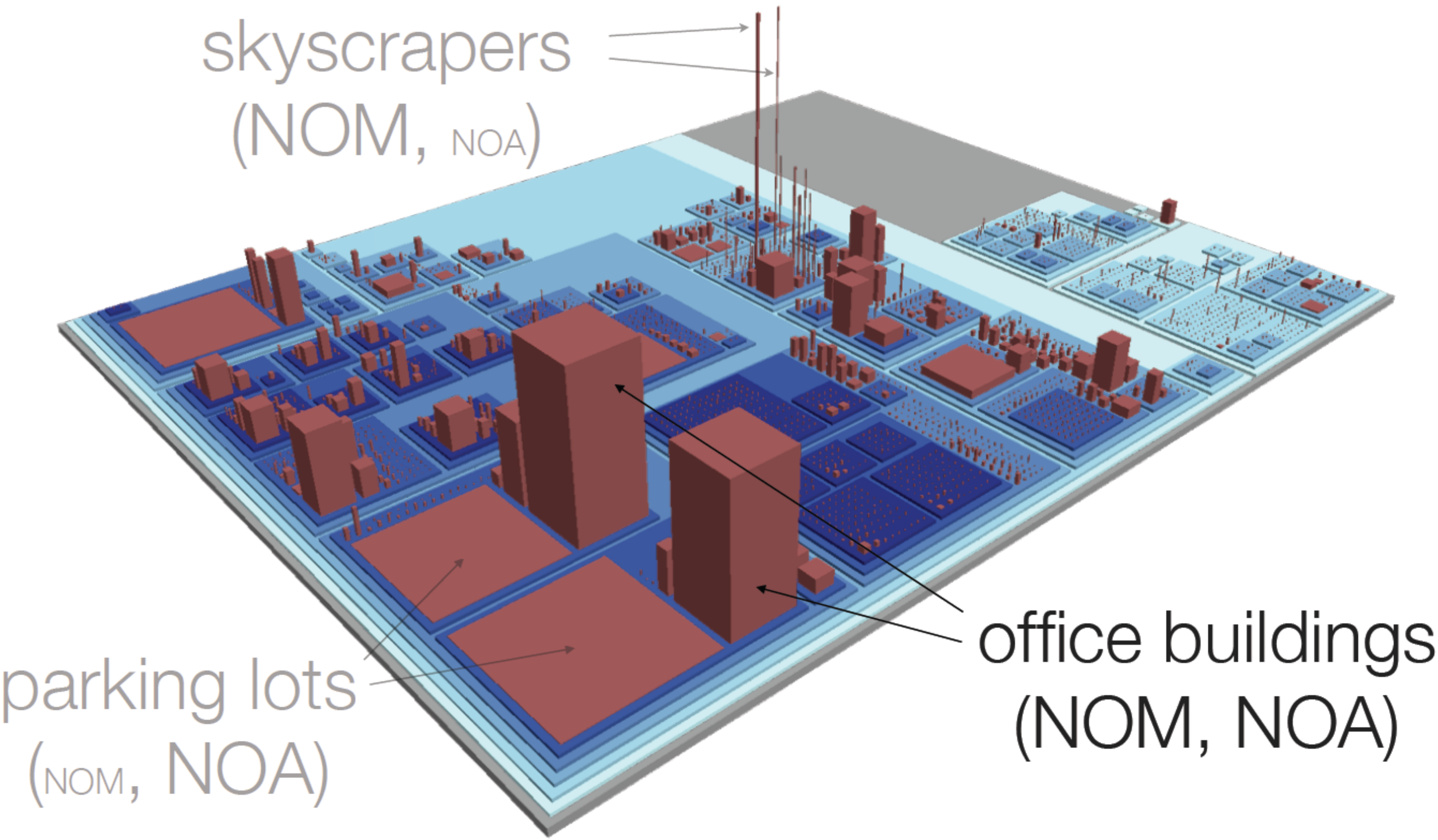
number of methods (NOM)	height
number of attributes (NOA)	base size



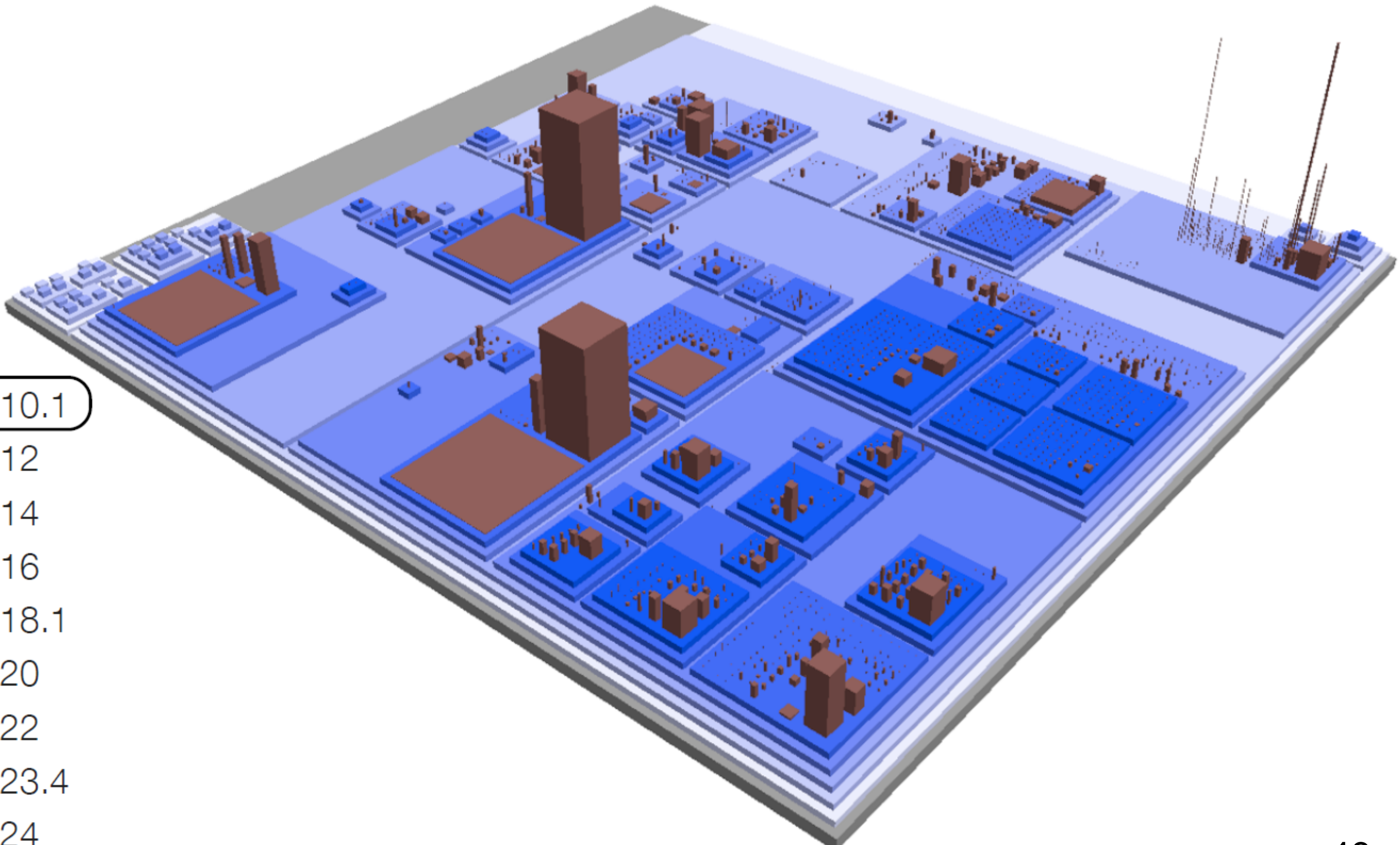
[Wettel & Lanza, ICPC 2007]

[Wettel & Lanza, VISSOFT 2007]

Decoding a city: ArgoUML

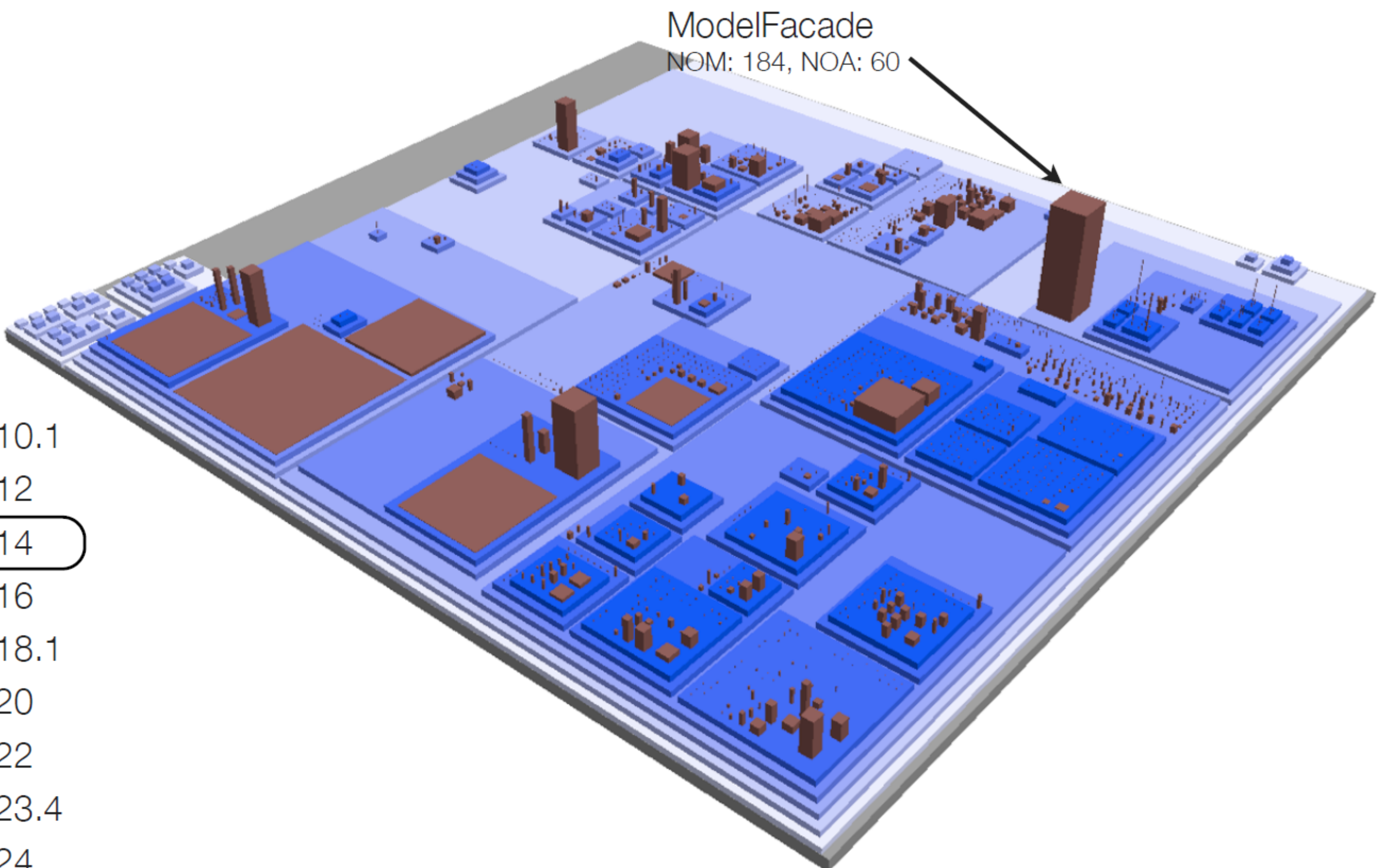


Travelling through ArgoUML's time



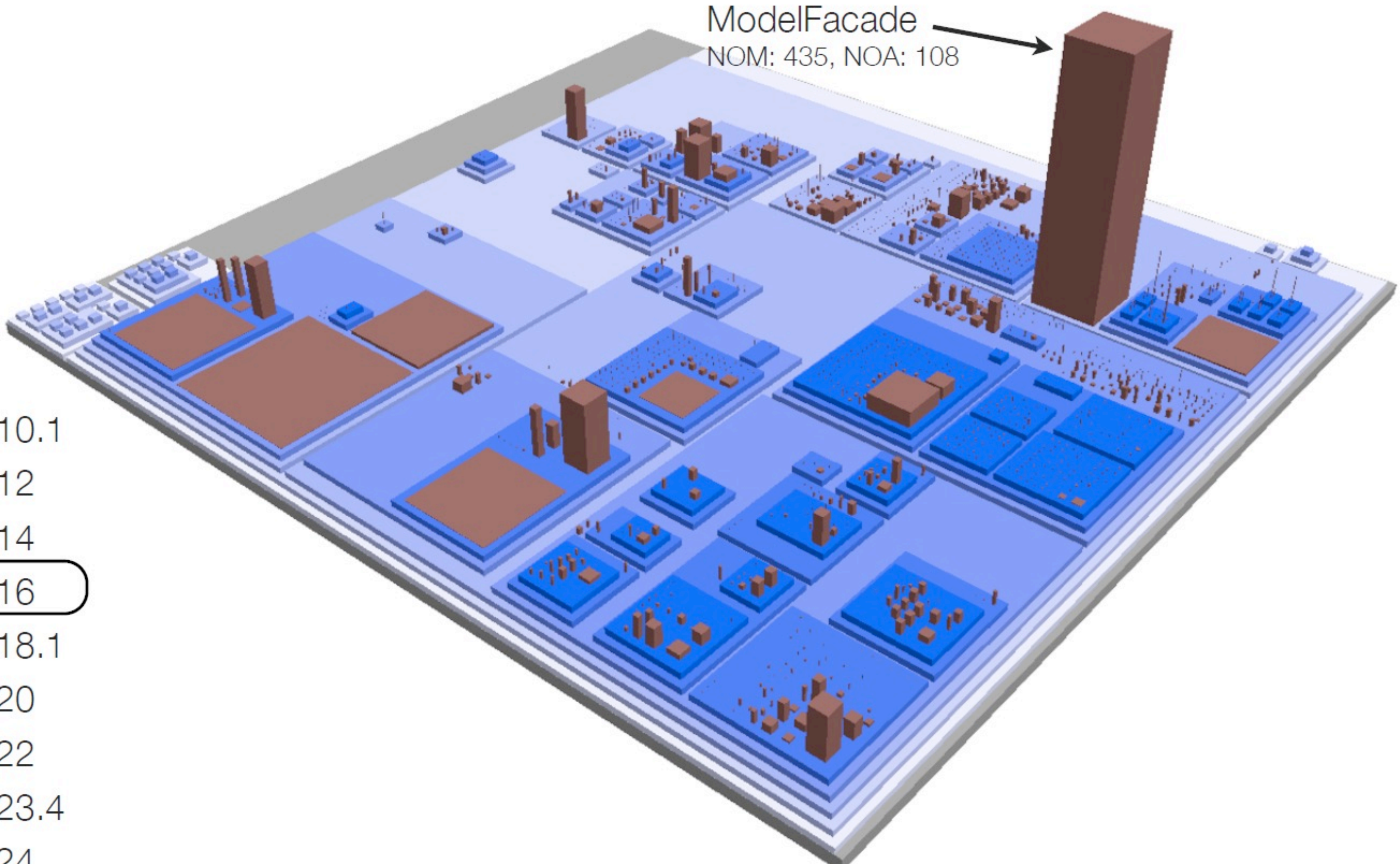
0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

Travelling through ArgoUML's time



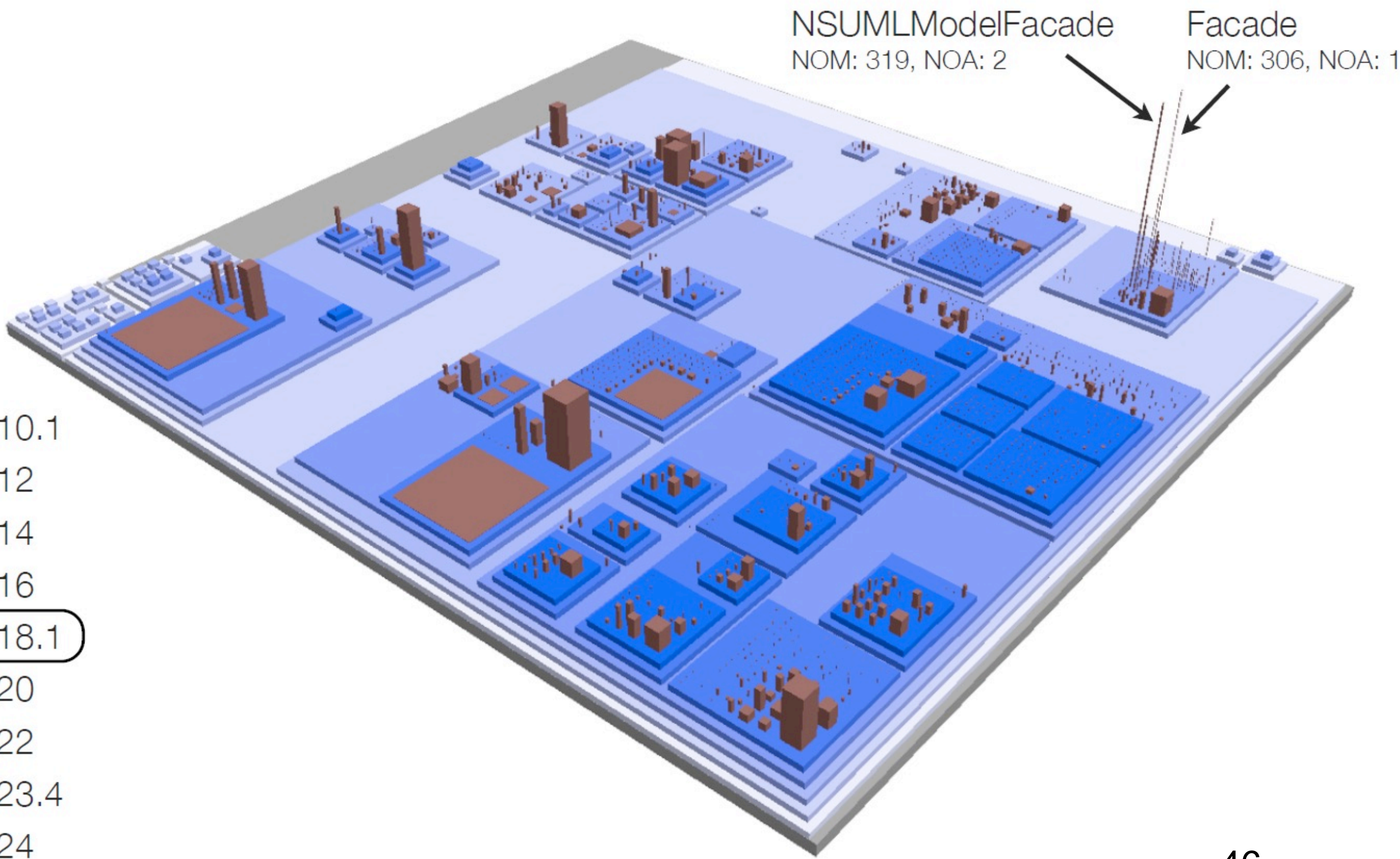
0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

Travelling through ArgoUML's time



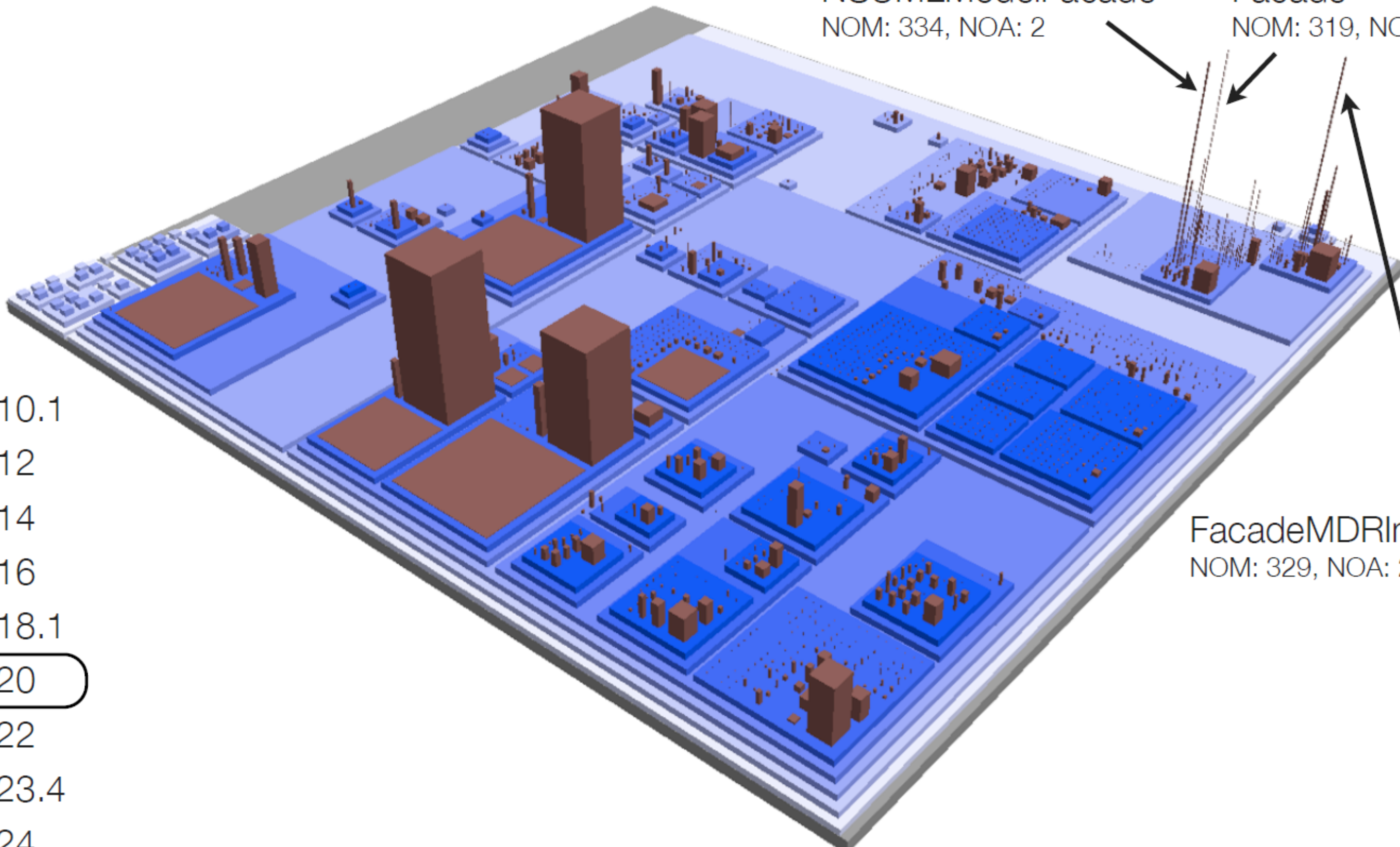
0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

Travelling through ArgoUML's time



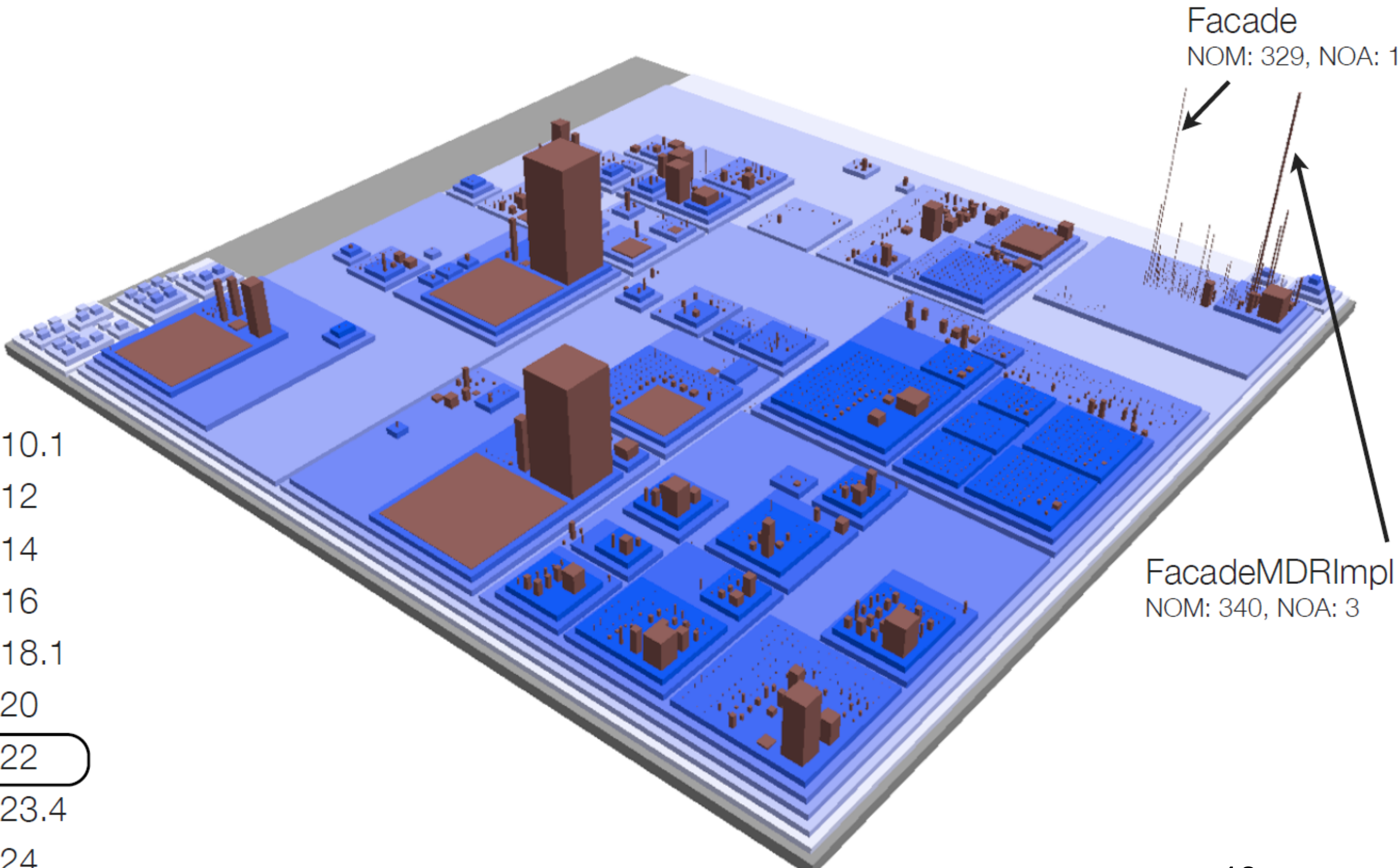
0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

Travelling through ArgoUML's time



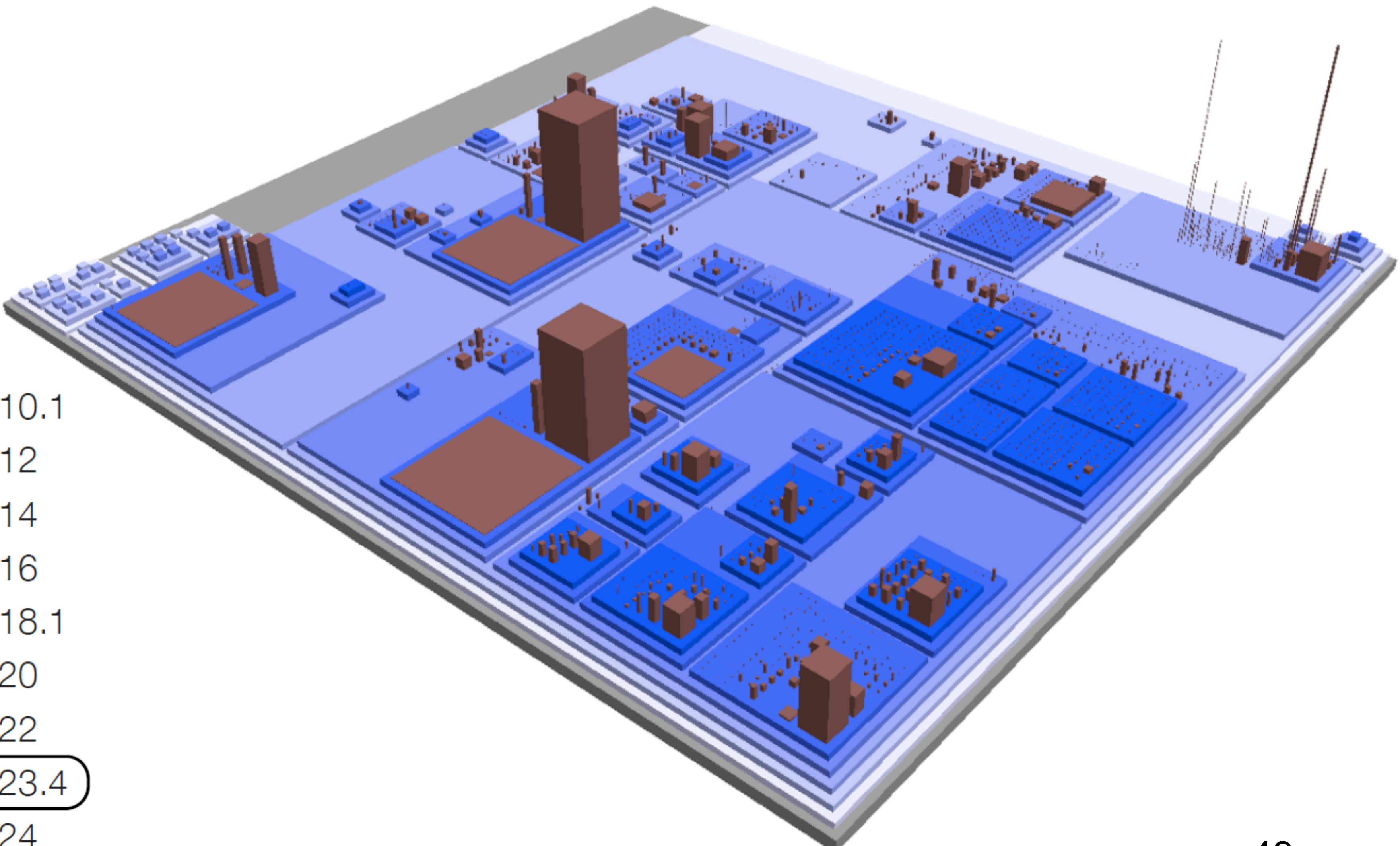
0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

Travelling through ArgoUML's time



0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

Travelling through ArgoUML's time



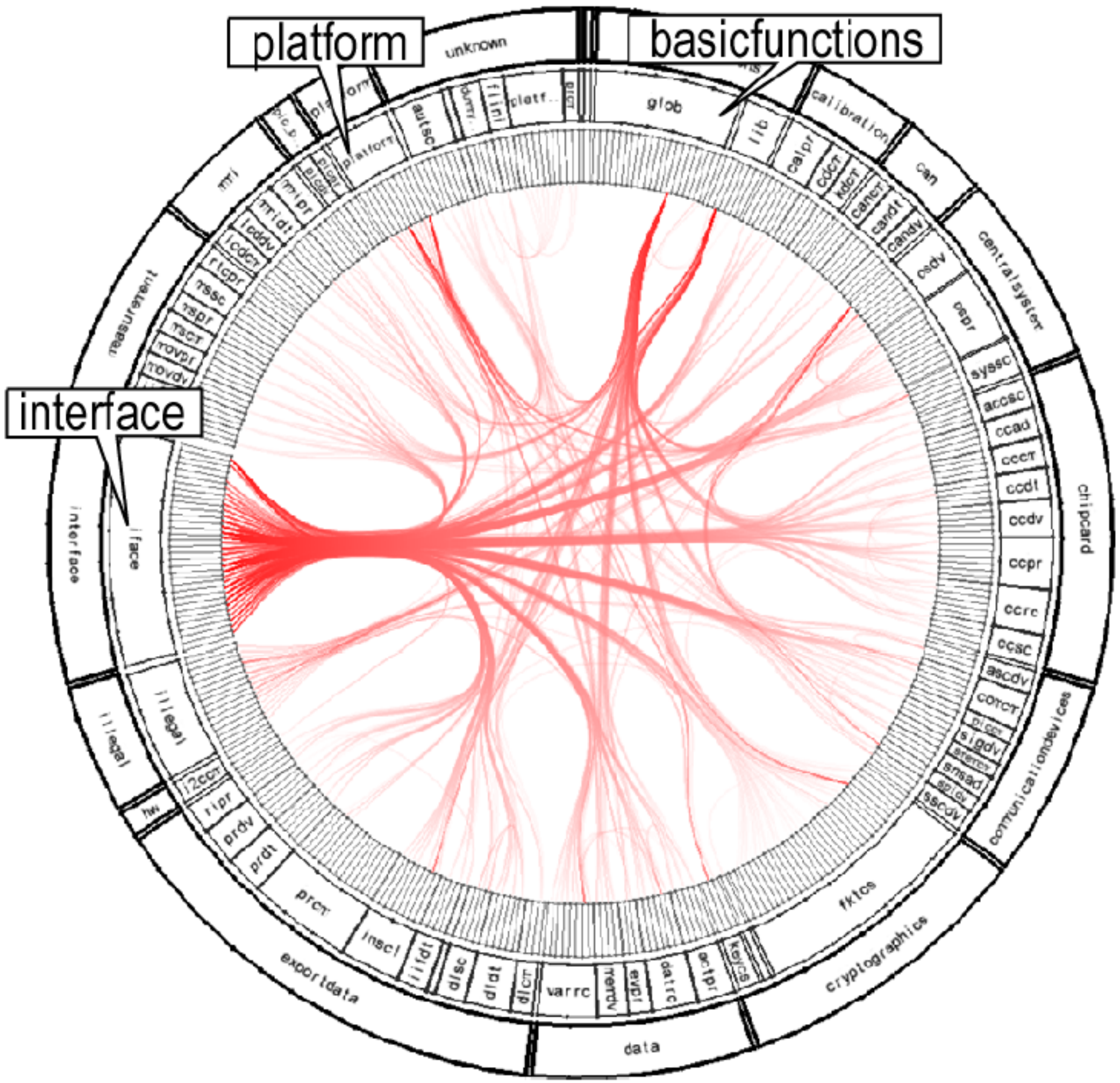
0.10.1
0.12
0.14
0.16
0.18.1
0.20
0.22
0.23.4
0.24

RoleViz [*]

[*] Ho-Quang, Truong, et al. "*Interactive Role Stereotype-Based Visualization To Comprehend Software Architecture*." 2020 Working Conference on Software Visualization (VISSOFT). IEEE, 2020.
Demo video: https://www.youtube.com/watch?v=1JYQMPMF9do&t=278s

Softagram (*)

(*) Demo video: https://www.youtube.com/watch?v=JNcHL5lnutc



Case Study: Visual Analytics in Software Product Assessments

Alexandru Telea*
Institute for Math. and Computer Science
University of Groningen, the Netherlands

Lucian Voinea†
SolidSource BV
Eindhoven, the Netherlands

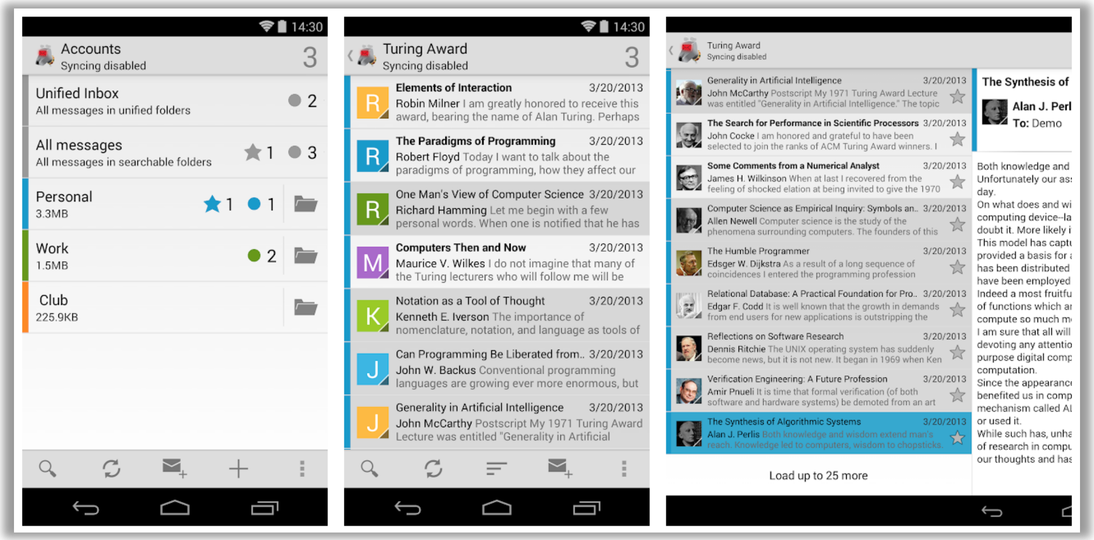
More "visualization" tools

- ObjectAid UML Explorer (an Eclipse plugin)
- StarUML
- Enterprise Architect (Sparx)
- srcML + srcUML
- PlantUML

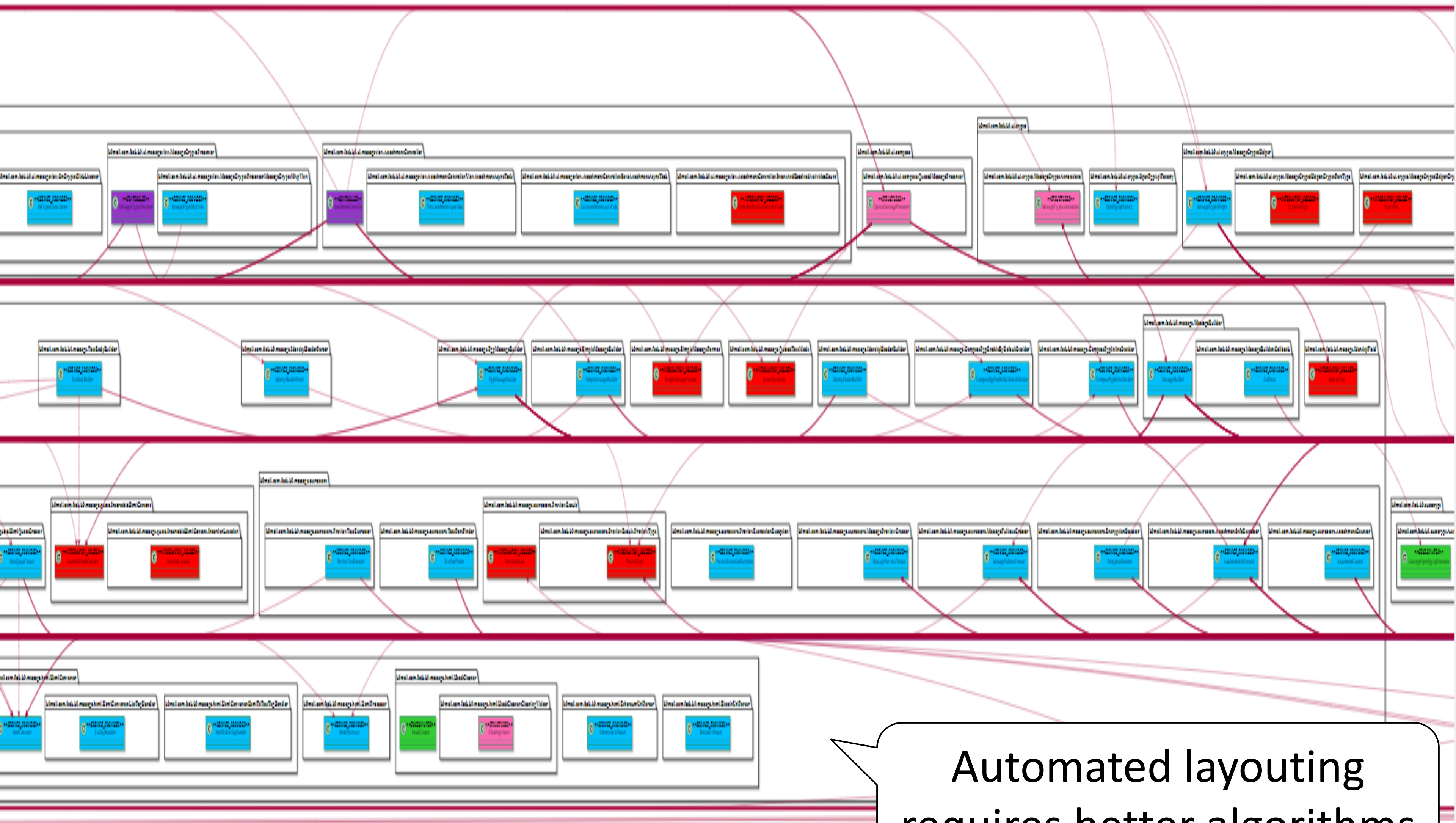
K-9 email

- Android open source
- 10 million downloads from Google Play Store
- 210 contributors

- https://github.com/k9mail/k-9



A Fragment of the 800 classes of K9



Automated layouting requires better algorithms

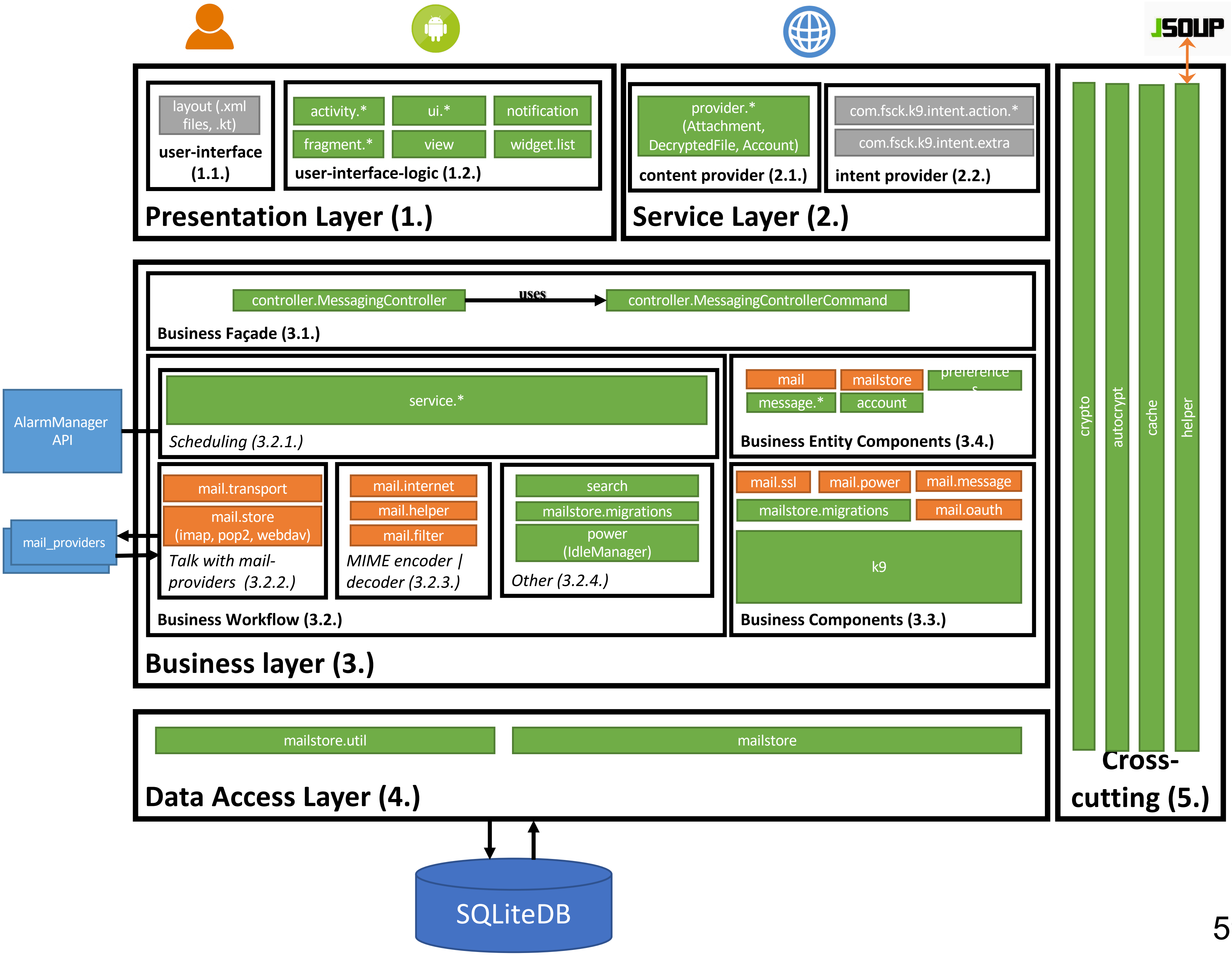
All packages

K9mail-library

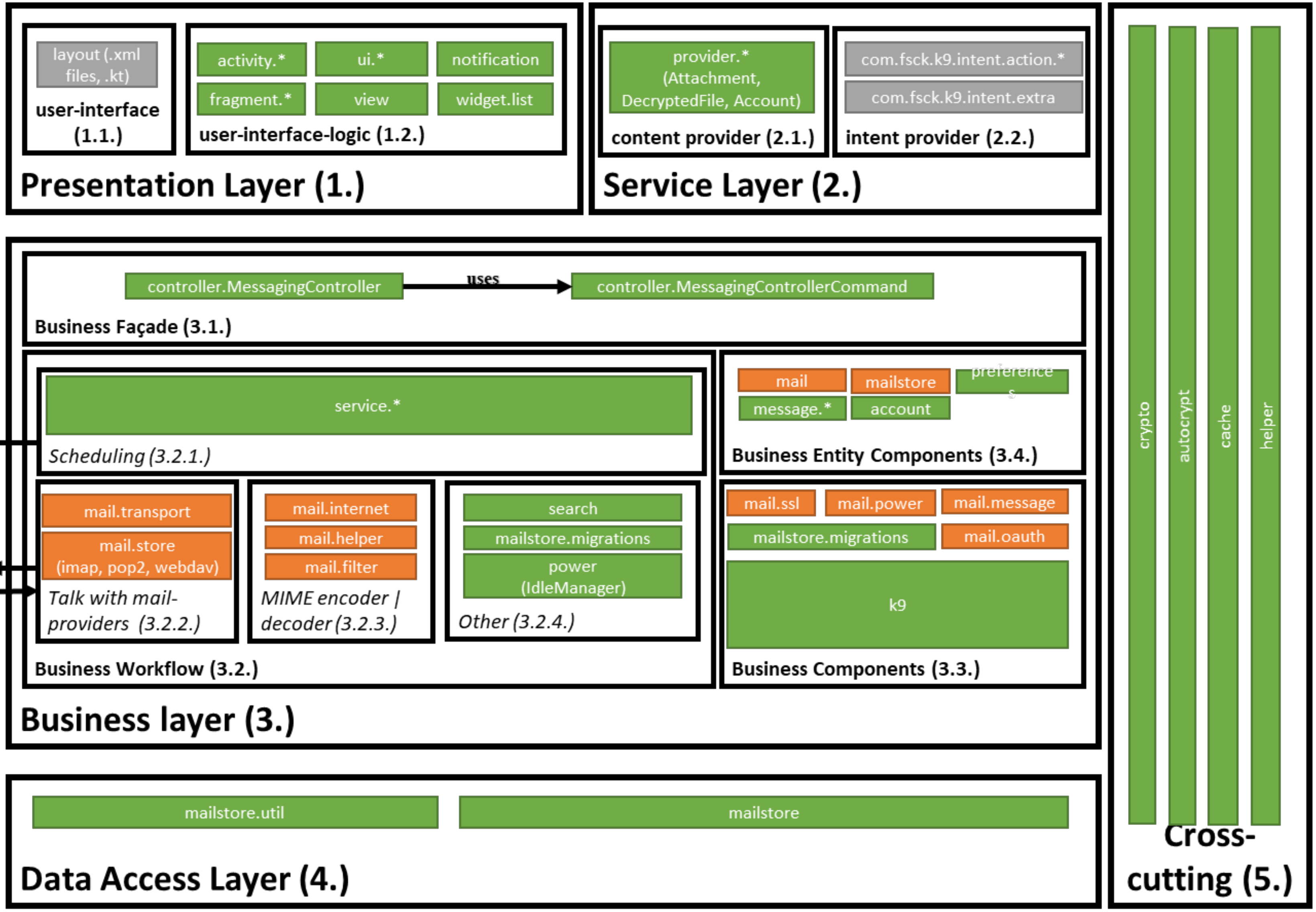
- mail
- mail.filter
- mail.helper
- mail.internet
- mail.message
- mail.oauth
- mail.power
- mail.ssl
- mail.store (imap, pop2, webdav)
- mail.store

k9mail

- k9
- account
- activity
- acitivity.compose
- acitivity.loader
- acitivity.misc
- acitivity.setup
- autocrypt
- cache
- controller
- crypto
- fragment
- helper
- helper.jsoup
- mailstore
- mailstore.migrations
- mailstore.util
- message
- message.extractors
- message.html
- message.quote
- message.signature
- ui
- ui.compose
- ui.crypto
- ui.dialog
- ui.message
- ui.messageview
- service
- setup
- search
- view
- widget.list
- notification
- power
- preferences
- provider
- remotecontrol
- search



JSOUP
layout (.xml files, .kt)
user-interface (1.1.)
activity.*
ui.*
notification
fragment.*
view
widget.list
user-interface-logic (1.2.)
Presentation Layer (1.)
provider.* (Attachment, DecryptedFile, Account)
content provider (2.1.)
com.fsck.k9.intent.action.*
com.fsck.k9.intent.extra
intent provider (2.2.)
Service Layer (2.)
controller.MessagingController
uses
controller.MessagingControllerCommand
Business Façade (3.1.)
AlarmManager API
service.*
Scheduling (3.2.1.)
mail
mailstore
message.*
account
Business Entity Components (3.4.)
mail_providers
mail.transport
mail.store (imap, pop2, webdav)
Talk with mail-providers (3.2.2.)
mail.internet
mail.helper
mail.filter
MIME encoder | decoder (3.2.3.)
search
mailstore.migrations
power (IdleManager)
Other (3.2.4.)
Business Workflow (3.2.)
mail.ssl
mail.power
mail.message
mailstore.migrations
mail.oauth
k9
Business Components (3.3.)
Business layer (3.)
crypto
autocrypt
cache
helper
mailstore.util
mailstore
Data Access Layer (4.)
Cross-cutting (5.)
SQLiteDB



layout (.xml files, .kt)
user-interface (1.1.)
activity.*
ui.*
notification
fragment.*
view
widget.list
user-interface-logic (1.2.)
Presentation Layer (1.)
provider.* (Attachment, DecryptedFile, Account)
content provider (2.1.)
com.fsck.k9.intent.action.*
com.fsck.k9.intent.extra
intent provider (2.2.)
Service Layer (2.)
controller.MessagingController
uses
controller.MessagingControllerCommand
Business Façade (3.1.)
service.*
Scheduling (3.2.1.)
mail.transport
mail.store (imap, pop2, webdav)
Talk with mail-providers (3.2.2.)
mail.internet
mail.helper
mail.filter
MIME encoder | decoder (3.2.3.)
search
mailstore.migrations
power (IdleManager)
Other (3.2.4.)
Business Workflow (3.2.)
mail
mailstore
preferences
message.*
account
Business Entity Components (3.4.)
mail.ssl
mail.power
mail.message
mailstore.migrations
mail.oauth
k9
Business Components (3.3.)
Business layer (3.)
mailstore.util
mailstore
Data Access Layer (4.)
crypto
autocrypt
cache
helper
Cross-cutting (5.)

K9 from:

Software Architectural Principles in Contemporary Mobile Software: from Conception to Practice

Hamid Bagheri[a], Joshua Garcia[a], Alireza Sadeghi[a], Sam Malek[a], Nenad Medvidovic[b]

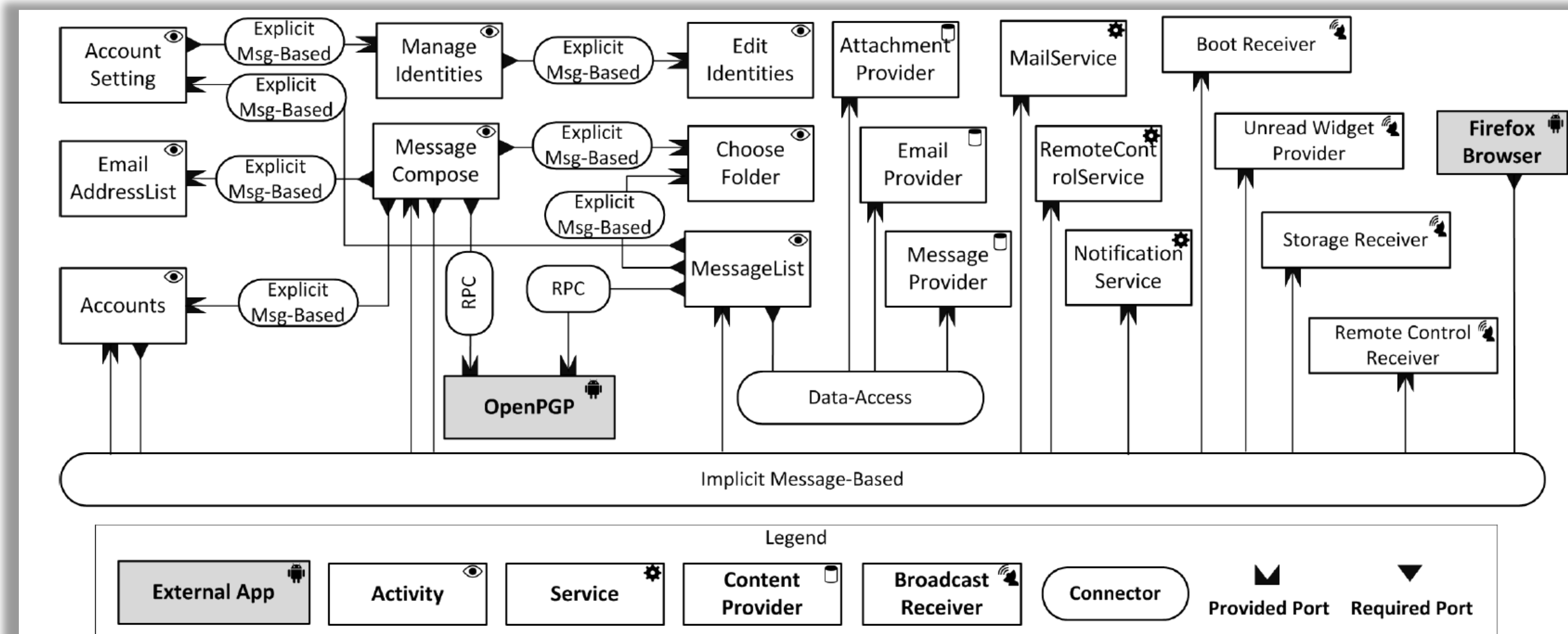
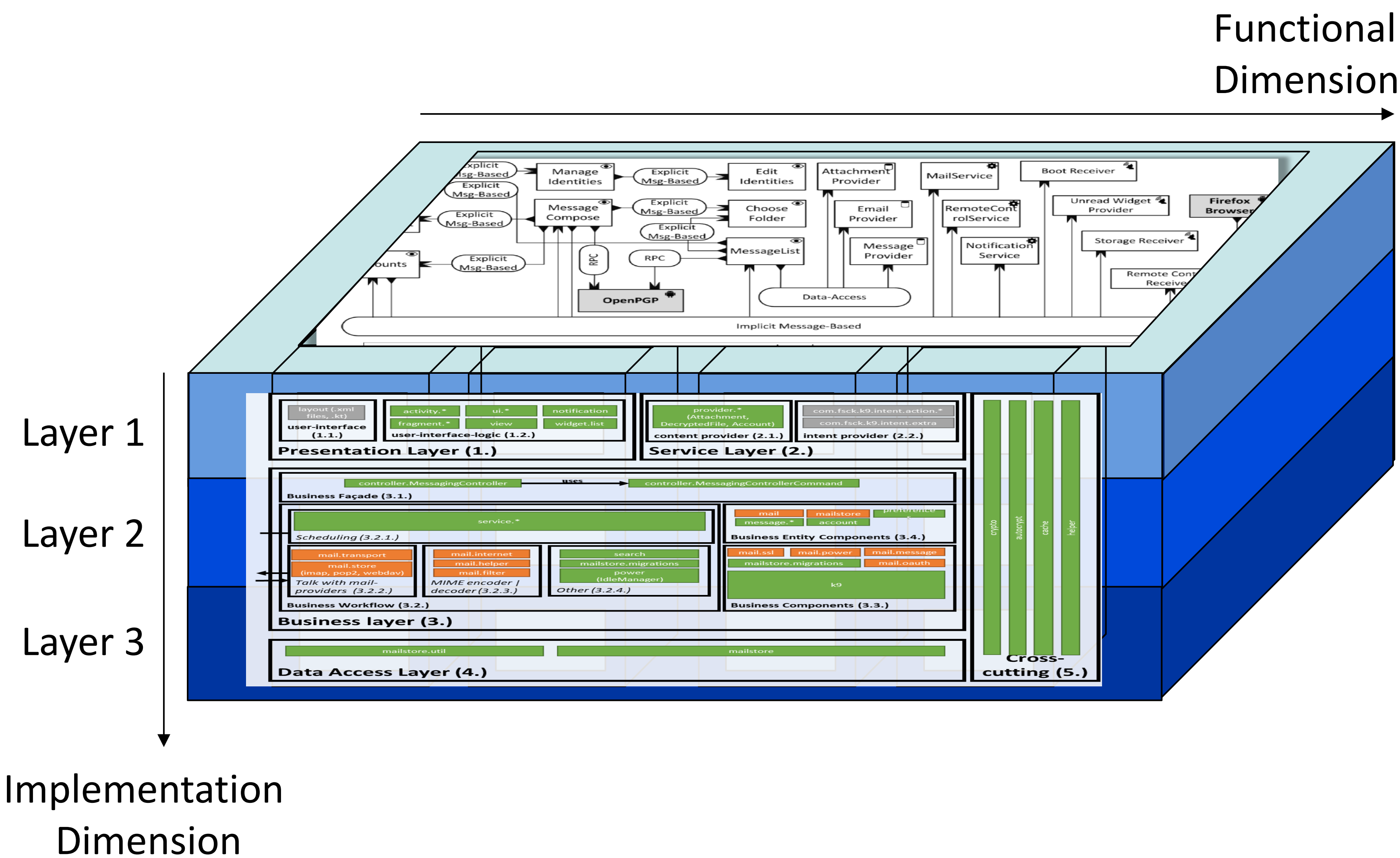
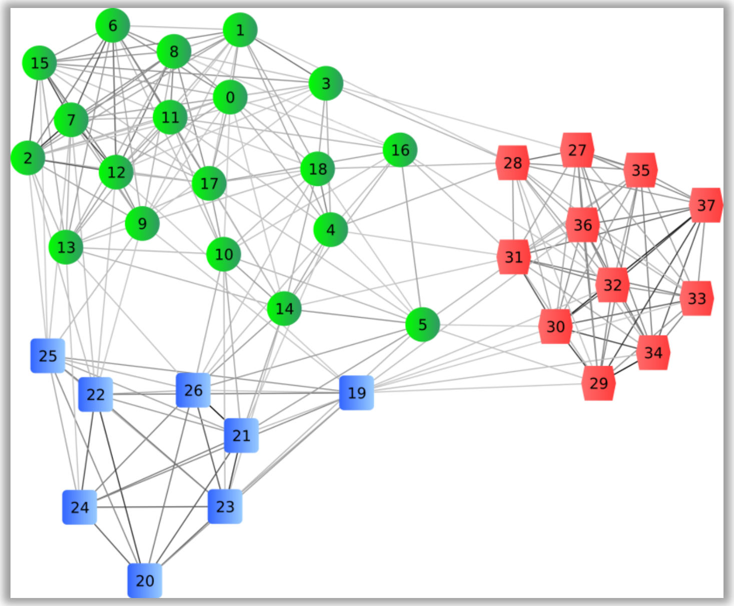


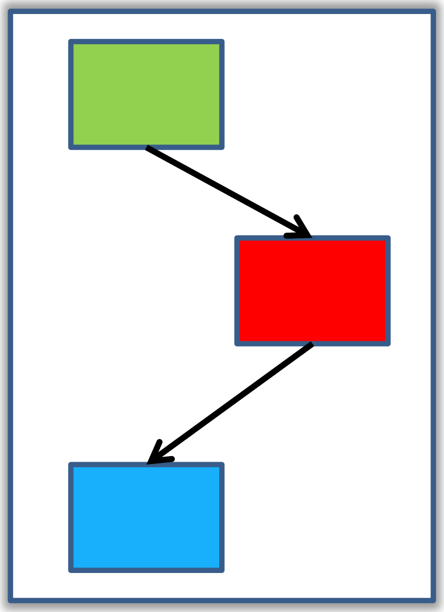
Figure 2: K-9 mail Android app architecture


Functional Dimension
Layer 1
Layer 2
Layer 3
Implementation Dimension
Presentation Layer (1.)
Service Layer (2.)
Business layer (3.)
Data Access Layer (4.)
Cross-cutting (5.)

How about more abstraction?

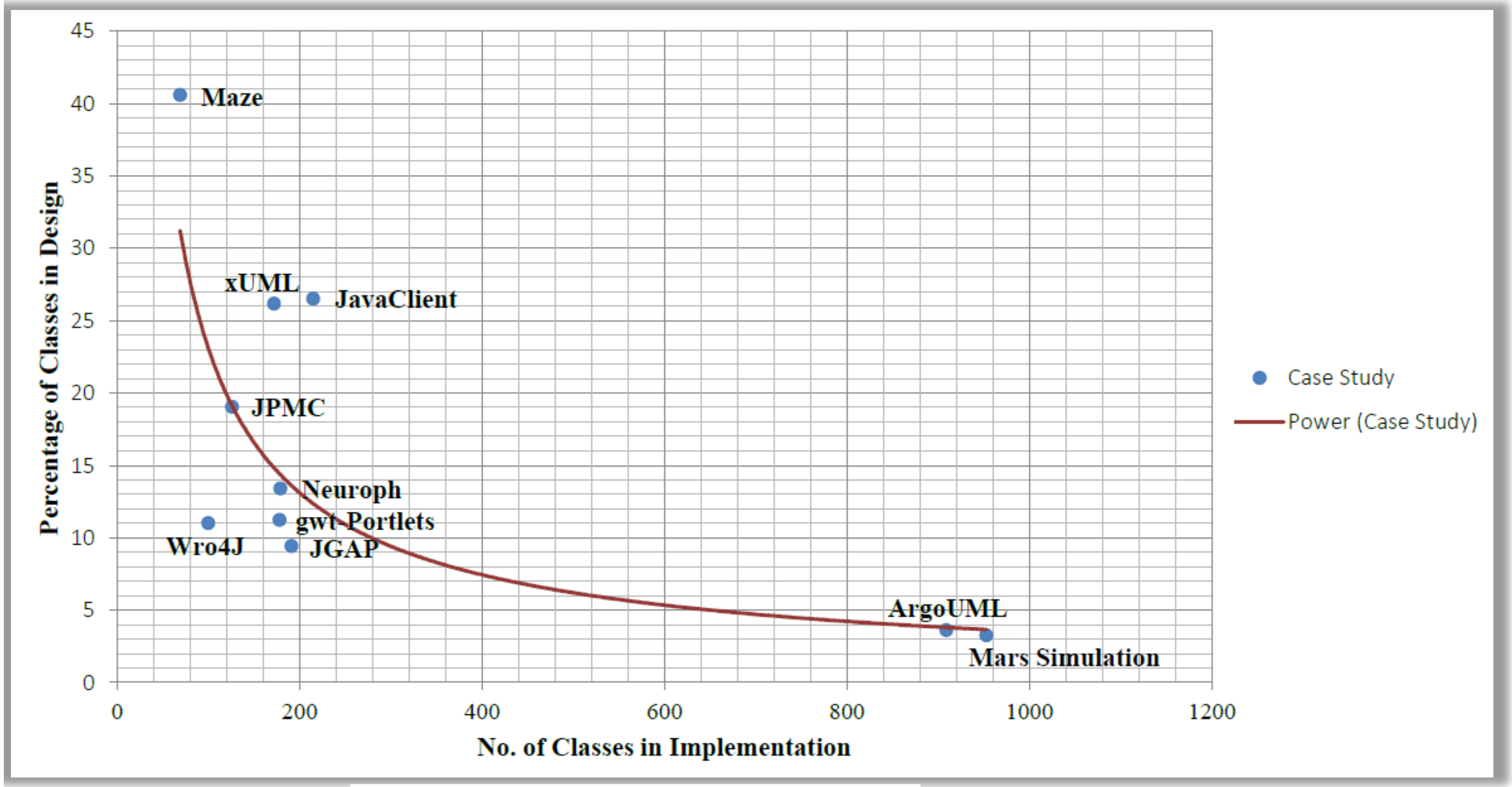
- Clustering





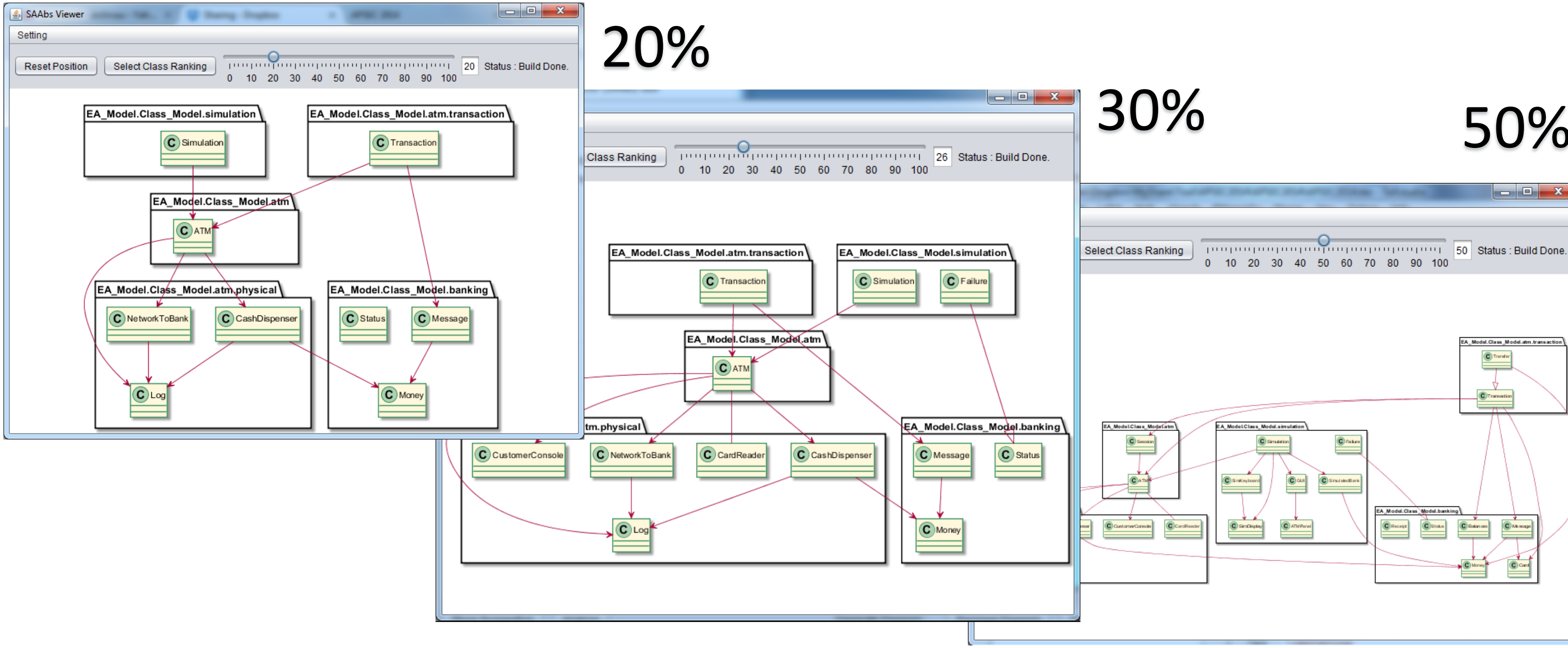
Economy of Modeling

% of system covered by model



Size of the system

Scaling Abstraction (*)



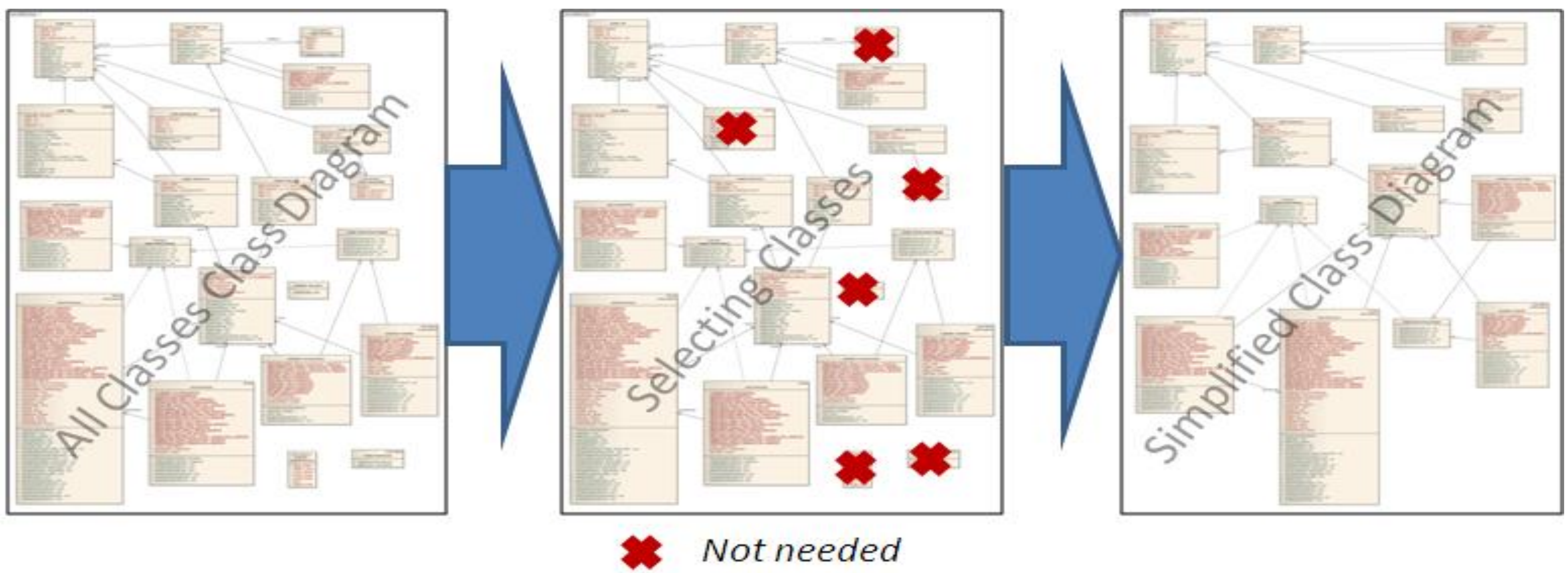
- Which criteria to use for abstraction?
- What is the relation between design and code?
- Different tasks require different parts/slices/views

(*) Osman, Mohd Hafeez, Michel RV Chaudron, and Peter Van Der Putten. "An analysis of machine learning algorithms for condensing reverse engineered class diagrams." 2013 IEEE International Conference on Software Maintenance. IEEE, 2013. Demo video: https://youtu.be/dHBB5wA2wDI

Class Diagram Simplification

Hafeez Osman

This research aims at **simplifying** class diagrams



Considering: - structural properties (coupling, size)

- semantic properties:
  ‘support’ vs ‘core functionality’

GUI / frameworks / gets&sets vs cruise control

- feature based

- Drop most of the operations in the classes mainly getters/setters/constructors.
=> But, do describe the pattern of creation & dependency handling.
- Add some layers (using dashed lines or so)

Part B: Practical Problems

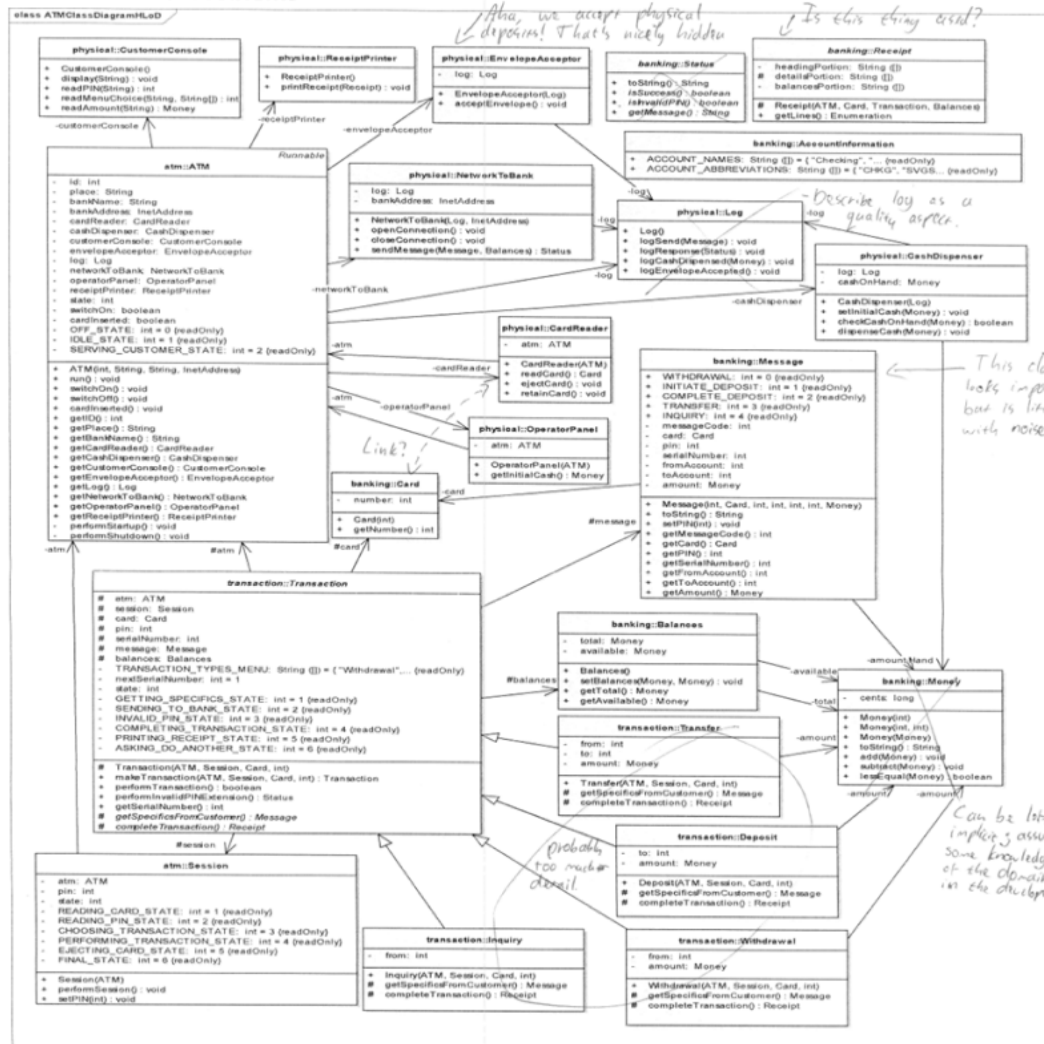
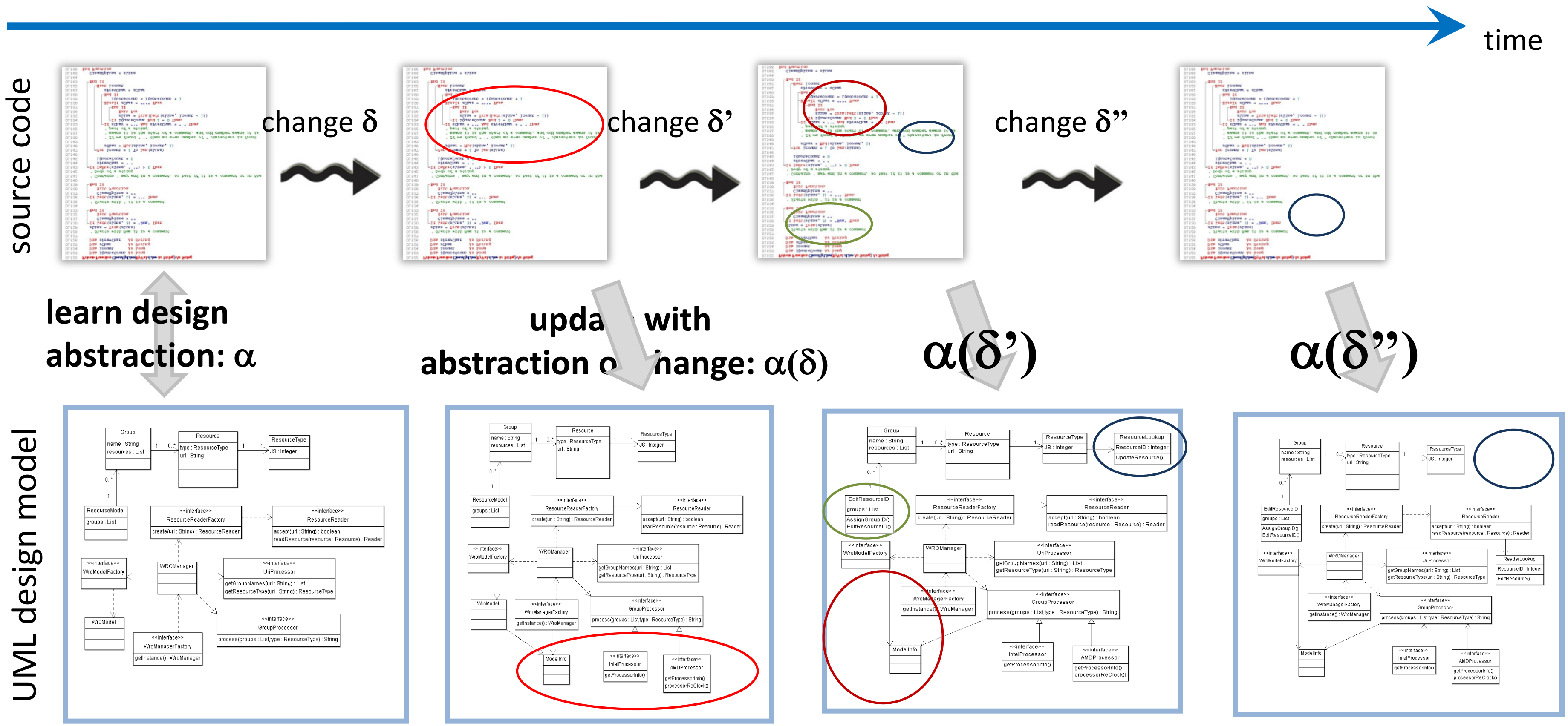


Figure 1: ATM Machine Class diagram

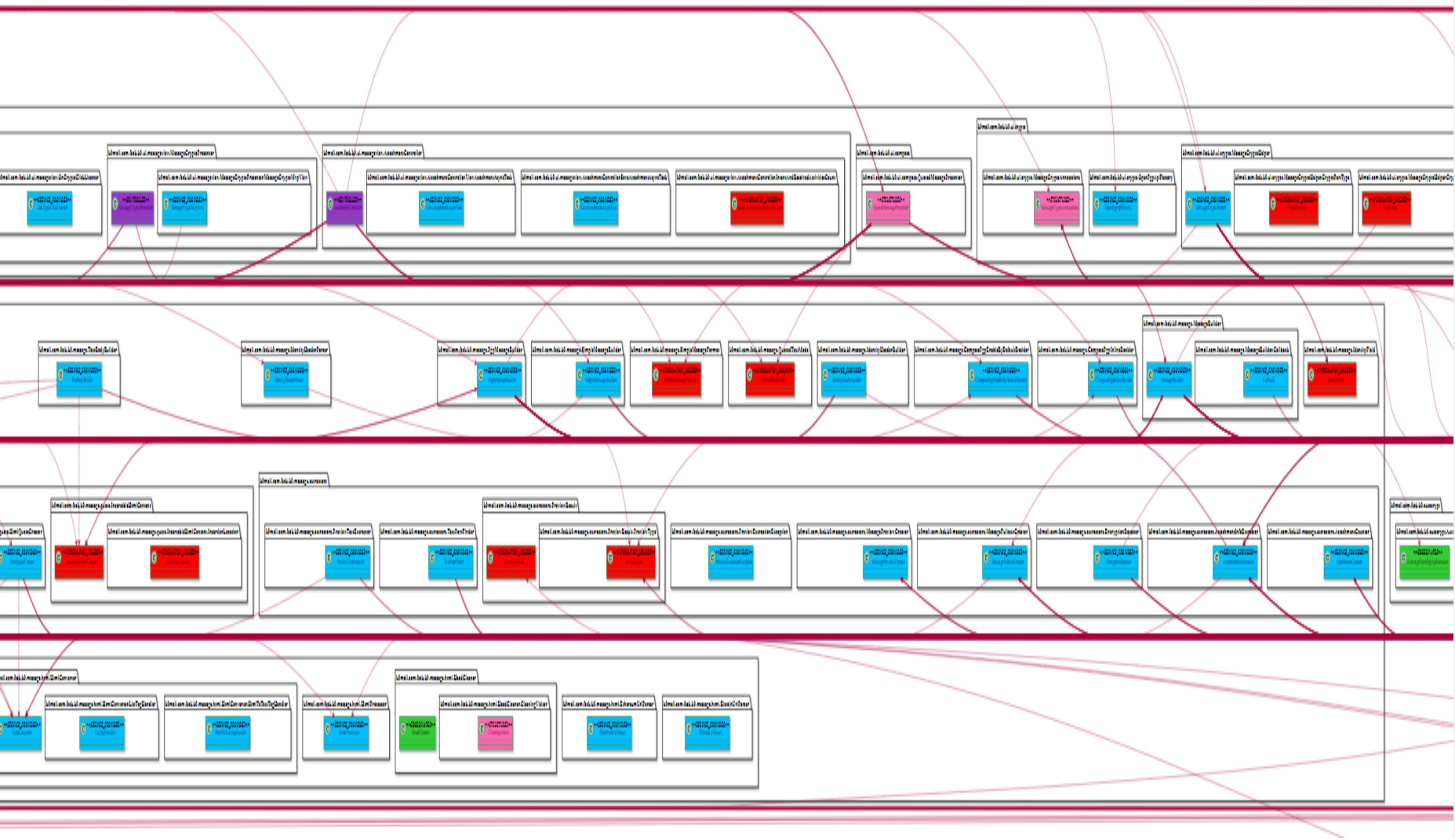
Question B1: Suppose you are new to the ATM system project and have to learn about the ATM system from the class diagram in Fig.1. Which information (class, method, relation, ...) do you think could be left out of the class diagram without affecting your understanding of the system?

University of Leiden, 2012

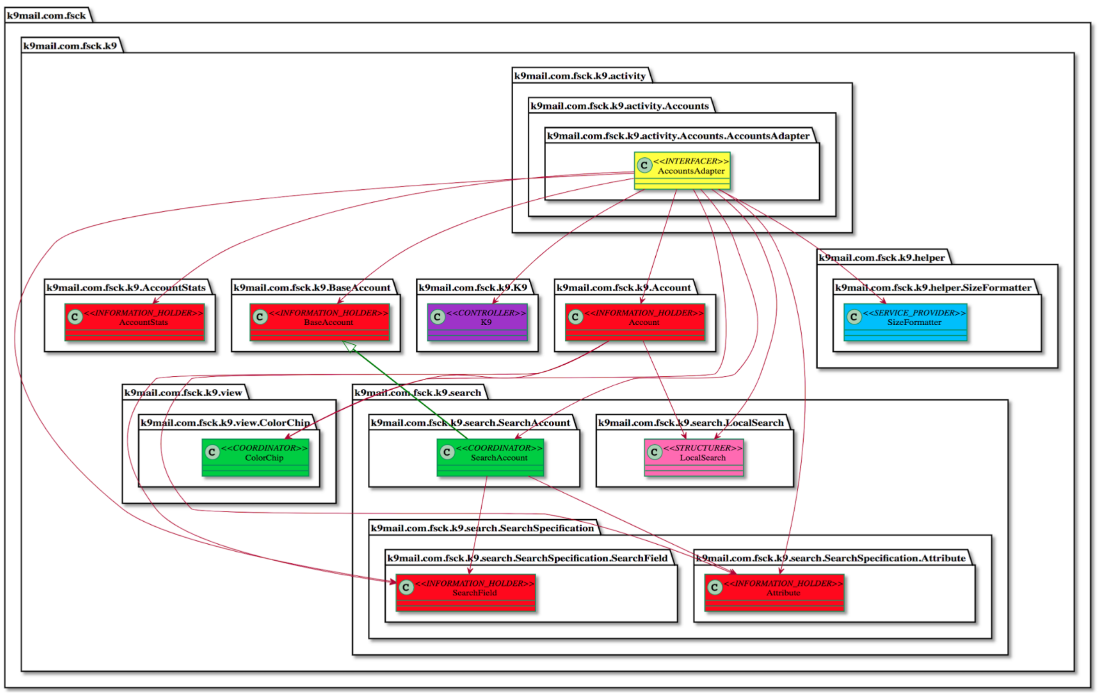
Automated Updating of Class Models



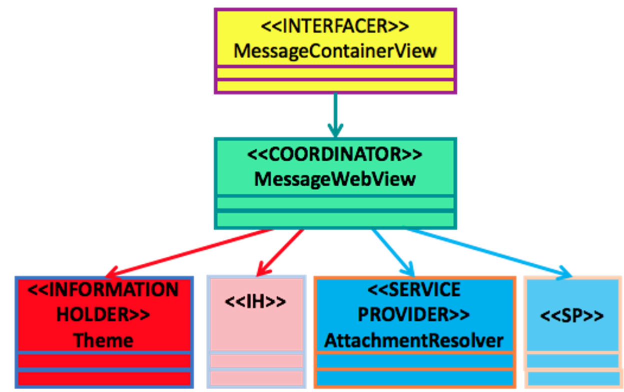
A change δ can be an addition, modification, removal.



Collaboration Pattern between Stereotypes



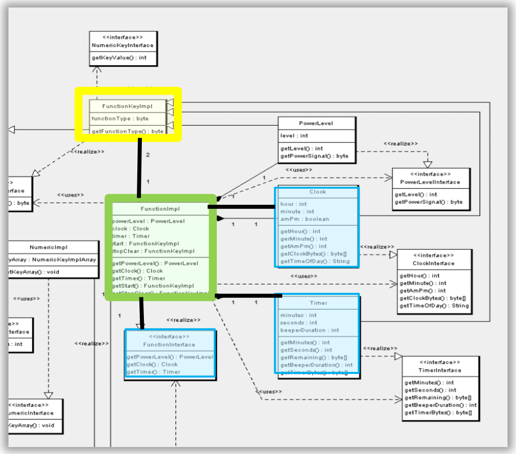
Through labelling of roles, we find recurring patterns in the design

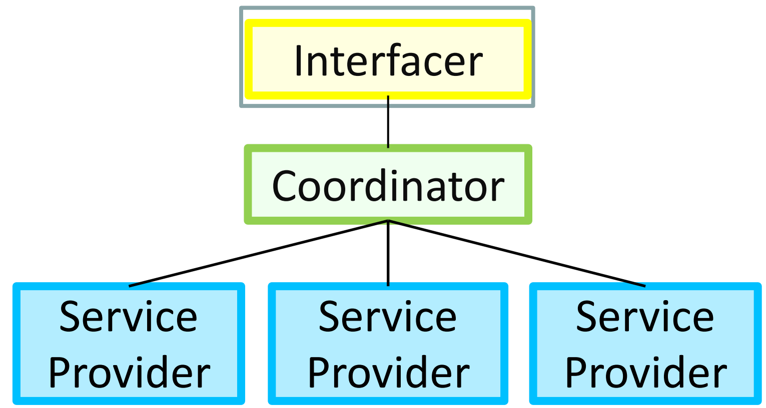


These patterns represent typical collaborations between responsibility-stereotypes

Common graph-patterns in Software Designs

Through labelling of roles, we find recurring patterns in the design



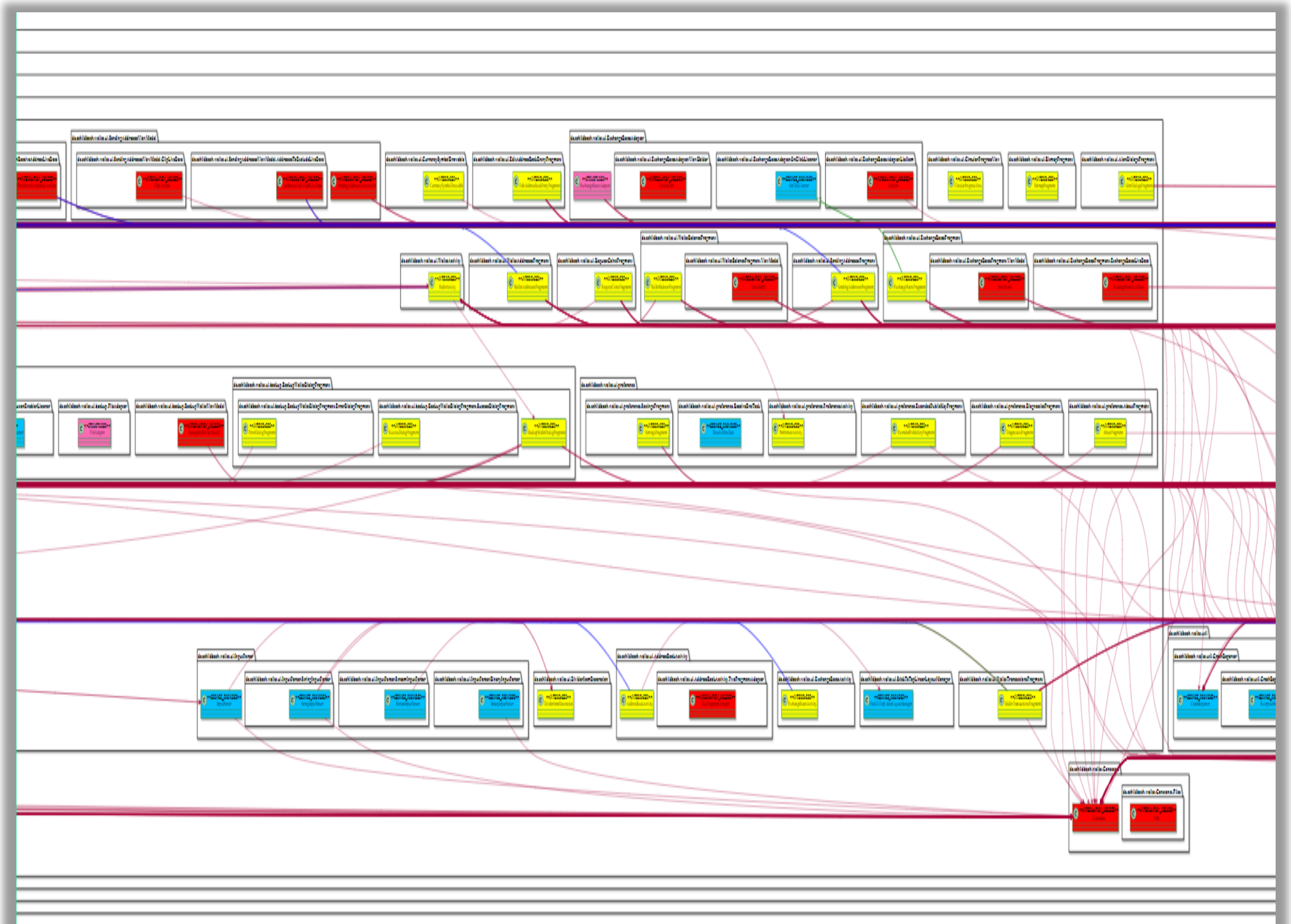


These patterns represent typical collaborations between role-stereotypes.

These patterns can be used for e.g.

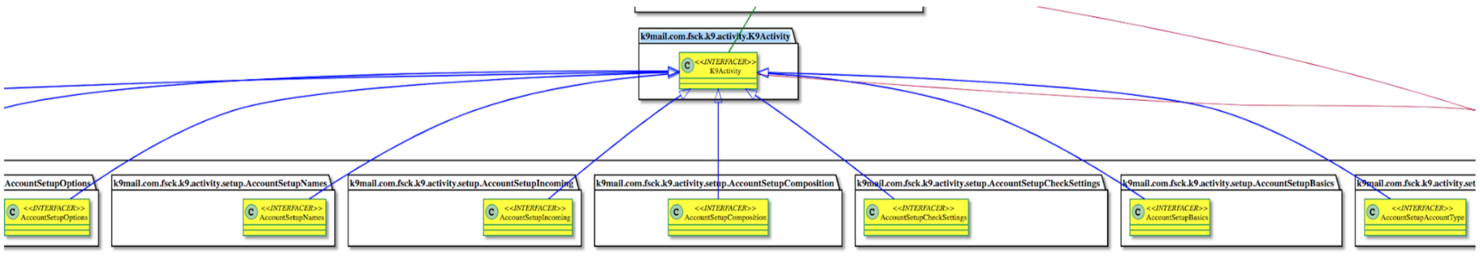
- checking designs (allowed dependencies; metric thresholds)
- synthesizing a design
- generating visualizations
- design summarization

Role-stereotypes in software design

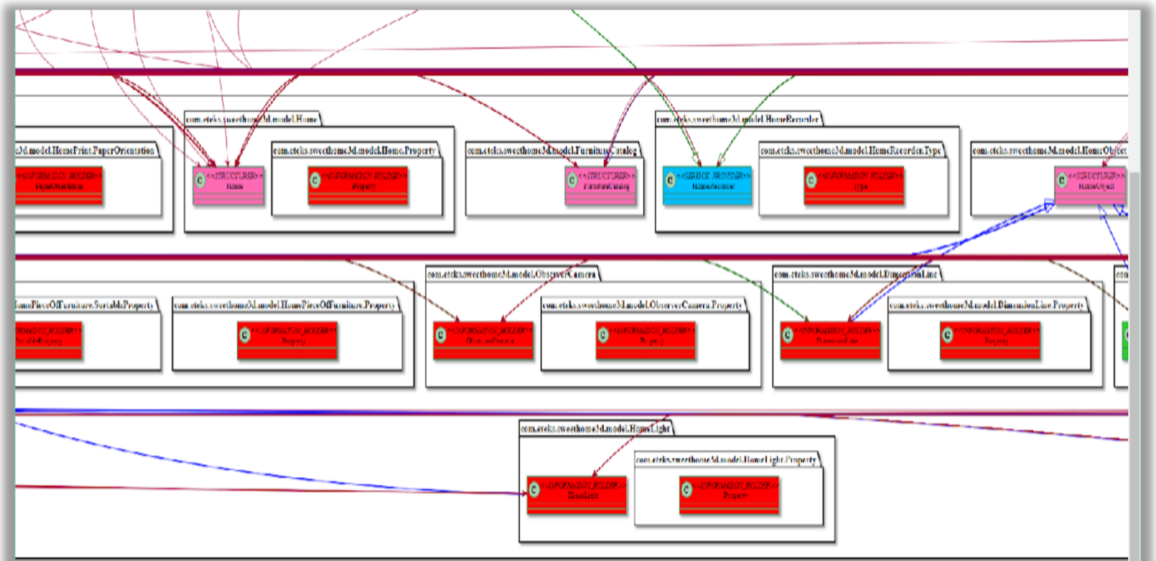


- Interfacer
- Controller
- Coordinator
- Service Provider
- Information Holder
- Structurer

Collaboration Patterns between Stereotypes



This shows expansion of roles



Interfacer

Controller

Coordinator

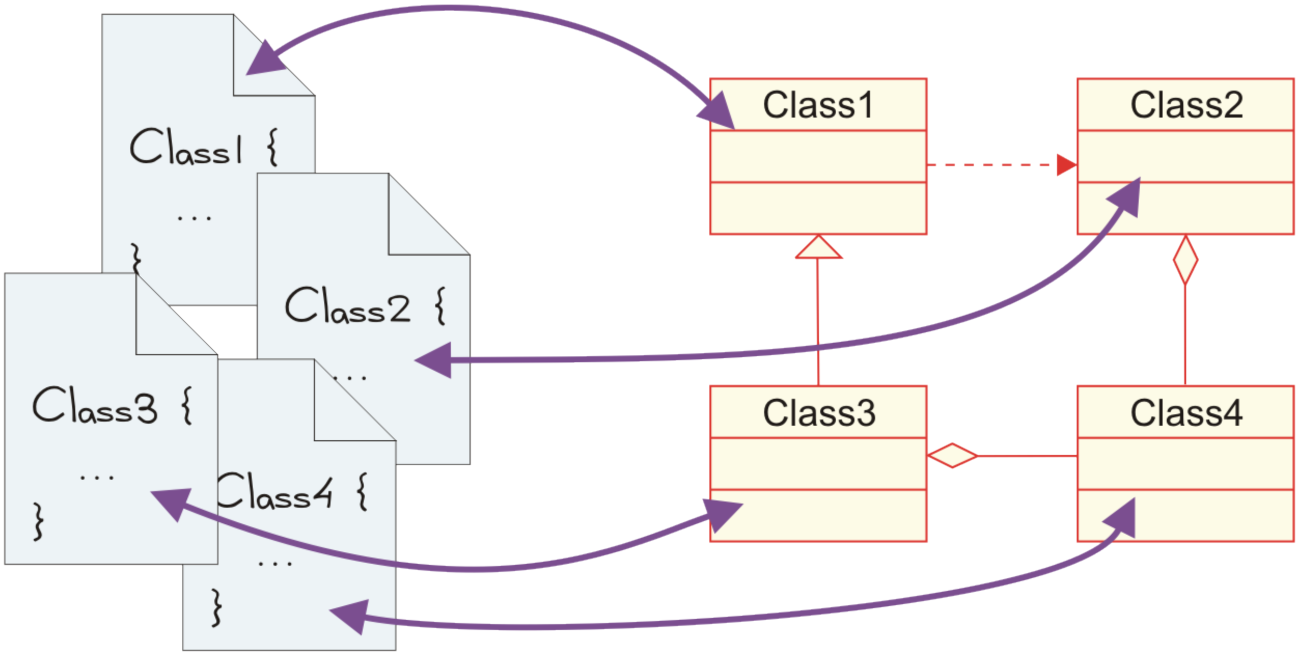
Service Provider

Information Holder

Structurer

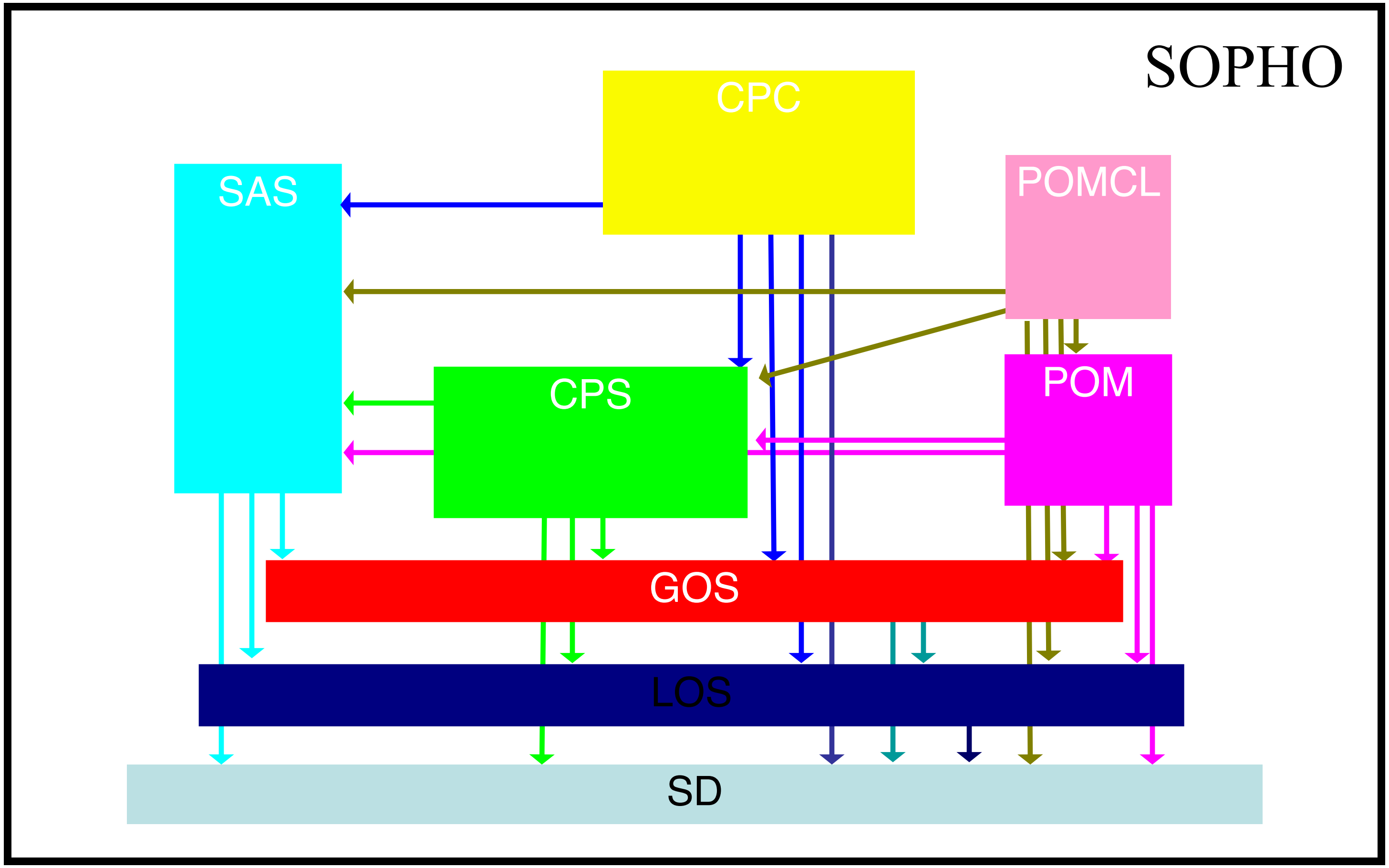
Checking Design-Code Correspondence using Relational Algebra

Managing Design – Code Correspondence



Reinder J. Bril, r.j.bril@tue.nl
TU/e Informatica, System Architecture and Networking

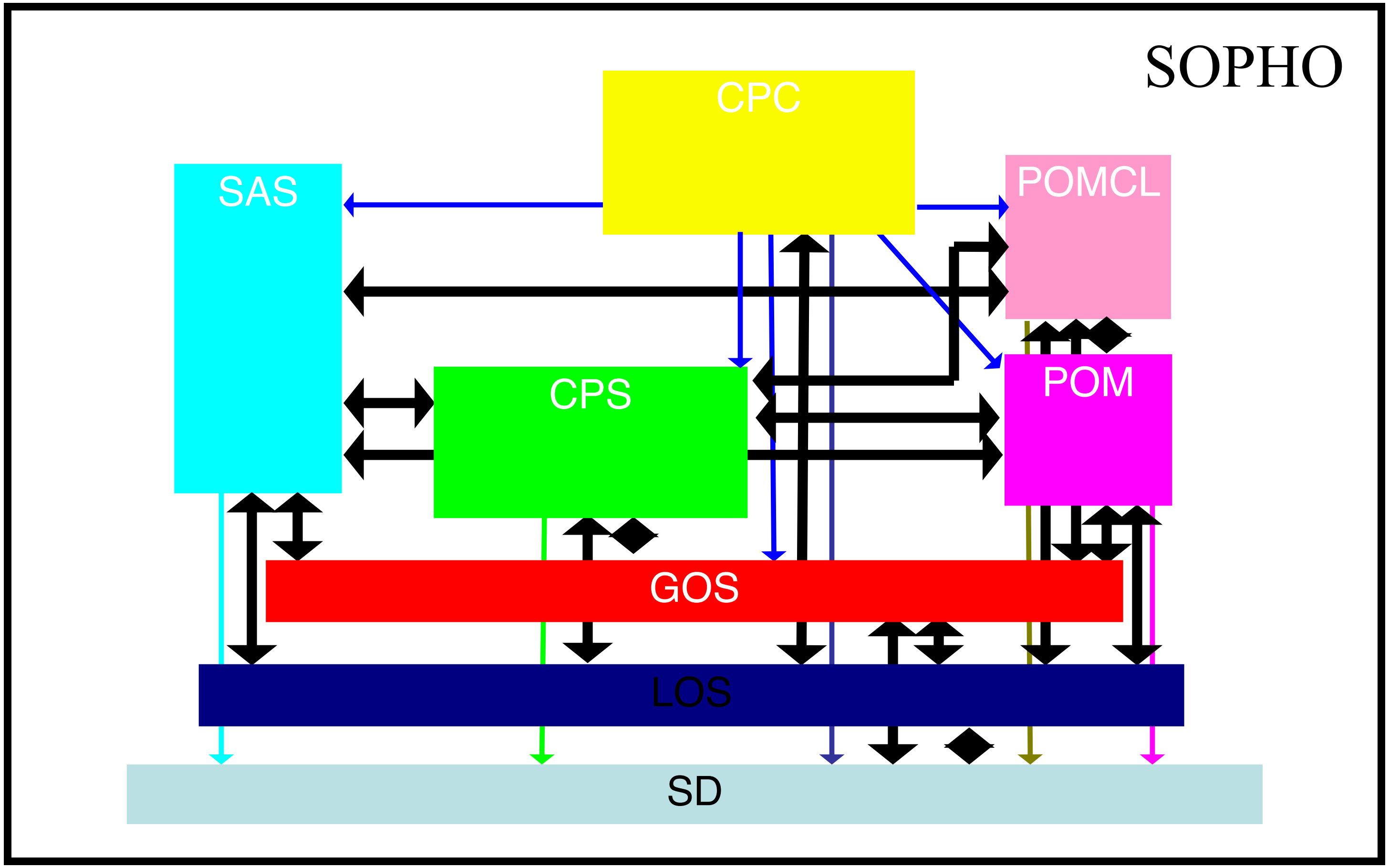
Application domain



"Intended" module architecture
(documentation + software architects)

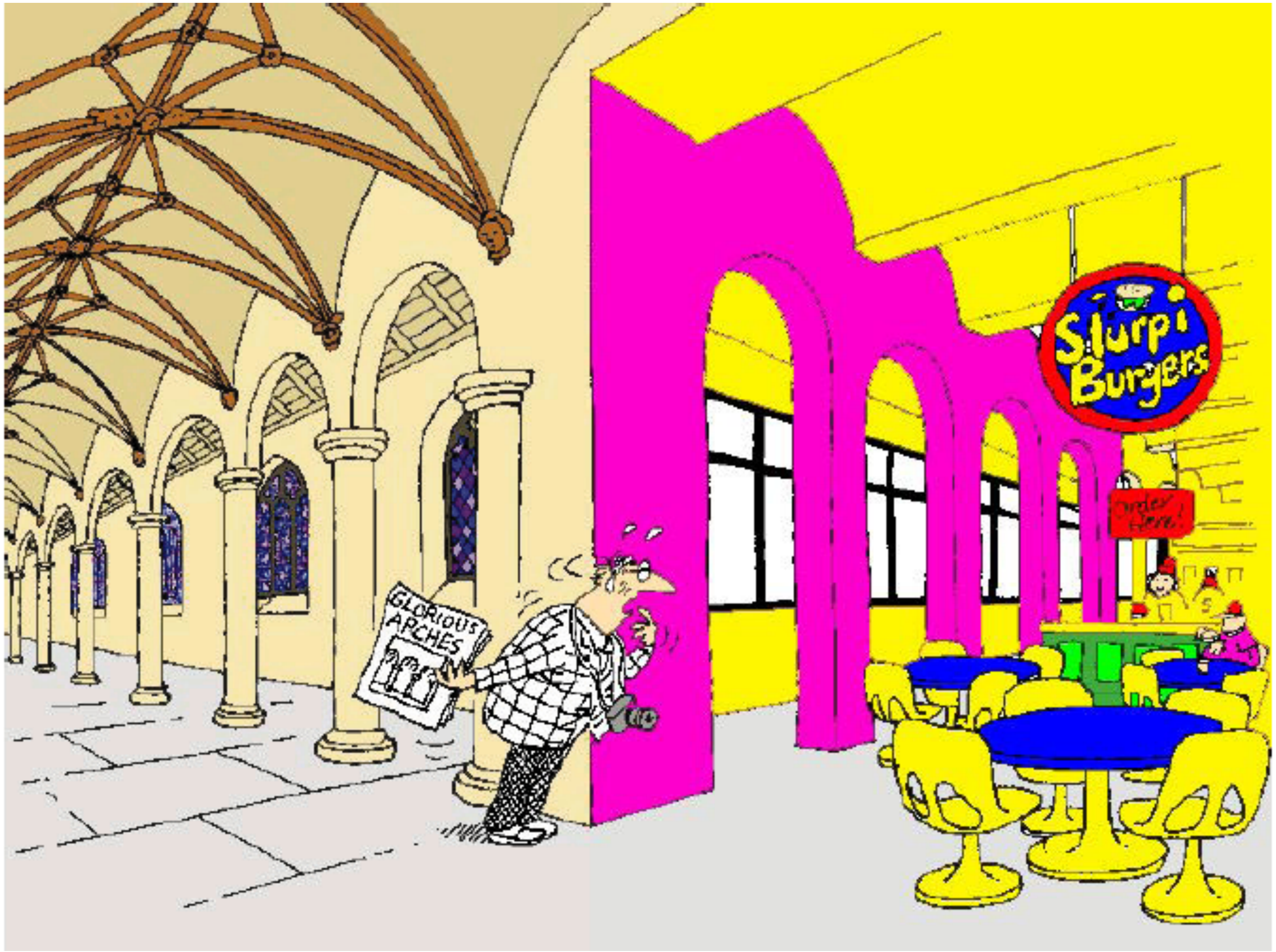
Reinder J. Bril, r.j.bril@tue.nl
TU/e Informatica, System Architecture and Networking

Application domain



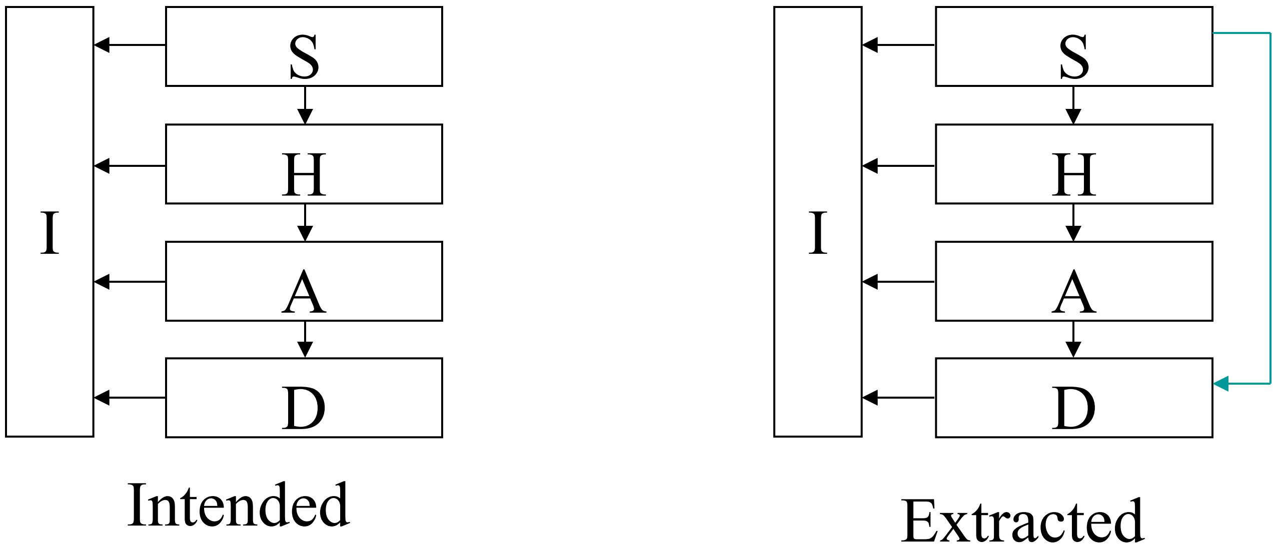
"Actual" module architecture
(extracted from the implementation)

Conceptual Integrity



Reinder J. Bril, r.j.bril@tue.nl
TU/e Informatica, System Architecture and Networking

Conformance



Causes when "intended" and "extracted" differ:

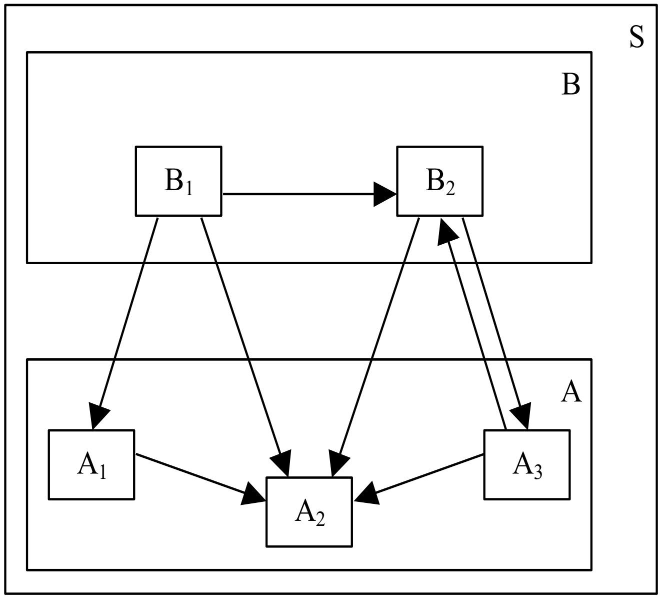
1. "intended" is wrong (e.g. out-of-date): improve;
2. "extracted" is wrong: improve;
3. implementation is optimized for, e.g., speed ⇒ refinement.

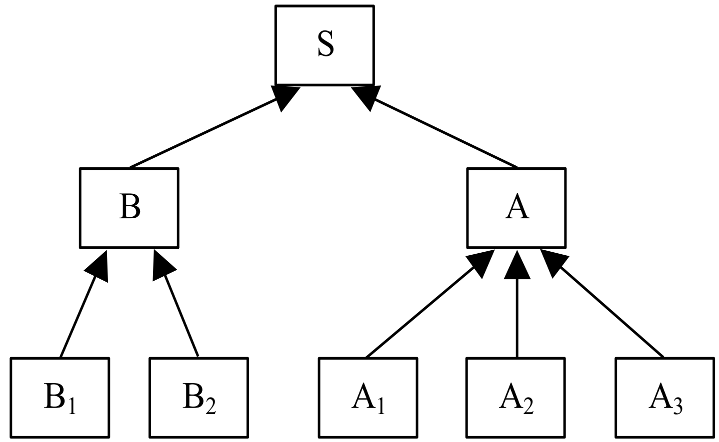
Reinder J. Bril, r.j.bril@tue.nl
TU/e Informatica, System Architecture and Networking

Application domain

- Ensure conformance to an architecture !
  - Keep the architecture up-to-date
- Approach using relation algebra (RPA):
  - Represent the “intended” architecture in RPA.
  - Extract the “derived” architecture from the implementation, and represent in RPA.
  - Express “conformance” in RPA.
  - Ensure conformance by means of verification (using RPA) and improvements (i.e. control).

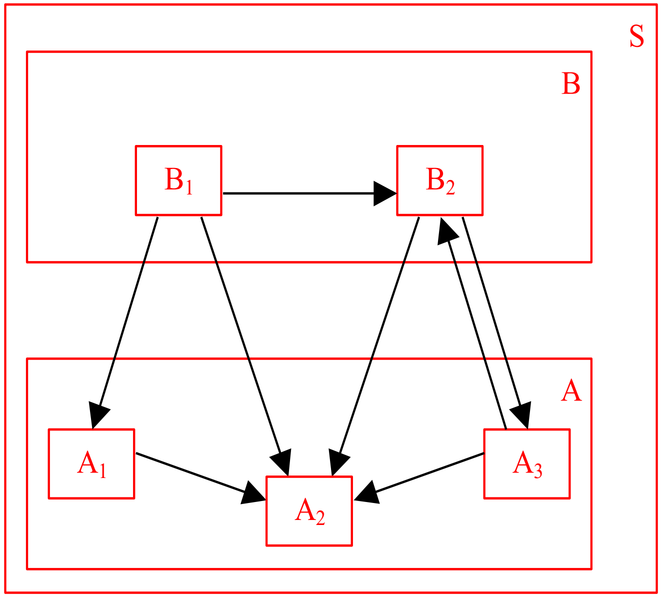
System Representation





System S is *balanced, and*
the decomposition tree has *3 levels*

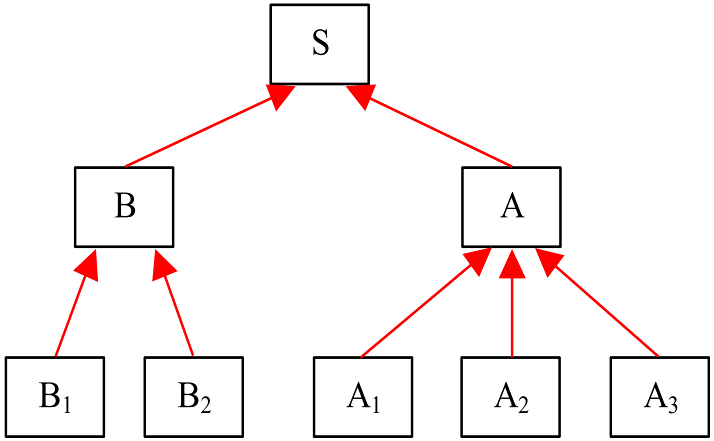
Relation Algebra: Example



Set of *Entities* E:

$E = \{ S, A, A_1, A_2, A_3, B, B_1, B_2 \}$

Relation Algebra: Example



Part-of relation P:

$$P = \{ \langle B, S\rangle, \langle A, S\rangle, , \langle B_1, B\rangle, \langle B_2, B\rangle , \langle A_1, A\rangle, \langle A_2, A\rangle, \langle A_3, A\rangle \}$$

A part-of relation:

- describes the decomposition tree;
- is both: *functional* and *a-cyclic*.

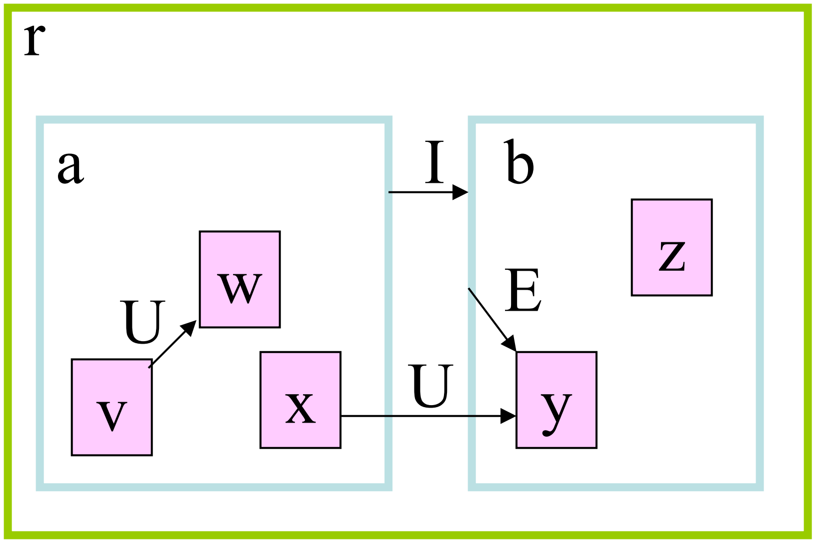
Example: Overview of Operations on Relations

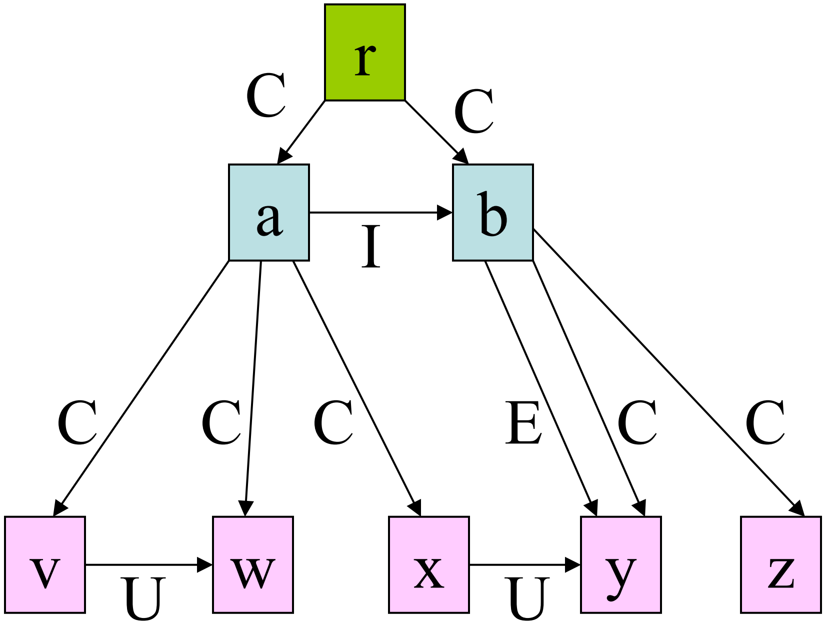
- $A^{-1} = \{\langle y,x\rangle \mid \langle x,y\rangle \in A\}$
- $A - B = \{\langle x,y\rangle \mid \langle x,y\rangle \in A \text{ and } \langle x,y\rangle \notin B\}$
- $A \cup B = \{\langle x,y\rangle \mid \langle x,y\rangle \in A \text{ or } \langle x,y\rangle \in B\}$
- $A \cap B = \{\langle x,y\rangle \mid \langle x,y\rangle \in A \text{ and } \langle x,y\rangle \in B\}$
- $A;B = \{\langle x,z\rangle \mid \langle x,y\rangle \in A \text{ and } \langle y,z\rangle \in B\}$
- $A^{+} = \bigcup_{n=1} R^{n}$, where $R^{n} = R;R^{n-1}$ for $n >= 2$
- $A^{*} = A^{+} \cup I$
- $A \emptyset B = \{\langle x,y\rangle \mid \langle x,v\rangle \in A \text{ and } \langle y,v\rangle \in B\}$
- $A \uparrow B = B^{-1} ; A ; B$ (lifting)
- $A \downarrow B = B ; A ; B^{-1}$ (lowering)

Operators in Relational Algebra

Union	I + E = {(a,b), (b,y)}
Intersection	E ^ C = {(b,y)}
Difference	C - E = {(r,a), (r,b), (a,v), (a,w), (a,x), (b,z)}
Inverse	inv E = {(y,b)}
Composition	I o E = {(a,y)}
Identity	id = {(r,r), (a, a), (b,b), (w,w) … }
Transitive Cl.	C+ = {(r,a), (r, b), (r,v), (r,w), (r,x), (r,y), (r,z), (a,v), (a,w), (a,x), (b,y), (b,z)}
Reflex. T.C.	C* = ID + C+

Example Typed Graph





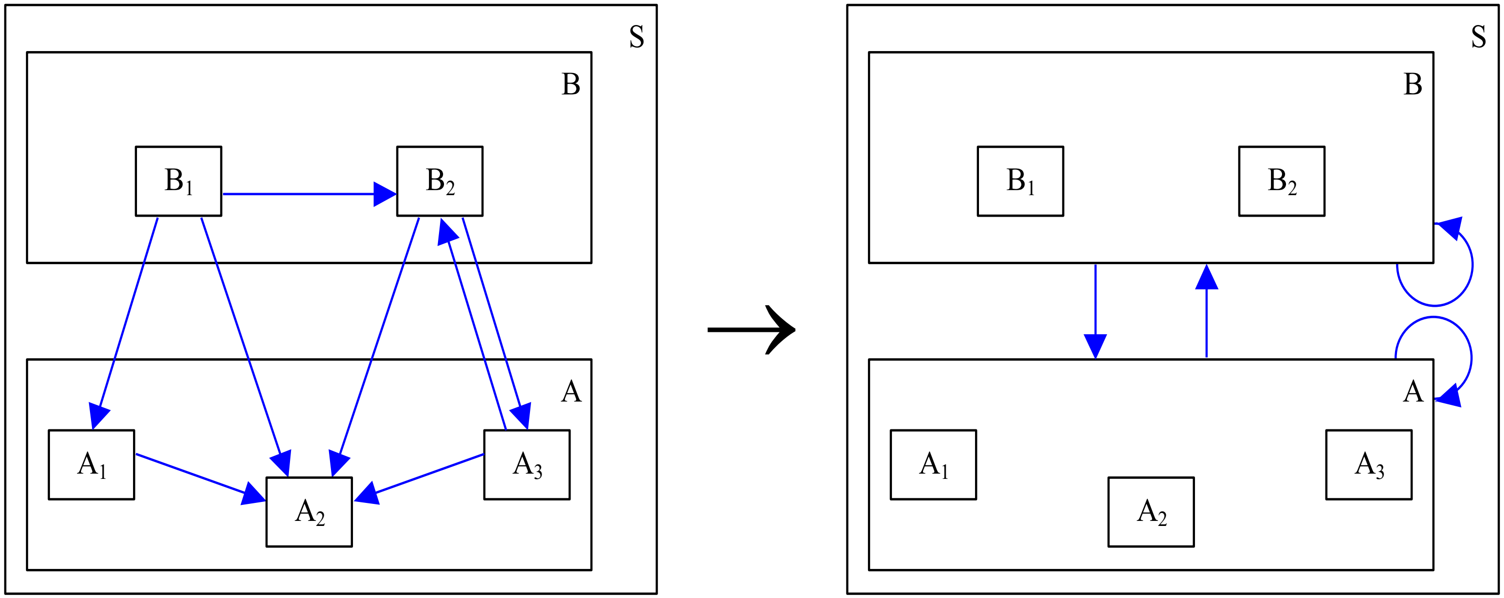
C = { (r,a), (r,b), (a,v), (a,w) (a,x), (b,y), (b,z) }

I = { (a,b) }

E = { (b,y) }

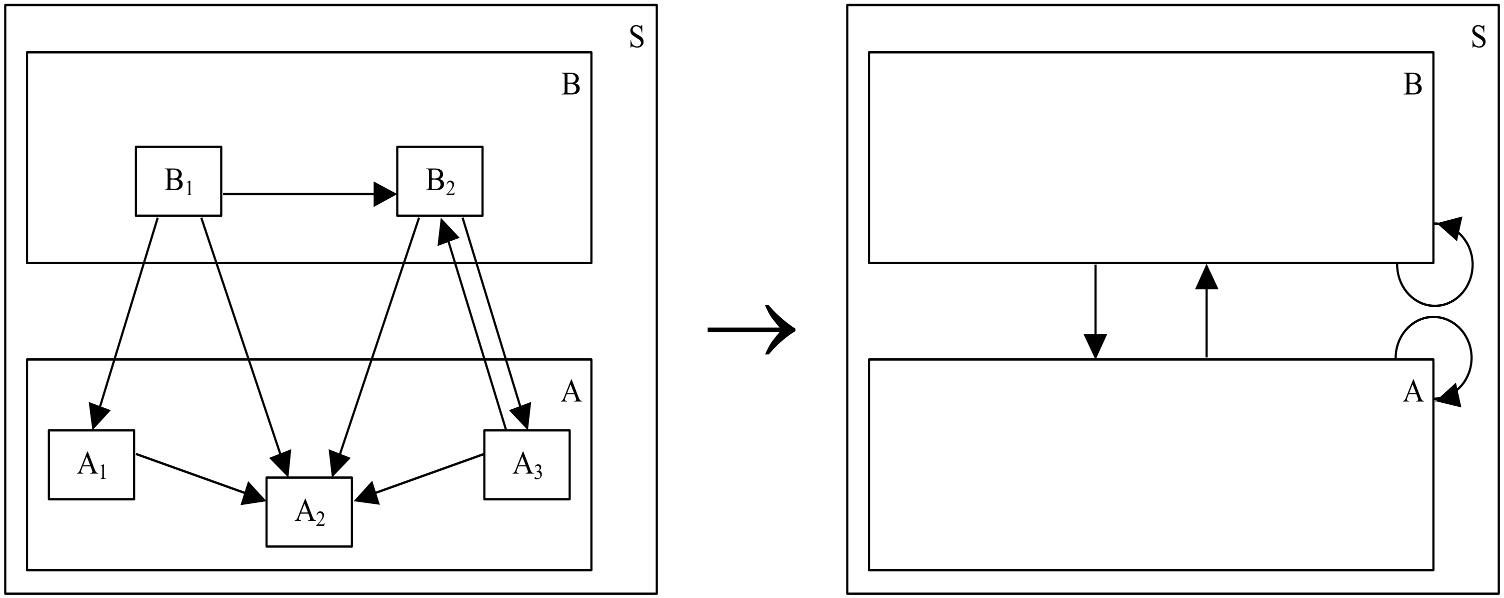
U = { (v,w), (x,y) }

Lifting (2)



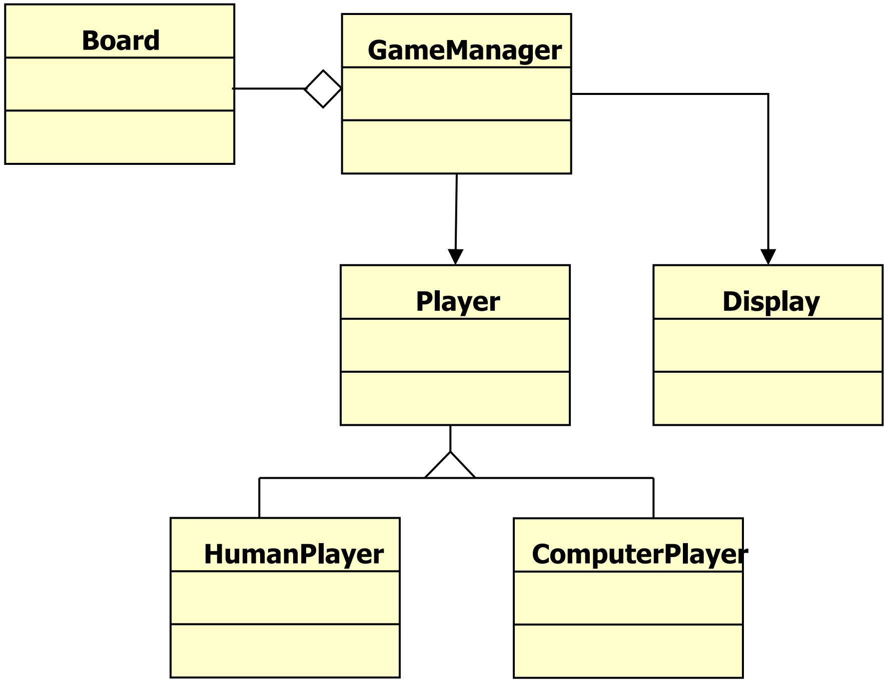
Reinder J. Bril, TU/e Informatica, System Architecture and Networking

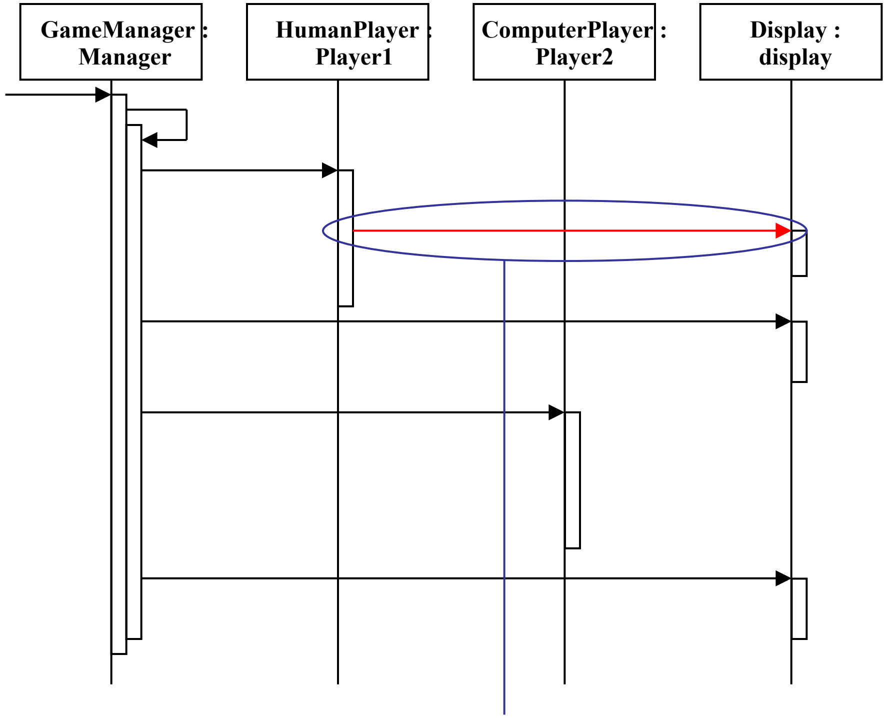
Hiding



Hiding the decomposition structure of both A and B

Example: “Class diagram – MSC”





There is no dependency between HumanPlayer and Display in the class diagram

Example: “Class diagram - MSC”

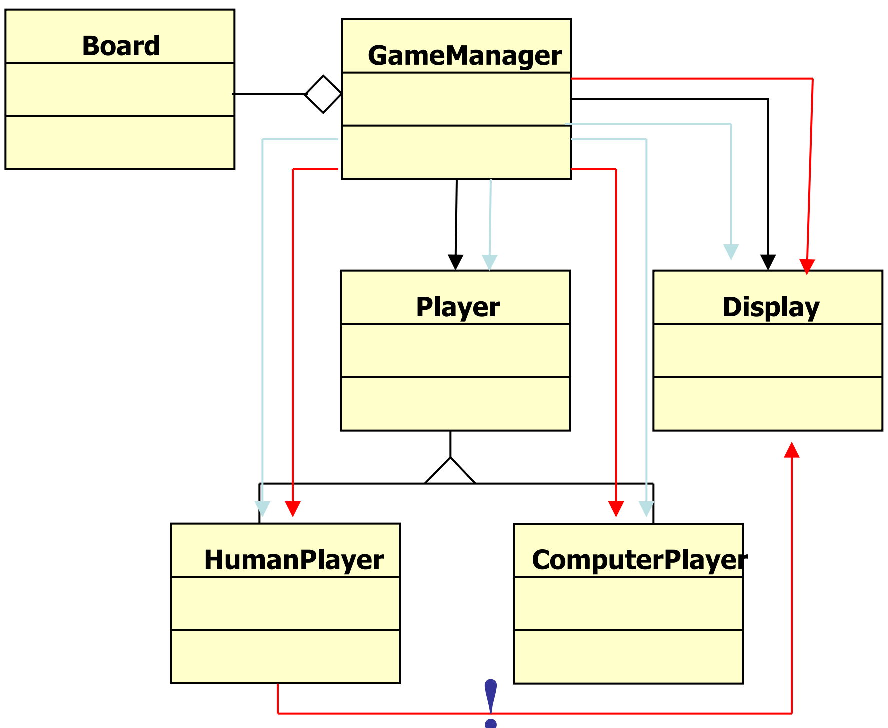
Class Diagram in RPA:

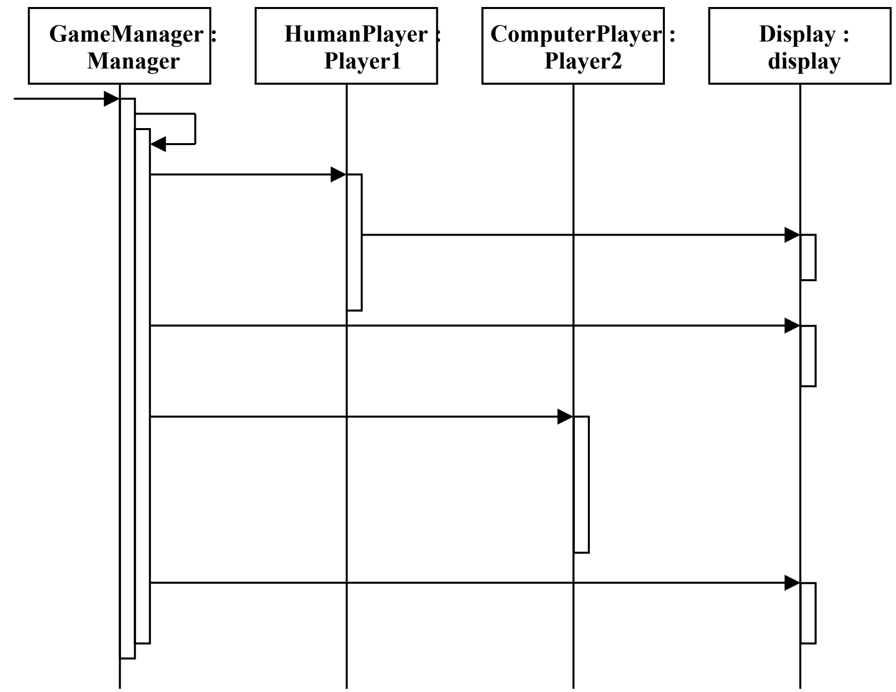
- CLASS = {GameManager, Board, Player, HumanPlayer, ComputerPlayer, Display}
- METHOD = {GameManager.play, GameManager.stop, Player.setToken, HumanPlayer.getNextMove, HumanPlayer.setToken, ComputerPlayer.getNextMove, ComputerPlayer.setToken, Display.printLn, Display.printBoard}
- IMPLEMENTS = {<GameManager.play,GameManager>, …. }
- INHERITANCE = {<HumanPlayer,Player>,<ComputerPlayer,Player>}
- DEPENDENCY = {<GameManager,Player>, <GameManager,Display>}
- AGGREGATION = {<GameManager,Board>}

Example: "Class diagram - MSC"

- MSC in RPA:
  - OBJECT = {Manager, Player1, Player2, display}
  - TYPE = {<Manager, GameManager>,
    <Player1,HumanPlayer>,
    <Player2,ComputerPlayer>,<display,Display>}
  - CALL = $\{c_1,c_2,c_3,c_4c_5\}$
  - NEXT = {<c_1,c_2>,<c_2,c_3>,<c_3,c_4>,<c_4,c_5>}
  - CALLER = {<Manager,c_1>,<Player1,c_2>,<Manager,c_3>,
    <Manager,c_4>,<Manager,c_5>}
  - CALLEE = {<Player1,c_1>,<display,c_2>,<display,c_3>
    <Player2,c_4>,<display,c_5>}
  - MESSAGE = {<HumanPlayer.getNextMove,c_1>,
    <Display.printLn,c_2>,
    <Display.printBoard,c_3>,
    <ComputerPlayer.getNextMove,c_4>,
    <Display.printBoard,c_5>}

Example: "Class diagram - MSC"





- Rule
  - (CALLER ® CALLEE) " TYPE µ (DEPENDENCY ↓ INHERITANCE*)

Uses of Relational Languages

1. Enforce architecture rules. **Holt 96, Feijs 98, Knodel 08**
2. Lift dependency edges. **Holt 98, Feijs 1998**
3. Find design pattern instances. **Consens 98, Beyer 02**
4. Find violations of patterns. **Guo 99**
5. Find anti-patterns. **vanEmden 02, Feijs 98**
6. Change impact analysis. **Feijs 98**
7. Specify extraction from syntax. **Lin 08**
8. Find source of dependency. **Fahmy 01, Feijs 98**
9. Locate uses of protocols. **Wu 01**
10. Type inference using transitive closure. **vanDeursen 99**

Slide by R. Holt

Time for Reflection

- What are the ***strengths*** and ***weaknesses*** of the Relational Algebra Approach towards checking conformance between Architecture and Implementation?

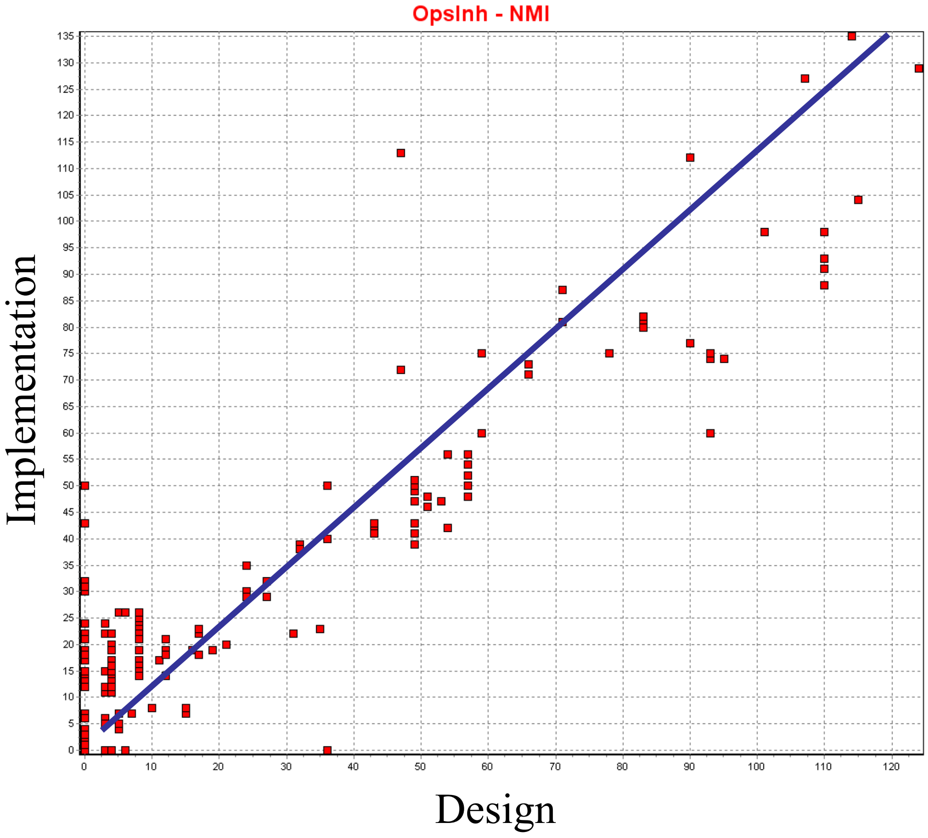
Checking Correspondence through metrics

- Horizontal axis:

  metric from design
- Vertical axis:

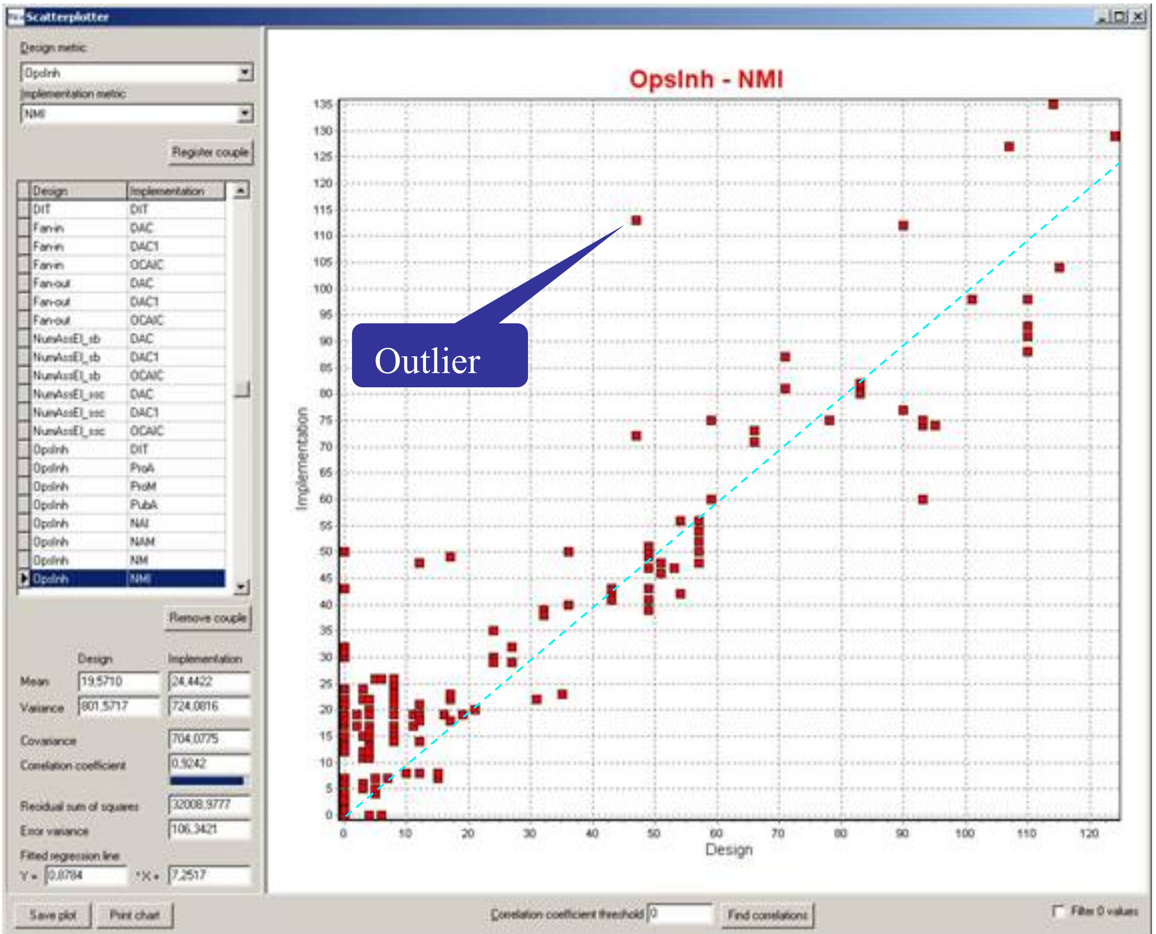
  metric from code
- Dots:

  (*metric*(class in design), *metric*(class in code))
- Here: Number of Methods Inherited



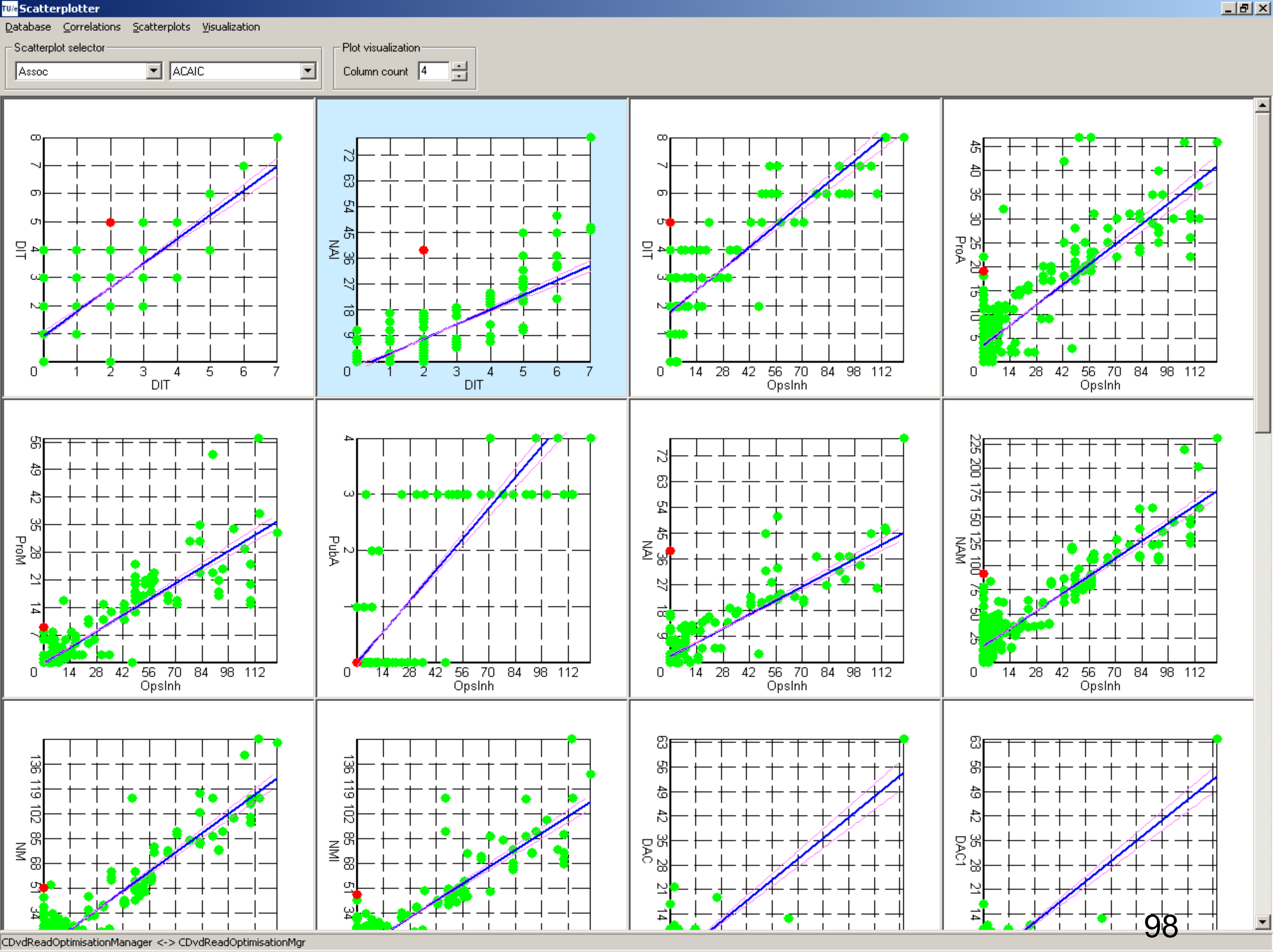
M.Sc. Thesis Dennis van Opzeeland TU Eindhoven

Tool for Correspondence Checking



- X-Axis
  - Metrics of Design
- Y-Axis
  - Metrics of Implementation

- Points represent Classes
- Points off the diagonal indicate (critical) outliers


Scatterplotter
Database
Correlations
Scatterplots
Visualization
Scatterplot selector
Assoc
ACAIC
Plot visualization
Column count
4
DIT
NAI
ProA
OpsInh
ProM
PubA
NAM
NM
NMI
DAC
DAC1
CDvdReadOptimisationManager <-> CDvdReadOptimisationMgr

Akerman & Tyree

- Architecture decisions are the primary representation of architecture

- Architecture results from effective decision making, not from architectural view construction*

- Architecting is primarily concerned with:
  - architecture assets
  - the business-driven decisions that transform these assets
  - the roadmap that implements these decisions

*J. Tyree and A. Akerman, "Architecture Decisions: Demystifying Architecture," IEEE Software, vol. 22, pp. 19-27, March. 2005

Akerman & Tyree:
Problems with current architecture development & descriptions methods

- Lack of Focus on What's Important
- Lack of Precision and Clarity
- Lack of Repository Support
- Lack of Support for Impact Analysis (Decisions to Concerns, Decisions to Decisions, and Decisions to Architecture Assets)
- Difficulty in Linking with the Views
- Lack of Support for Temporal Mapping

Akerman & Tyree: Solution

- Architecture meta-model
- Focus on “information about architecture that an organization cares about” instead of diagrams and views. Architecture is captured as an ontology.
- Tool support to enable effective decision making and “on-demand” view creation

Beyond structure views

How about a views for?

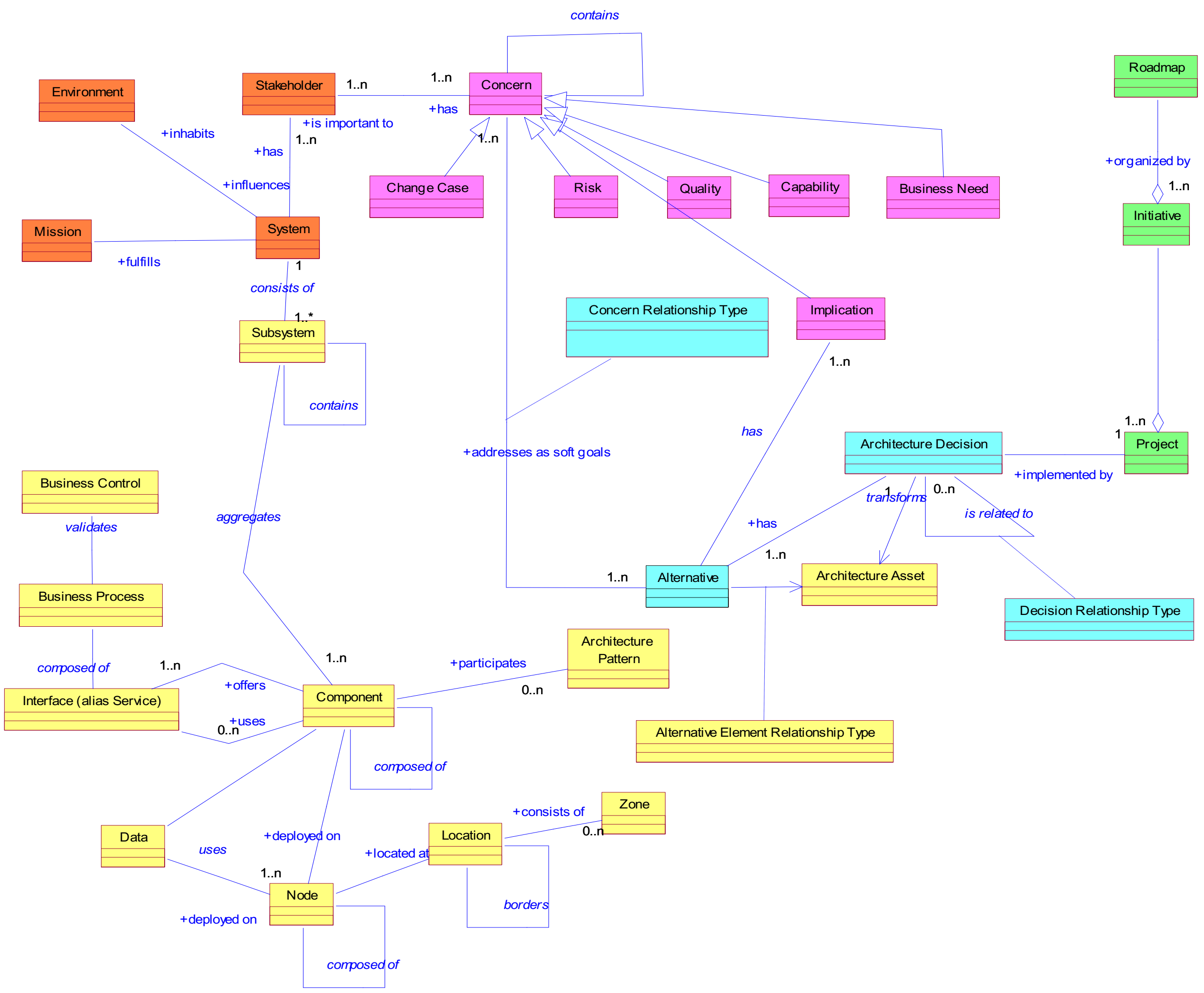
- security
- safety
- performance?

Beyond Views

- How about recovery of
  - Design principles
  - Design rationale

Architecture Meta-Model (Details)

- Concerns
- Decisions
- Roadmap
- Assets



Summary Reverse Architecting

- Rev. Arch. is a labour intensive activity
  - Manual (re)discovery of abstractions
  - Have a purpose in mind

- Rev. Arch. is a step in managing:
  - conformance of the implem. to the architecture
  - conceptual integrity

- Need a method for focussing on what is important

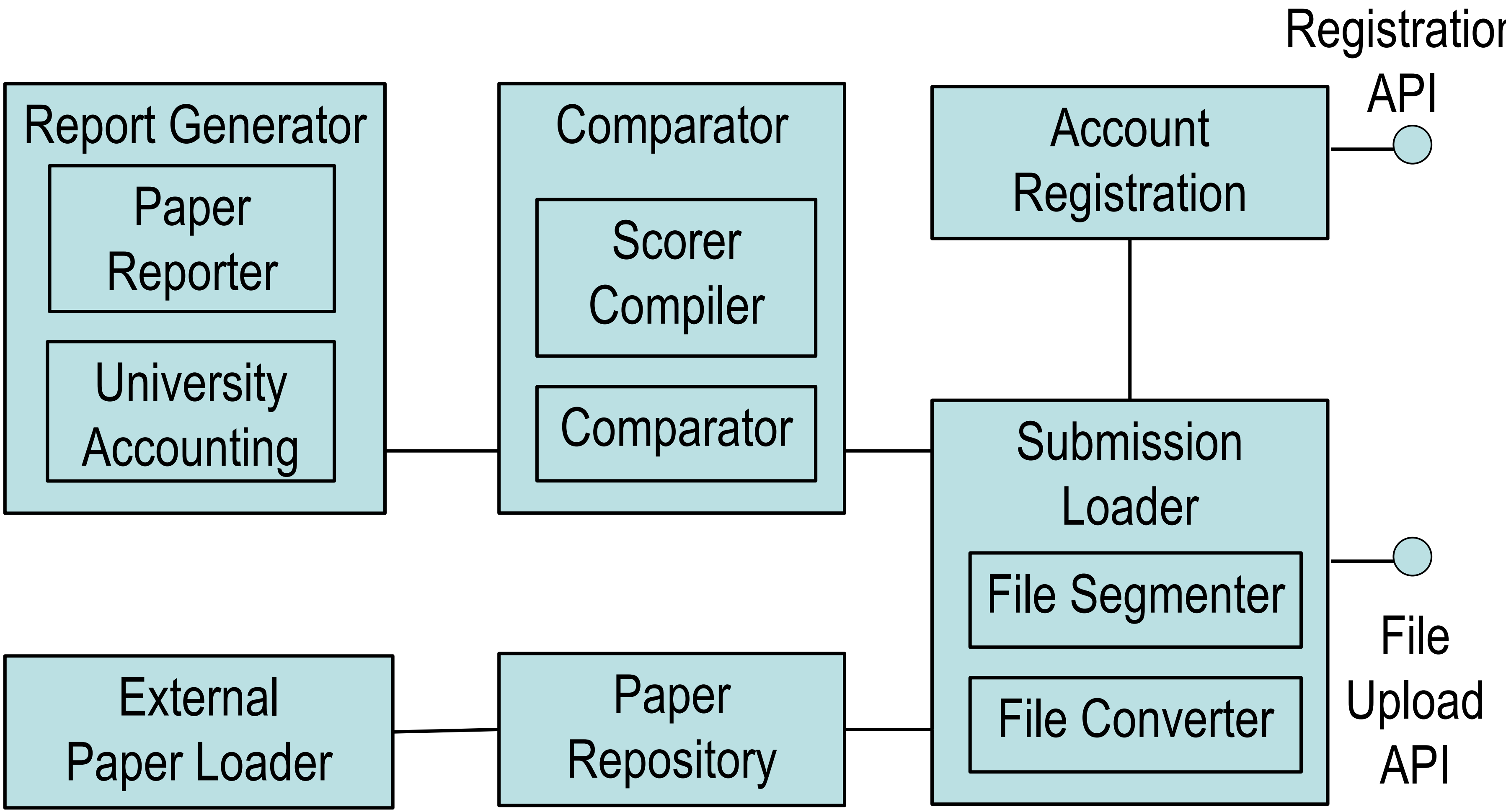
1-slide Summary of Best Architecting Practices

- Get stakeholder involvement & feedback early and frequently
- Understand the drivers for the project (business, politics)
- Understand the requirements incl. quality properties
  - SMART & prioritized
- Develop iteratively and incrementally
- Describe architecture using multiple views
  - abstract, but precise, design decisions & rationale
- Design for change (modularity, low coupling, information hiding, separation of concerns)
- Monitor that architecture is implemented
- **Simplify, simplify, simplify**
- Analyze in an early stage (use maths! and scenarios)
- Regularly update planning and risk analysis
- Get good people, make them happy, set them loose

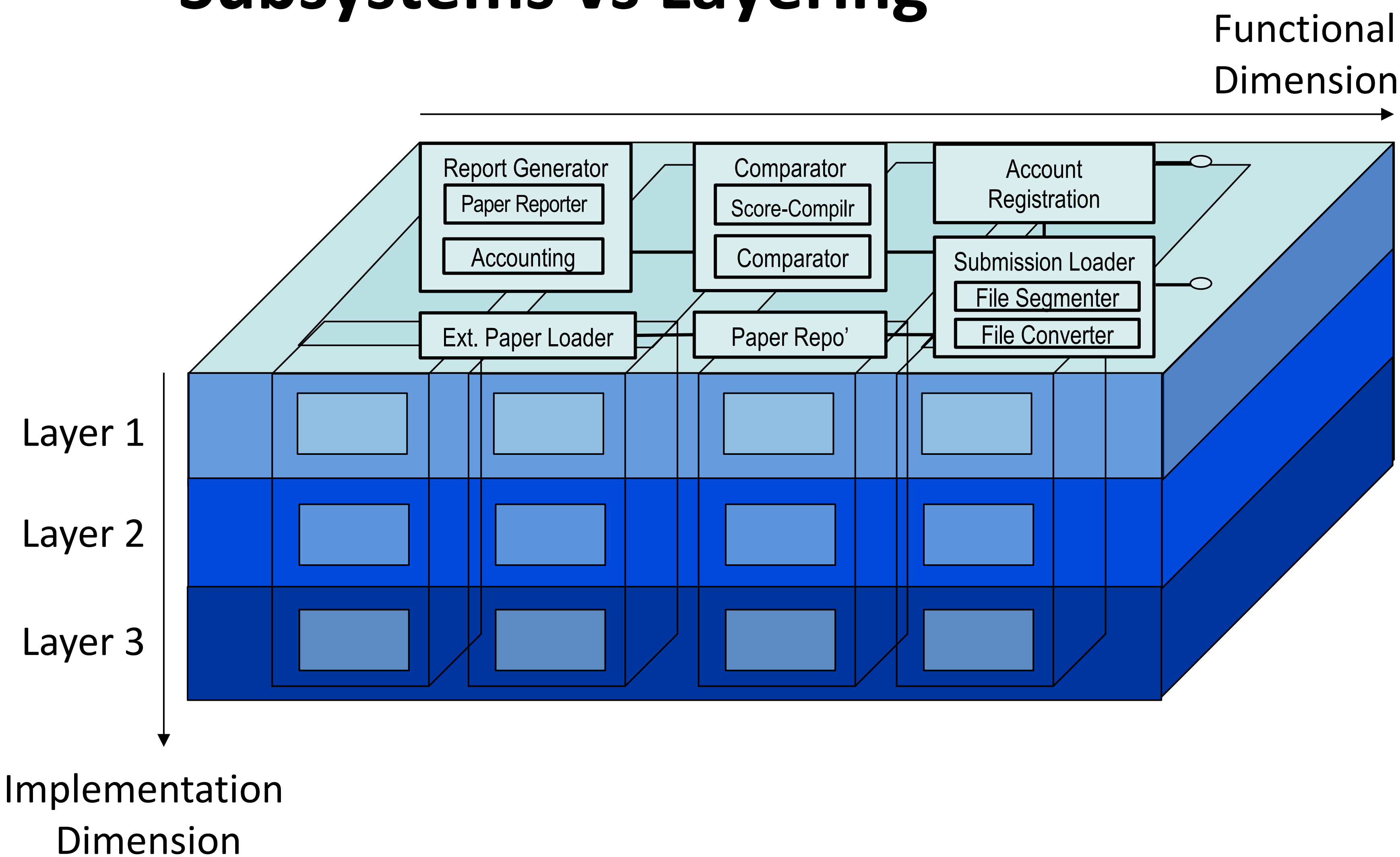
Questions

Example: Automated Plagiarism Checking System

- University can have subscriptions
- University-faculty can make accounts
- Faculty can send in documents for checking
  - Documents are turned into a standard internal format
  - The document is segmented (chapters, section, sentences, …)
  - Document is compared on a sentence by sentence basis.
  - A plagiarism score is produced
  - A report is sent to the person that sent in the document
- The system keeps records of use for producing yearly accounting reports


Registration
API
Report Generator
Paper
Reporter
University
Accounting
Comparator
Scorer
Compiler
Comparator
Account
Registration
Submission
Loader
File Segmenter
File Converter
File
Upload
API
External
Paper Loader
Paper
Repository

Subsystems vs Layering



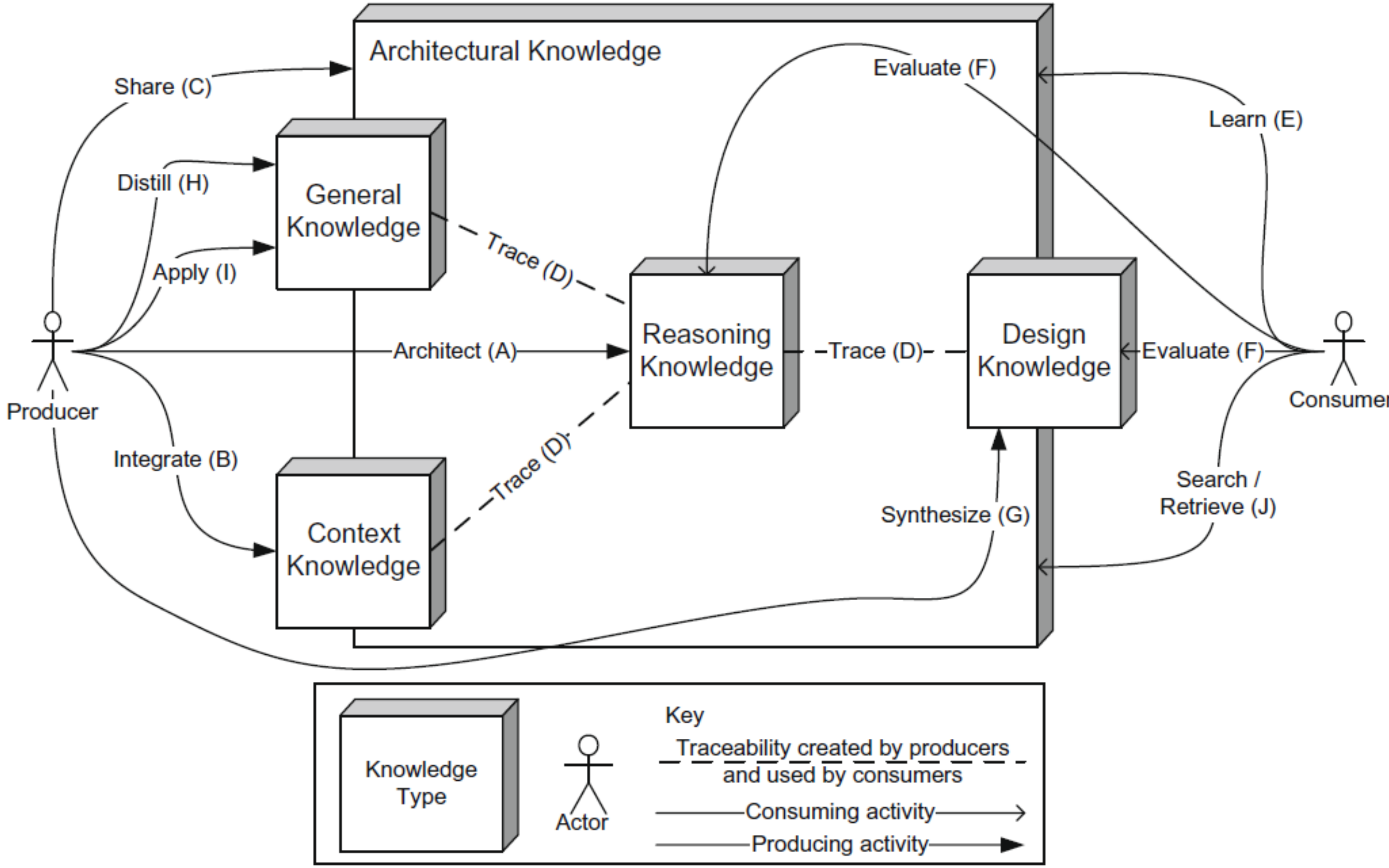
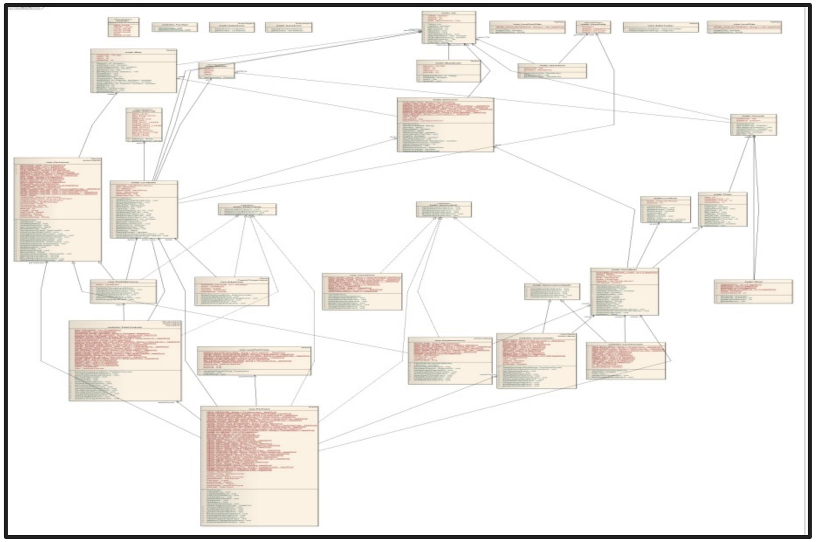


Fig. 2. Architectural knowledge activities.

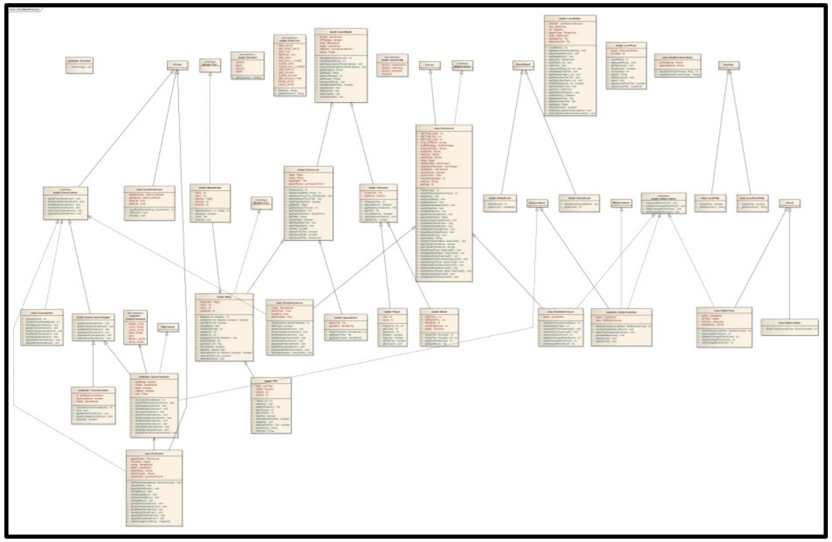
A comparative study of architecture knowledge management tools ☆

Antony Tang [a,*], Paris Avgeriou [b], Anton Jansen [b], Rafael Capilla [c], Muhammad Ali Babar [d]

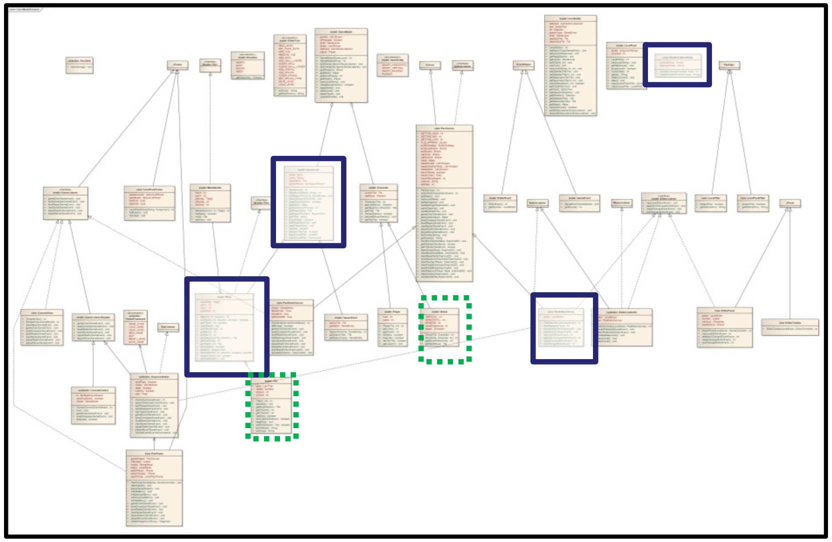
Comparing Reverse and Forward UML



Reverse Engineered Design



Forward Class Design

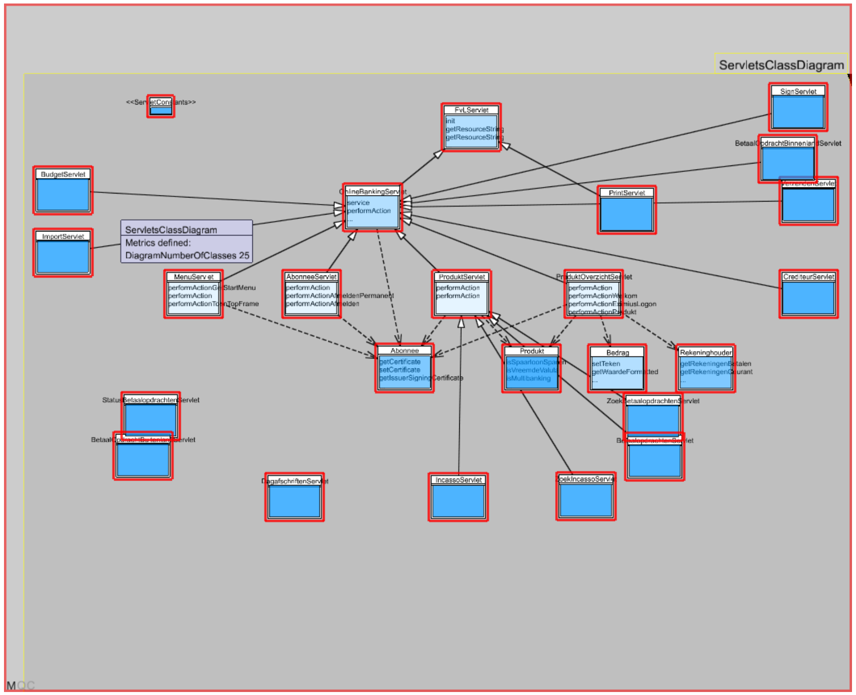


False Positive

False Negative

How is Level of Detail distributed in a diagram?

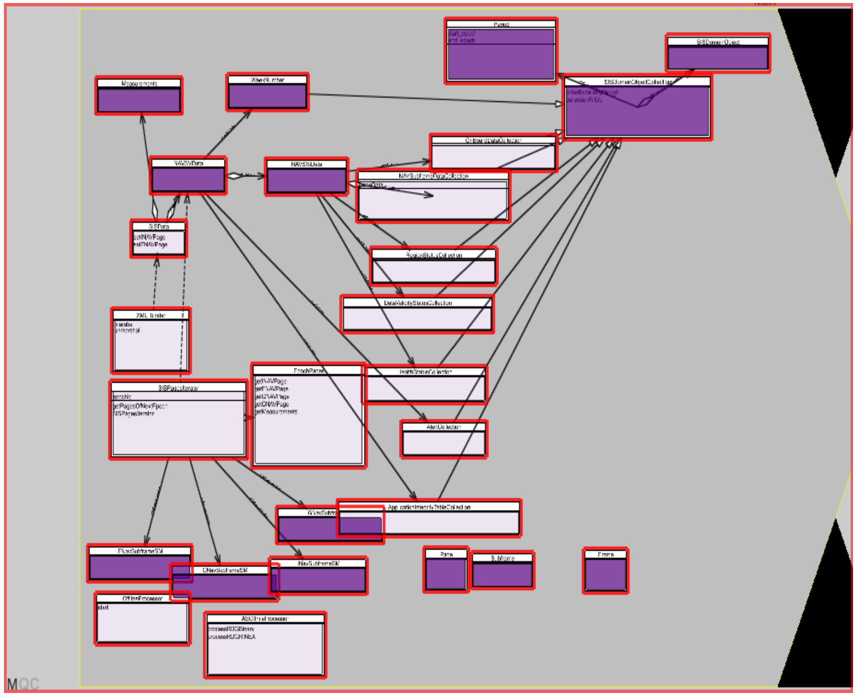
Case 1




High detail


Low detail

Case 2

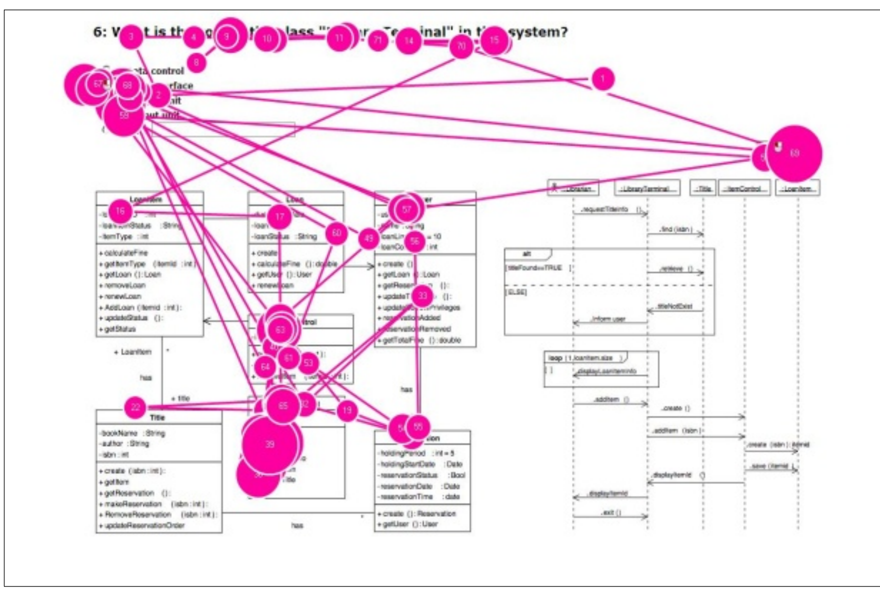



High detail


Low detail

Robin van den Broek B.Sc. Thesis in CS, 2009, Leiden University

What do developers look at anyway?



Example 4+1 Views model