Why Developers Overlook Architecture Degradation Symptoms?

Alessandro Garcia

SEIF Workshop 2013
Rio de Janeiro

LES | DI |PUC-Rio - Brazil

OPUS Group

Software has an “architecture” too!

How to achieve good architectures?

Keep it simple!

Component addresses a single concern

Loosely coupled components

Simple interfaces

It does not matter...

... if the intended architecture is well defined:

but...

... but the program is not compliant to it!

The actual architecture is in the source code:

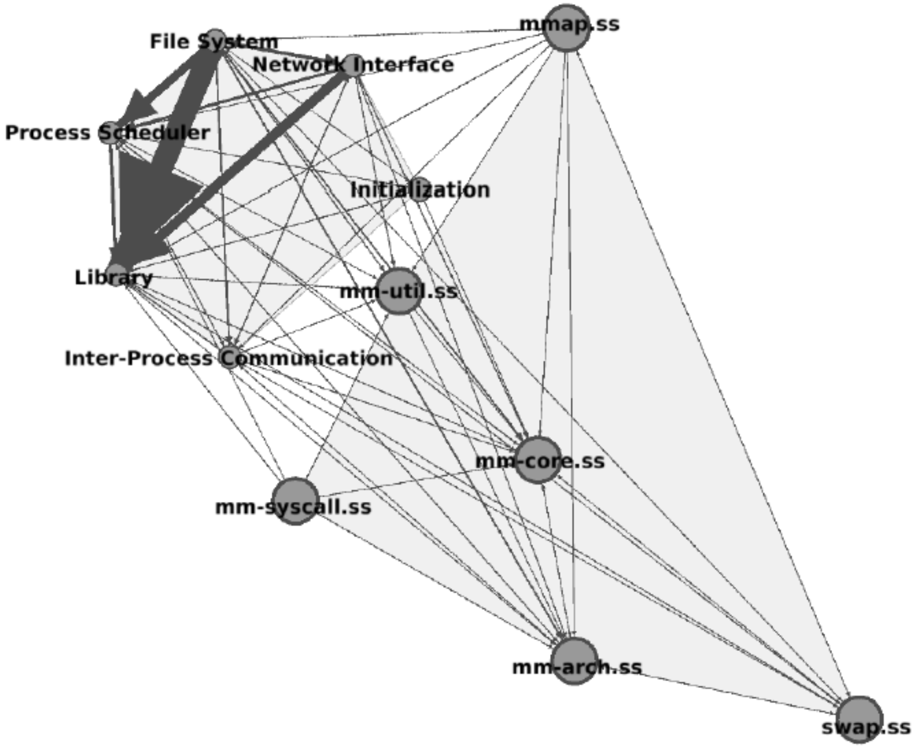
Software Architecture Degradation is…

- continuous quality decay of architecture design
  - evolving systems: changes are made everyday

Architecture software degradation ...

... Why do we care?

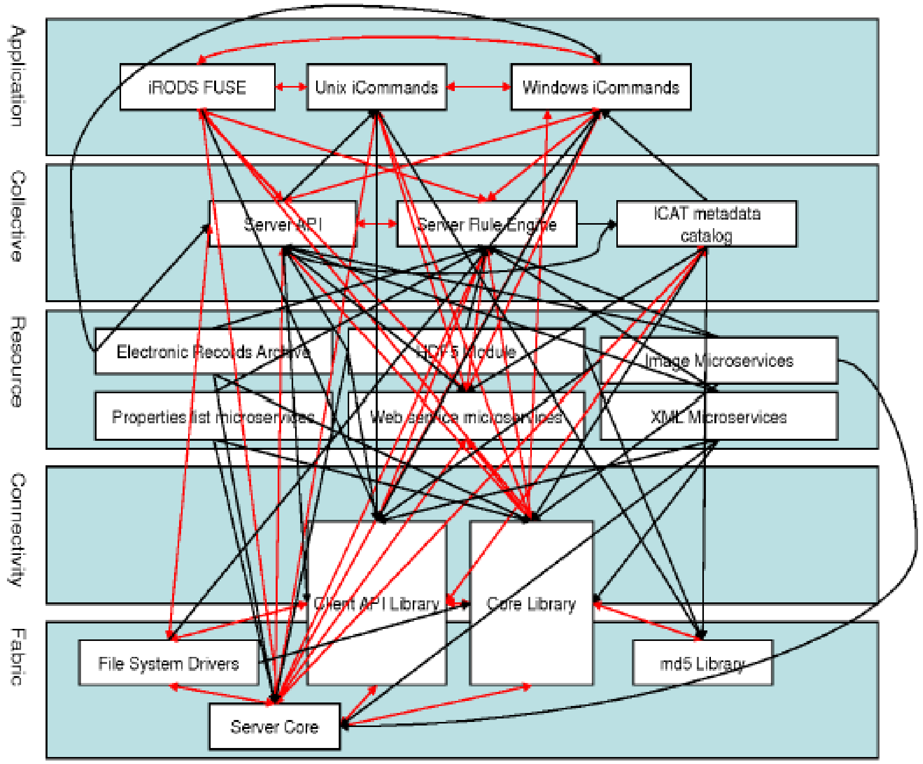
Memory manager component - Linux



Taylor, R. et al. **Software Architecture: Foundations, Theory and Practice**. Wiley Publishing. 2009
Nenad Medvidović. **When, Where, and Why Do Software Systems Architectures Decay?** March 2013.

Why do we care?

Actual architecture - iRODS

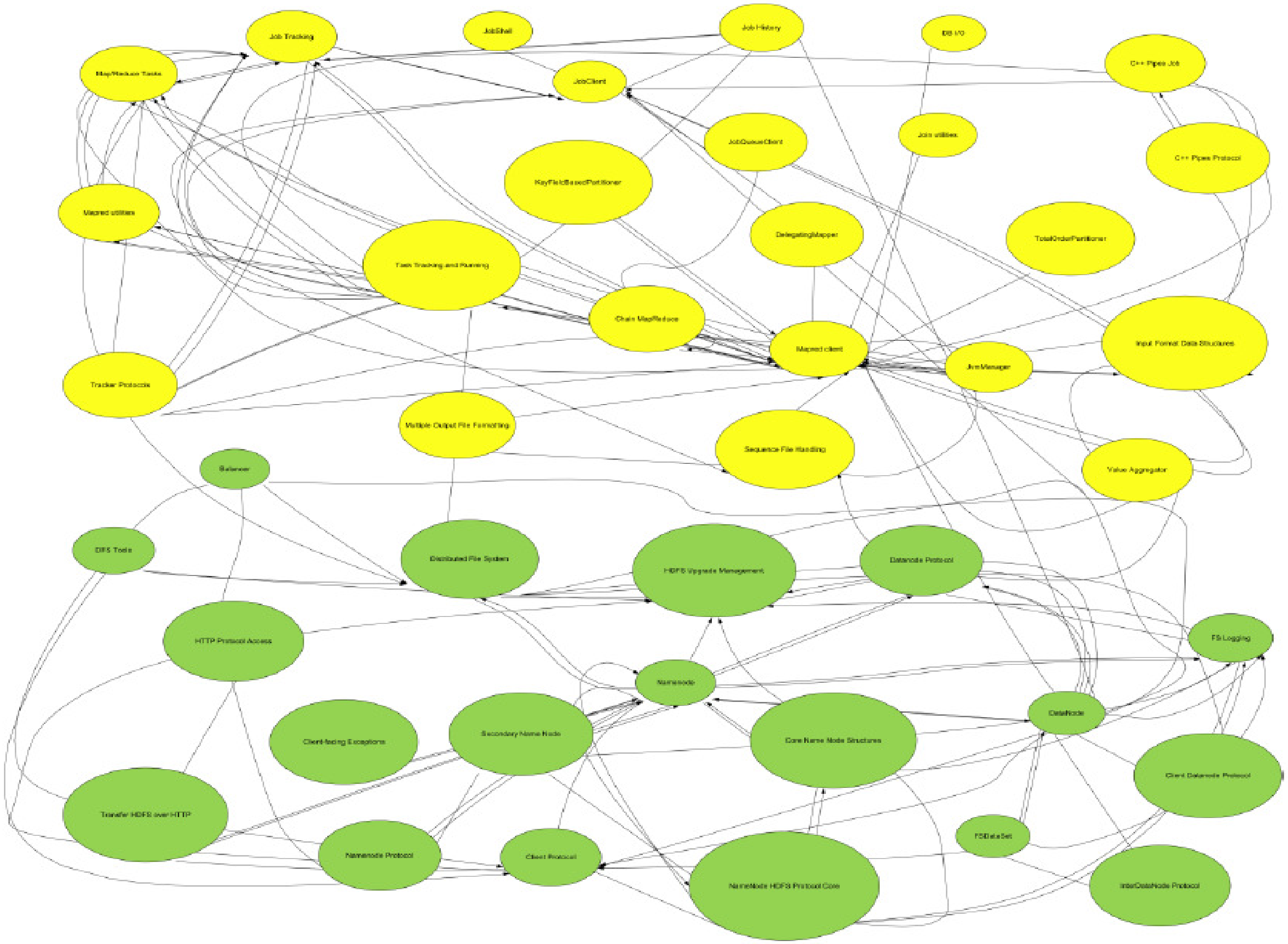


— violations of the intended architecture

Taylor, R. et al. **Software Architecture: Foundations, Theory and Practice**. Wiley Publishing. 2009
Nenad Medvidović. **When, Where, and Why Do Software Systems Architectures Decay?** March 2013.

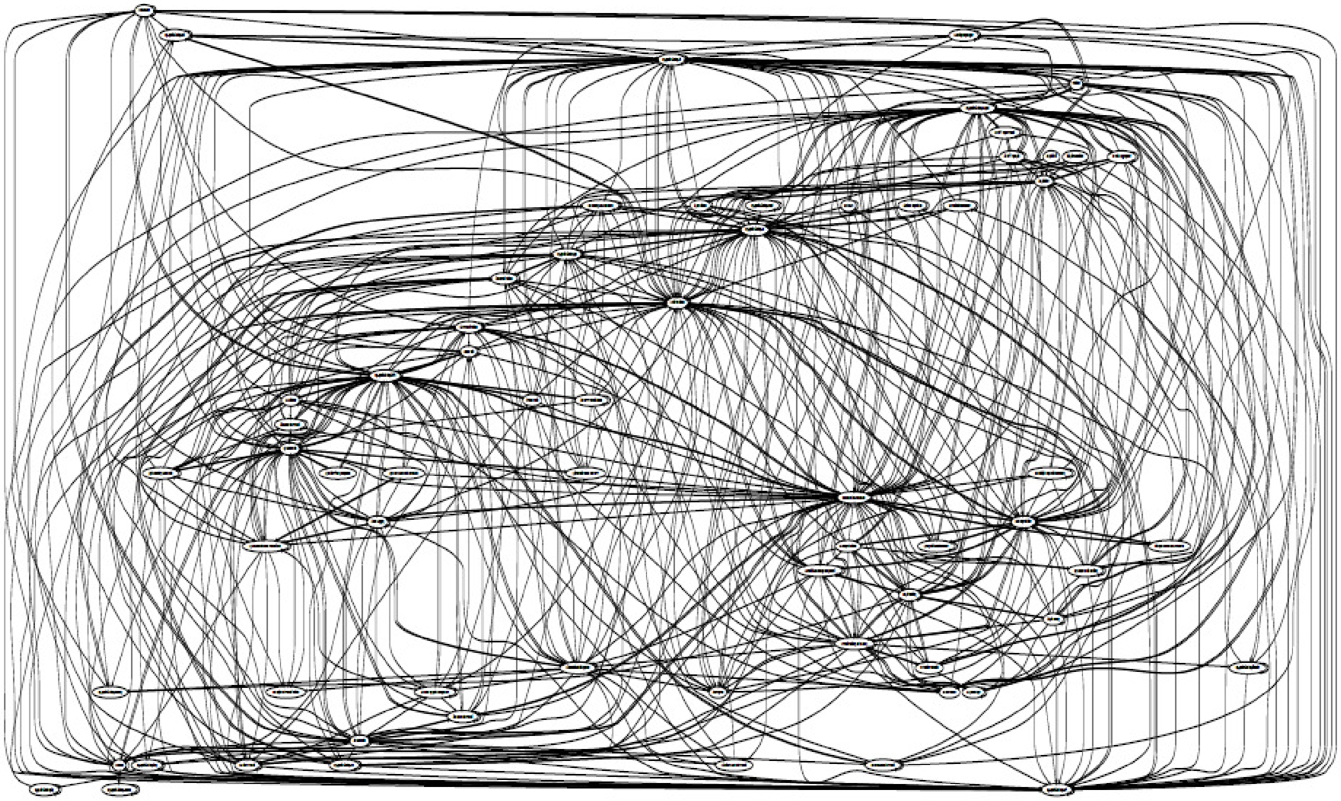
Why do we care?

- **Hadoop**



Why do we care?

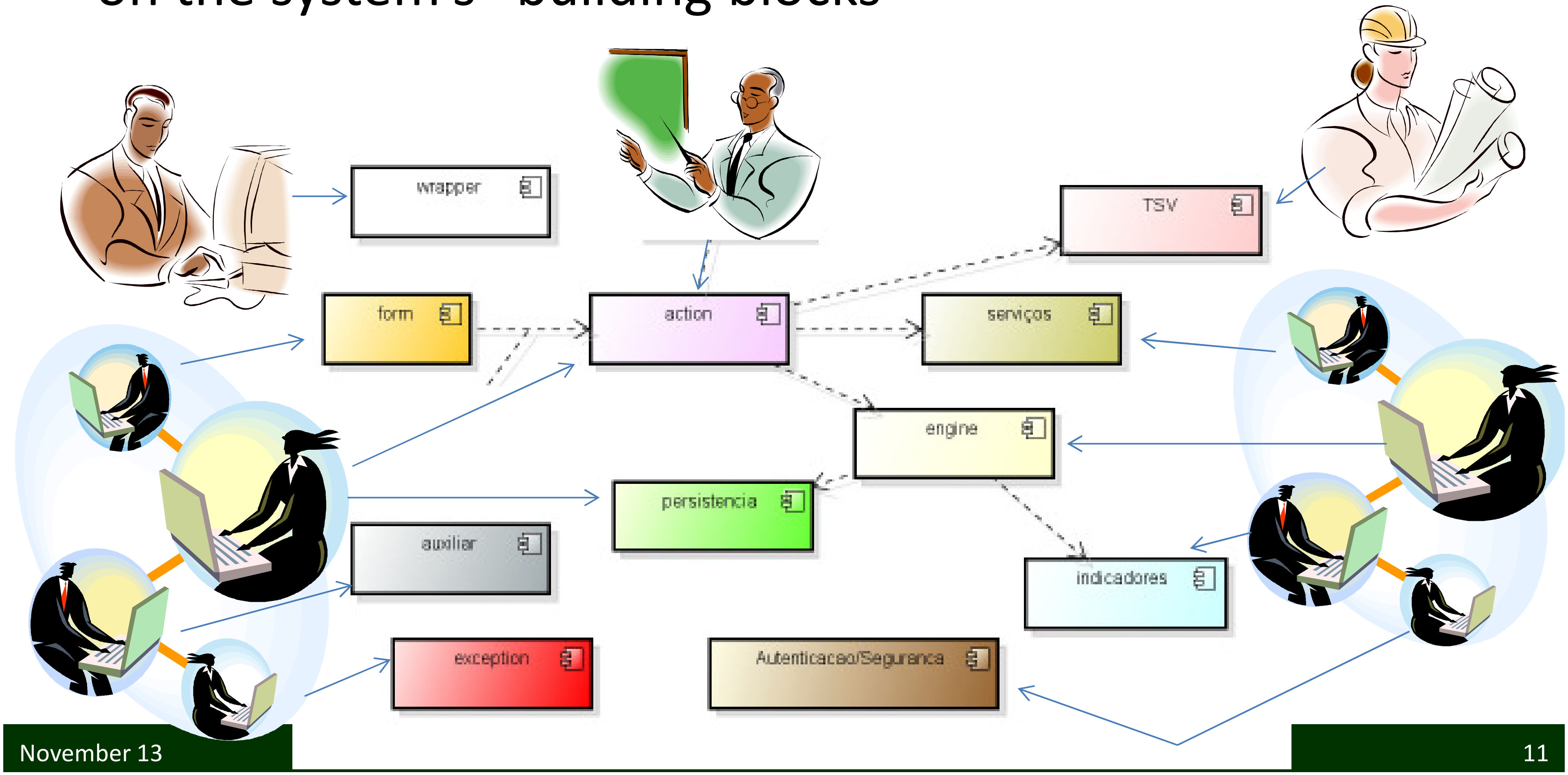
Actual architecture - Hadoop



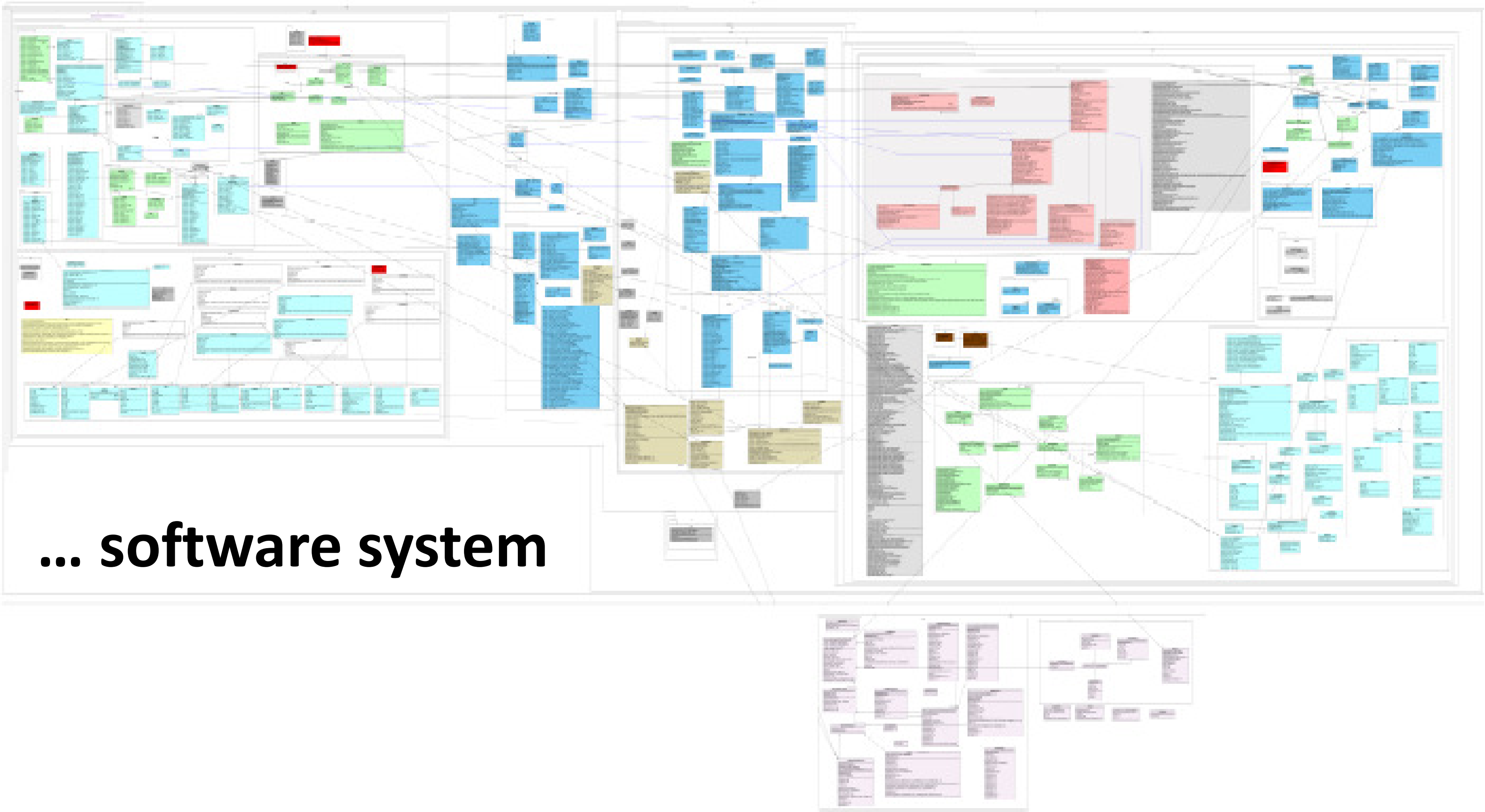
Nenad Medvidović. **When, Where, and Why Do Software Systems Architectures Decay?** March 2013.

Intended architecture of a software system

- … defines how developers actually communicate and work on the system's "building blocks"

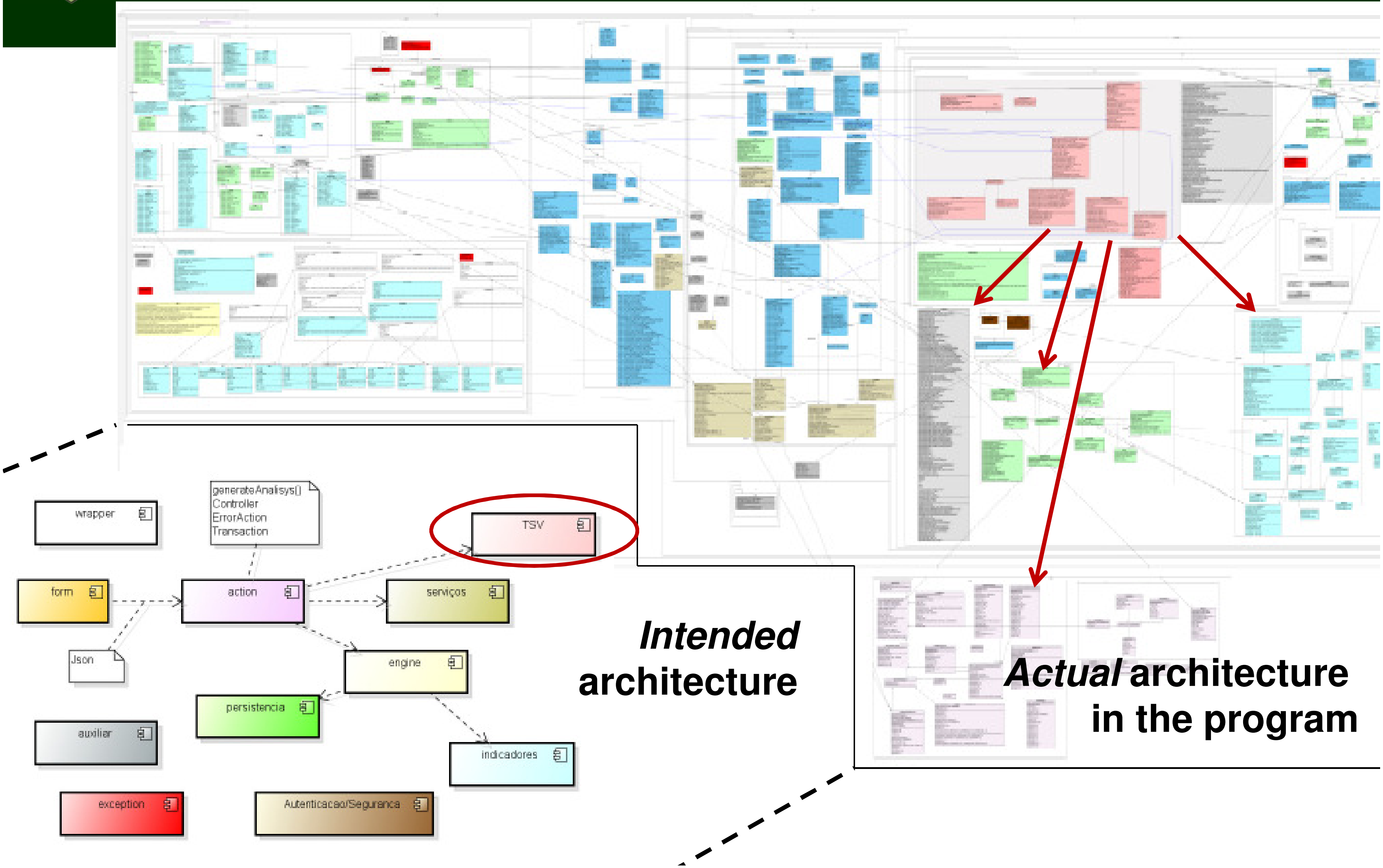


Actual architecture of the same…



… software system

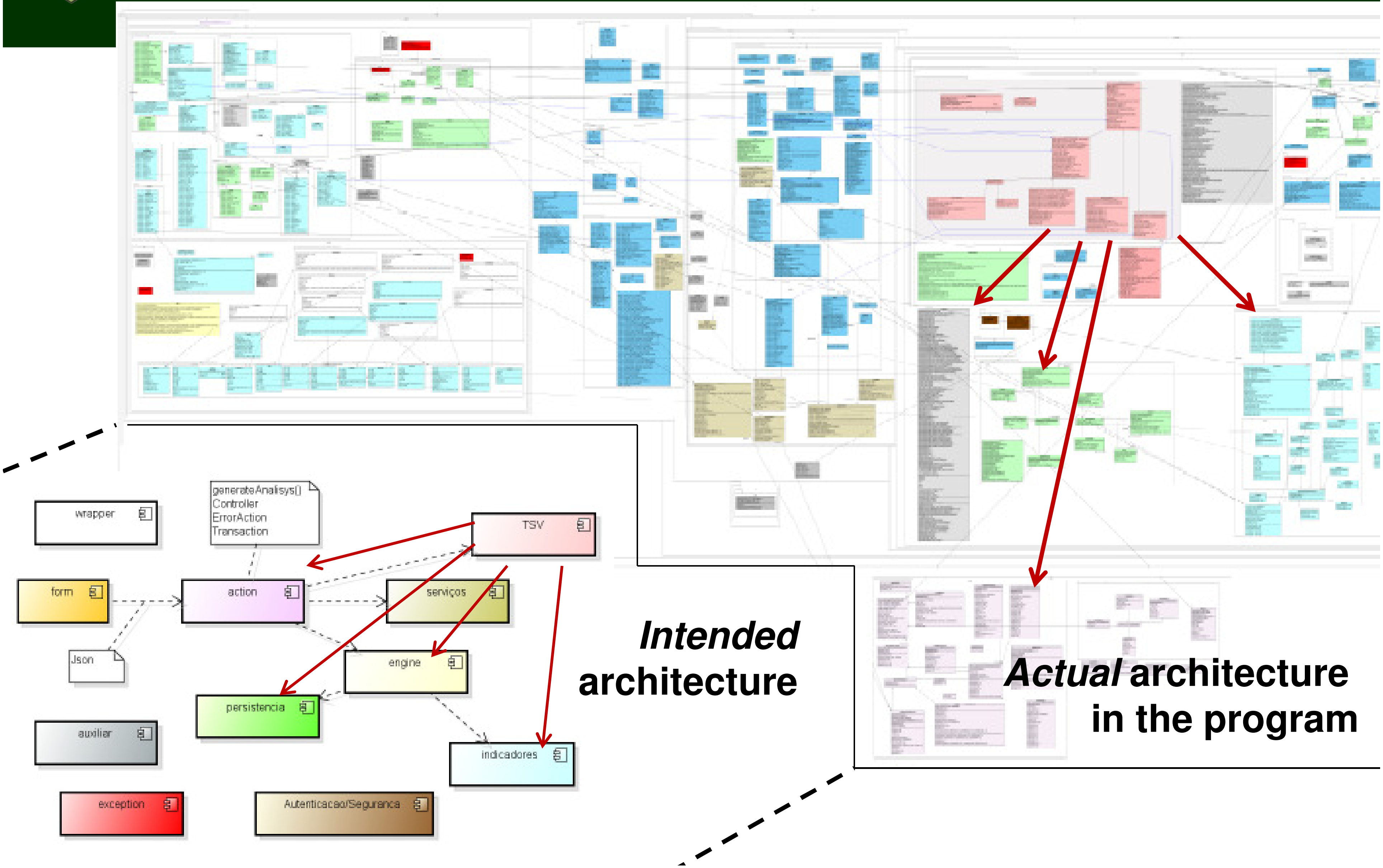
Architectural Erosion



***Intended* architecture**

***Actual* architecture in the program**

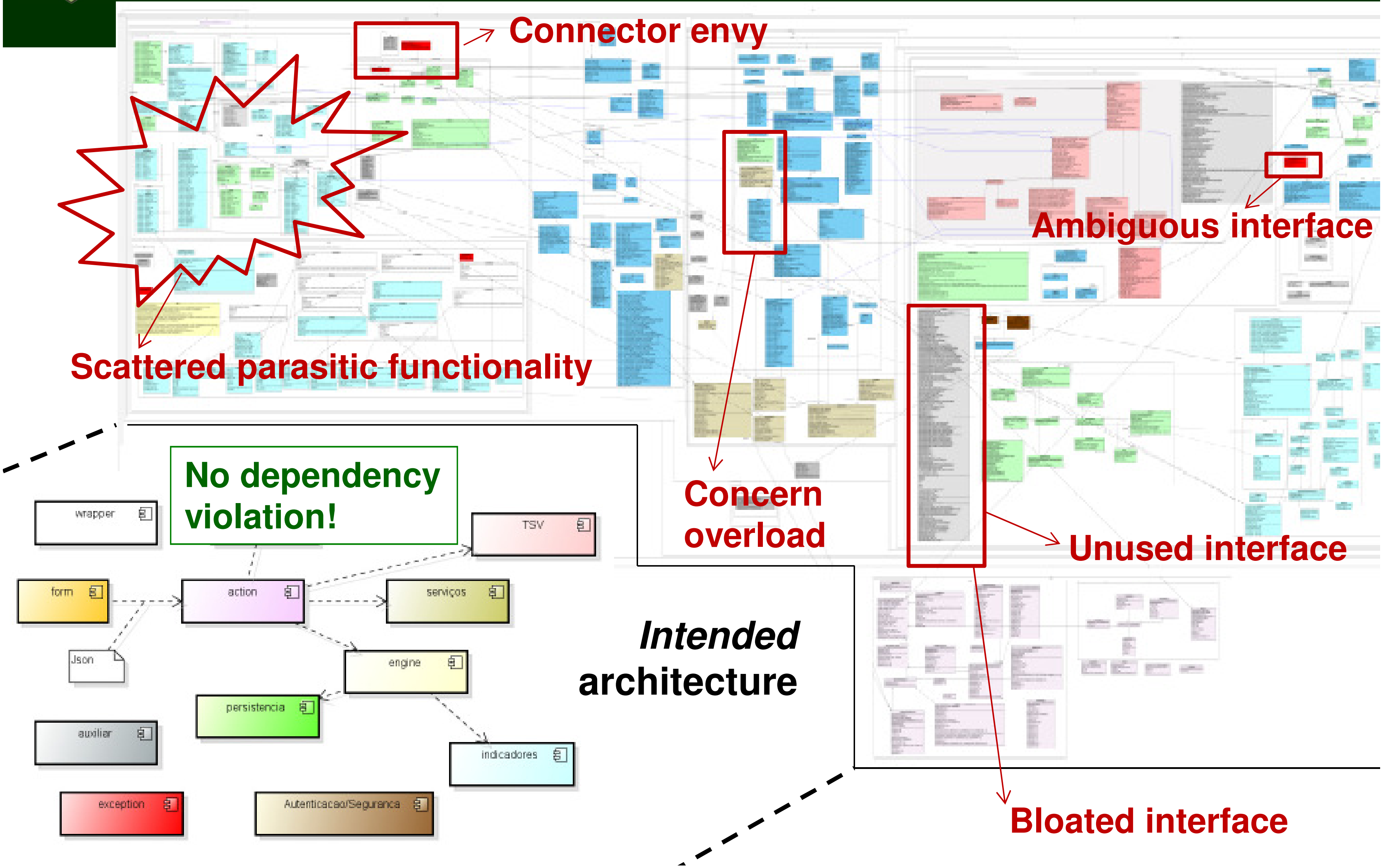
Architectural Erosion



***Intended* architecture**

***Actual* architecture in the program**

Architectural Drift



Connector envy

Ambiguous interface

Scattered parasitic functionality

No dependency violation!

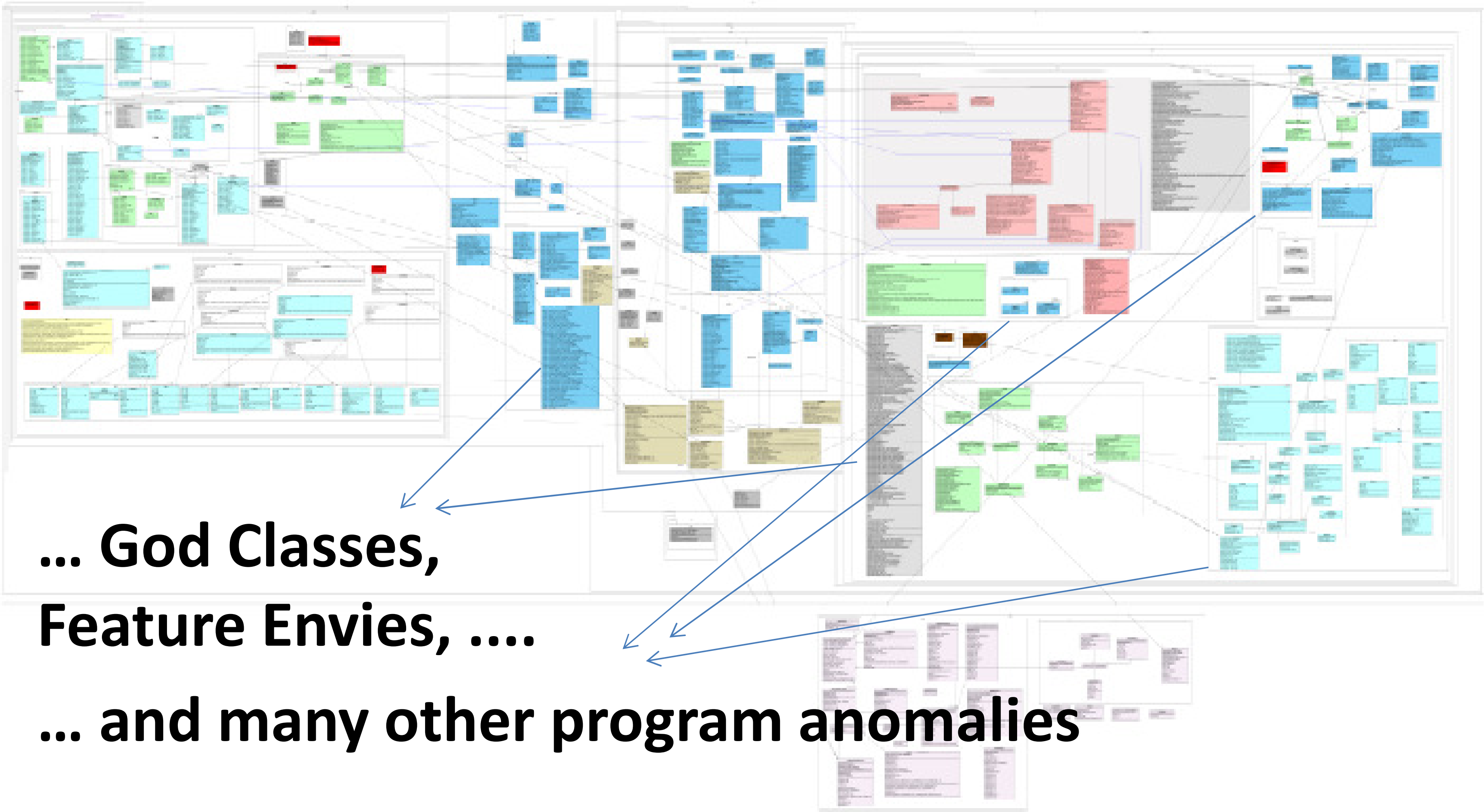
Concern overload

Unused interface

Intended architecture

Bloated interface

Drift often manifest as code anomalies…



… God Classes,

Feature Envies, ….

… and many other program anomalies

Why do we care?

- Netscape, Mozilla, EJB, FindBugs and ArgoUML
  - several years of production
- These projects involved US$ millions
  - ... millions and millions of users in many countries
  - ... dozens of developers
- Degradation affects several software domains:
  - Health care, mobile applications, banking, finantial market analysis, ...

Recent Advances...

- **Architecture recovery** techniques
  - Recovery of actual architectures from source code
- **Drift detection** in actual architectures
  - Metrics-based strategies for programs
- **Erosion detection**
  - Use of DSLs to descrive and check architecture rules

Architecture Recovery techniques are…

- … useless to support detection of architectural problems in the program in these cases
  - they retrieve components, which do not correspond to actual components
    - **reason**: intended software architecture is already diffused; packages do not match architectural components
  - they do not retrieve enough information: interfaces, dependencies, etc…
    - **reason**: intended software architecture is already diffused

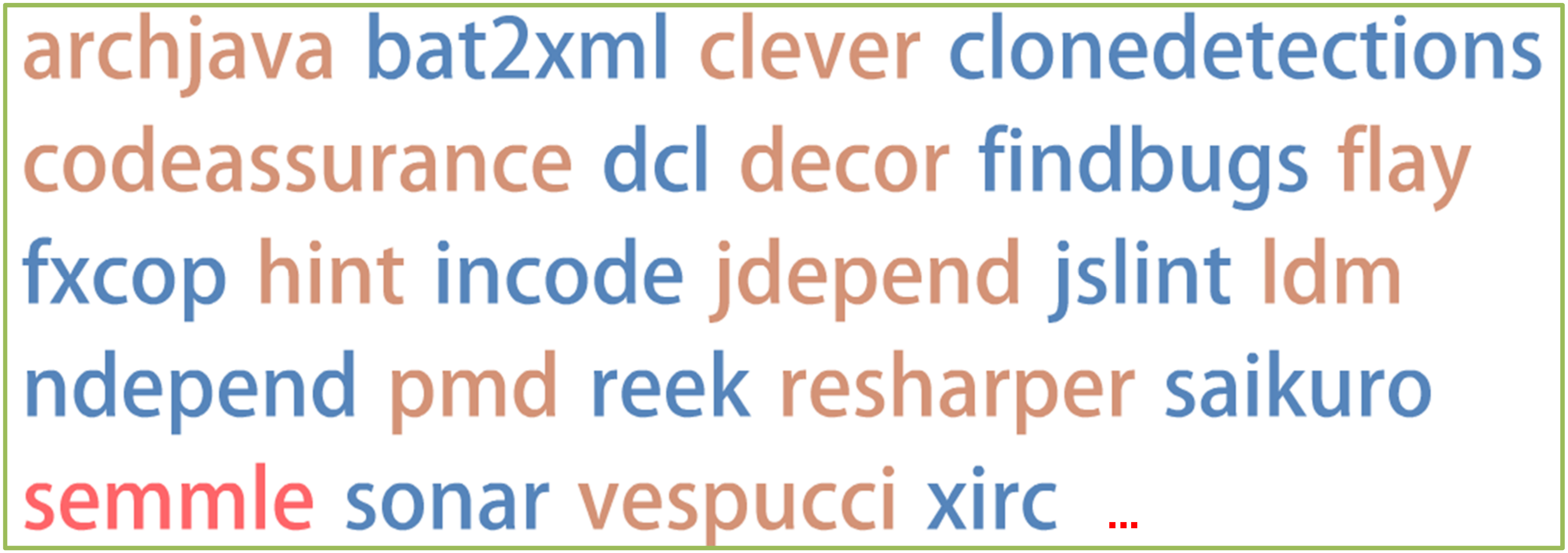
Recent Advances...

- **Architecture recovery** techniques
  - Recovery of actual architectures from source code

- **Erosion detection** in source code
  - DSLs to describe (and check) anti-erosion rules

- **Drift detection** in source code
  - DSLs to describe (and check) anti-drift rules
  - Metrics-based strategies

Existing Anti-Degradation Techniques



archjava bat2xml clever clonedetections codeassurance dcl decor findbugs flay fxcop hint incode jdepend jslint ldm ndepend pmd reek resharper saikuro semmle sonar vespucci xirc ...

... supports either drift- **OR** erosion-prevention rules

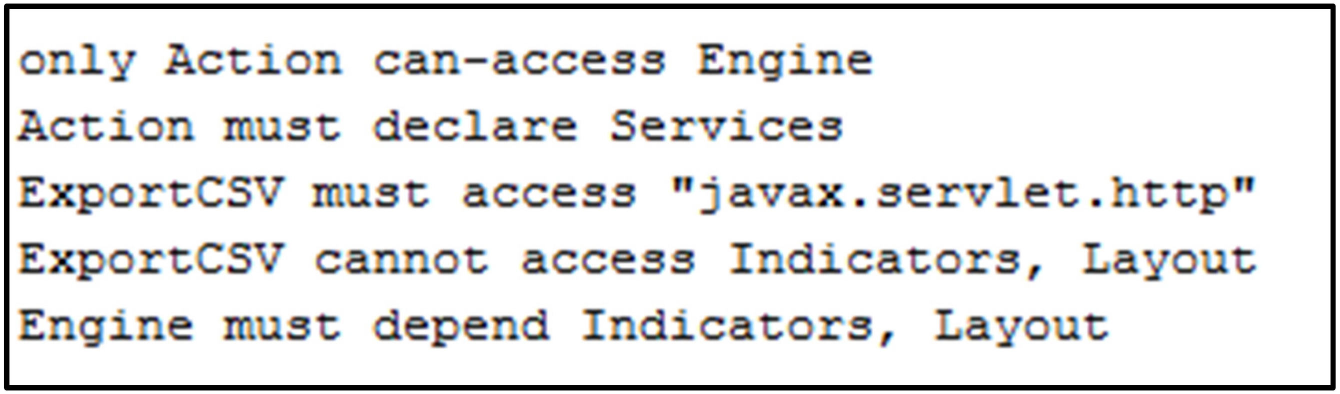
... for different program languages

Anti-Erosion and Anti-Drift Rules

Architectural Mapping

```
ArchitecturalConcept Action { parent AbsAppAction}
ArchitecturalConcept Engine { suffix Engine }
…
```

Anti-Erosion Rules



```
only Action can-access Engine
Action must declare Services
ExportCSV must access "javax.servlet.http"
ExportCSV cannot access Indicators, Layout
Engine must depend Indicators, Layout
```

Bloated Interfaces, Ambiguous Interfaces

Anti-Drift Rules

```
GodClass {
   Coupling > 7
   Cohesion, TopValues(25%)
   MethodComplexity, TopValues(25%)
}
```

Why Developers Overlook …

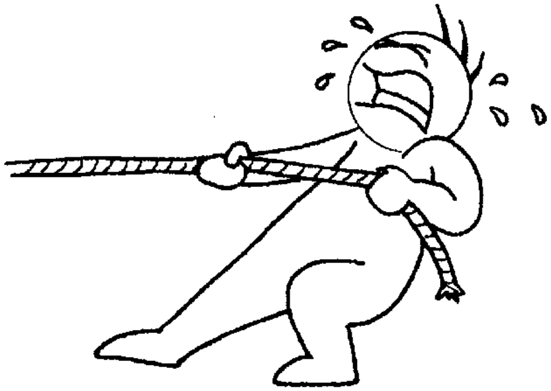
… Architecture Degradation Symptoms?

Why Developers Overlook …

… Architecture Degradation Symptoms?

Detection Accuracy

Detection Effort



Empirical Methods

- Exploratory quantitative studies
  - 7 software projects, such as:
    - PDP – Company X
    - Platform for financial market analysis – Company Y
    - OODT – NASA/Apache
    - MIDAS – Bosch
    - Logistics Framework – Company Z
- Case studies (*in situ*)
  - 7 software projects in the same domain
  - Observations, questionaries and interviews with architects and developers

Why Developers Overlook …

… Architecture Degradation Symptoms?

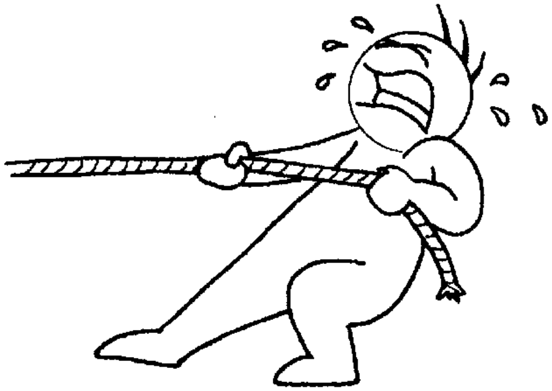
7 Lessons Learned

Why Developers Overlook …

… Architecture Degradation Symptoms?

Detection Accuracy

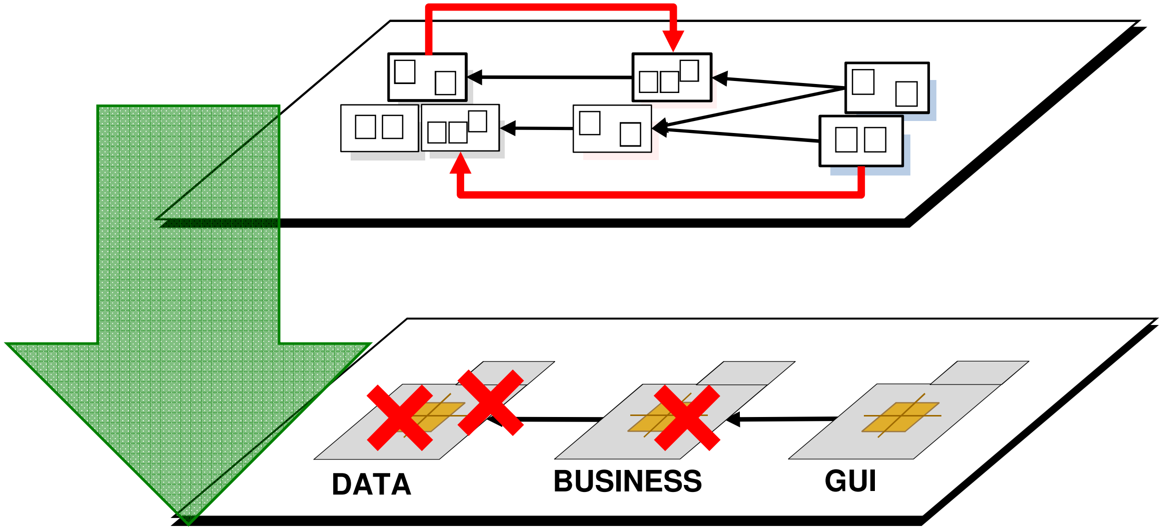
Detection Effort



Downstream Analysis

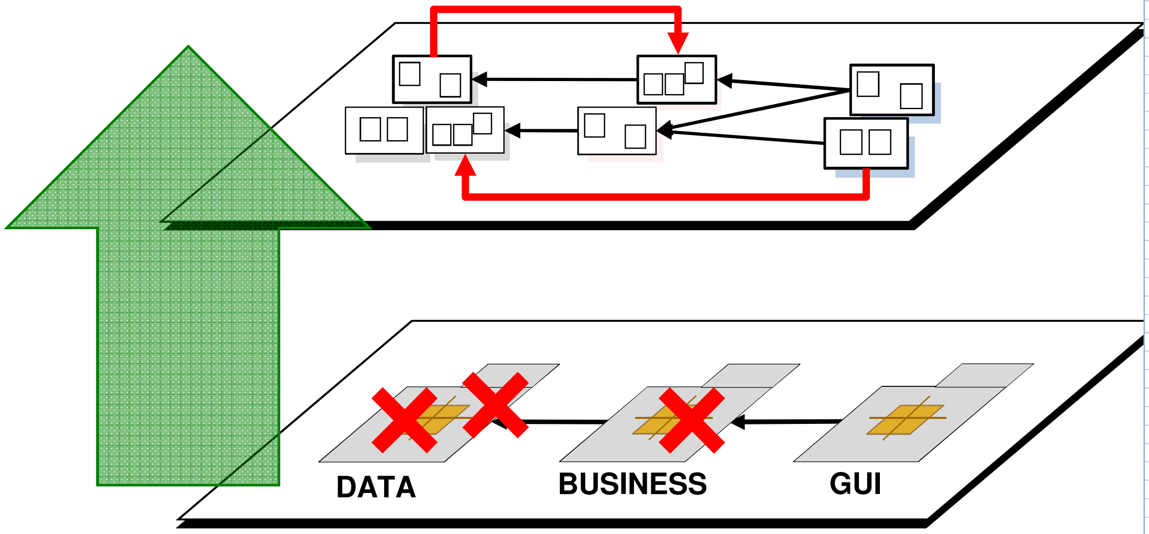
Architecture problems and code anomalies were related in

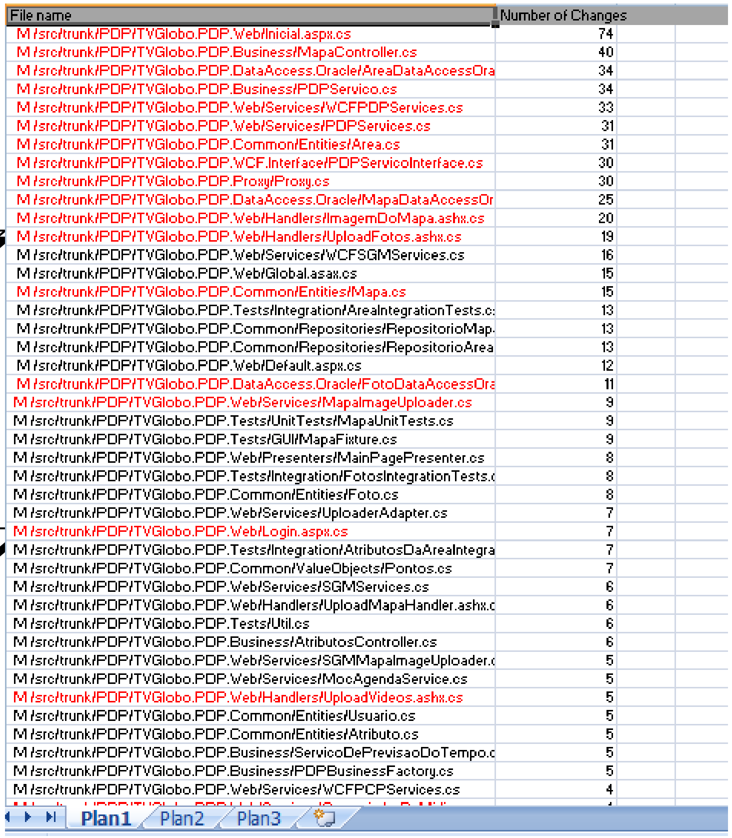
>80%



Upstream Analysis

Lack of ranking support

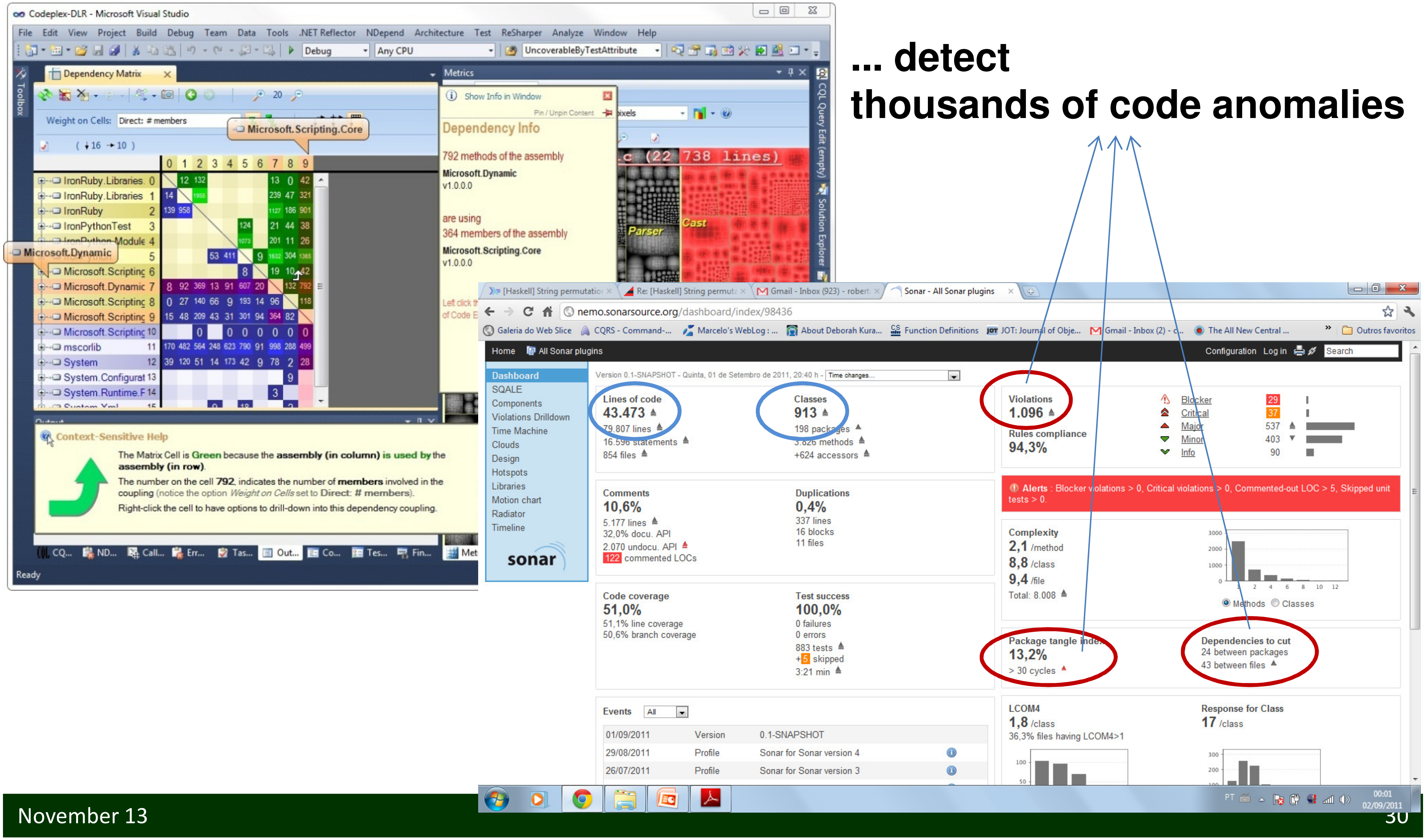




File name	Number of Changes
M /src/trunk/PDP/TVGlobo.PDP.Web/Inicial.aspx.cs	74
M /src/trunk/PDP/TVGlobo.PDP.Business/MapaController.cs	40
M /src/trunk/PDP/TVGlobo.PDP.DataAccess.Oracle/AreaDataAccessOra	34
M /src/trunk/PDP/TVGlobo.PDP.Business/PDPServico.cs	34
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/WCFPDPServices.cs	33
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/PDPServices.cs	31
M /src/trunk/PDP/TVGlobo.PDP.Common/Entities/Area.cs	31
M /src/trunk/PDP/TVGlobo.PDP.WCF.Interface/PDPServicoInterface.cs	30
M /src/trunk/PDP/TVGlobo.PDP.Proxy/Proxy.cs	30
M /src/trunk/PDP/TVGlobo.PDP.DataAccess.Oracle/MapaDataAccessOr	25
M /src/trunk/PDP/TVGlobo.PDP.Web/Handlers/ImagemDoMapa.ashx.cs	20
M /src/trunk/PDP/TVGlobo.PDP.Web/Handlers/UploadFotos.ashx.cs	19
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/WCFSGMServices.cs	16
M /src/trunk/PDP/TVGlobo.PDP.Web/Global.asax.cs	15
M /src/trunk/PDP/TVGlobo.PDP.Common/Entities/Mapa.cs	15
M /src/trunk/PDP/TVGlobo.PDP.Tests/Integration/AreaIntegrationTests.c:	13
M /src/trunk/PDP/TVGlobo.PDP.Common/Repositories/RepositorioMap.	13
M /src/trunk/PDP/TVGlobo.PDP.Common/Repositories/RepositorioArea	13
M /src/trunk/PDP/TVGlobo.PDP.Web/Default.aspx.cs	12
M /src/trunk/PDP/TVGlobo.PDP.DataAccess.Oracle/FotoDataAccessOra	11
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/MapaImageUploader.cs	9
M /src/trunk/PDP/TVGlobo.PDP.Tests/UnitTests/MapaUnitTests.cs	9
M /src/trunk/PDP/TVGlobo.PDP.Tests/GUI/MapaFixture.cs	9
M /src/trunk/PDP/TVGlobo.PDP.Web/Presenters/MainPagePresenter.cs	8
M /src/trunk/PDP/TVGlobo.PDP.Tests/Integration/FotosIntegrationTests.c	8
M /src/trunk/PDP/TVGlobo.PDP.Common/Entities/Foto.cs	8
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/UploaderAdapter.cs	7
M /src/trunk/PDP/TVGlobo.PDP.Web/Login.aspx.cs	7
M /src/trunk/PDP/TVGlobo.PDP.Tests/Integration/AtributosDaAreaIntegra	7
M /src/trunk/PDP/TVGlobo.PDP.Common/ValueObjects/Pontos.cs	7
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/SGMServices.cs	6
M /src/trunk/PDP/TVGlobo.PDP.Web/Handlers/UploadMapaHandler.ashx.c	6
M /src/trunk/PDP/TVGlobo.PDP.Tests/Util.cs	6
M /src/trunk/PDP/TVGlobo.PDP.Business/AtributosController.cs	6
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/SGMMapaImageUploader.c	5
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/MocAgendaService.cs	5
M /src/trunk/PDP/TVGlobo.PDP.Web/Handlers/UploadVideos.ashx.cs	5
M /src/trunk/PDP/TVGlobo.PDP.Common/Entities/Usuario.cs	5
M /src/trunk/PDP/TVGlobo.PDP.Common/Entities/Atributo.cs	5
M /src/trunk/PDP/TVGlobo.PDP.Business/ServicoDePrevisaoDoTempo.c	5
M /src/trunk/PDP/TVGlobo.PDP.Business/PDPBusinessFactory.cs	5
M /src/trunk/PDP/TVGlobo.PDP.Web/Services/WCFPCPServices.cs	4

Too many **DRIFT** candidates to inspect...

... detect thousands of code anomalies



Dependency Info

792 methods of the assembly

Microsoft.Dynamic
v1.0.0.0

are using
364 members of the assembly

Microsoft.Scripting.Core
v1.0.0.0

Context-Sensitive Help

The Matrix Cell is Green because the **assembly (in column) is used by** the **assembly (in row)**.

The number on the cell **792**, indicates the number of **members** involved in the coupling (notice the option *Weight on Cells* set to **Direct: # members**).

Right-click the cell to have options to drill-down into this dependency coupling.

Version 0.1-SNAPSHOT - Quinta, 01 de Setembro de 2011, 20:40 h - Time changes...

Lines of code
43.473
79.807 lines
16.596 statements
854 files

Classes
913
198 packages
3.826 methods
+624 accessors

Violations
1.096

Rules compliance
94,3%

Blocker	29
Critical	37
Major	537
Minor	403
Info	90

Alerts : Blocker violations > 0, Critical violations > 0, Commented-out LOC > 5, Skipped unit tests > 0.

Comments
10,6%
5.177 lines
32,0% docu. API
2.070 undocu. API
122 commented LOCs

Duplications
0,4%
337 lines
16 blocks
11 files

Complexity
2,1 /method
8,8 /class
9,4 /file
Total: 8.008

Code coverage
51,0%
51,1% line coverage
50,6% branch coverage

Test success
100,0%
0 failures
0 errors
883 tests
+5 skipped
3:21 min

Package tangle index
13,2%
> 30 cycles

Dependencies to cut
24 between packages
43 between files

Events All

01/09/2011	Version	0.1-SNAPSHOT
29/08/2011	Profile	Sonar for Sonar version 4
26/07/2011	Profile	Sonar for Sonar version 3

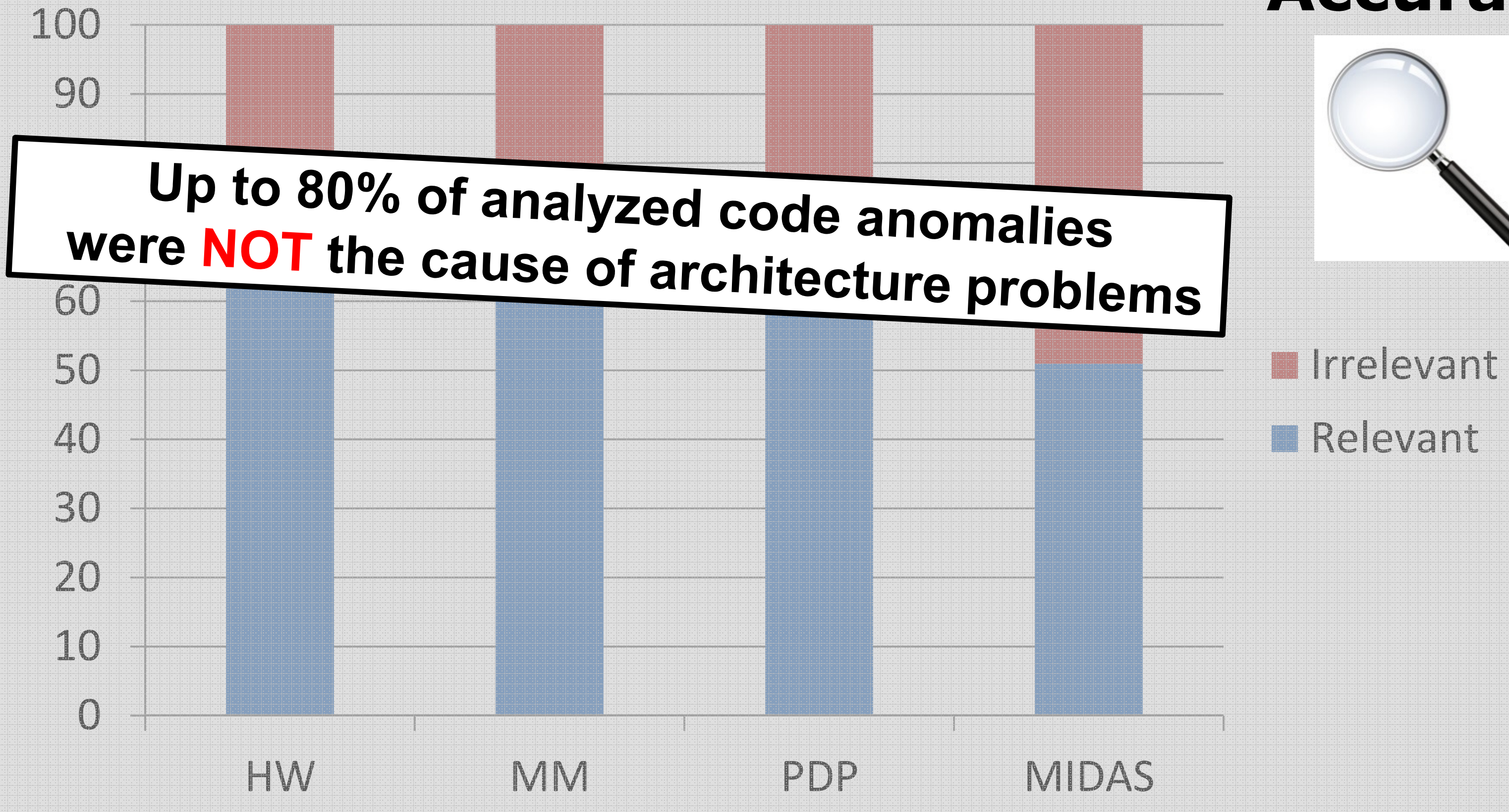
LCOM4
1,8 /class
36,3% files having LCOM4>1

Response for Class
17 /class

But many irrelevant code anomalies

- Upstream Analysis

Detection Accuracy

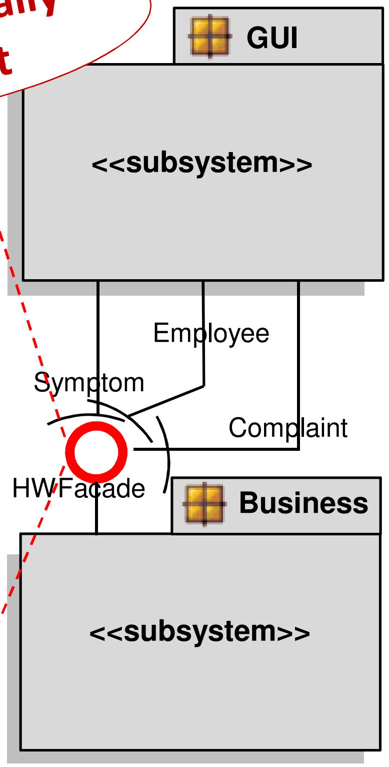


Up to 80% of analyzed code anomalies were NOT the cause of architecture problems

Architectural Relevance of Code Anomalies

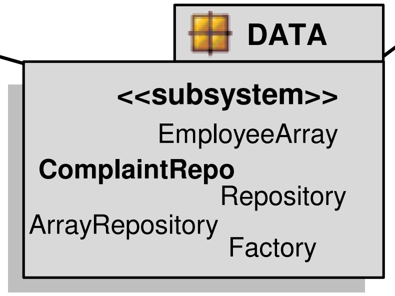
```
public class HWFacade{

   public void updateComplaint(..){..}
   public Complaint searchComplaint(..){..}
   public void insertComplaint(..){..}

   public void insertEmployee(..){..}
   public Employee searchEmployee(..){..}
   public void updateEmployee(..){..}

   public void insertSymptom(..){..}
   public Symptom searchSymptom(..){..}
   public void updateSymptom(..){..}

   ...
}
```

Architecturally Relevant



```
public class ComplaintRepo{
   ...
   public int insert(..){..}
   public void updat
   public int get
   public boolea
   public Complaint search(..){..}
   public void reset(..){..}
   public Object next(..){..}
   public void remove(..){..}
   public List getList(..){..}
   public boolean hasNext(..){..}
   public void updateTimestamp(..){..}
   public int searchTimestamp(..){..}
    ...
}
```

Architecturally Irrelevant



7 Lessons – Why Developers Overlook ...

... Architecture Degradation Symptoms?

1. Lack of prioritization support

Studying prioritization models

- Which other characteristics could be explored for detecting architecturally-relevant code anomalies ?
  - Change density
  - Error density
  - Anomaly density
  - Code anomaly type
  - Etc...

Roberta Arcoverde et al – **RSSE/ICSE 2012**: Automatically Detecting Architecturally-Relevant Code Anomalies

Roberta Arcoverde et al – **SBES 2013**: ***Prioritization of Code Anomalies Based on Architecture Sensitiveness.*** SBES'13) Brasília, Brazil, September 2013.

Prioritization heuristics

Change density

System	# of Ranked CE	Arch. Relevant	%
HW	14	10	71%
MM	10	7	70%
PDP	**10**	**10**	**100%**

Error density

System	# of Ranked CE	Arch. Relevant	%
HW	**14**	**12**	**85%**
MM	10	8	70%
PDP	10	8	70%

Anomaly density

System	# of Ranked CE	Arch. Relevant	%
HW	10	7	60%
MM	10	9	70%
PDP	10	8	70%
MIDAS	**10**	**6**	**90%**

7 Lessons – Why Developers Overlook ...

... Architecture Degradation Symptoms?

1. Lack of prioritization support
2. **There is no ‘universal’ prioritization model**
3. **Prioritization models: satisfactory results too late**

Prioritization heuristics

Change density

System	# of Ranked CE	Arch. Relevant	%
HW	14	10	
MM	10	7	
PDP	**10**	**10**	**100%**

Version 12

Error density

System	# of Ranked CE	Arch. Relevant	%
HW	**14**	**12**	**85%**
MM	10	8	
PDP	10	8	

Version 9

Anomaly density

System	# of Ranked CE	Arch. Relevant	%
HW	10	7	
MM	10	9	
PDP	10	8	
MIDAS	**10**	**6**	**90%**

Version 10

Earliness of Anomaly

- Early anomalies often appear in the 1st version

18%

Of all architecturally-relevant code anomalies were identified as **early anomalies**

Earliness of Architectural Problems

- Early anomalies often appear in the 1st version

18%

Of all architecturally-relevant code anomalies were introduced as

early

→

Related to almost

40%

of all the

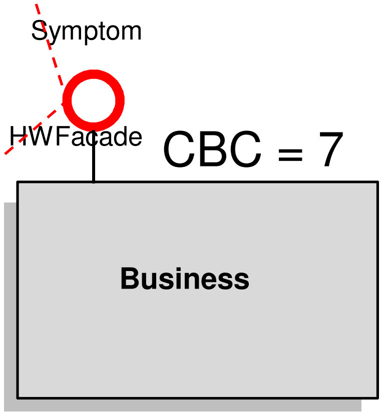
architectural problems

Example

- 1^{st} version

```
public class HWFacade{

   public void updateComplaint(..){..}
   public Complaint searchComplaint(..){..}
   public void insertComplaint(..){..}

   public void insertEmployee(..){..}
   public Employee searchEmployee(..){..}
   public void updateEmployee(..){..}
    ...
}
```

Architecturally Relevant

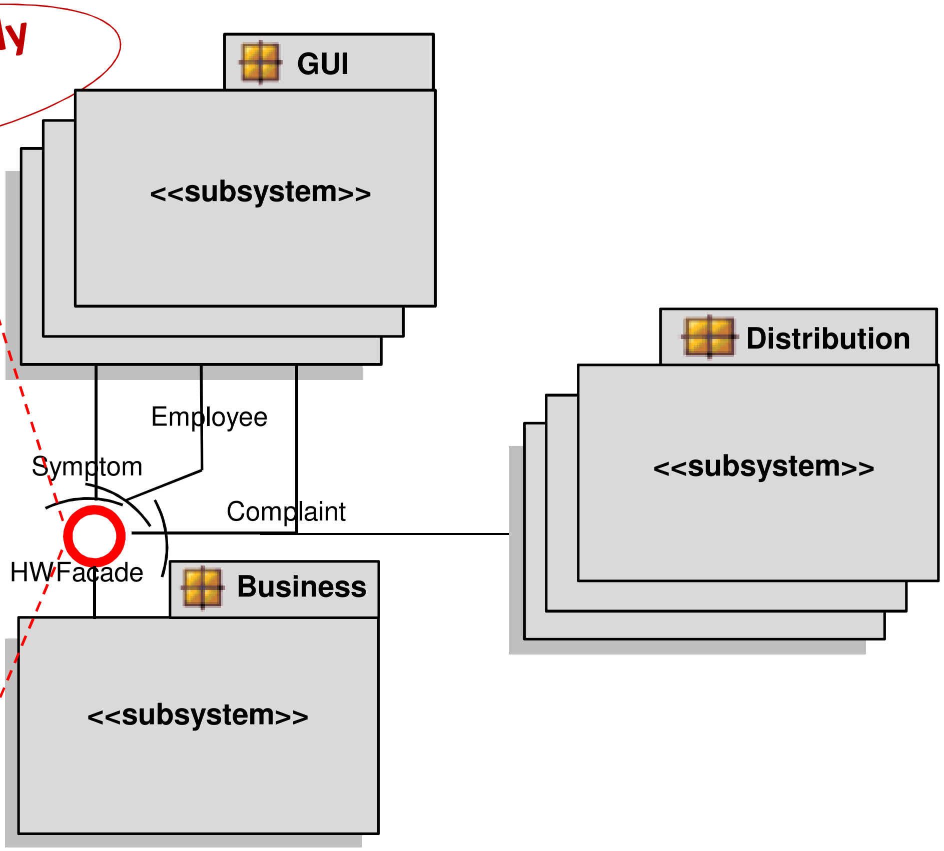


Example: fixing here is expensive

- Version 10

Architecturally Relevant

```
public class HWFacade{

    public void updateComplaint(..){..}
    public Complaint searchComplaint(..){..}
    public void insertComplaint(..){..}

    public void insertEmployee(..){..}
    public Employee searchEmployee(..){..}
    public void updateEmployee(..){..}

    public void insertSymptom(..){..}
    public Symptom searchSymptom(..){..}
    public void updateSymptom(..){..}

    ...
}
```



Priorization Heuristics: conclusions

- Heuristics proposed were able to correctly outline architecturally-relevant anomalies
  - Ranked elements were architecturally relevant in 75%-85% average
- Anomaly density heuristic presented very good results
  - Code modules infected by multiple code anomalies were often related to architectural problems
  - Identification of code anomaly patterns
- Mapping-based prioritization was even better
  - … but there is a cost involved to produce and maintain these architecture-code mappings

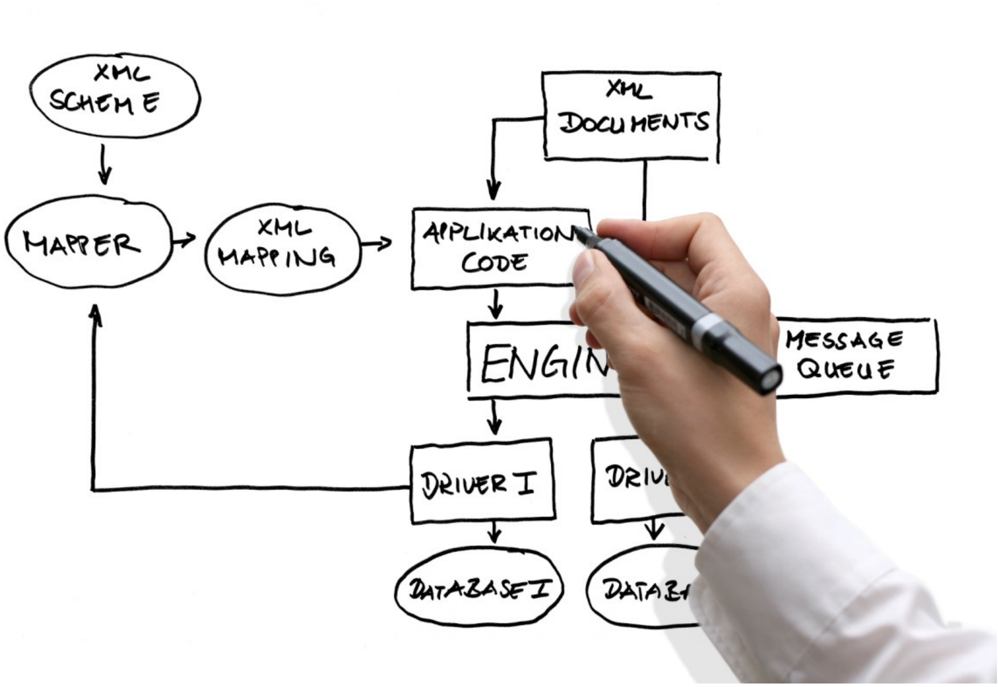
7 Lessons – Why Developers Overlook ...

... Architecture Degradation Symptoms?

1. Lack of prioritization support
2. There is no ‘universal’ prioritization model
3. **Prioritization: satisfactory results too late**
4. **Critical code anomalies are often introduced early**

What about Upfront Detection?

... when developers write their own architectural rules?



- **2nd stage -** Case studies (*in situ*): 7 software projects

  Observations, questionaries and interviews

Empirical Methods

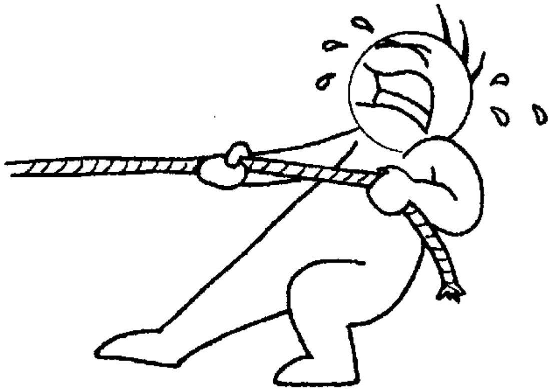
- **1st Stage** - Exploratory quantitative studies
  - 7 software projects, such as:
    - PDP – Radix Engenharia
    - Platform for financial market analysis – Minds@Work
    - OODT – NASA/Apache
    - MIDAS – Bosch
    - Logistics Framework – Petrobras/PUC-Rio
- Case studies (*in situ*)
  - 1 case study: accuracy vs. effort
  - 6 software projects in the same domain: reuse of rules
  - Observations, questionaries and interviews with architects and developers

Why Developers Overlook …

… Architecture Degradation Symptoms?

Detection Accuracy

Overall Effort



Architectural Mapping

Rule Description

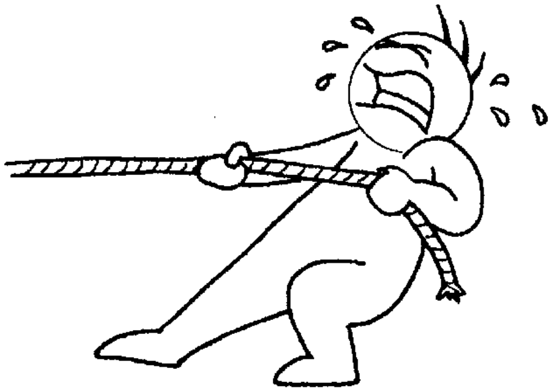
Architecture Problem Detection

What about Upfront Detection?

- Expl ntitative studies
  - 7 , suc
    - aria
    - al m k
    - he
    - Logistics Framework – F

Accuracy

Effort



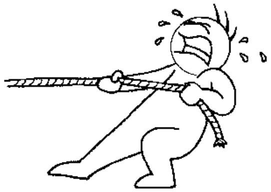
- Comparison:

Specification and Detection of Architectural Rules

vs.

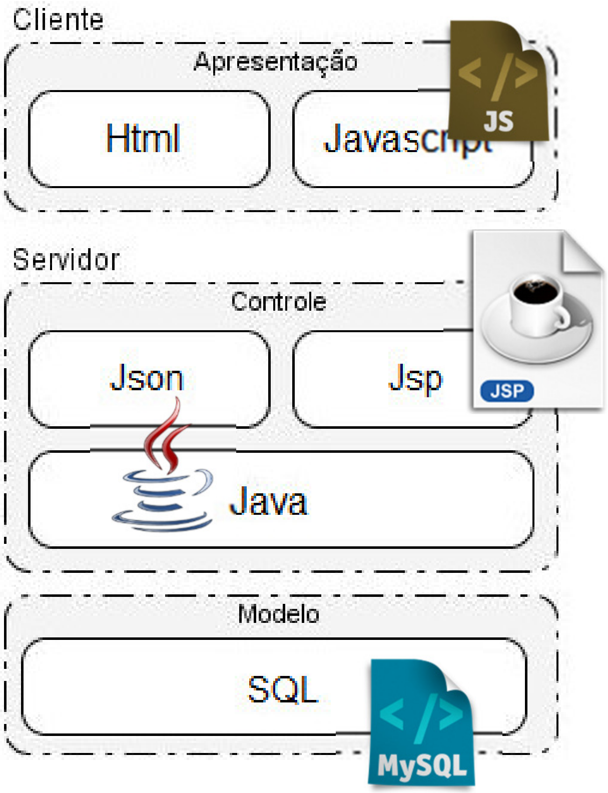
Code Inspection

What about Upfront Detection?

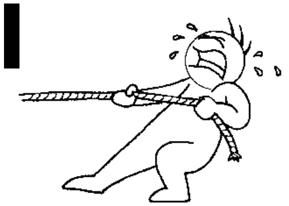
	Accuracy	Effort 
Architectural Rules	85%.. 95% (... but a few ‘universal’ drift rules could be reused)	
Code Inspection	85%.. 95%	

False positives were related to ...

...the nature of multiparadigm of software projects



What about Upfront Detection?

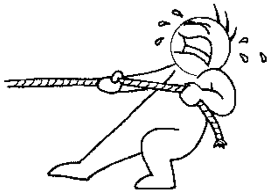


	Accuracy	Overall Effort (per subsystem)
Architectural Rules	90%.. 100%	**22 hours** -37,5%
Code Inspection	90%.. 100%	**16 hours**

Effort is Too High

Configuration
Effort

(per subsystem)



Strategy	Configuration (hour)	Detection (hour)	Total (hour)
Code inspection	0	16	16
Architectural rules	**20**	2	22

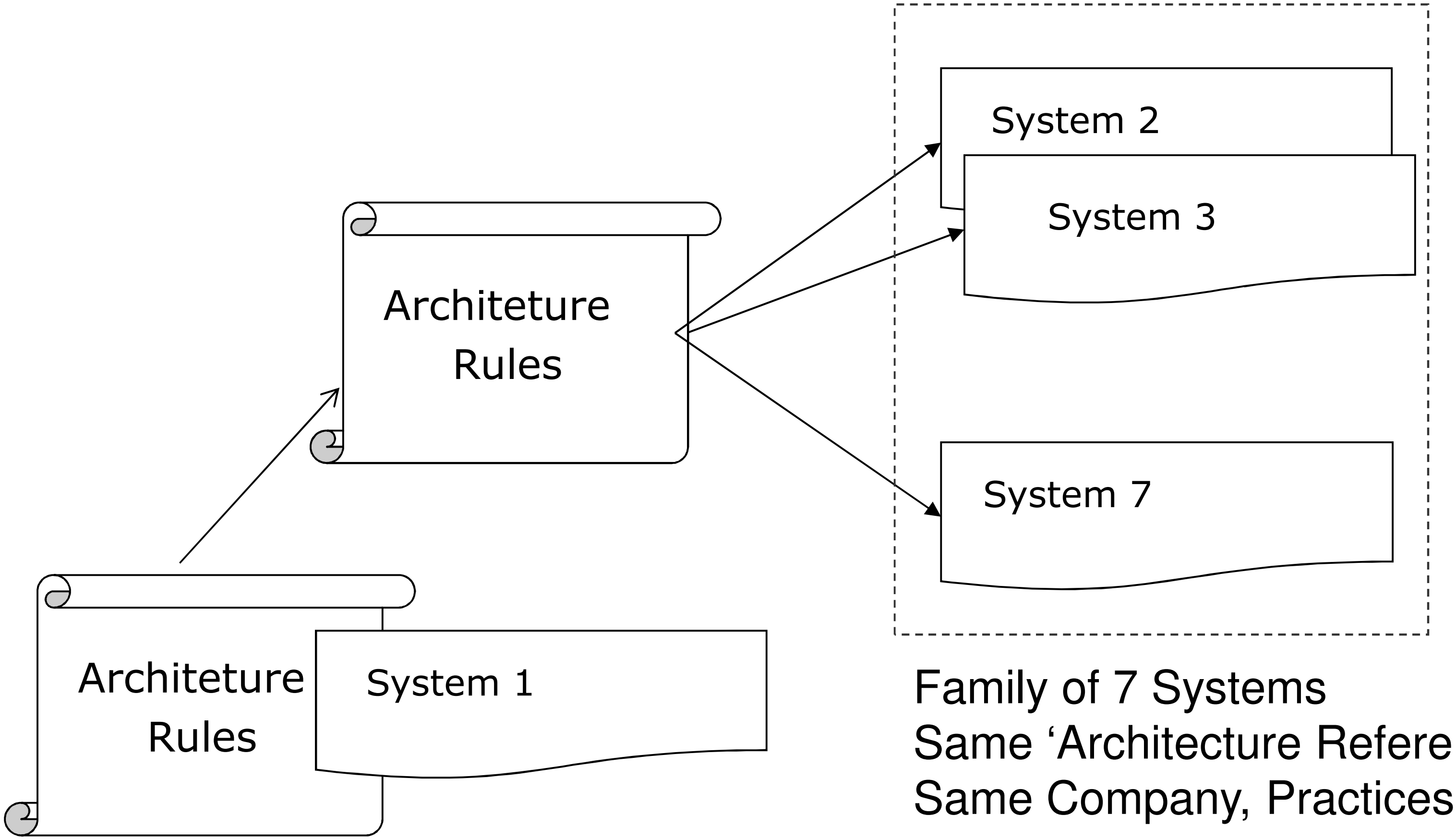
	Architectural Mapping (hour)	Rules Tailoring (hour)	Total (hour)
Architectural rules	12	8	20

(anti-drift rules)

Reuse to pay off the upfront effort?

- Reuse of architectural rules

Same Domain:
Financial Market Analysis



Family of 7 Systems
Same ‘Architecture Reference’
Same Company, Practices, Frameworks

7 Lessons – Why Developers Overlook ...

... Architecture Degradation Symptoms?

1. Lack of prioritization support
2. There is no ‘universal’ prioritization model
3. Prioritization models tend to yield satisfactory results too late
4. Critical code anomalies are often introduced early
5. **Effort on upfront detection is costly or prohibitive**
6. **False negatives in multi-paradigm software projects**
7. **Reuse of anti-drift rules are hard**

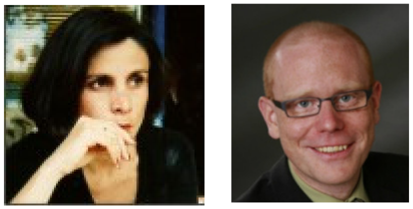
Possible solutions

- Better support for reuse of architectural rules
  - Per concerns in a domain
  - Our initial results are promising
- Synthesizing code anomalies -> architectural problems
- Further study degradation symptoms in multi-paradigm projects
- Exploit informal architectural blueprints to improve static analysis and early detection

The Opus Team and Collaborators

Examples of Collaborators...

USC
USA



TU Darmstadt
Germany

Why Developers Overlook Architecture Degradation Symptoms?

Alessandro Garcia

SEIF Workshop 2013
Rio de Janeiro

LES | DI |PUC-Rio - Brazil

OPUS Group