



# Qualidade Contínua de Software

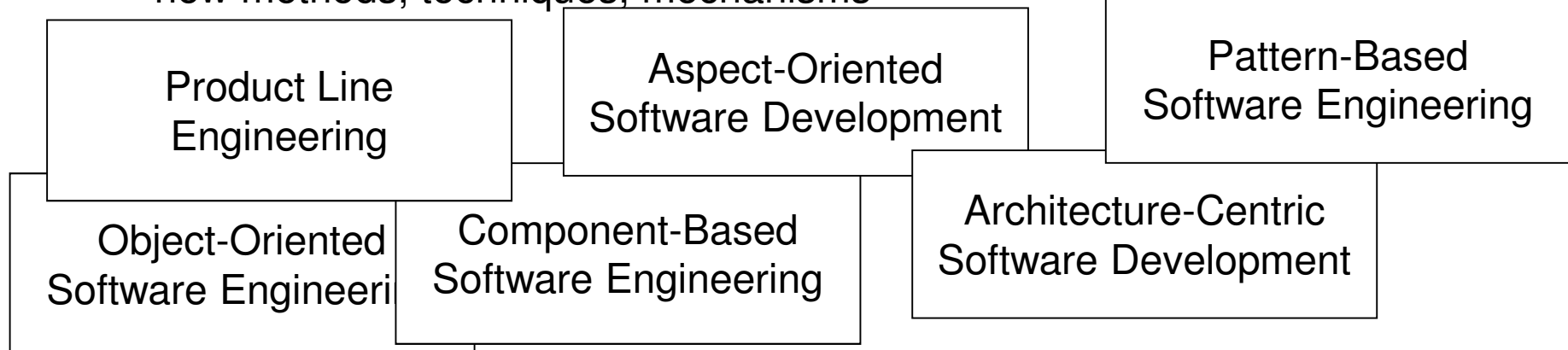
Alessandro Garcia  
afgarcia@inf.puc-rio.br



# Plethora of (SE) Software Engineering Paradigms promoting “quality by construction”

- syndrome of \*-based software engineering approaches
  - underpinning principles
    - simple design
    - low coupling and high cohesion
    - separation of concerns
    - narrow interfaces
    - ....
  - abstractions and decomposition mechanisms
    - classes, objects, inheritance, delegation, ...
  - new methods, techniques, mechanisms

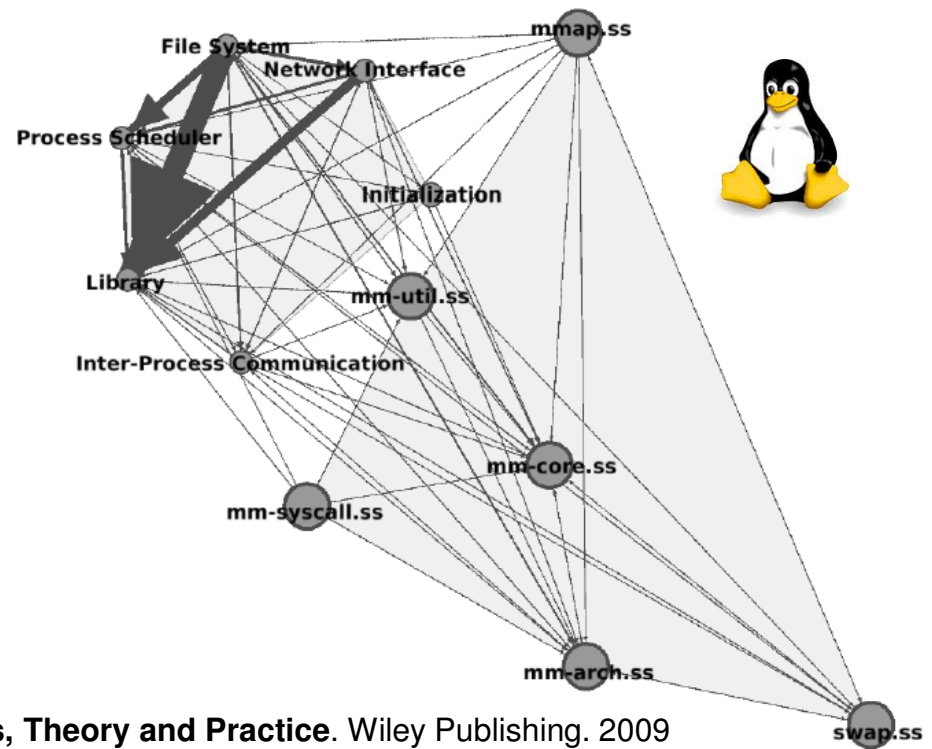
Are they enough?



# However ...

**... Spite of developers applying those principles and approaches by construction**

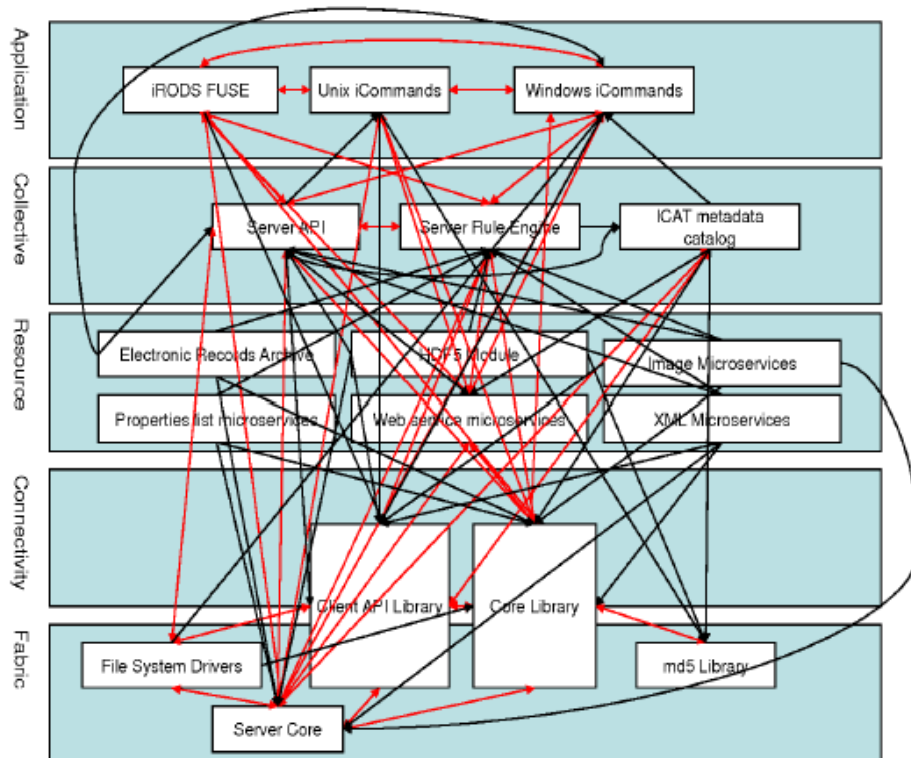
Memory manager component - Linux



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.

# Quality Degradation over Time

Actual architecture - iRODS

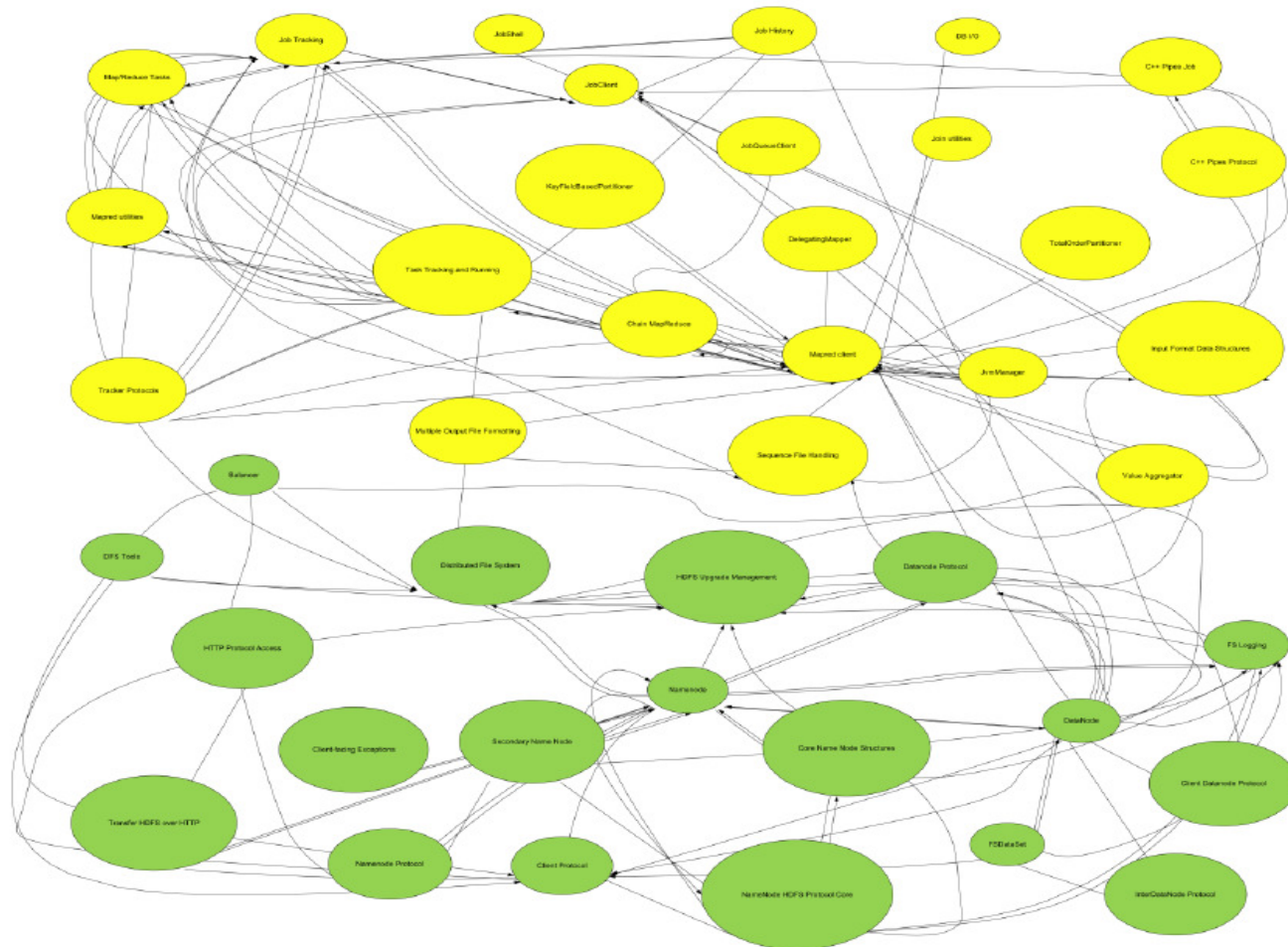


— violations of the intended architecture

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.

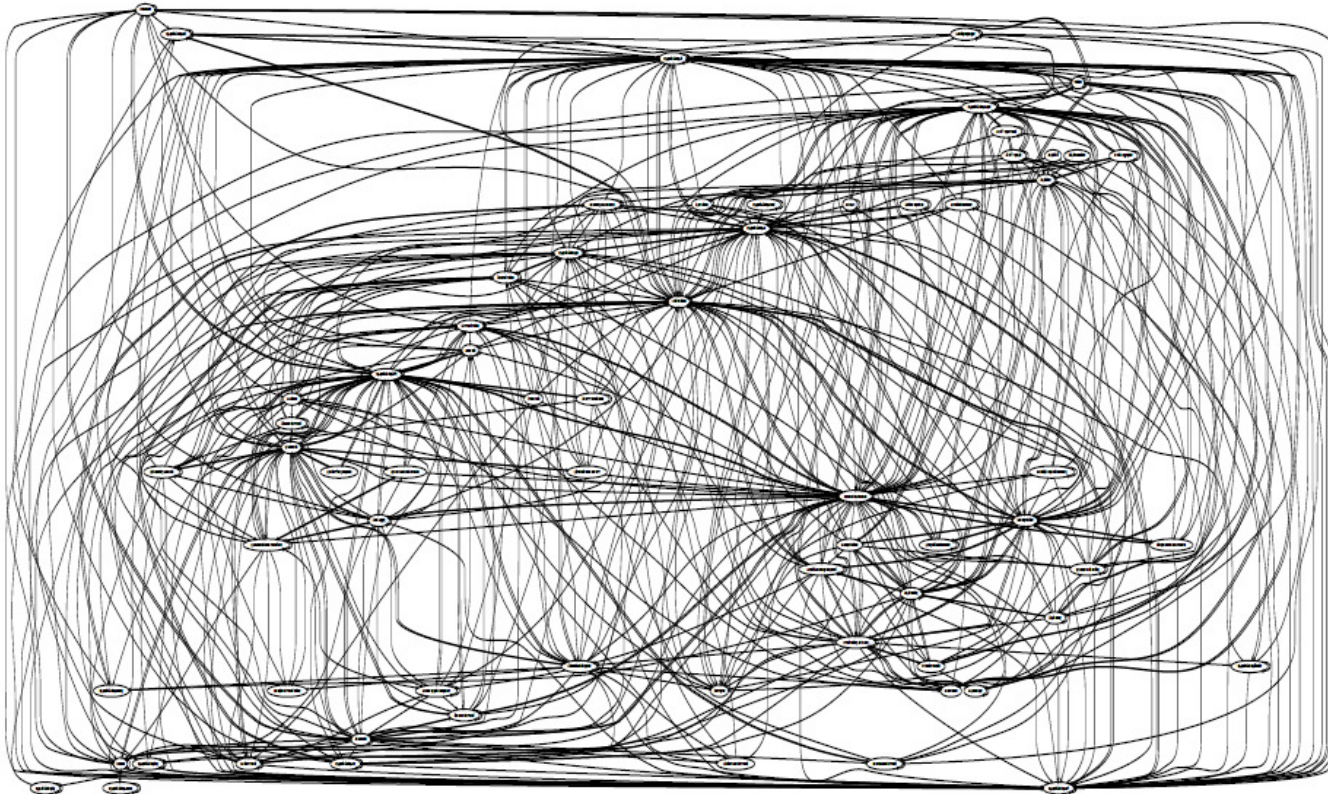
# Quality Degradation over Time

- Hadoop



# Quality Degradation over Time

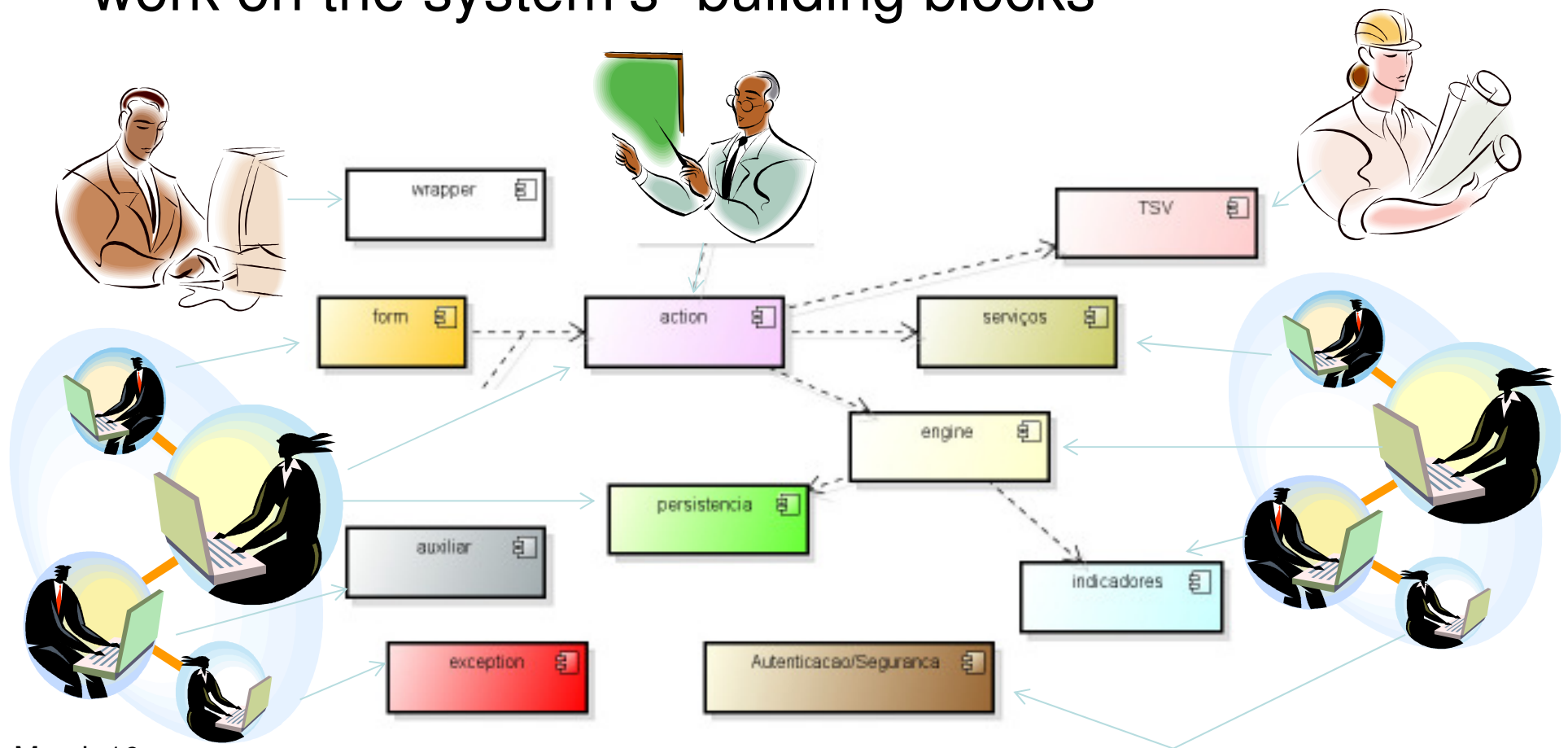
Actual architecture - Hadoop



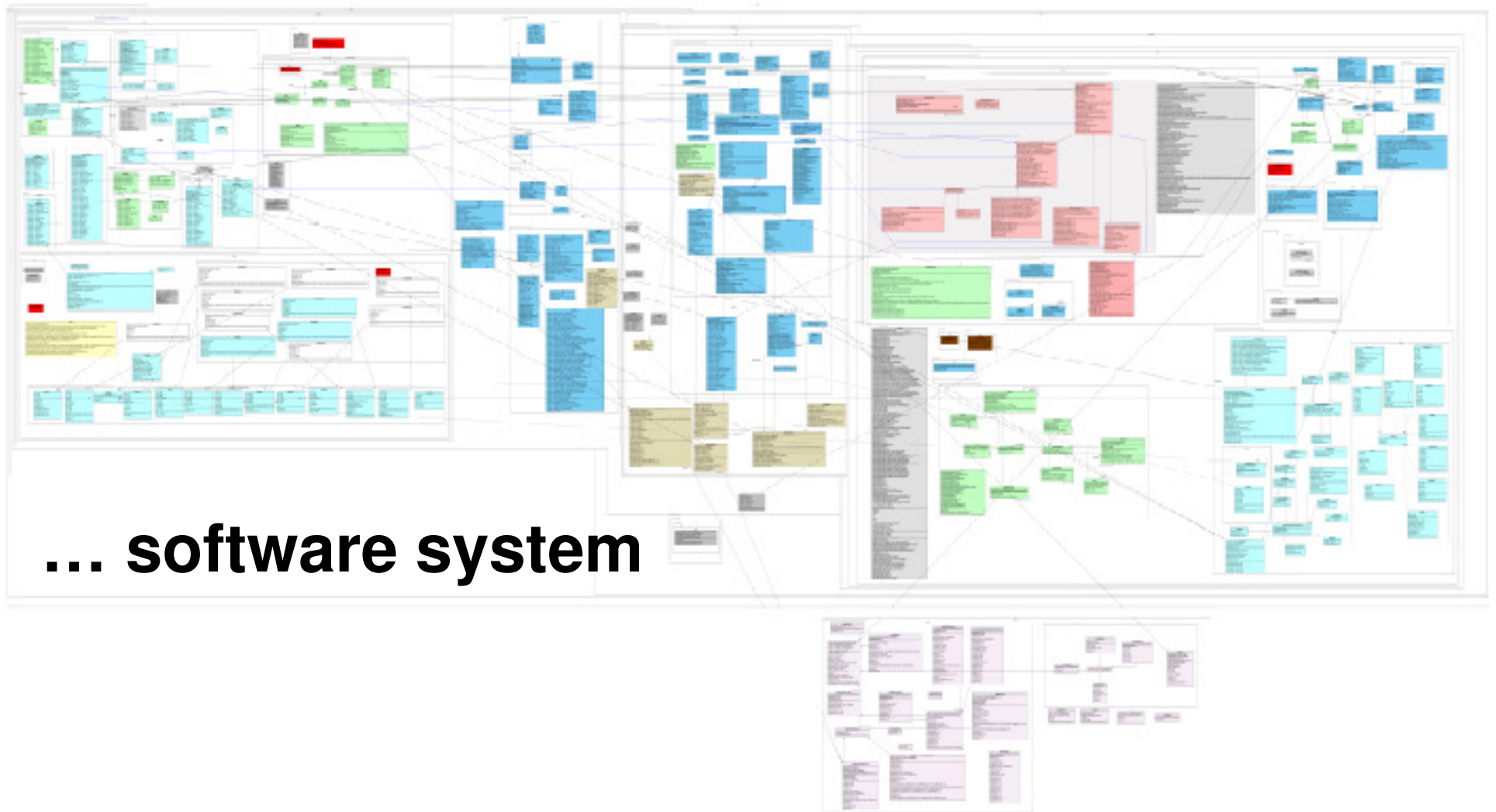
Nenad Medvidović. **When, Where, and Why Do Software Systems Architectures Decay?** March 2013.

# *Intended* architecture of a software system

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

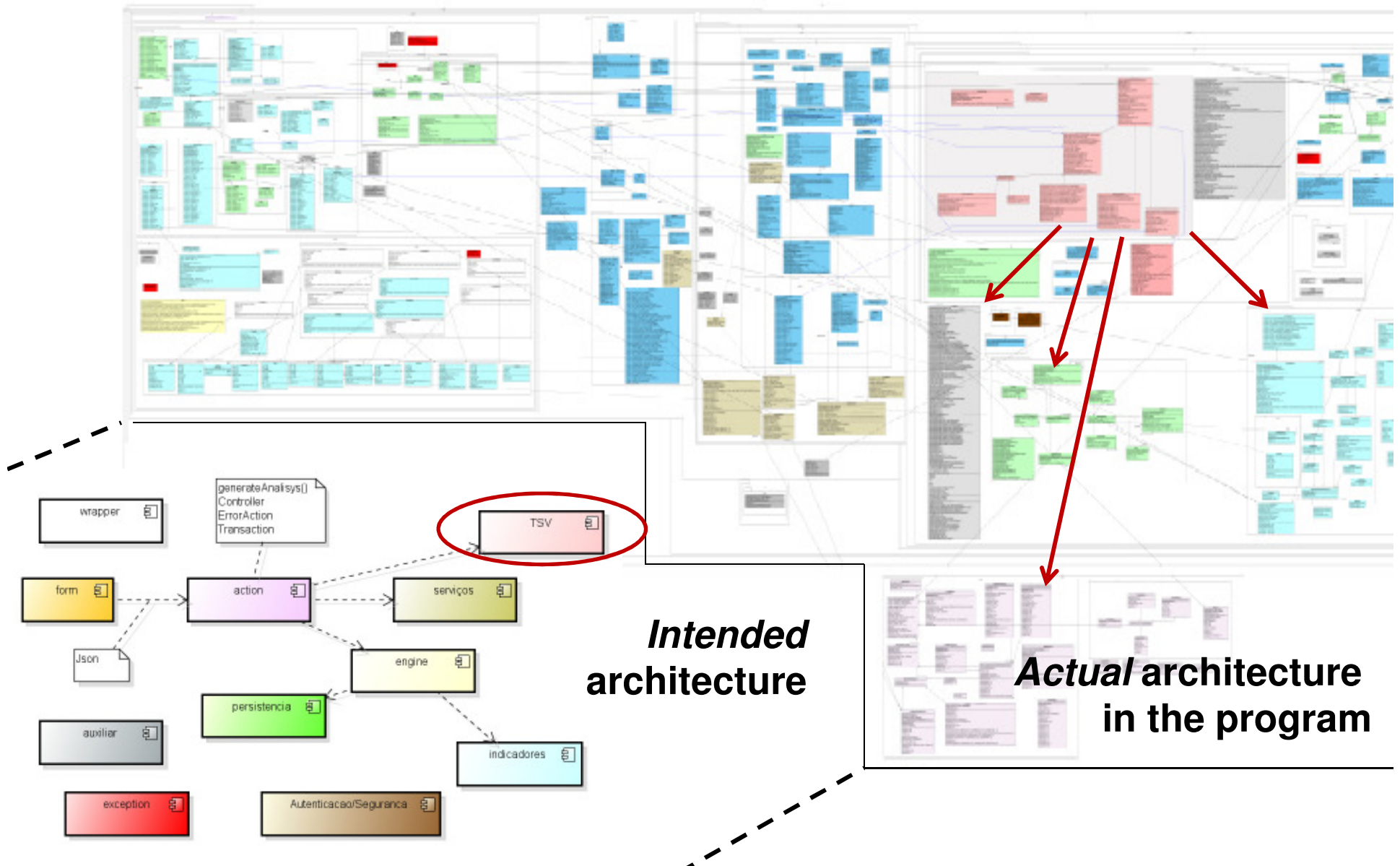


# *Actual* architecture of the same...

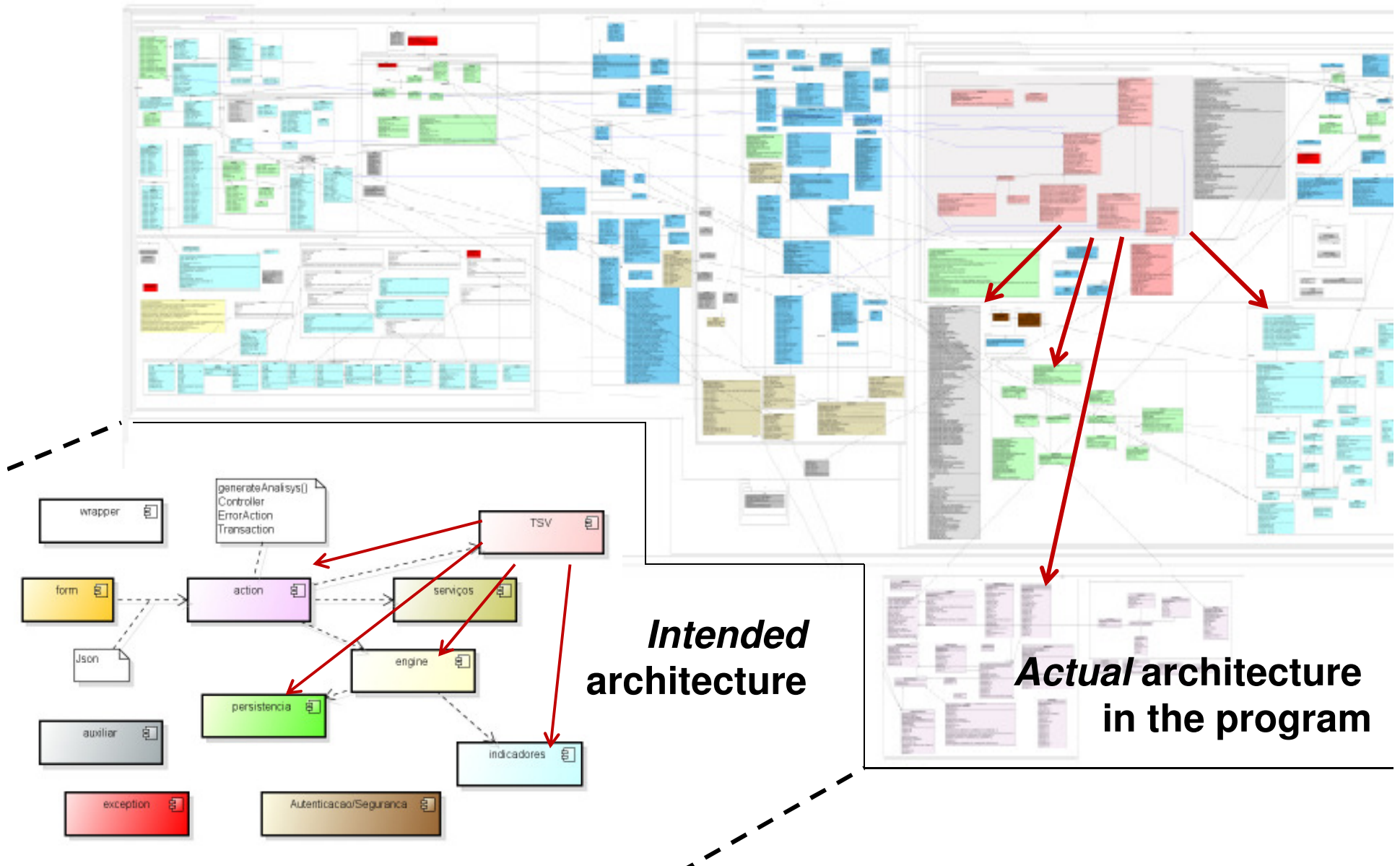




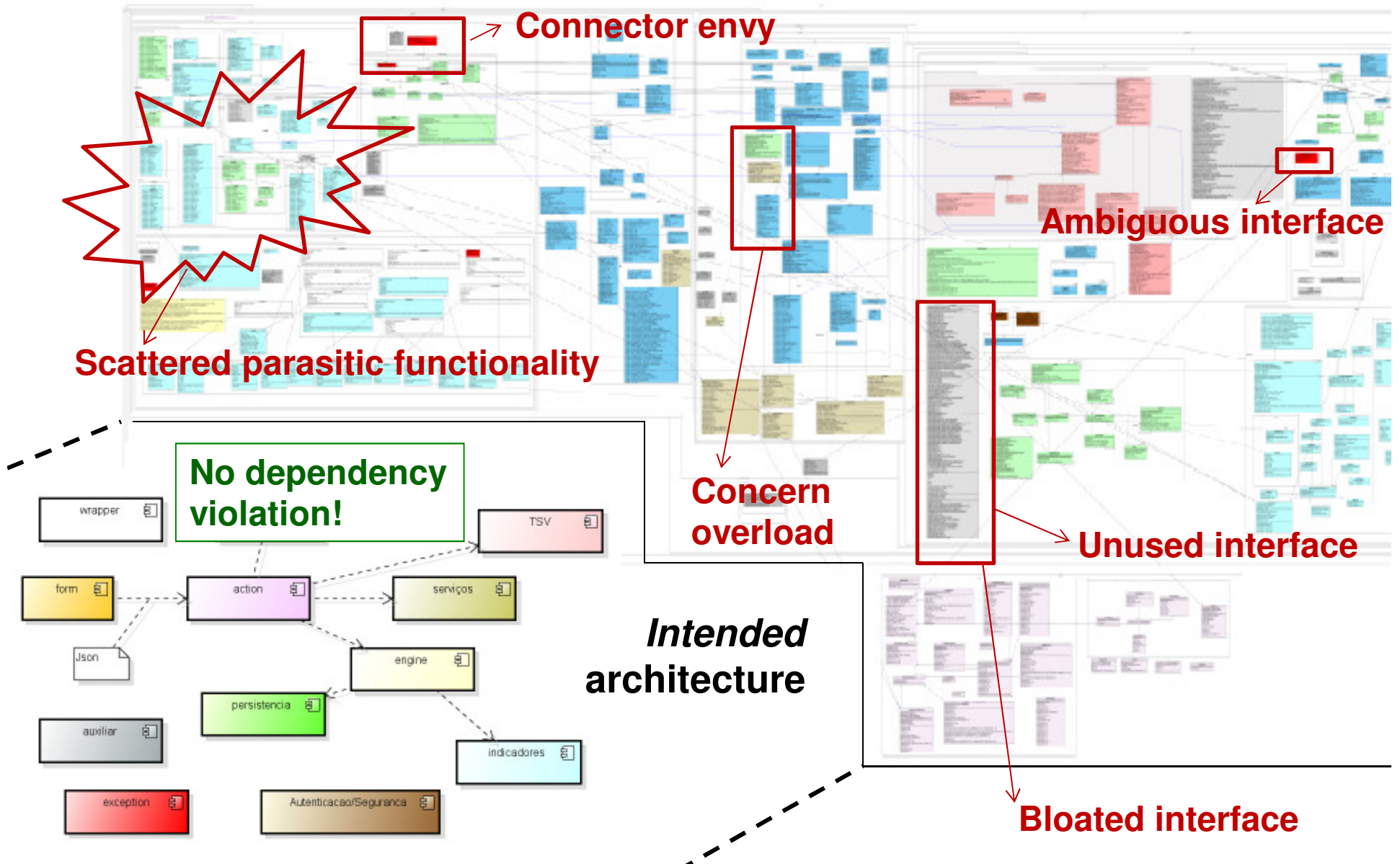
# Architectural Erosion



# Architectural Erosion



# Architectural Drift



# Retrospective Techniques in SE

- Help to understand when, how and why certain quality problems were introduced
  - architecture recovery techniques
  - bug localization techniques
  - system's reviews
- However, then repairing the problems afterwards might be too late...

# Continuous Code Quality

How is the modularity 'health' of my code?

Did I violate any design rule?

Is my list of assertions ok?

How should I refactor my code?

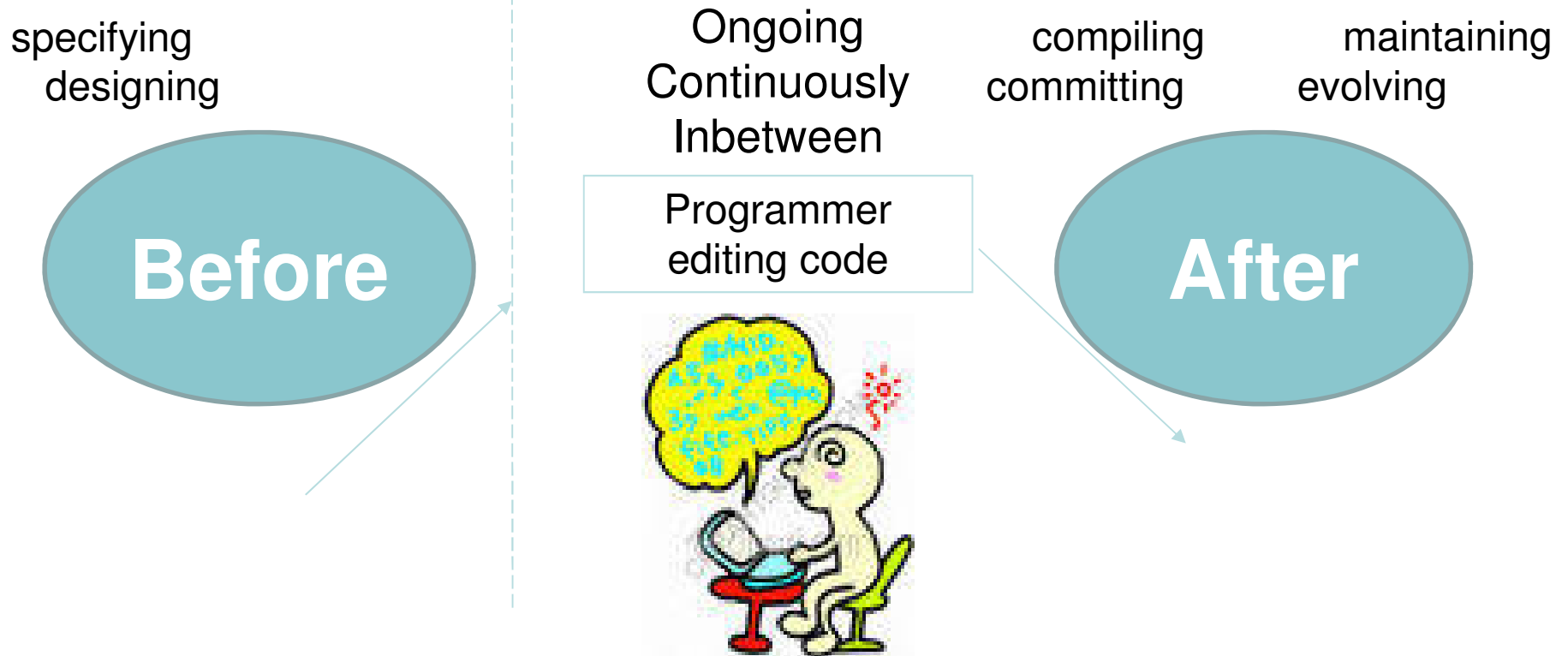
Will I introduce merge conflicts?

Did I introduce security vulnerabilities?

How to implement this 'catch' block?



# Continuous Code Quality



- support: warn and provide feedback to developers when editing the code

# Examples of emerging SE techniques

- Continuous Maintainability
- Continuous Robustness
- Continuous Security

# Ementa e Formato da Disciplina

- Série de seminários/aulas
- Parte 1: Técnicas para qualidade de software por construção/retrospecção
  - Arquitetura de software
    - Recuperação de arquitetura
  - Linhas de produto de software
  - Desenvolvimento orientado a aspectos
  - Localização de bugs
  - Etc...



# Ementa e Formato da Disciplina

- Parte 2: Taxonomia para problemas de qualidade de software
  - Anomalias de modularidade
  - Anomalias de robustez
  - Vulnerabilidades de segurança
  - Outros problemas de qualidade
- Estudos de caso: por que e como estes problemas ocorrem em projetos de software?
- Limitações de técnicas prospectivas e retrospectivas de análise de qualidade de software

# Ementa e Formato da Disciplina

- Parte 3: Qualidade contínua em software
  - Robustez contínua em software
  - Modularidade contínua em software
  - Segurança contínua em software
  - Suporte a colaboração para qualidade contínua em software

Apresentações dos alunos e aulas

# Trabalho

- Apresentação
  - Cada aluno escolhe uma técnica de qualidade contínua para estudar e apresentar
    - Apresentação de aprox. 50 minutos
  - Monografia pode abordar qualquer um destes temas:
    - Revisão conceitual daquela área
    - Revisão sistemática de trabalhos naquela área
    - Comparação de técnicas naquela área
    - Um estudo de caso de uma ferramenta
    - Etc...

# Assessment

- Participation – 10%
- Presentation – 45%
- Article – 45%

# Monografia e Apresentação

- **Critério:**
  - Clareza e Objetividade
  - Originalidade
  - Uso de exemplos
  - Reflexão crítica
  - Se houver avaliação/comparação, espera-se que o critério esteja bem definido
  - Largura e profundidade
  - Se envolver implementação:
    - criatividade no projeto
    - uso elegante de princípios e mecanismos de modularidade
    - qualidade da documentação do projeto

# Atendimento

- Todos sempre muito bem-vindos para discutir proposta e refinamento do trabalho, apresentação e outras questões
- Via email: [afgarcia@inf.puc-rio.br](mailto:afgarcia@inf.puc-rio.br)
- Minha sala: RDC – 506, marcar horário com antecedência

# Próximas duas aulas

- 15/3, 16:00, terça: sala 9º. Andar, Prédio Couve-Flor
- 22/3, 16:00, terça: sala 9º. Andar, Prédio Couve-Flor

## Professores Convidados:

- 1) architecture-centric development,  
Apache Ant (Prof. Marcio Barros)
- 2) product-line engineering (Prof. Jaejoon Lee)



# Qualidade Contínua de Software

Alessandro Garcia  
afgarcia@inf.puc-rio.br

