

Software and systems engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

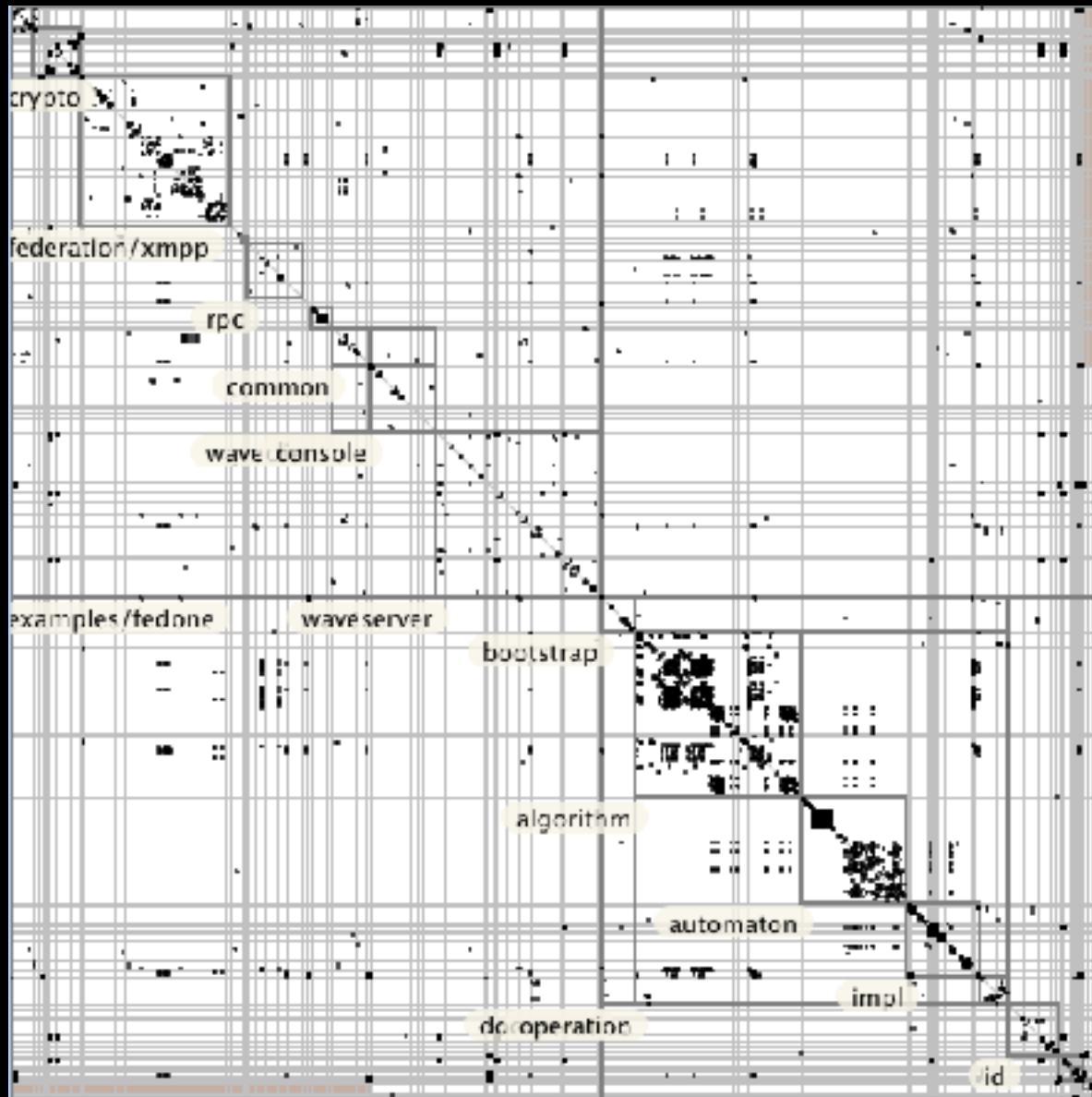
phmb@cin.ufpe.br ♦ twitter.com/pauloborba

To do before class

- Watch videos
- Read chapter 9 in the textbook
- Send questions and opinions through slack

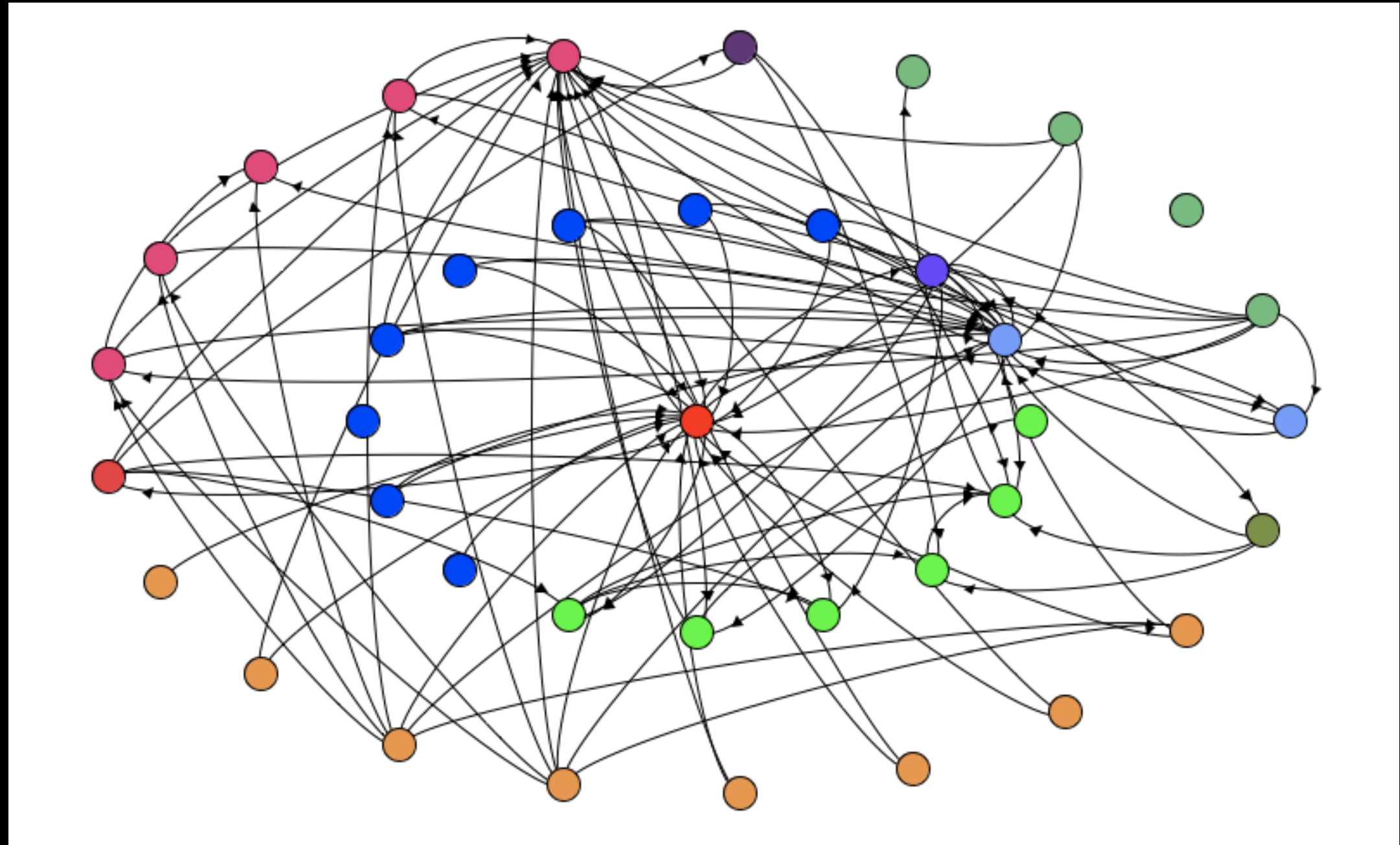
Refactoring

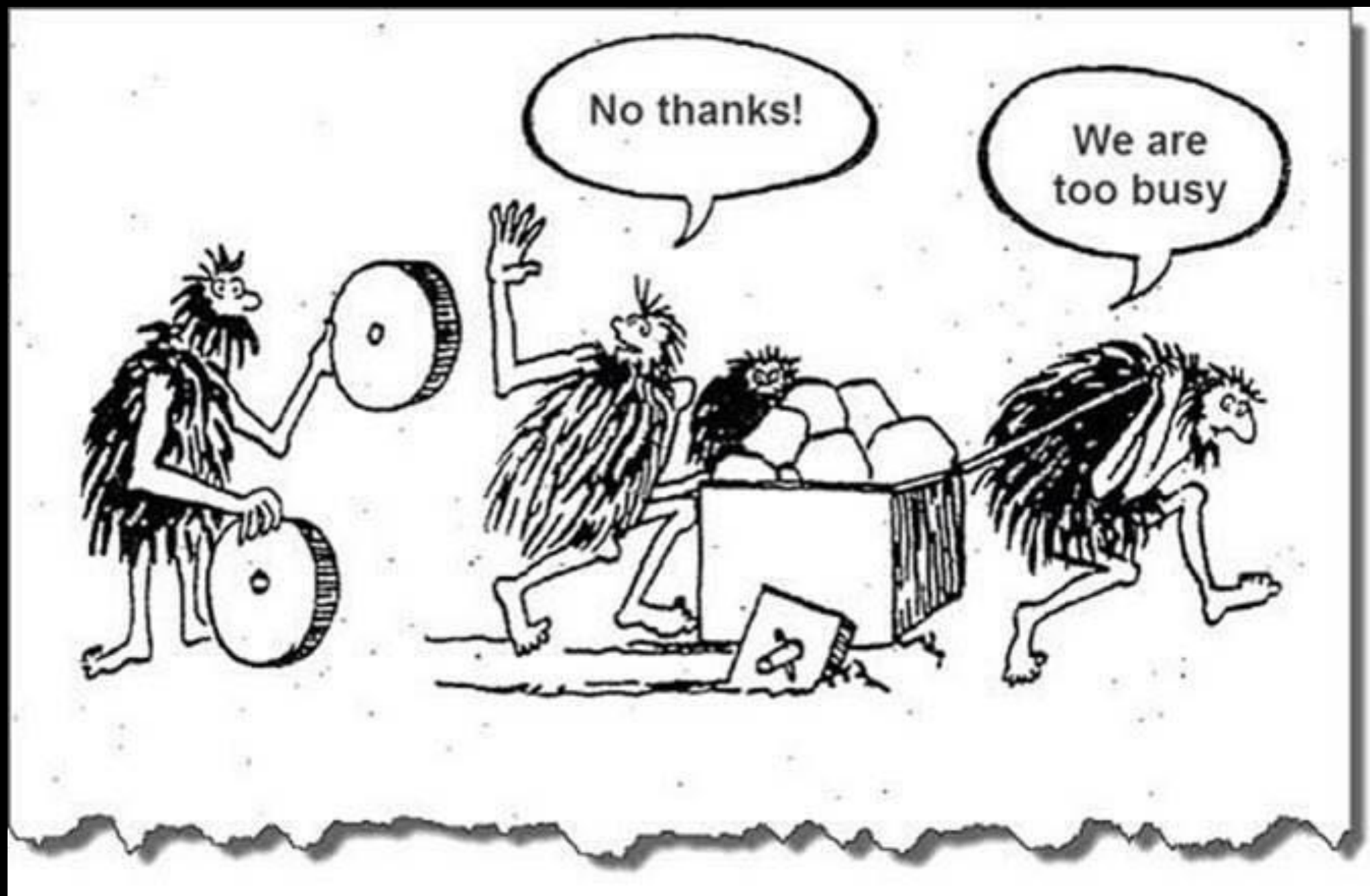
Reuse problems



```
181  
182 /**  
183  * Return a dummy VersionedWaveletDelta instance used  
184  * use in searches within a NavigableSet of deltas.  
185  *  
186  * @param version the version with which to return the delta  
187  * @return a dummy versioned delta with a null delta  
188  */  
189 private static VersionedWaveletDelta emptySerializedDelta(  
190     return new VersionedWaveletDelta(null, HashedVersionedWaveletDelta.  
191 )  
192  
193 /** A comparator to be used in a TreeSet for deserializing  
194 private static final Comparator<VersionedWaveletDelta>  
195     new Comparator<VersionedWaveletDelta>() {  
196         @Override  
197         public int compare(VersionedWaveletDelta first, VersionedWaveletDelta second) {  
198             if (first == null && second != null) { return -1; }  
199             if (first != null && second == null) { return 1; }  
200             if (first == null && second == null) { return 0; }  
201             return Long.valueOf(first.version.getVersion()).compareTo(second.version.getVersion());  
202         }  
203     }  
204  
205 /** A comparator to be used in a TreeSet for transforming  
206 @VisibleForTesting  
207 static final Comparator<ProtocolWaveletDelta> transform  
208     new Comparator<ProtocolWaveletDelta>() {  
209         @Override  
210         public int compare(ProtocolWaveletDelta first, ProtocolWaveletDelta second) {  
211             if (first == null && second != null) { return -1; }  
212             if (first != null && second == null) { return 1; }  
213             if (first == null && second == null) { return 0; }  
214             return Long.valueOf(first.getHashedVersion().getVersion()).compareTo(second.getHashedVersion().getVersion());  
215         }  
216     }  
217  
117 readLock.unlock();  
118 }  
119  
120 protected void acquireWriteLock() {  
121     writeLock.lock();  
122 }  
123  
124 protected void releaseWriteLock() {  
125     writeLock.unlock();  
126 }  
127  
128 /** A comparator to be used in a TreeSet for applied deltas  
129 protected static final Comparator<ProtocolAppliedWaveletDelta>  
130     new Comparator<ProtocolAppliedWaveletDelta>() {  
131         @Override  
132         public int compare(ProtocolAppliedWaveletDelta first, ProtocolAppliedWaveletDelta second) {  
133             if (first == null && second != null) { return -1; }  
134             if (first != null && second == null) { return 1; }  
135             if (first == null && second == null) { return 0; }  
136             return Long.valueOf(first.getVersionAppliedAt().getVersion()).compareTo(second.getVersionAppliedAt().getVersion());  
137         }  
138     }  
139 }  
140  
141 /**  
142  * Return a dummy ProtocolWaveletDelta instance used as a  
143  * boundary for use in searches within a NavigableSet of  
144  *  
145  * @param version the version to return the delta applied at  
146  * @return the generated dummy delta  
147  */  
148 private static ProtocolWaveletDelta emptyDeltaAt(VersionedWaveletDelta version) {  
149     return ProtocolWaveletDelta.newBuilder()  
150         .setAuthor("dummy")  
151         .setHashedVersion(WaveletOperationSerializer.serialize(version))  
152         .build();  
153 }
```

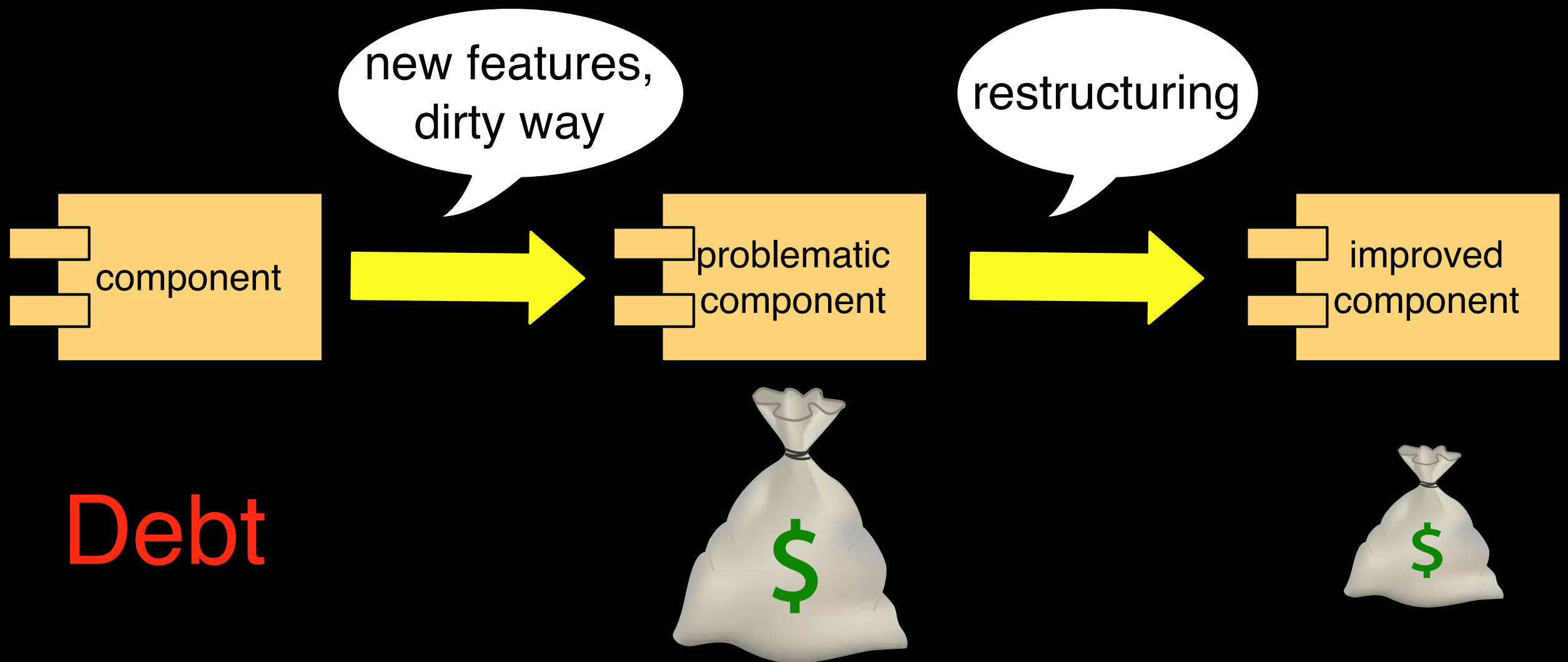
Modularity problems





How and
when to
solve them?

Technical debt



Maintenance costs more

Much of the early work in program restructuring was inspired by the need to reduce the cost of maintaining programs. It has been shown that the principle cost of any software development is the maintenance after the software is “done.” A study of one Air Force system revealed that it cost \$30 per line to develop and \$4,000 per line to maintain over its lifetime [Boe75]. By analyzing the IBM OS/360 project, Belady and Lehman determined that the cost of a change rose exponentially with respect to a system’s age. They attributed this rising cost to the decay of the software’s structure [BL71, BL76]. Gerald Weinberg maintains a private list of the world’s most expensive program errors. The top three errors involved the change of exactly one line of code [Wei83]. His theory as to why these errors happened is that since the actual change was so small, the programmers did not take the time to fully test the code or consider the ramifications of the change.

Lehman's first law

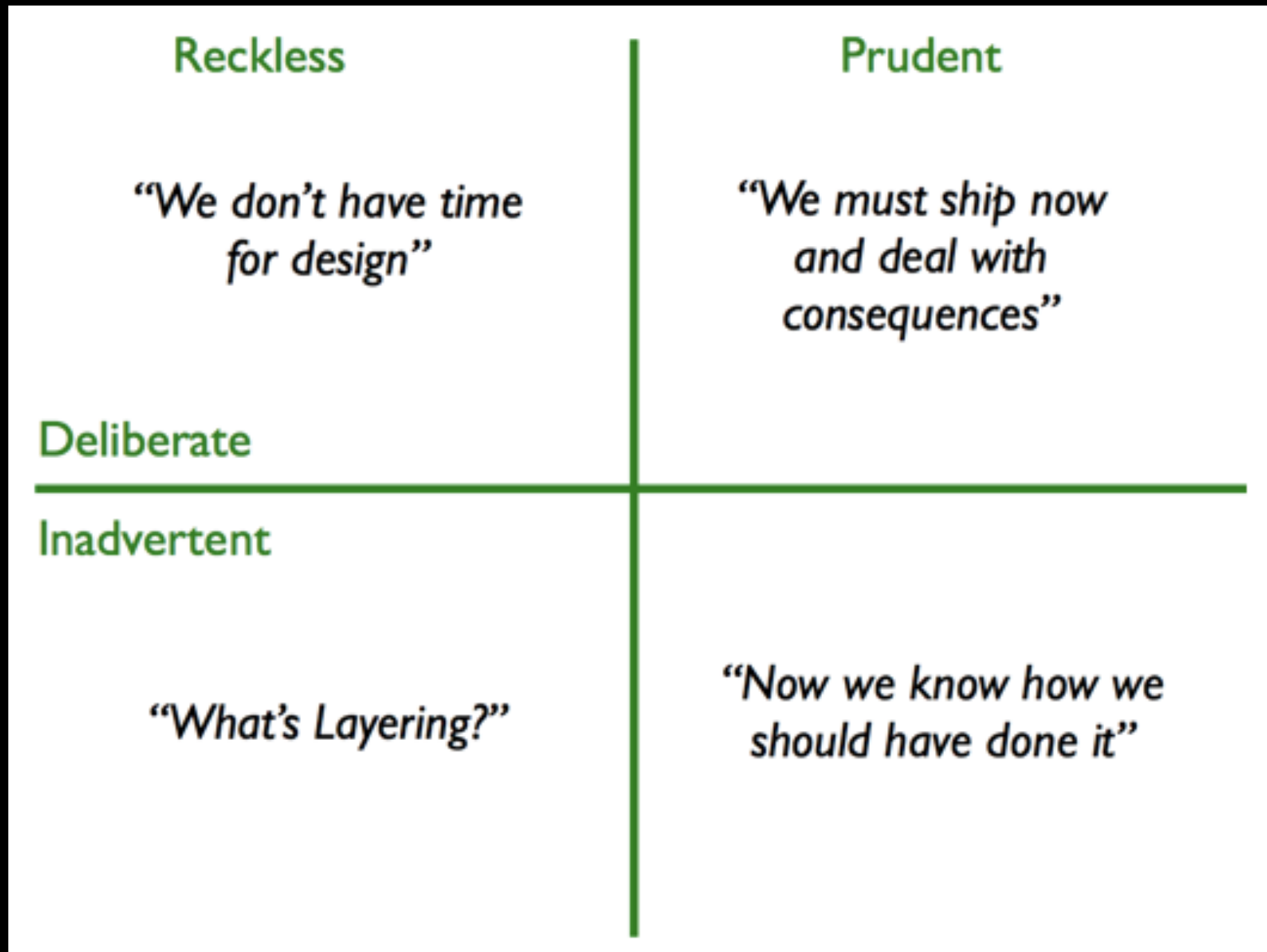
“A large program that is used undergoes continuing change or becomes progressively less useful”

“The change process continues until it is judged more cost effective to replace the system with a recreated version”

Lehman's second law

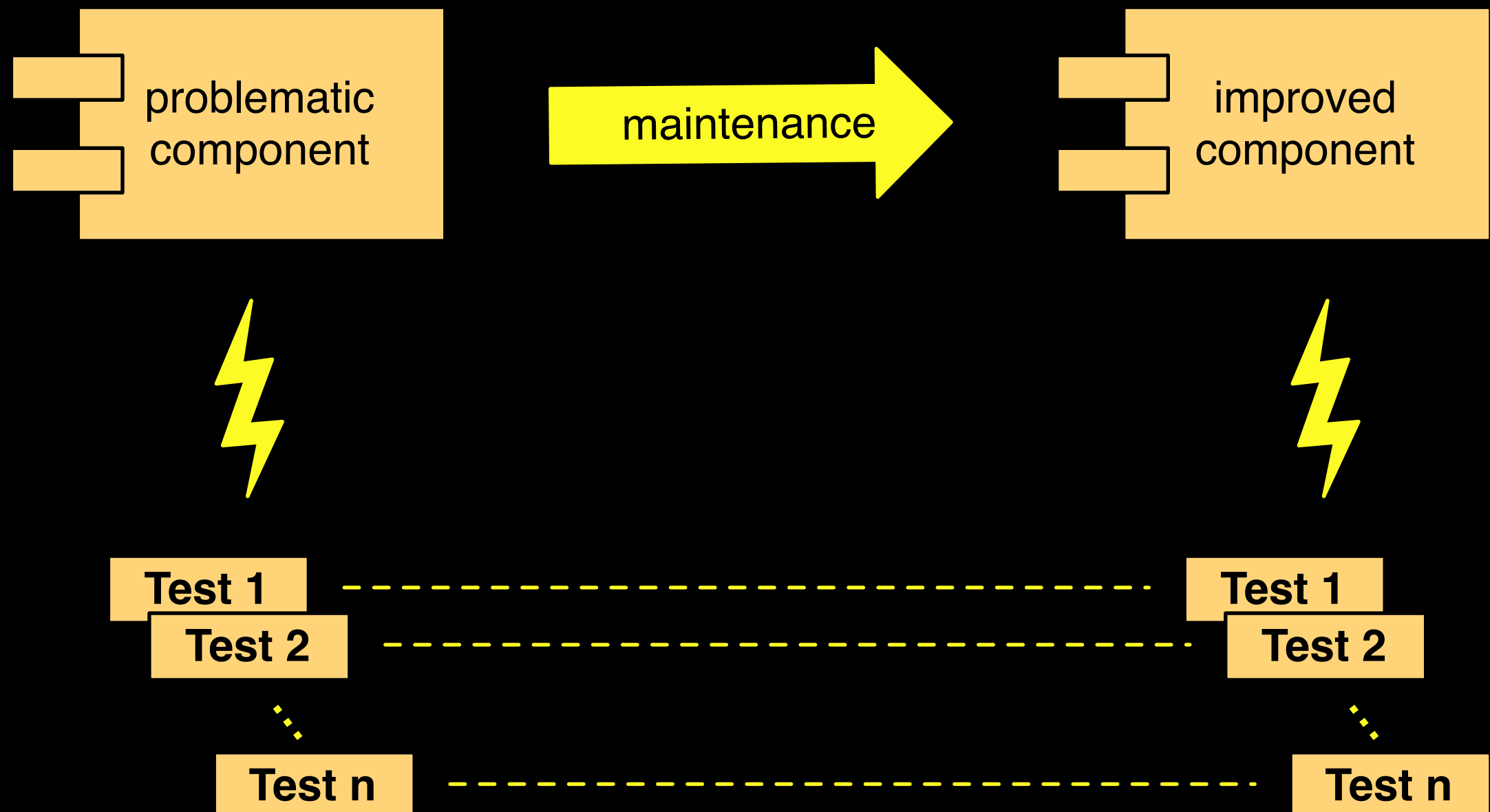
“As a large program is continuously changed, its complexity, which reflects deteriorating structure, increases unless work is done to maintain or reduce it”

Technical debt quadrant



How to solve
the problems
and reduce the
debt?

Preserving behavior



Refactorings are...

*behavior-preserving
source-to-source
transformations that
improve some internal quality
factors*

Reuse and modularity problem

```
if(objeto.getNome() == null ||  
    objeto.getNome().equals("") ||  
    objeto.getSobrenome() == null ||  
    objeto.getSobrenome().equals("") ||  
    objeto.getTipo() == null ||  
    objeto.getTipo().equals("") ||  
    ...
```


Extract method refactoring

```
boolean nullOrEmpty(Membro s) {  
    return s.getNome() == null ||  
           s.getNome().equals("");  
}
```

```
boolean nullOrEmpty(String s) {  
    return s == null ||  
           s.equals("");  
}
```

Debt reduced

```
if(nullableEmpty(objeto.getNome()) ||  
    nullableEmpty(objeto.getSobrenome()) ||  
    nullableEmpty(objeto.getTipo()) ||  
    ...
```

```
if(objeto.getNome() == null ||  
    objeto.getNome().equals("")) ||  
    objeto.getSobrenome() == null ||  
    objeto.getSobrenome().equals("")) ||  
    objeto.getTipo() == null ||  
    objeto.getTipo().equals("")) ||  
    ...
```

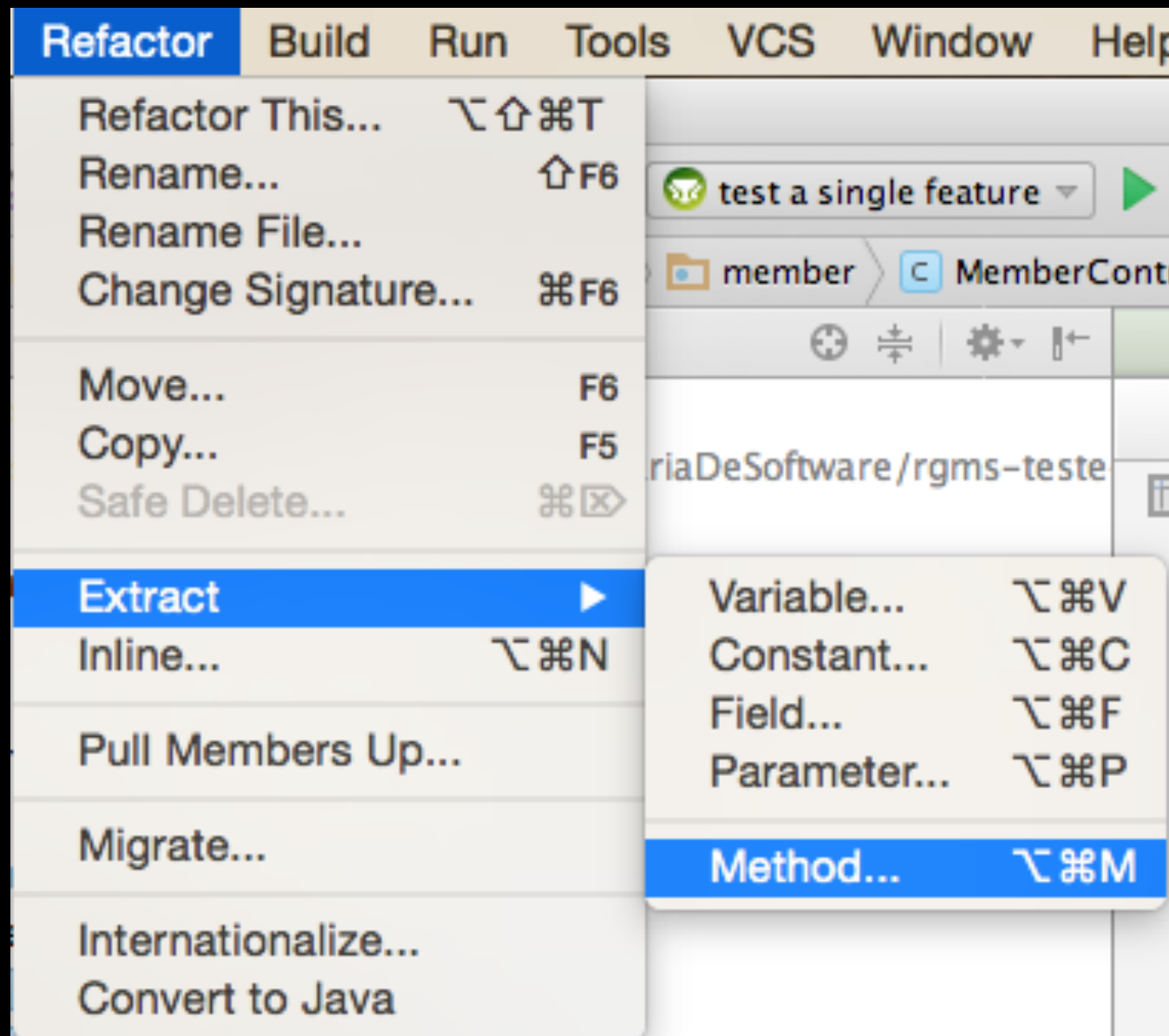
Debt eliminated

```
if(invalidStringFields(objeto)
```

```
...
```

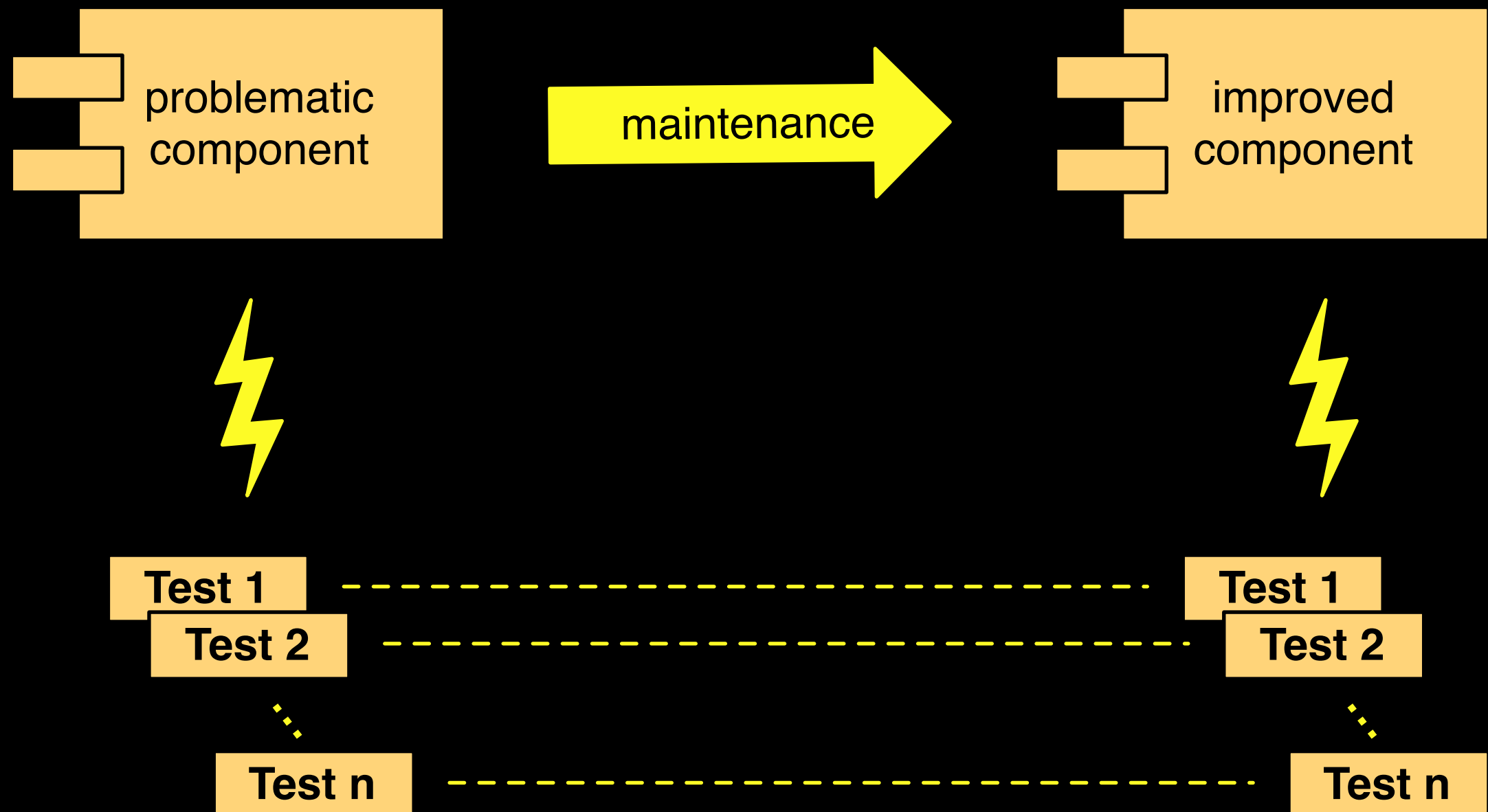
```
    if(nullOrEmpty(objeto.getNome()) ||  
        nullOrEmpty(objeto.getSobrenome()) ||  
        nullOrEmpty(objeto.getTipo()) ||  
        ...
```

Automatic refactorings



Refactor	Navigate	Search	Project	Run
Rename...				⌘ R
Move...				⌘ V
Change Method Signature...				⌘ C
Extract Method...				⌘ M
Extract Local Variable...				⌘ L
Extract Constant...				
Inline...				⌘ I
Convert Anonymous Class to Nested...				
Convert Member Type to Top Level...				
Convert Local Variable to Field...				
Extract Superclass...				
Extract Interface...				
Use Supertype Where Possible...				
Push Down...				
Pull Up...				
Extract Class...				
Introduce Parameter Object...				
Introduce Indirection...				
Introduce Factory...				
Introduce Parameter...				
Encapsulate Field...				
Generalize Declared Type...				
Infer Generic Type Arguments...				
Migrate JAR File...				
Create Script...				
Apply Script...				
History...				

Tools give no behavior preservation guarantee

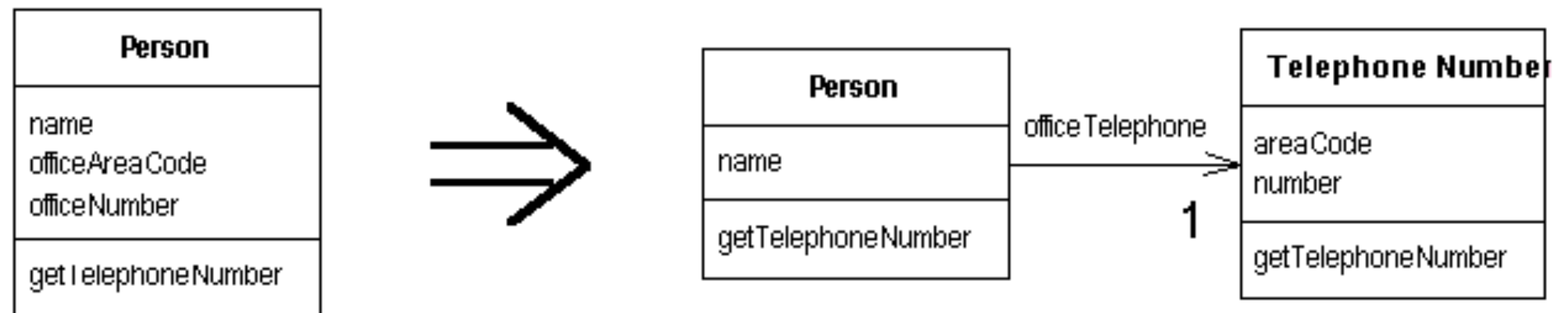


Refactoring catalogue

Extract Class

You have one class doing work that should be done by two.

Create a new class and move the relevant fields and methods from the old class into the new class.



<http://www.refactoring.com/catalog/index.html>

Black belt
parametrization!

Smells good?

functions as
parameters

```
artigos.each{|a|
  map = {}
  a.elements.each("DADOS-BASICOS-DO-ARTIGO"){|d|
    atts = d.attributes
    map["TITULO-DO-ARTIGO"] = atts["TITULO-DO-ARTIGO"]
    map["ANO-DO-ARTIGO"] = atts["ANO-DO-ARTIGO"]
  }
  a.elements.each("DETALHAMENTO-DO-ARTIGO"){|d|
    atts = d.attributes
    map["TITULO-DO-MEIO"] = atts["TITULO-DO-MEIO"]
    map["VOLUME"] = atts["VOLUME"]
    map["FASCICULO"] = atts["FASCICULO"]
    map["PAGINA-INICIAL"] = atts["PAGINA-INICIAL"]
    map["PAGINA-FINAL"] = atts["PAGINA-FINAL"]
  }
}
```

...

Not only for articles...

```
...  
a.elements.each("AUTORES"){ |author|  
  map["AUTORES"] = (if map["AUTORES"] then  
    map["AUTORES"]  
    else [] end) +  
    [author.attributes["NOME-PARA-CITACAO"]]  
}  
}
```

XML structure as parameter

```
structure = {  
  "DADOS-BASICOS-DO-ARTIGO" =>  
    ["TITULO-DO-ARTIGO", "ANO-DO-ARTIGO"],  
  "DETALHAMENTO-DO-ARTIGO" =>  
    ["TITULO-DO-MEIO", "VOLUME", "FASCICULO",  
    "PAGINA-INICIAL", "PAGINA-FINAL"],  
  "AUTORES" =>  
    ["*", "NOME-PARA-CITACAO"]  
}
```

Abstracting structure details

```
structure.keys.each {|e|
  a.elements.each(e) {|d|
    atts = d.attributes
    if structure[e].include?("*") then
      (structure[e] - ["*"]).each {|att|
        map[e] = (if map[e] then map[e] else [] end) +
          [atts[att]]
      }
    else
      structure[e].each {|att|
        map[att] = atts[att]
      }
    end
  }
}
```

Improvement is more
often needed than not,
so abstract to see it!

How to choose
refactoring
targets?

Bad smells based on...

- Values
- Principles
- Patterns

Strategy

- Identify problem
- Any pattern for problem in the given context?
- New solution based on values and principles?
- Apply appropriate refactorings
- Test

Checklist

- Design and implementation conforms to discussed principles and patterns
- Most well known refactoring have been applied

Take notes,
now!

Refactoring research at CIn

- Refactoring of software product lines: Paulo, Leopoldo
- Formal refactoring: Márcio, Augusto

Hands-on!
Check assignment

To do after class

- Answer questionnaire (check classroom assignment), study correct answers
- Finish exercise (check classroom assignment), study correct answers
- Read, again, chapter 9 in the textbook
- Evaluate classes (check classroom assignment)
- Study questions from previous exams

To do after class, optional

- estudar material (definição, quadrate) sobre débito técnico
- estudar catálogo de refactorings, e assistir vídeo se você não é familiar com a noção de refactoring
- ler resumos ou assistir vídeos do debate Is TDD dead?

Questions from previous exams

- Explique (a) o que é teste de regressão e (b) porque eles são úteis para atividades de refatoração.
- Cite (a) um ``mau cheiro" associado à refatoração “extract class”, e (b) explique qual a mudança sugerida por essa refatoração.
- Explique brevemente quais as vantagens de refatorar o código e porque algumas empresas não realizam esta atividade rotineiramente.
- Cite um ``mau cheiro" associado à refatoração extract method, e explique qual a mudança sugerida por essa refatoração.

Software and systems engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

phmb@cin.ufpe.br ♦ twitter.com/pauloborba