Software and systems engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

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

To do before class

- Watch video
- Read chapters 8 and 11 in the textbook
- Send questions and opinions through slack

Design, implementation, and maintenance

Model driven development (MDD)

Low level design and implementation

Should be guided by...

Values

Principles

Patterns

architectural, design, implementation

Coding values, Beck

- Communication: focus on the reader
- Simplicity: eliminate excess complexity
- Flexibility: easy to change, in expected ways
- Precision: names aligned to behavior

SOLID design principles

- A class should have a single responsibility (single responsibility)
- A module should be open for extension but closed for modification (open closed)
- Subclasses should be substitutable for their base classes (Liskov Substitution)
- Dependencies should be injected into a dependent object (dependency injection)
- A method of class C should only call methods of C or of the classes of C fields (Demeter)

More design principles

- Depend upon Abstractions, do not depend upon concretions (dependency inversion)
- Reusable code calls custom code, abstractions call concretions (inversion of control)
- Many client specific interfaces are better than one general purpose interface (interface segregation)

Package cohesion principles

- The granule of reuse is the granule of release (release reuse equivalency)
- Classes that change together, belong together (common closure)
- Classes that aren't reused together should not be grouped together (common reuse principle)

Package coupling principles

- The dependencies betwen packages must not form cycles (acyclic dependencies)
- Depend in the direction of stability (stable dependencies)
- Stable packages should be abstract packages (stable abstractions)

Other rules

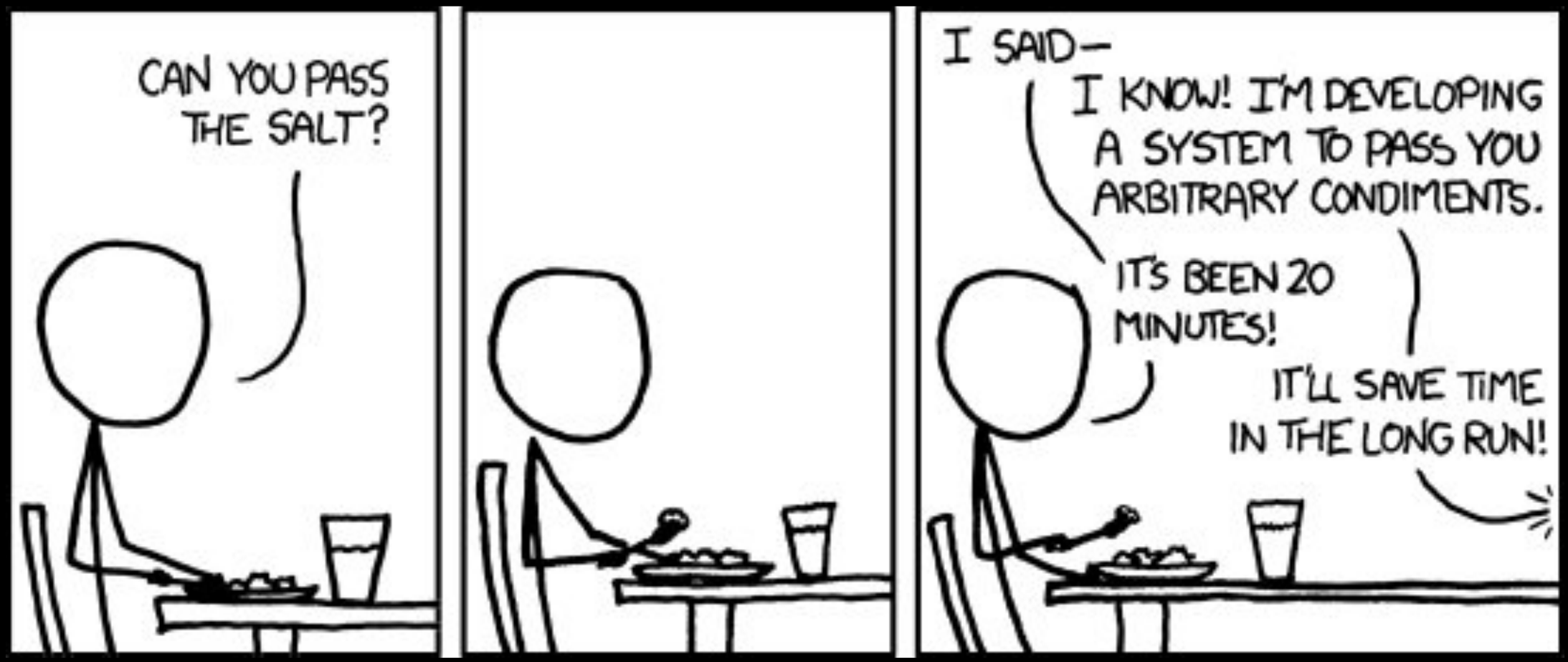
Rule of Modularity: Write simple parts connected by clean interfaces.
Rule of Clarity: Clarity is better than cleverness.
Rule of Composition: Design programs to be connected to other programs.
Rule of Separation: Separate policy from mechanism; separate interfaces from engines.
Rule of Simplicity: Design for simplicity; add complexity only where you must.
Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.
Rule of Transparency: Design for visibility to make inspection and debugging easier.
Rule of Robustness: Robustness is the child of transparency and simplicity.
Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.

The Art of Unix Programming, Eric Raymond

Other rules

Rule of Least Surprise: In interface design, always do the least surprising thing.
Rule of Silence: When a program has nothing surprising to say, it should say nothing.
Rule of Repair: When you must fail, fail noisily and as soon as possible.
Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.
Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.
Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
Rule of Diversity: Distrust all claims for “one true way”.
Rule of Extensibility: Design for the future, because it will be here sooner than you think.

The Art of Unix Programming, Eric Raymond

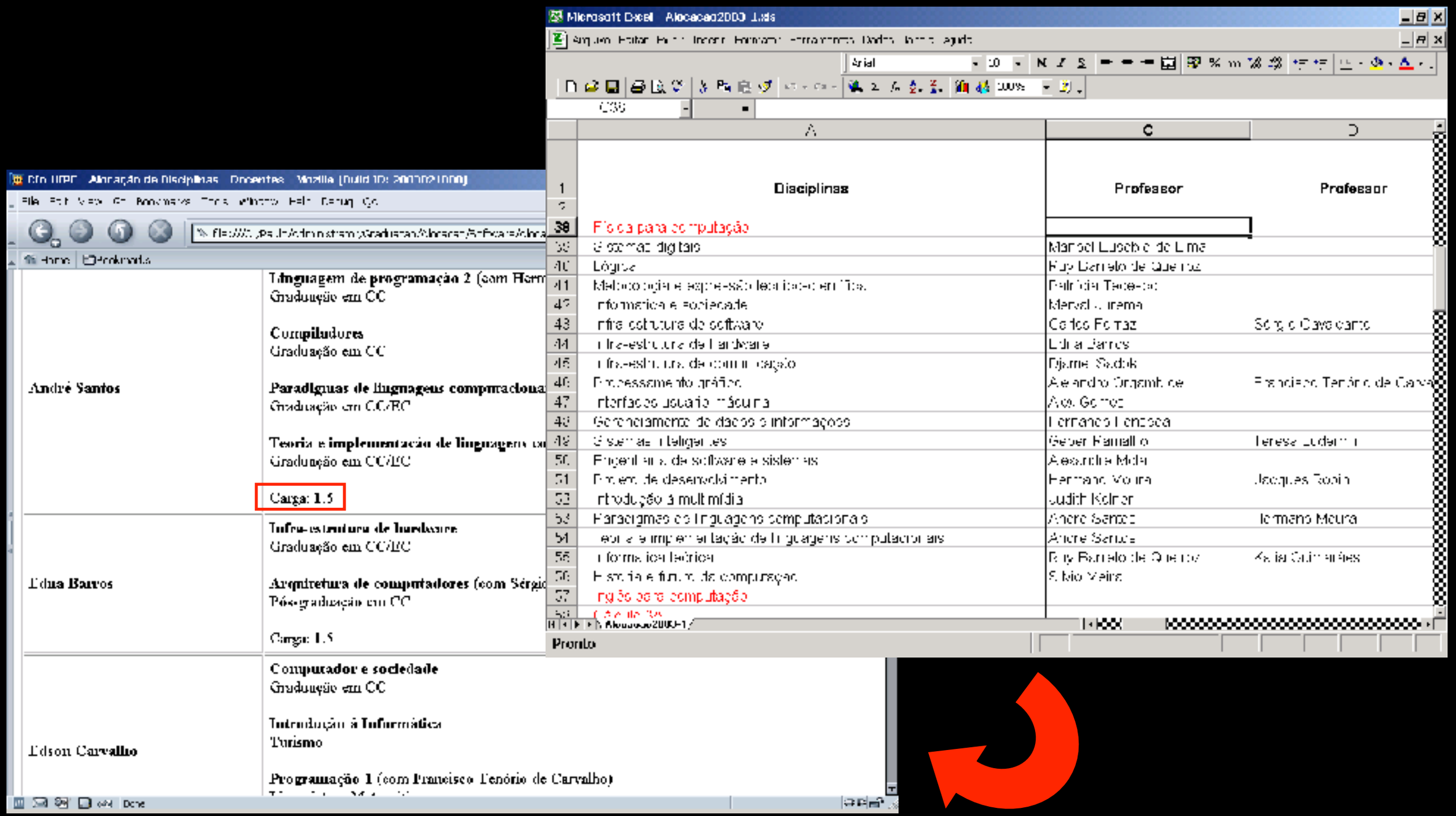


https://xkcd.com/

Patterns

- Context
- Problem
- Forces
- Solution
- Consequences
- Known uses

Professors and courses

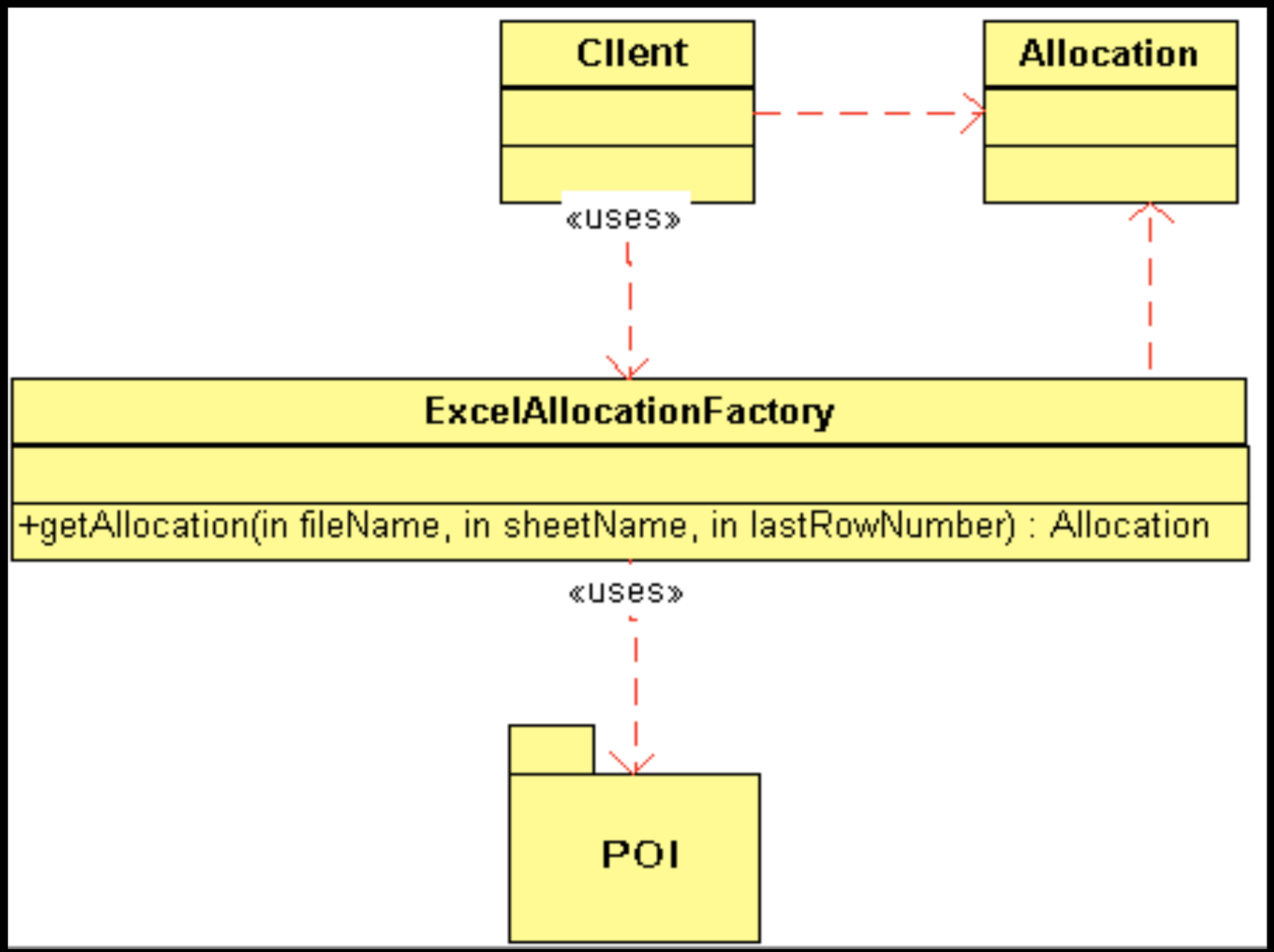


Java POI

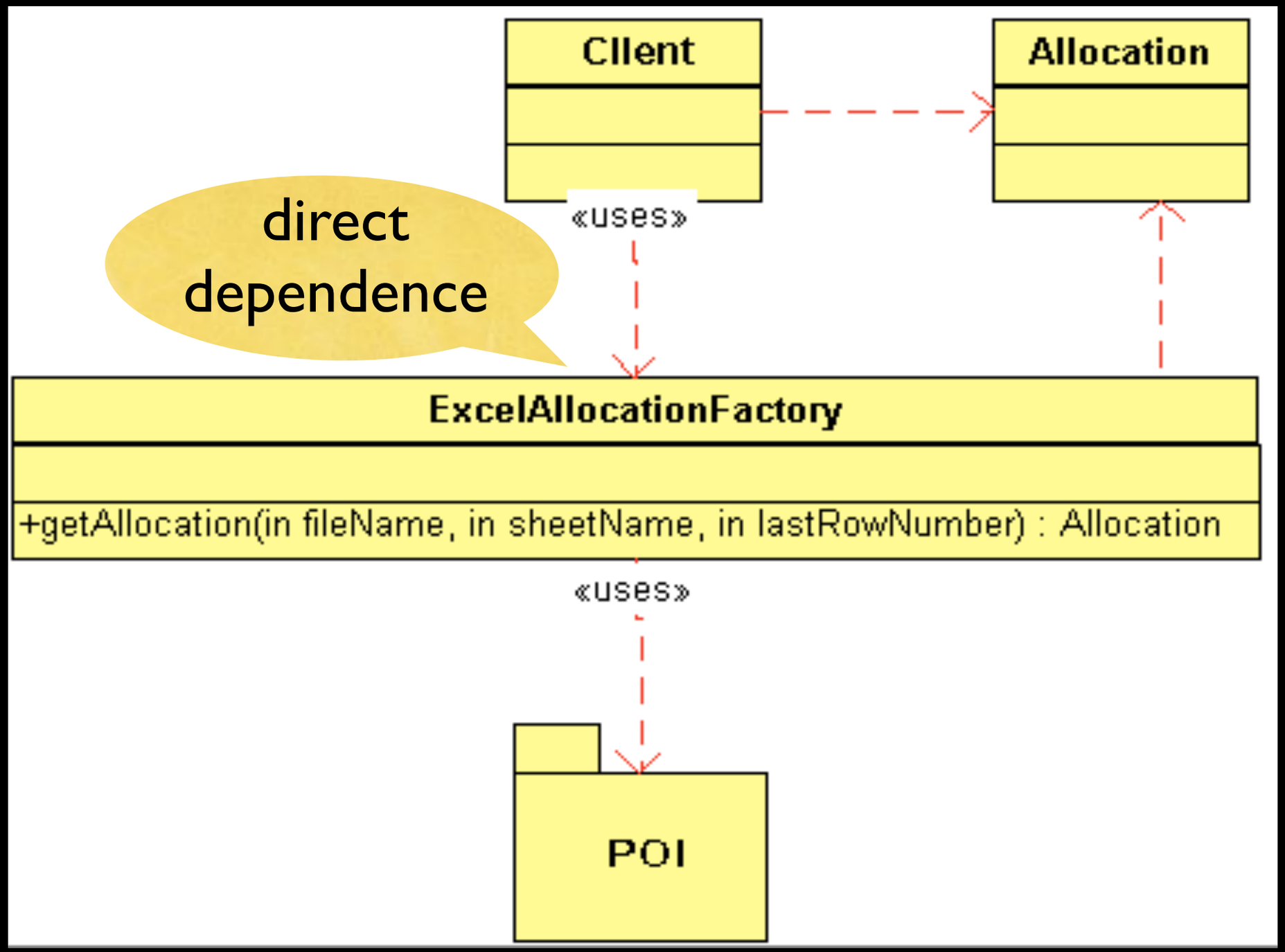
Java APIs Java for manipulating “OLE 2 Compound Document” files, such as xls e doc:

- HSSF (Horrible Spreadsheet Format)
- HDF (Horrible Document Format)
- HPSF (Horrible Property Set Format)

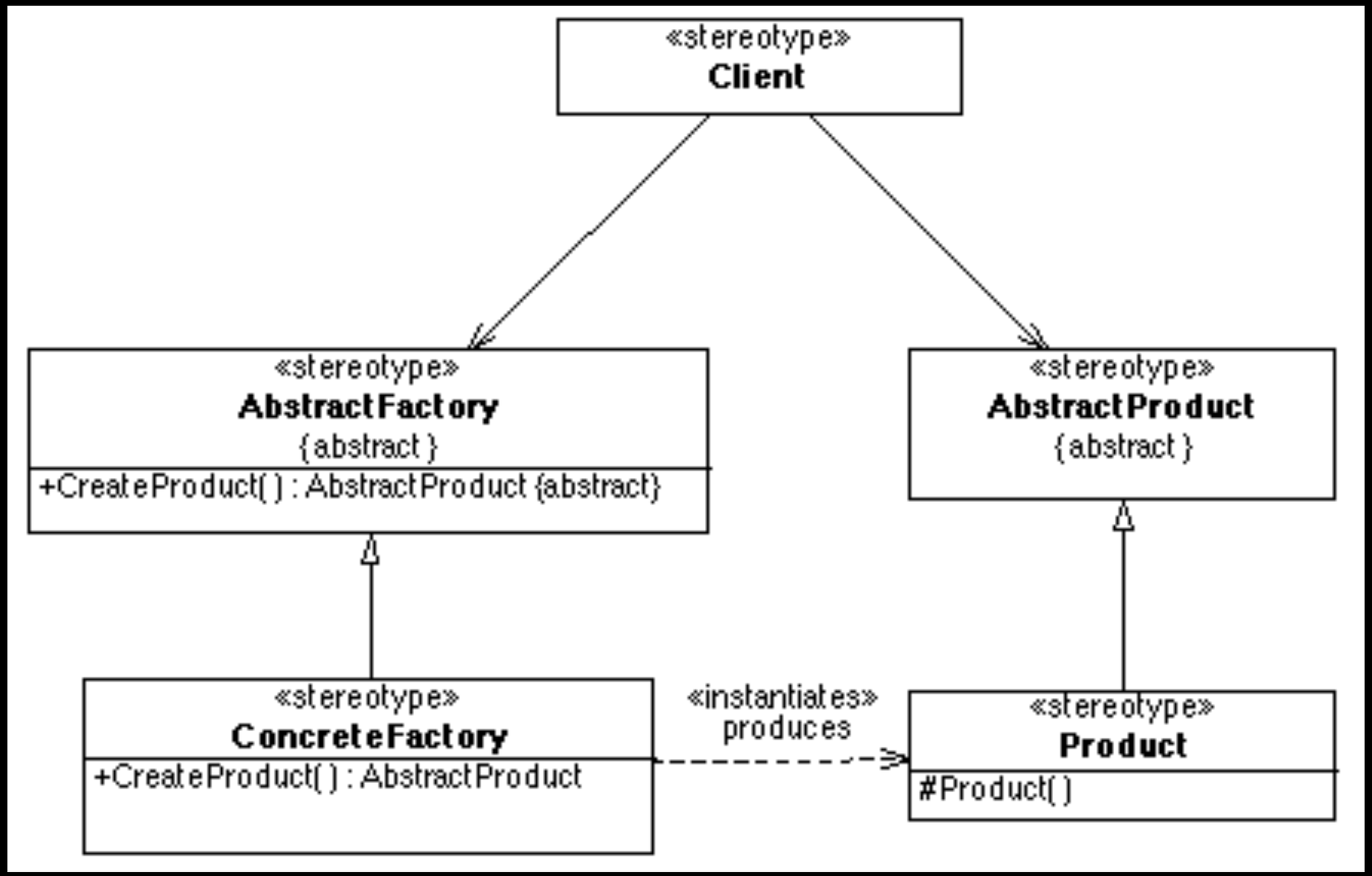
Factory pattern for using POI



No interface?

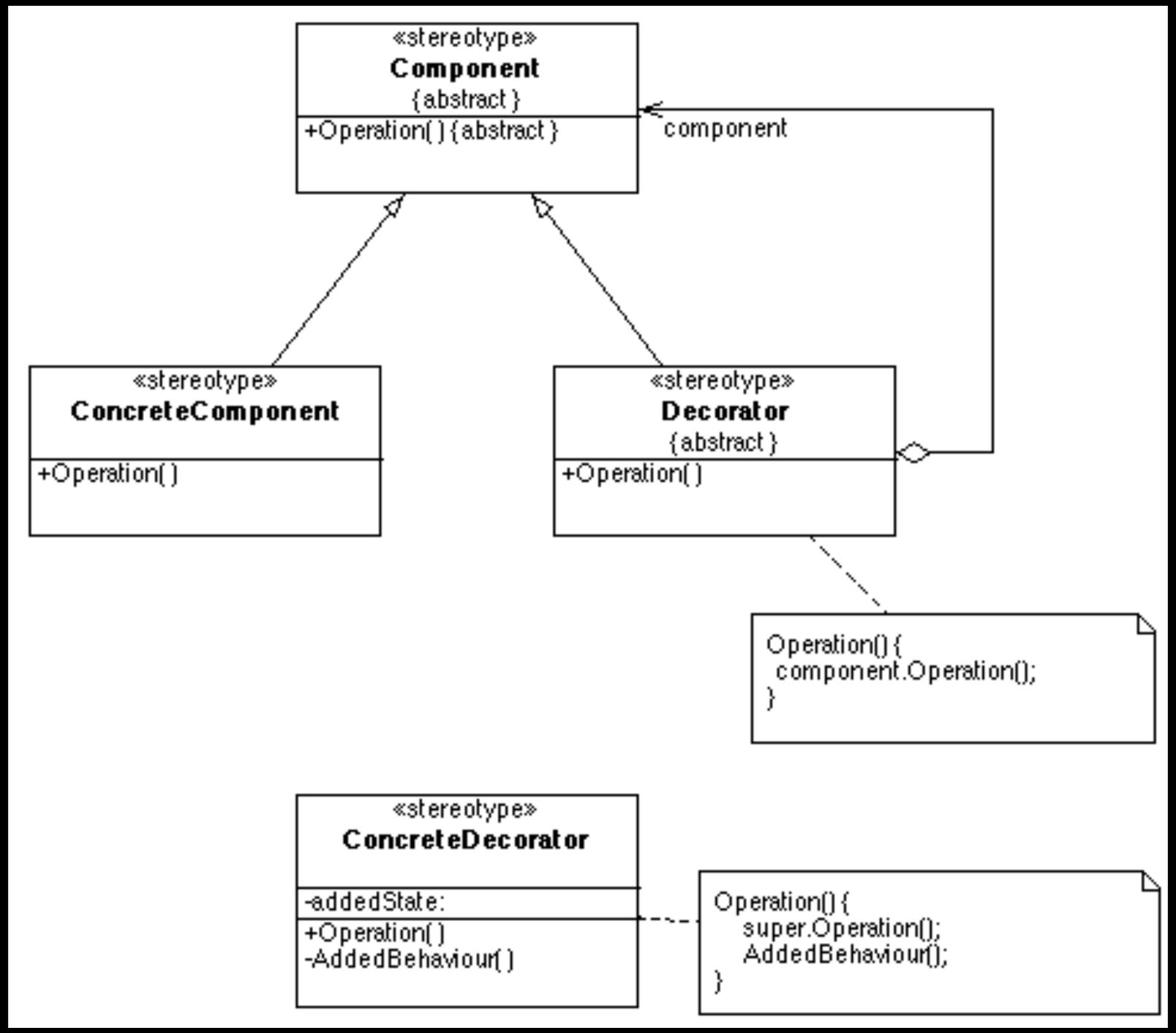


Abstract factory



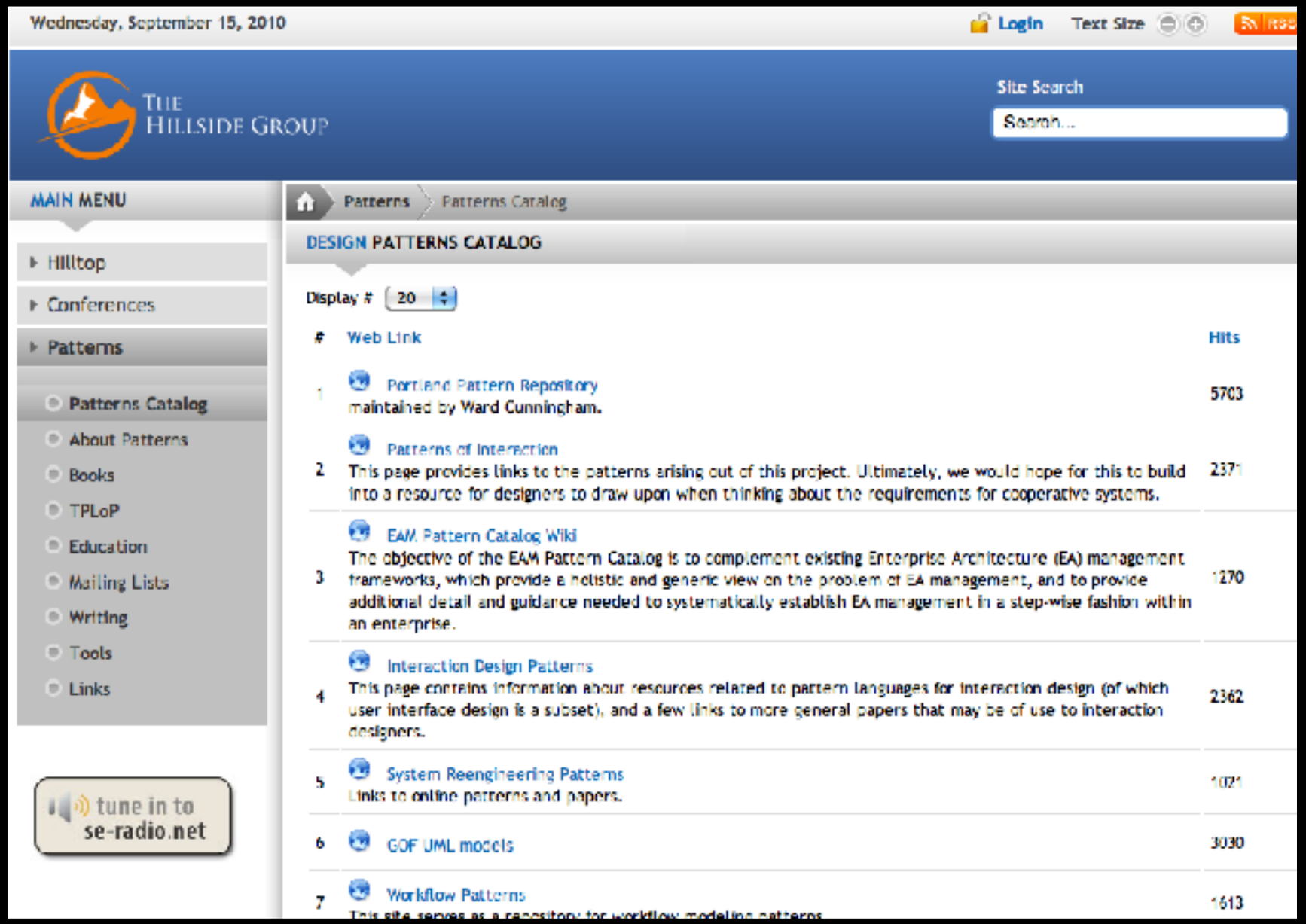
http://www.tml.tkk.fi/~pnr/Tik-76.278/gof/html/Abstract-factory.html

Decorator



http://www.tml.tkk.fi/~pnr/Tik-76.278/gof/html/Decorator.html

Pattern catalogues



http://hillside.net/patterns/patterns-catalog

Strategy

- Analyze change request
- Any pattern for the given context?
- New solution based on values and principles?
- Apply appropriate pattern or principles
- Check whether new functionality breaks existing tests

For now, focus on bad design smells, related to classes and their relationships

Later we will focus on bad code smells, related to method implementations

Applies for testing code too!

Checklist

- Design and implementation conforms to discussed principles and patterns

Take notes,

now!

Design, implementation and evolution research at CIn

- Software product lines and evolution: Leopoldo, Paulo
- Formal specification methods: Augusto, Alexandre Mota, Juliano, Henrique, Márcio
- Concurrent systems and green design: Fernando Castor
- Distributed systems architecture: Vinicius, Kiev

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, chapters 8 and 11 in the textbook
- Evaluate classes (check classroom assignment)
- Study questions from previous exams

To do after class, optional

- Estudar material sobre princípios de projeto
- Estudar princípios de codificação no Capítulo 3 do livro Implementation patterns, Kent Beck, 2008 (slides úteis para ter uma visão geral do livro)
- Navegar pelo catálogo de padrões

Questions from previous exams

- Descreva o princípio "Dependency Injection". Explique as vantagens de seguir este princípio.
- Descreva o objetivo do padrão de projeto "Adapter", e desenhe um diagrama de classes representando a sua estrutura.
- O princípio "Acyclic dependencies"estabelece que "The dependencies between packages must not form cycles". Explique as vantagens de seguir esse princípio.
- Desenhe um diagrama de classes que ilustre o padrão de projeto "Decorator", e explique as suas vantagens.

Questions from previous exams

- Descreva o princípio "Single Responsibility Principle". Explique as vantagens de seguir este princípio.
- Descreva o princípio de projeto "Demeter". Explique as vantagens de seguir este princípio.
- Desenhe um diagrama de classes que ilustre o padrão de projeto "Abstract Factory".

Software and systems engineering

Paulo Borba
Informatics Center
Federal University of Pernambuco

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