

# IWDPTP'05

International Workshop on Design Pattern Theory and Practice

Yann-Gaël Guéhéneuc  
Giuliano Antoniol

Budapest, Hungary

2005/09/25



**Ptidej Team – OO Programs Quality Evaluation and Enhancement using Patterns**

Group of Open, Distributed Systems, Experimental Software Engineering

Department of Informatics and Operations Research

University of Montreal

© Giuliano Antoniol, Yann-Gaël Guéhéneuc



# Participants

- Giuliano Antoniol `antoniol@ieee.org`
- Francesca Arcelli `arcelli@disco.unimib.it`
- Mario Luca Bernardi `mlberna@unisannio.it`
- Lajos Fülöp `flajos@inf.u-szeged.hu`
- Yann-Gaël Guéhéneuc `guehene@iro.umontreal.ca`
- Christian Lange `c.f.j.lange@tue.nl`
- Giuseppe di Lucca `dilucca@unisannio.it`
- Claudia Raibulet `raibulet@disco.unimib.it`



# Papers

1. Improving Design Patterns Modularity using Aspect Orientation – Mario L. Bernardi, Giuseppe A. Di Lucca
2. Semantics of a Pattern System – Ashraf Gaffar and Naouel Moha
3. Evaluating the Use of Design Patterns during Program Comprehension: Experimental Setting – Yann-Gaël Guéhéneuc, Stefan Monnier, and Giuliano Antoniol
4. Enhancing Software Evolution with Pattern Oriented Software Product Life Cycle – Jayadev Gyani and P.R.K. Murti
5. A Taxonomy and a First Study of Design Pattern Defects – Naouel Moha, Duc-loc Huynh, and Yann-Gaël Guéhéneuc
6. The Role of Design Pattern Decomposition in Reverse Engineering Tools – Claudia Raibulet and Francesca Arcelli



# Putting It All Together

- Understanding patterns
  - Catalogue
  - Taxonomy (DPD)
  - Decomposition
- Using patterns
  - Quantitative evaluation of the impact  
evaluation of their impact
  - Use of patterns by maintainers
  - Pattern-based software product life cycle

# Research Questions

1. How can we decide the intent of DP with respect to its functioning?
2. How to represent DP in (UML) models and source code?
3. Where do we find programs using DP with available (UML) models?
4. What is the variation of classes in DP wrt. classes not in DP?
5. How much DP ease program comprehension (in absence of models, doc.)?
  - Do DP promote/prevent software evolution?
  - Do DP complexify code or not?
6. What are similarities among DP?
7. Does AOP make DP implementation easier/better?
8. What terminology must we use when talking about DP?
9. Does it make sense to decompose DP?
  - How do we identify sub-components?
10. Can we detect DP (from a catalogue) with good precision/recall?
  - Static vs. Dynamic vs. Hybrid analyses?
11. Can we build a complete catalogue of variants of DP?
12. Is there defects in DP?
13. What about DP in imperative languages?
  - Can we use object recovery techniques?



# Discussions

2. How to represent DP in (UML) models and source code?
  - Lightweight extensions
    - Naming conventions
    - Stereotypes
    - Colours/symbols → **DP views**
    - Annotations (comments, JDK 1.5.0)
  - Extension of the UML meta-model (profile)
  - Lexical vs. Structural vs. Semantic info.



# Follow-up

- Mailing list

- Site web

`ptidej.iro.umontreal.ca/organisations/iwdptp/`

- Bulletin Board

`www-etud.iro.umontreal.ca/~guehene/phpBB2/`

- Next workshop?