

DEQUALITE: A method to build quality models considering software design

Foutse Khomh

Yann-Gaël Guéhéneuc

foutsekh@iro.umontreal.ca, guehene@iro.umontreal.ca

MOSART'08



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

Context

- Quality models usually use metrics of classes (such as number of methods) or of relationships between classes (for example coupling) to measure internal attributes of programs.
- The quality of object-oriented programs does not depend on classes solely: It also depends on the organisation of classes also, i.e., the program architecture.
- We present a method DEQUALITE to build quality models while taking into account the design of systems.

Related work

- Our method DEQUALITE relates to three distinct areas of research:
- quality models, design pattern detection and combination of models

- ◆ Quality models;

Wydaeghe *et al.* assess the quality characteristics of the architecture of an OMT editor through the study of 7 design patterns and concluded on the flexibility, modularity, reusability, and understandability of the architecture and the patterns. However, they do not link their assessment with any quality model.

Wendorff *et al.* in "Assessment of design patterns during software reengineering: Lessons learned from a large commercial project" Provided evidences that design patterns do not always improve the quality of systems.

- *Design of systems have an impact on their quality*

Related work

◆ Design pattern detection

Antoniol et al. developed a technique to identify structural patterns in a system in order to examine how useful a design pattern recovery tool could be in program understanding and maintenance.

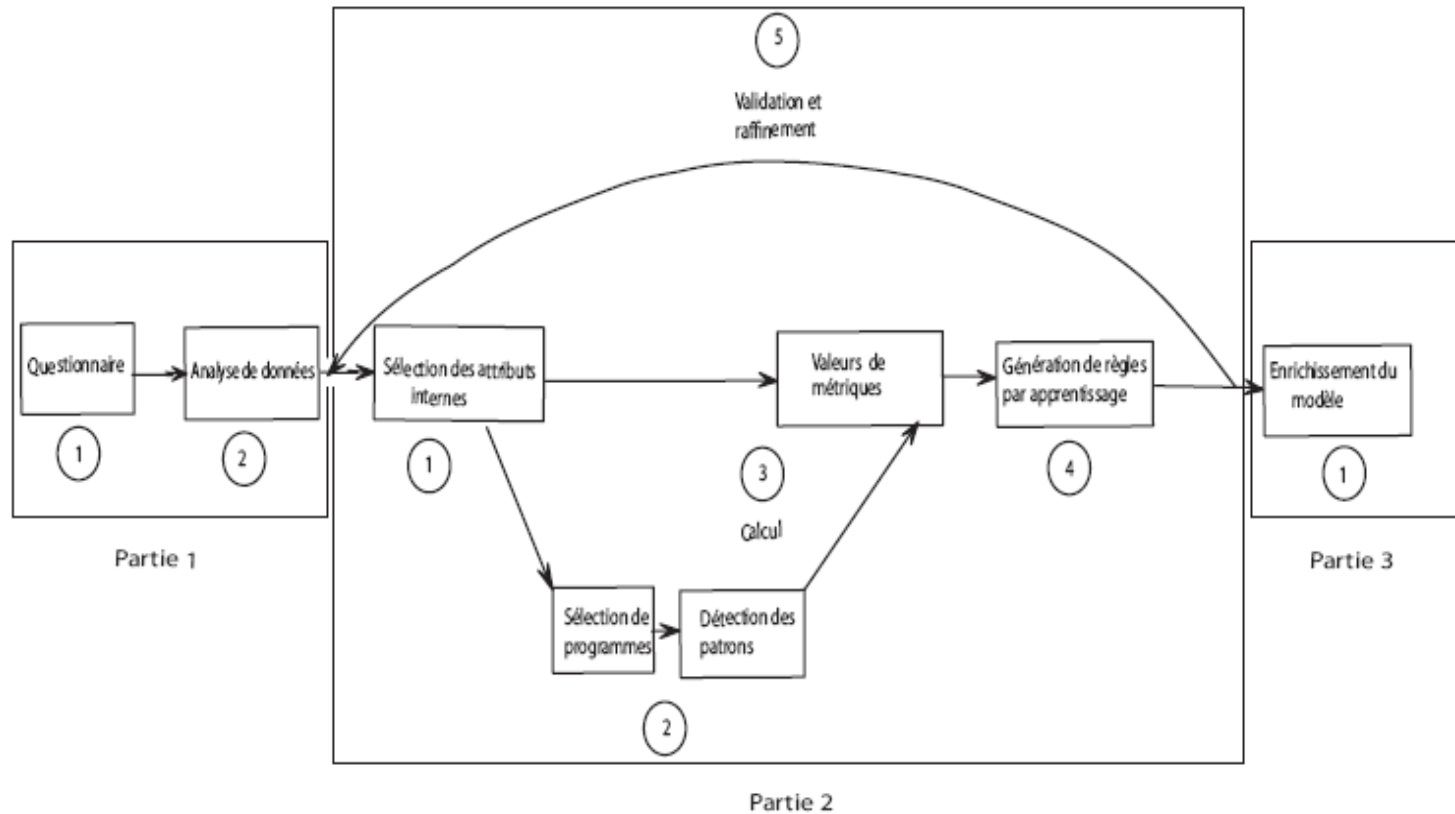
Geheneuc et al. Presented a technique that uses explanation-based constraint programming to identify design patterns from systems.

• Combination of models

Bouktif et al. Presented a method to combine quality models in order to improve their predictive power.

Our method is built on these previous efforts

Method:DEQUALITE





Method: DEQUALITE

- Part 1: Study of the impact of design patterns on 23 quality attributes
- Part 2: Build a quality model PQMOD that takes in account design patterns
- Part 3: We use a technique of mixture of experts to combine quality models from the literature with the quality model built in part 2.

Method: DEQUALITE

Part1: Design patterns and quality of systems.

This part of the method is organised in 2 steps

Step 1: Definition of the questionnaire

Choice of design patterns of interest

- 23 design patterns in the GoF book

Identification and definition of quality characteristics of interest

- 10 characteristics

Design	Implementation	Runtime
Expandability	Learnability	Generality
Simplicity	Understandability	Modularity at runtime
Reusability	Modularity	Scalability
		Robustness

Definition of a scale

- Likert scale of 5 points + "I don't know"

Method: DEQUALITE

■ Step 2: Data Collection and processing

- Questionnaire send out to colleagues with a good knowledge of design patterns between January and May 2007 and posted on three lists: refactoring, patterns-discussion and gang-of-4-patterns.
- Number of received questionnaires: 30
 - Number of complete questionnaires: 18
 - Number of almost complete (90%) questionnaires: 2
 - Incomplete questionnaires (or "I don't know"):10
- Number of kept questionnaires: 20
- Answers A and B aggregated in Positive, C in Neutral, and D and E in Negative
 - Some questionnaire were clearly generally more positive/negative than others

Result of data processing (1/3)

Design Patterns	Expendability(%)		Simplicity(%)		Generality(%)		Modularity(%)		Mod. at runtime(%)		
	E	R(%)	E	R(%)	E	R(%)	E	R(%)	E	R(%)	
A.Factory	+	0.00	+	30.36	+	1.76	+	0.37	-	30.36	
Builder	+	0.15	+	30.36	+	30.36	+	0.15	-	30.36	
F.Method	+	1.76	+	30.36	+	30.36	+	5.92	-	30.36	
Prototype	+	30.36	+	30.36	+	1.76	-	15.09	+	30.36	
Singleton	-	0.15	+	0.15	-	5.92	-	0.37	-	0.37	
Adapter	+	30.36	-	30.36	+	15.09	+	1.76	+	30.36	
Bridge	+	0.37	-	0.37	+	5.92	+	1.76	+	50.00	
Composite	+	0.00	+	5.92	+	1.76	+	5.92	+	30.36	
Decorator	+	0.15	-	5.92	+	0.15	+	5.92	+	5.92	
Facade	+	30.36	+	0.37	+	50.00	+	50.00	-	15.09	
Flyweight	-	1.76	-	0.00	-	0.15	-	5.92	-	0.15	
Proxy	-	30.36	+	15.09	+	15.09	+	1.76	-	15.09	
Ch.Of.Resp	+	0.15	+	5.92	+	0.37	+	5.92	+	30.36	
Command	+	5.92	-	30.36	+	15.09	+	15.09	-	30.36	
Interpreter	+	5.92	+	50.00	+	15.09	+	30.36	-	30.36	
Iterator	+	0.15	+	5.92	+	5.92	+	30.36	+	50.00	
Mediator	+	30.36	-	5.92	-	30.36	+	50.00	-	5.92	
Memento	-	5.92	+	50.00	-	30.36	-	0.15	-	0.15	
Observer	+	0.15	+	15.09	+	1.76	+	1.76	+	5.92	
State	+	5.92	+	15.09	+	15.09	+	15.09	+	30.36	
Strategy	+	1.76	+	5.92	+	5.92	+	0.37	+	1.76	
T.Method	+	0.37	+	5.92	+	1.76	-	5.92	-	5.92	
Visitor	+	5.92	-	0.15	+	1.76	+	1.76	+	30.36	
		19 + / 4 -		16 + / 7 -		19 + / 4 -		18 + / 5 -		11 + / 12 -	

Design Patterns	Learnability(%)		Understandability(%)		Reusability(%)		Scalability(%)		Robustness(%)		
	E	R(%)	E	R(%)	E	R(%)	E	R(%)	E	R(%)	
A.Factory	-	15.09	-	15.09	+	50.00	-	1.76	-	0.00	
Builder	+	30.36	+	0.37	-	15.09	-	0.00	-	0.00	
F.Method	+	50.00	-	30.36	+	15.09	-	5.92	-	0.00	
Prototype	-	30.36	+	30.36	+	30.36	-	5.92	-	0.15	
Singleton	+	0.37	+	0.15	-	0.37	-	15.09	-	0.37	
Adapter	+	5.92	-	30.36	+	5.92	-	0.00	-	0.00	
Bridge	-	5.92	+	50.00	-	30.36	-	0.15	-	0.15	
Composite	+	1.76	+	5.92	+	15.09	-	30.36	-	0.15	
Decorator	-	30.36	-	30.36	-	5.92	-	0.37	-	0.00	
Facade	+	5.92	+	1.76	-	5.92	-	1.76	-	1.76	
Flyweight	-	0.00	-	0.00	-	15.09	+	1.76	-	1.76	
Proxy	-	30.36	-	5.92	+	50.00	-	5.92	-	0.15	
Ch.Of.Resp	+	15.09	-	5.92	+	30.36	-	0.15	-	1.76	
Command	-	15.09	-	5.92	-	5.92	-	1.76	-	5.92	
Interpreter	+	5.92	+	5.92	+	30.36	-	5.92	-	0.15	
Iterator	+	50.00	+	50.00	+	5.92	-	0.37	-	30.36	
Mediator	+	30.36	+	30.36	-	1.76	-	1.76	-	0.15	
Memento	-	30.36	-	30.36	-	15.09	-	0.00	-	0.15	
Observer	+	50.00	-	30.36	+	50.00	-	0.37	-	0.15	
State	+	30.36	+	30.36	-	1.76	-	0.37	-	0.15	
Strategy	+	1.76	+	15.09	-	30.36	-	1.76	-	5.92	
T.Method	+	30.36	-	15.09	+	30.36	-	1.76	-	0.37	
Visitor	-	0.37	-	1.76	-	1.76	-	1.76	-	0.00	
		14 + / 9 -		11 + / 12 -		11 + / 12 -		1 + / 22 -		0 + / 23 -	

Result of data processing (2/3)

- Globally, design patterns have a positive impact on quality attributes

Quality Characteristics	Expandability	Simplicity	Reusability	Learnability	Understandability
Number of Positive/Negative Patterns	19+ / 4-	16+ / 7-	11+ / 12-	14+ / 9-	11+ / 12-

Quality Characteristics	Modularity	Generality	Mod. at runtime	Scalability	Robustness
Number of Positive/Negative Patterns	18+ / 5-	19+ / 4-	11+ / 12-	1+ / 22-	0+ / 23-

Result of data processing (3/3)

Patrons de conception	Caractéristiques de qualité									
	Extensibilité	Simplicité	Généralité	Modularité	Modularité à l'exécution	Facilité d'apprentissage	Compréhensibilité	Réutilisabilité	Passage à l'échelle	Robustesse
Fabrique abstraite	G	G	G	G	G	B	G	G	N	N
Monteur	G	G	G	G	G	G	G	N	N	N
Fabrique	G	G	G	G	G	G	G	G	N	N
Prototype	G	G	G	N	G	G	G	G	N	N
Singleton	B	G	N	N	N	G	G	N	N	N
Adaptateur	G	G	G	G	G	G	G	G	B	N
Pont	G	B	G	G	G	B	G	B	B	N
Objet composite	G	G	G	G	G	G	G	G	G	N
Décorateur	G	N	G	G	G	G	B	B	B	N
Façade	G	G	G	G	B	G	G	N	N	N
Poids-mouche	N	B	B	B	N	B	B	B	G	N
Proxy	N	G	G	G	N	G	N	G	B	N
Chaîne de responsabilité	G	G	G	G	G	G	B	G	N	N
Commande	G	G	G	G	G	G	B	B	B	N
Interpréteur	G	G	G	G	G	G	G	G	B	N
Itérateur	G	G	G	G	G	G	G	G	B	N
Médiateur	G	B	G	G	N	G	G	N	N	N
Memento	N	G	N	N	N	G	N	G	B	N
Observateur	G	G	G	G	G	G	G	G	N	N
État	G	G	G	G	G	G	G	B	N	N
Stratégie	G	G	G	G	G	G	G	G	N	N
Patron de méthode	G	G	G	N	N	G	G	G	N	N
Visiteur	G	B	G	G	G	B	B	B	B	N

Method: DEQUALITE

- Part2: Quality model: PQMOD
- *This part of the method is organised in 5 steps*
- *Step 1: We Choose internal attributes relevant to patterns and quality characteristics of systems and corresponding metrics.*
- *Step2: We Identify a set of systems implementing patterns BP.*

Method: DEQUALITE

Systèmes	Liste des patrons implantés	Nombre d'occurrences
QuickUML 2001	Fabrique abstraite, Monteur, Commande, Objet composite, Observateur, Singleton	7
Lexi v0.1.1	Monteur, Observateur et Singleton	5
JRefactory v2.6.24	Adaptateur, Monteur, Fabrique, Singleton, État, Visiteur	26
Netbeans v1.0	Fabrique abstraite, Adaptateur, Commande, Itérateur, Observateur	28
JUnitv3.7	Objet composite, Décorateur, Itérateur, Observateur, Singleton	8
JHotDraw v5.1	Adaptateur, Commande, Objet composite, Décorateur, Fabrique, Observateur, Prototype, Singleton, État, Stratégie, Patron de méthode	24
MapperXML v1.9.7	Fabrique abstraite, Adaptateur, Objet composite, Facade, Fabrique, Observateur, Singleton, Stratégie, Patron de méthode	16
Nutch v0.4	Singleton, Pont , Commande, Memento, Patron de méthode, Adaptateur, Stratégie, Itérateur	16
PMD v1.8	Adaptateur, Monteur, Objet composite, Fabrique	8

Method: DEQUALITE (Part 2)

- Step 3: We compute metrics on the patterns of systems from BP
- Step 4: We apply machine learning algorithms Jrip and J48 to find the rules between quality characteristics and values of the metrics, We selected the rules with the best classifications and obtained the quality model PQMOD:
- In this step, we have obtained 10 rules which happened to be in-line with the best practices recommended by the object-oriented programming paradigm.

Method: DEQUALITE (Part 2)

Example of rule: Rule of scalability

```
(DCMEC >= 0.1)
  et (NOC <= 0.65)
    et (DCMEC <= 0.2) => Scal=G
(AID >= 2.83)
  et (AID <= 2.92) => Scal=G
(connectivity <= 0.66)
  et (NOC >= 0.75)
    et (ACMIC <= 0.2) => Scal=B
(LCOM5 <= 0.63)
  et (ICHClass >= 0.25)
    et (NMA <= 5.88) => Scal=B
=> Scal=N
```

Validation of the model PQMOD

- We applied the model PQMOD on five systems and obtained the following results

Attributs	Nutch v0.4		Xerces v 1.4.4		Ant v1.7.0		GanttP. v2.0.4		PADL	
	Attendu	Prédit	Attendu	Prédit	Attendu	Prédit	Attendu	Prédit	Attendu	Prédit
Extensibilité	G	G	G	G	G	G	B	G	G	G
Généralité	N	G	G	G	G	G	B	G	N	G
Modularité	G	G	G	G	G	G	N	G	G	G
Mod. à l'exécution	G	G	N	G	G	N	B	N	N	G
Compréhensibilité	N	G	N	G	G	N	N	G	N	G
Réutilisabilité	G	G	N	G	G	G	B	G	N	G
Pass. à l'échelle	G	B	G	N	G	N	G	B	B	B

Tableau 4. Résultats de l'application du modèle de qualité PQMOD sur quelques systèmes.

Method: DEQUALITE

Part 3: Mixture of Experts

- We combined the model PQMOD with the model QMOOD (Bansiya et al.) as illustrated on this figure

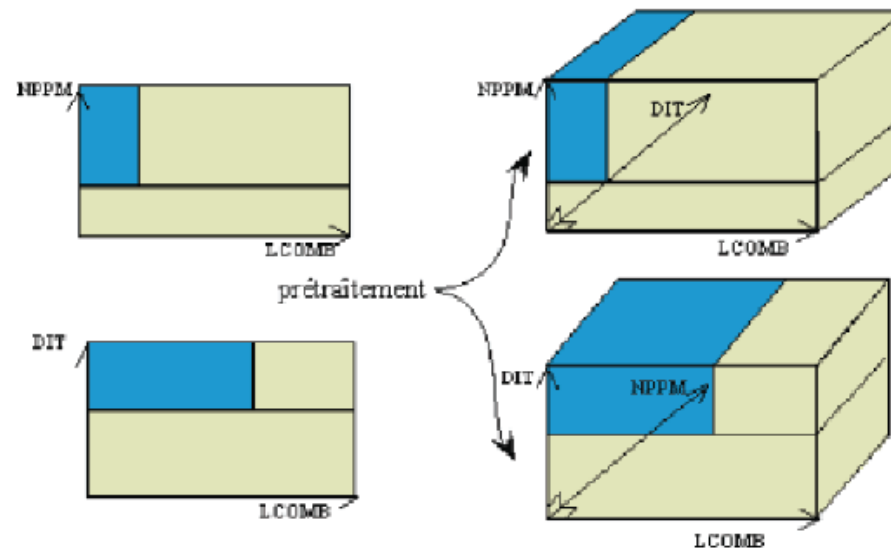


Figure 4. Unification des espaces d'entrée des arbres de décision en vue de leur fusion (cette figure est tirée de (Bouktif, 2005))

Method: DEQUALITE

Mixture of Experts: Application on PQMOD

- For the rule on extensibility we obtained the following result:

```
NOA <= 2
| AID <= 0.5: N
| AID > 0.5: M
NOA > 2
| (0.5)* ANA-(0.5)* DCC+(0.5)* MFA+(0.5)* NOPM
```

Conclusion

- We have presented a method DEQUALITE to build quality models while taking in account design patterns;
- The method:
 - Assess the impact of design patterns on the quality of systems;
 - The method build a quality model PQMOD from design patterns;
 - The method combine PQMOD to models from the litterature to improve their predictive power.



Future work

- We plan to improve quality models by taking in account

Design defects

Composition of patterns

- We also plan to explore other techniques of composition of models.