

Tyrannie

1972

David Parnas

structuration du code

modularité

séparation des préoccupations

Les choix de conception tendent à favoriser une décomposition "dominante"

```

check Nil eval Nil tester Nil
check Const eval Const tester Const
check Add eval Add tester Add
check Mult eval Mult tester Mult
check Fun eval Fun tester Fun
    
```

...ce qui affaiblit les décompositions secondaires

2011

JE N'Y ARRIVE PAS

MES OPÉRATIONS DE MAINTENANCE NE SONT PAS MODULAIRES

équipe de maintenance

HELP!

QUI PEUX M'AIDER?

JE SUIS SUPER-LAMBDA

JE VAIS T'AIDER

OH.

T'AIMERAIIS QUE MES PROGRAMMES PUISSENT CHANGER DE FORME

OH.

TU VEUX MANIPULER TES STRUCTURES DE DONNÉES DE FAÇONS DIFFÉRENTES ?

OH.

J'AI DES VUES À TE PROPOSER

JE NE PARLE PAS DES STRUCTURES DE DONNÉES, JE PARLE DES PROGRAMMES

data Expr =
 Const Int
 Add Expr Expr
 eval (Add e1 e2) =
 eval e1 + eval e2
 tester (Const i) = show i
 toStr (Add e1 e2) =
 toStr e1 ++ "+" ++
 toStr e2

data Expr =
 Const Int
 Add Expr Expr
 fold f g (Const i) = f i
 fold f g (Add e1 e2) =
 g (fold f g e1) (fold f g e2)

module Const where
 module Add where
 eval x y = x + y
 toStr x y =
 x ++ "+" ++ y

eval = fold Const.eval Add.eval
toString = fold Const.toStr Add.toStr

J'AIMERAIS QUE MES PROGRAMMES AIENT LES BONNES PROPRIÉTÉS DE TOUTES LES ARCHITECTURES POSSIBLES.

OH.

IL S'AGIT DU PROBLÈME DE L'EXPRESSION

J'AI PLEIN DE SOLUTIONS

IL Y A PLEIN D'ASTUCES À APPLIQUER

BON, REGARDONS, MAIS JE N'AIME PAS TROP LES ASTUCES.

ET MON CHEF ENCORE MOINS

VOIS-LA

MAIS TU M'AS POURRI MON ARCHITECTURE INITIALE

OK, MONTRE-MOI

UN PEU PLUS TARD

ÇA Y EST, JE PEUX PASSER D'UNE STRUCTURE DE CODE À UNE AUTRE

ON A RÉUSSI

SALUT BOB, AS-TU RÉSOLU TON PROBLÈME?

NON, PAS ENCORE

AS-TU ESSAYÉ DE TRANSFORMER LA PREMIÈRE ARCHITECTURE VERS LA SECONDE?

COMMENT ÇA?

SI TON PROGRAMME A LE MÊME COMPORTEMENT DANS LES DEUX ARCHITECTURES, TU DOIS POUVOIR LE PROUVER.

ET DONC, TU DOIS POUVOIR DÉFINIR UNE TRANSFORMATION DE LA PREMIÈRE ARCHITECTURE VERS LA SECONDE

ALORS, TU PEUX TRANSFORMER TON PROGRAMME POUR QU'IL SOIT DANS L'ARCHITECTURE IDÉALE POUR TON OPÉRATION DE MAINTENANCE

ET ALORS?

JE COMMÈNCE À COMPRENDRE J'AURAI RÉALISÉ MA MAINTENANCE, JE POURRAI TRANSFORMER LE PROGRAMME VERS SON ARCHITECTURE INITIALE

TRÈS COMPLEXES

MAIS COMMENT JE FAIS?

PAS TANT QUE ÇA, IL EXISTE PLEIN D'OUTILS POUR TRANSFORMER DES PROGRAMMES

MAIS ÇA NE MARCHE PAS BIEN LES OUTILS DE REFACTORIZING

MAIS SI, IL Y EN A DES BONS

MAIS SI, JE NE VAIS PAS POUVOIR CHANGER D'ARCHITECTURE JUSTE AVEC UN OUTIL DE REFACTORIZING

MAIS SI

AVEC UNE POIGNÉE DE TRANSFORMATIONS ÉLÉMENTAIRES ON PEUT CONSTRUIRE DES TRANSFORMATIONS TRÈS COMPLEXES

TU ES PROGRAMMEUR, NON?

BOB A TROUVÉ UNE SOLUTION À SON PROBLÈME

MAIS CELLE-CI LUI A ÉTÉ CÔUTEUSE

QUELQU'UN POURRA-T-IL RENDRE CETTE SOLUTION PRATICABLE?

BOB POURRA-T-IL UTILISER DES OUTILS PLUS AUTOMATISÉS LA PROCHAÎNE FOIS QU'IL AURA BESOIN DE...

VUES DE PROGRAMMES

Views, Program Transformations, and the Evolutionary Problem in a Functional Language

Julien Cohen & Rémi Douence

1 Introduction

2 Modularity and Evolution

The Edit Operation, Babel, Tools, Haskell, Hugs

data Expr =
 Const Int
 Add Expr Expr
 fold f g (Const i) = f i
 fold f g (Add e1 e2) =
 g (fold f g e1) (fold f g e2)

module Const where
 module Add where
 eval x y = x + y
 toStr x y =
 x ++ "+" ++ y

eval = fold Const.eval Add.eval
toString = fold Const.toStr Add.toStr

References

[1] Julien Cohen and Rémi Douence, Views, Program Transformations, and the Evolutionary Problem in a Functional Language, Research Report hal-00484441, version 2, January 2010.

[2] Huiqing Li and Simon Thompson, Tool Support for Program Manipulation, Jan. 2008.

[3] David L. Parnas, On the criteria to be used in the analysis of programs to facilitate their modification, IBM Systems Journal, vol. 10, no. 3, pp. 181-212, 1965.

[4] Ben Taylor, Harold Oskey, William Harrison, and Stanley M. Joz, Sulfon, A degree of separation: multi-dimensional separation of concerns, In ICSE, 1998, ACM.

[5] Philip Wadler, The expression problem. Note to Java genericity mailing list, Nov., 1998.

[6] Philip Wadler, Views: A way for pattern matching to combat with data abstraction, In POPPL, 1999.

[7] Johannes M. M. Leuz of Program Evolution - Rules and Topics for Program Management, Infotech State of the Art Conf., Why Software Projects Fail, 1998.