Intensional Source-code Views and Relations

Kim Mens

Andy Kellens
BankAccount
  owner
  amount
  increase(amount)
  decrease(amount)
  calculateInterest()
  saveToDisk()

SavingsAccount
  calculateInterest()
  saveToDisk()

CheckingAccount
  calculateInterest()
  saveToDisk()

YouthAccount
  calculateInterest()
  expenseLimit()
  saveToDisk()

BankCard
  account
  pay(amount)
  checkPin()
  saveToDisk()

CreditCard
  pay(amount)
  enoughCredit()
  saveToDisk()

Source-code View
Collection of source-code entities
Extensional vs Intensional

Extensional View:
- CheckingAccount
- SavingsAccount
- YouthAccount

Intensional View:
- "All subclasses of BankAccount"
- if subclass(?entity,[BankAccount])
Intensional Views: alternatives

"All subclasses of BankAccount"

if subclass(?entity,[BankAccount])

"All classes in package BankAccount"

if classInPackage(?entity,BankAccount)

Exclude: BankAccount

Extensional Consistency
classInTheHierarchyOf(?class,[ConceptAnalysis.ContextCreator]),
or(methodInClass(?entity,?class),
and(metaClassOf(?meta,?class),
methodInClass(?entity,?meta)))
Intensional Relations

Accounts

reference

Bankcards

implement

Account changing Methods

implement

call

Persistence Methods

Intensional Relation
Intensional Relations

Account changing Methods → Persistence Methods

All account changing methods must call a persistence method

∀ x ∈ "Account changing methods"
∃ y ∈ "Persistence Methods"
x calls y

Q₁, Q₂ ∈ {∀, ∃, ∃!, ∃!, . . .}
V₁, V₂ ∈ Views
r = predicate over source-code entities
Deducing Relations

INPUT

Collection of source views
Collection of target views
Set of predicates
\( \forall \) and \( \exists \)

OUTPUT

Set of relations

Approach:
All combinations of views, predicates and quantors

Redundancy
Subset relations
"Weak" Relations
Subsumed Relations

Prune \( \exists - \exists \) by threshold on min. number of tuples

\( Q_1 x \in V_1; Q_2 y \in V_2: x \rightarrow y \)
Deducing Tool

Sources
- Root
- Licor Views
- AttributeCreator:Classes
- attributeFilterClass
- AttributeFilters
- BasicAnalyzerMethods
- CAFramework
- CAMainMethods
- ClassifierConcepts
- ConceptClasses
- ConceptCombines
- ConceptFilters
- ConceptLatticeClasses
- ContextCreationClasses
- ContextCreatorMetaClasses
- ContextCreatorMethods

Targets
- Filters
- GenericParseTreeAttributeGenerators
- HelperClasses
- MethodsOfContextCreator
- ObjectCreationMethods
- ObjectCreatorClasses
- ParseTreeAttributeCreators
- PredefinedAnalyzers
- PredefinedContextCreators
- PrivateConceptAnalysisMethods
- SimpleConcepts
- testTo5
- ToBeUnderstoodByAnalyzers
- ToBeUnderstoodByAttributeFilters
- ToBeUnderstoodByConceptFilters
- ToBeUnderstoodByContextCreatorMetaClass

Predicates
- equals
- dirtyOverridden
- classHasInstanceOfType
- classUnderstands
- methodReferencesClass
- methodInClass
- classReturnsClass
- classChainReturnsClass
- classChainReturnsInstanceOfClass
- classHasInstanceOfClass
- subclassOf
- methodSendsTransToMethod
- classReferencesClass
- methodInClassOrMetaClass
- methodSendsToMethod
- methodSendsToClass

Relations
<relation name> : for all x in attributeFilterClass : exists y in ToBeUnderstoodByAttributeFilters : x classUnderstands y
Case study: DelfStoF framework

- Initial Understanding
- Verification and refinement
- Deducing new relations
- Co-evolution

Encode assumptions using views and relations
Verify assumptions with code; refine where necessary
Deduce new relations
Apply documentation to newer versions
Initial Understanding

- Manually try to understand the framework
- Encode initial assumptions
  - 34 Views
  - 30 Relations

Restrictions

- No element of View A may refer an element of View B

Framework Interface

- All classes in view A must implement all selectors in view B

Encode assumptions using views and relations
Verification and refinement

- Inconsistencies: code - documentation
- Fix inconsistencies:
  - Assumptions were wrong
    - Refine documentation
  - Assumptions right; code inconsistent
    - Update/refactor code

FAILED!!!
All elements of view A must call a method from view B

Verify assumptions with code; refine where necessary
Deducing new relations

- Given:
  - 30 views
  - 15 predicates
- Deduced: 2335 relations
  - 90% automatic pruned
  - +/- 200 relations remained
- 17 non-trivial

Need for optimization
Co-evolution

- Use documentation with newer versions
- Find and fix inconsistencies
  - Update documentation
  - Update implementation

Extensions done by master student

Relations concerning coding conventions broken

Apply documentation to newer versions

Bigger cases needed!
IVs as Active Documentation

- Active Documentation
- Co-evolution between design and implementation
- Non-intrusive
- Explicit Formal Documentation
- Feedback on inconsistencies
- Crosscutting nature
- Manual declaration of Views
- Efficiency
- Need for more automation
Ongoing Work

- Formalism: Description Logics
- Mining of views
  - Formal Concept Analysis
  - Inductive Logic Programming
- Improving deducing of relations
- More and larger case studies
  - KAVA (Pharmacists society)
Questions