Towards an Application of Update Propagation on Logic Programs Representing Java Source Code

Richard Tantius
Daniel Speicher
Andreas Behrend
Improve and Extend my Project.

We already use tools, to evaluate Quality
I found a strange class. Maybe I should do some refactoring. Make a new class. Move some methods. I wonder what’s the effect on software quality?
Okay let's start.

This might take some time.

Done! Let’s take a look at the results.

..., and only 300 classes left.
Isn’t there a faster way?

Compute the refactoring impact, without executing the changes.
Content

• Basics
• Update Propagation
• Implementation
• Demo
Software Analyses

Help to evaluate software quality

Common Analysis: Software Metrics

Cohesion
Metric (example LCOM1)

Definition $LCOM1$:

Connected Pair
- M and N in C are connected
  - If M accesses a field F and N also accesses F
  - F is in C

$LCOM1$ counts the number of non connected method pairs
Cohesion Meta Model *

Prolog Model

Cohesion Model and Metrics
  • Good Prolog Adaptation
  • Generate from JT

Existing Infrastructure
  • JTransformer
    http://sewiki.iai.uni-bonn.de/research/jtransformer/start

  • Cultivate
    http://sewiki.iai.uni-bonn.de/research/cultivate/start
Prolog Model

class($C_{id}$).
class_field($C_{id}$, $a_{id}$).
class_method($C_{id}$, $P_{id}$).
class_method($C_{id}$, $Q_{id}$).
class_method($C_{id}$, $T_{id}$).
method_accesses_field($P_{id}$, $a_{id}$).
method_accesses_field($Q_{id}$, $a_{id}$).

...
Prolog Model

Class C

\[
\begin{align*}
  c(C_{id}). \\
  cf(C_{id}, a_{id}). \\
  cm(C_{id}, P_{id}). \\
  cm(C_{id}, Q_{id}). \\
  cm(C_{id}, T_{id}). \\
  mf(P_{id}, a_{id}). \\
  mf(Q_{id}, a_{id}). \\
  \ldots
\end{align*}
\]
**LCOM1** Prolog Query

### Connected Pairs
- M and N in C are connected
  - If M accesses a field F and N also accesses F
  - F is in C

```prolog
cp(C, M, N) :-
    mf(M, F), cf(C, F), mf(N, F).
```

### Non Connected Method Pairs
```prolog
lp(C, M, N) :-
    cm(C, M), cm(C, N), not(cp(C, M, N)).
```
lcom1(C, R):-
    findall([M,N], (cp(C, M, N), not(M=N)), E),
    length(E, T),
    R is T/2.

Query

cp(C, M, N) :-
    mf(M,F), cf(C, F), mf(N,F).

lp(C, M, N) :-
    cm(C, M), cm(C, N), not(cp(C, M, N)).
Refactoring Impact

Abstract Model

Fact Base

Source Code Level

Java Code

Refactoring

Metric Values

Abstract Model

Fact Base

Java Code

Metric Values

10/1/2012

Richard Tantius
Abstract Refactoring

Class C (LCOM1 = 0)

Class D

Model facts

\[ \text{cm}(D_{id}, T_{id}). \]

\[ \text{cm}(C_{id}, T_{id}). \]

\[ \text{cm}(C_{id}, T_{id}). \]
Refactoring

Metric Values

Abstract Model

Fact Base

Java Code

Update

Metric Values

Abstract Model

Fact Base

Java Code

Refactoring
Refactoring

Abstract Model

Update

Abstract Model

Metric Values

Metric Values

Manual Approach

Shortcut via Update Propagation

Abstract Model

Fact Base

Java Code

Update Propagation

Metric Values

Update

Refactoring

Update

Metric Values
Update Propagation

Generic approach

- Adapted from deductive databases.
- Not adapted for prolog until now
  - Reduced well-formed subset
Update Propagation

- Simulates new database state
  - Post refactoring
- Delta facts instead of updates
- Employs additional rules
  - Augmented rule set
Update Propagation

U. Griefahn.
Reactive Model Computation,

A. Behrend.
Soft Stratification for Transformation-Based
Approaches to Deductive Databases,
University of Bonn: PhD thesis, 2004
Abstract Refactoring

Class C ($LCOM1 = \emptyset$)

Class D

Model facts

$cm(C^{id}, T^{id})$.  
$del_{cm}(C^{id}, T^{id})$.  
$add_{cm}(D^{id}, T^{id})$.  

Richard Tantius 17
Update Propagation
Rule schema

Delta facts:

\[ \text{add}_q(Y, X) \]
\[ \text{del}_q(Y, X) \]
...

Update Propagation

Rule schema

Delta facts:

\texttt{add}_q(Y, X)

\texttt{del}_q(Y, X)

...

Propagation rules:

\[ p(X) :\neg q(X, Y), \quad r(Y), \quad \text{not}(s(Y)) \]
Update Propagation

Rule schema

Delta facts:

\[ \texttt{add}_q(Y, X) \]
\[ \texttt{del}_q(Y, X) \]
...

Propagation rules:

\[ p(X) \leftarrow q(X, Y), r(Y), \text{not}(s(Y)). \]
\[ \texttt{add}_p(X) \leftarrow \texttt{add}_q(X, Y), \texttt{new}_r(Y), \text{not} (\texttt{new}_s(Y)). \]
\[ \texttt{add}_p(X) \leftarrow \texttt{new}_q(X, Y), \texttt{add}_r(Y), \text{not} (\texttt{new}_s(Y)). \]
\[ \texttt{add}_p(X) \leftarrow \texttt{new}_q(X, Y), \texttt{new}_r(Y), \text{del}_s(Y). \]
Update Propagation

Rule schema

Delta facts:

\[ \text{add}_q(Y, X) \]
\[ \text{del}_q(Y, X) \]
...

Propagation rules:

\[ p(X) : - q(X, Y), \ r(Y), \ \text{not}(s(Y)). \]
\[ p(X) : - ... \]

\[ \text{add}_p(X) : - \text{add}_q(X, Y), \ \text{new}_r(Y), \ \text{not}(\text{new}_s(Y)), \ \text{not}(p(X)). \]

\[ \text{add}_p(X) : - \text{new}_q(X, Y), \ \text{add}_r(Y), \ \text{not}(\text{new}_s(Y)), \ \text{not}(p(X)). \]

\[ \text{add}_p(X) : - \text{new}_q(X, Y), \ \text{new}_r(Y), \ \text{del}_s(Y), \ \text{not}(p(X)). \]

\[ \text{del}_p(X) \ ... \]
Update Propagation

Rule schema

Delta facts:

\begin{align*}
\text{add}_q(Y, X) \\
\text{del}_q(Y, X)
\end{align*}

Propagation rules:

\begin{align*}
p(X) & : = q(X, Y), \quad r(Y), \quad \text{not}(s(Y)). \\
\text{add}_p(X) & : = \text{add}_q(X, Y), \quad \text{new}_r(Y), \quad \text{not}(\text{new}_s(Y)), \quad \text{not}(p(X)). \\
\text{add}_p(X) & : = \text{new}_q(X, Y), \quad \text{add}_r(Y), \quad \text{not}(\text{new}_s(Y)), \quad \text{not}(p(X)). \\
\text{add}_p(X) & : = \text{new}_q(X, Y), \quad \text{new}_r(Y), \quad \text{del}_s(Y), \quad \text{not}(p(X)). \\
\text{del}_p(X) \quad & \text{...}
\end{align*}

Transition rules:

\begin{align*}
\text{new}_A & : = A, \quad \text{not}(\text{del}_A). \\
\text{new}_A & : = \text{add}_A.
\end{align*}
Transition / Propagation Rules (LCOM1)

Mapping

\[ \text{cp}(C, M, N) \, \text{:-} \, \text{mf}(M, F), \, \text{cf}(C, F), \, \text{mf}(N,F). \]
\[ \text{lp}(C, M, N) \, \text{:-} \, \text{cm}(C, M), \, \text{cm}(C, N), \, \text{not}(\text{cp}(C, M, N)). \]

Query

\[ \text{lcom1}(C, R) \, \text{:-} \]
\[ \text{findall}([M,N], \, (\text{lp}(C, M, N), \, \text{not}(M=N)), \, E), \]
\[ \text{length}(E, L), \, R \, \text{is} \, L / 2. \]
Transition / Propagation Rules (LCOM1)

\[ \text{lp}(C, M, N) :- \text{cm}(C, M), \text{cm}(C, N), \text{not}(\text{cp}(C, M, N)). \]

\[ \text{add}_{\text{lp}}(C, M, N) :- \text{add}_{\text{cm}}(C, M), \quad \text{nwd}_{\text{cm}}(C, F), \text{not}(\text{nwd}_{\text{cp}}(C, M, N)). \]

\[ \text{add}_{\text{lp}}(C, M, N) :- \text{add}_{\text{cm}}(C, N), \quad \text{nwd}_{\text{cm}}(C, M), \quad \text{not}(\text{nwd}_{\text{cp}}(C, M, N)). \]

\[ \text{add}_{\text{lp}}(C, M, N) :- \text{del}_{\text{lp}}(C, M, N), \quad \text{nwd}_{\text{cm}}(C, M), \quad \text{nwd}_{\text{cm}}(C, N). \]

\[ \text{del}_{\text{lp}}(C, M, N) :- \text{del}_{\text{cm}}(C, M), \quad \text{cm}(C, N), \text{not}(\text{cp}(C, M, N)). \]

\[ \text{del}_{\text{lp}}(C, M, N) :- \text{del}_{\text{cm}}(C, N), \quad \text{cm}(C, M), \quad \text{not}(\text{cp}(C, M, N)). \]

\[ \text{nwi}_{\text{lp}}(C, M, N) :- \quad \text{nwd}_{\text{cm}}(M, F), \quad \text{nwd}_{\text{cm}}(C, F), \quad \text{not}(\text{nwd}_{\text{cp}}(C, M, N)). \]

\[ \text{cp}(C, M, N) :- \text{mf}(M, F), \quad \text{cf}(C, F), \quad \text{mf}(N, F). \]

\[ \text{add}_{\text{cp}}(C, M, N) :- \text{add}_{\text{mf}}(M, F), \quad \text{nwd}_{\text{cf}}(C, F), \quad \text{nwd}_{\text{mf}}(N, F), \quad \text{not}(\text{cp}(C, M, N)). \]

\[ \text{add}_{\text{cp}}(C, M, N) :- \text{add}_{\text{cf}}(C, F), \quad \text{nwd}_{\text{mf}}(M, F), \quad \text{nwd}_{\text{mf}}(N, F), \quad \text{not}(\text{cp}(C, M, N)). \]

\[ \text{add}_{\text{cp}}(C, M, N) :- \text{add}_{\text{mf}}(N, F), \quad \text{nwd}_{\text{mf}}(M, F), \quad \text{nwd}_{\text{cf}}(C, F), \quad \text{not}(\text{cp}(C, M, N)). \]

\[ \text{del}_{\text{cp}}(C, M, N) :- \text{del}_{\text{mf}}(M, F), \quad \text{cf}(C, F), \quad \text{mf}(N, F), \quad \text{not}(\text{nwi}_{\text{cp}}(C, M, N)). \]

\[ \text{del}_{\text{cp}}(C, M, N) :- \text{del}_{\text{cf}}(C, F), \quad \text{mf}(M, F), \quad \text{mf}(N, F), \quad \text{not}(\text{nwi}_{\text{cp}}(C, M, N)). \]

\[ \text{del}_{\text{cp}}(C, M, N) :- \text{del}_{\text{mf}}(N, F), \quad \text{mf}(M, F), \quad \text{cf}(C, F), \quad \text{not}(\text{nwi}_{\text{cp}}(C, M, N)). \]

\[ \text{nwi}_{\text{cp}}(C, M, N) :- \quad \text{nwd}_{\text{mf}}(M, F), \quad \text{nwd}_{\text{cf}}(C, F), \quad \text{nwd}_{\text{mf}}(N, F). \]

From \( \text{lp}/3, \text{cp}/3 \).

\[ \text{nwd}_{\text{lp}}(C, M, N) :- \text{add}_{\text{lp}}(C, M, N). \]

\[ \text{nwd}_{\text{lp}}(C, M, N) :- \quad \text{lp}(C, M, N), \text{not}(\text{del}_{\text{lp}}(C, M, N)). \]

\[ \text{nwd}_{\text{cp}}(C, M, N) :- \text{add}_{\text{cp}}(C, M, N). \]

\[ \text{nwd}_{\text{cp}}(C, M, N) :- \quad \text{cp}(C, M, N), \text{not}(\text{del}_{\text{cp}}(C, M, N)). \]

10/1/2012
Richard Tantius
20
Implementation

- PEF
- Metric rule definition
- Model Facts Converter
- Rule Analyser
- Rule Asserter
- UP Rules Generator
- Model facts
- UP rules
- Cache
- Metric rules
Conclusion

• Update Propagation in Prolog
  o Transforms well-formed programs
  o No limitation to
    • Metrics
    • Refactorings
Future Work

- Refactorings
- Metrics
- Bad Smells
- more

when to use UP

- UP Rules

improve

- User Interface

Evaluation
Thank you for your attention.