UNIVERSITÀ CA' FOSCARI DI VENEZIA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

Corso di Analisi e Verifica Programmi
Approfondimento

Mining Control Flow Graphs for Crosscutting Concerns

Studente: Marco Lionello

Anno Accademico 2006-2007

Introduzione (1/3)

- La programmazione orientata agli aspetti è un **paradigma di programmazione** basato sulla creazione di entità software(aspetti) che soprintendono alle iterazioni fra oggetti finalizzate ad eseguire un compito comune.

- Il vantaggio rispetto alla tradizionale Programmazione orientata agli oggetti consiste nel non dover implementare separatamente in ciascun oggetto il codice necessario ad eseguire questo compito comune.

- Un programma aspect-oriented è costituito essenzialmente da due insiemi di costrutti:
  - gli aspetti
  - gli oggetti.
  - Gli aspetti sono delle entità esterne agli oggetti che osservano il flusso del programma generato dalle interazioni tra oggetti, modificandolo quando opportuno.

- Metafora dello spettacolo teatrale

- Logging

Introduzione (2/3)

- **Scattered Code:** codice che esiste varie volte ma che non può essere incapsulato da moduli separati in quanto attraversa trasversalmente l'intero sistema
  - Codice difficile da mantenere,capire, estendere
- **Aspect Mining:** trovare e isolare i tagli trasversali di interesse
  - I tagli possono essere reimplementati come *aspetti* aumentando mantenibilità estensibilità e diminuendo la complessità
  - Quali tagli trasversali rifare?
    - Discorso molto lungo e articolato
    - Solo i comportamenti di tagli trasversali super imposti devo essere rifatti (es. comportamenti che non sono "core" del sistema)
    - Altri credono che tutto dovrebbe essere sotto forma di aspetti (non c'è ancora abbastanza esperienza in questo campo)
- **Aspect mining:** ci permette anche di trovare e classificare differenti funzioni che occorrono in diversi software
- **Aspect Mining:** è veramente utile al fine della buona comprensione di un sistema

Introduzione (3/3)

- Si è sviluppata in un primo momento una **tecnica dinamica** [2]:
  - Durante l'esecuzione del programma vengono generate delle tracce di programma che rispecchiano il comportamento a tempo di esecuzione
  - Le tracce vengono investigate per trovare i pattern di esecuzione ricorrenti
  - Diversi vincoli dicono quando un pattern è ricorrente, questo include il requisito che il pattern deve esistere in diversi contesti di chiamata

- **Problema principale:** la tecnica dinamica tiene conto del comportamento specifico e non del comportamento potenziale.
  - Sviluppo di una **tecnica di analisi statica** [12]→ analizza il grafo di flusso per i pattern di esecuzione ricorrenti

- Due punti essenziali
  - I risultati dell'analisi statica e dinamica sono differenti → ovvio!
  - I tagli trasversali di interesse sono molto spesso "buoni", perché derivano da delegation o da "stili di buona programmazione".

Aspect mining basato sulle relazioni di esecuzione

- [2] è basate su tracce di programma che rispecchiano l'andamento del programma in una sua esecuzione.
- I pattern ricorrenti sono potenziali tagli di interesse che descrivono funzionalità ricorrenti e possono essere rifatti in aspetti
- **Execution relation:** descrive in che relazione sono due metodi nelle tracce di programma
- In alcune circostanze il programmi possono essere analizzati ma non eseguiti quindi si usa l'analisi statica dal control flow graph(CFG).

Notazioni:

- CFG
  - Start node n^s
  - Exit node n^e

$$G = (N, E, n^s, n^e)$$

- Ogni procedura o metodo $p \in \mathcal{P}$ di un programma ha il suo CFG:

$$G_p = (N_p, E_p, n_p^s, n_p^e)$$

$$\forall p, q : p \neq q \Rightarrow N_p \cap N_q = \emptyset \wedge E_p \cap E_q = \emptyset$$

- CFG dell'intero programma $G^\star = (N^\star, E^\star)$

$$N^\star = \bigcup_p N_p, \; E^\star = \bigcup_p E_p$$

- In linguaggi (es. java) con più uscite (exception) si assume che ci sia un'unica uscita come join di tutte.

Classificazione di Relazioni di esecuzione
(1/2)

- I tagli di interesse sono riflessi in due tipi di Relazioni di esecuzione:
  - *Outside execution relation* (Esterna cioè la chiamata di un metodo precede o segue la chiamata di un altro metodo)
  - *Inside execution relation* (Interna se nel corpo di una c'è la chiamata all'altra)

- **Outside-before**: $u \rightharpoonup v$ con $u, v \in \mathcal{P}$. se c'è un percorso da $n_u \rightarrow^{\star} n_v$ in $G^{\star}$,dove n_u è una chiamata a *u*, n_v è una chiamata a *v* e non ci sono altre chiamate nel percorso ("*u eseguita prima di v*").
- $S^{\rightharpoonup}(G)$ è l'insieme delle relazioni di outside-before execution in G.

- La relazione può essere vista specchiata: $v \leftharpoonup u$ quindi abbiamo una **outside after** execution ("*v eseguita dopo u*").
- L'insieme delle relazioni outside after execution in G è: $S^{\leftharpoonup}(G)$

Classificazione di Relazioni di esecuzione
(2/2)

- $u \in_{\top} v$ con $u, v \in \mathcal{P}$ è chiamata **inside first** execution relation se c'è un percorso $n_v \rightarrow^{\star} n_u$ in G_u tale che $n_v = n_v^s$ è il nodo Start, n_u è una chiamata a *u* e non ci sono altre chiamate nel percorso.("*u* è la prima esecuzione in *v*")
- $u \in_{\perp} v$ è chiamata **inside last** execution relation se c'è un percorso $n_u \rightarrow^{\star} n_v$ in G_u tale che $n_v = n_v^s$ è il nodo EXIT, n_u è una chiamata a *u* e non ci sono altre chiamate nel percorso("*u* è l'ultima esecuzione in *v*").

- $S^{\in \top}(G)$ è l'insieme delle relazioni inside first
- $S^{\in \perp}(G)$ è l'insieme delle relazioni inside last

- Caso speciale quando c'è un percorso $n_p^s \rightarrow^{\star} n_p^e$ che non contiene chiamate le seguenti relazioni vengono automaticamente generate (catturano la possibilità che nessuna chiamata a metodo occorra durante l'esecuzione di un metodo *p*)

$$\epsilon \in_{\top} p \qquad \epsilon \in_{\perp} p$$

- Epsilon relation
  - Un'inside first execution $u \in_{\top} v$ genera $\epsilon \rightharpoonup u$ in quanto non ci sono altre chiamate prima che *u* sia chiamata
  - Un'inside last execution $u \in_{\perp} v$ genera $\epsilon \leftharpoondown u$ in quanto non ci sono altre chiamate dopo che *u* è chiamata

Esempio

```
void m1() {
  a();
  b();
  a();
}
void m2() {
  a();
  if (...) {
    b();
    a();
  }
}
void m3() {
  a();
  c();
}
```

- Outside relation:

  $a \rightharpoonup b$, $b \rightharpoonup a$, $a \rightharpoonup c$, $b \leftharpoonup a$, $c \leftharpoonup a$. $a \leftharpoonup b$

- Inside relation:

  $a \in_{\top} m1$,
  $a \in_{\top} m2$, $a \in_{\top} m3$, $a \in_{\bot} m1$, $a \in_{\bot} m2$ and $c \in_{\bot} m3$.

- Epsilon relation:

  $\epsilon \rightharpoonup a$, $\epsilon \leftharpoonup a$, and $\epsilon \leftharpoonup c$.

Vincoli nelle relazioni di esecuzione

- Le relazioni di esecuzione possono essere definite grazie a questo vincolo:
  - **Vincolo di uniformità**: $s = u \circ v \in S^\circ, \circ \in \{\rightharpoonup, \leftharpoonup, \in_\top, \in_\bot\}$ se $\forall w \circ v \in S^\circ : u = w$ con $u, v, w \in \mathcal{P} \cup \{\epsilon\}$
  - $\widehat{U}^\circ$ è l'insieme delle relazioni di esecuzione che soddisfano il vincolo $s \in S^\circ$

- Ulteriore vincolo: :
  - Una relazione di esecuzione $s = u \circ v \in U^\circ = \widehat{U}^\circ \setminus \{u \circ v \mid u = \epsilon \vee v = \epsilon\}$ è chiamata **crosscutting** se $\exists s' = u \circ w \in U^\circ : w \neq v$ con $u, v, w \in \mathcal{P}$ (se *s* ha più contesti di chiamata nel grafo di flusso)
  - Per le inside-execution relation $u \in_| v$, $u \in_\bot v$ il contesto di chiamata è il metodo che lo circonda *v*.
  - Per le outside-execution relation $u \rightharpoonup v$, $u \leftharpoonup v$ il contesto di chiamata è il metodo che è invocato rispettivamente prima o dopo il metodo *u*.
  - R° è l'insieme delle relazioni di esecuzione $s \in U^\circ$ che vengono chiamate ***aspect-candidate*** e rappresentano i potenziali ***crosscutting concerns*** (tagli di interesse).

Esempio

```
void m1() {
  a();
  b();
  a();
}
void m2() {
  a();
  if (...) {
    b();
    a();
  }
}
void m3() {
  a();
  c();
}
```

- Execution relation : $a \rightharpoonup b$,

  $a \rightharpoonup c$, $a \leftharpoonup b$, $a \in_{\top} m1$, $a \in_{\top} m2$, $a \in_{\top} m3$, $a \in_{\bot} m1$, $a \in_{\bot} m2$, $c \in_{\bot} m3$, $\epsilon \leftharpoonup c$.

- Aspect Candidate (o relazioni uniformi):

  $a \rightharpoonup b$, $a \rightharpoonup c$, $a \in_{\top} m1$, $a \in_{\top} m2$, $a \in_{\top} m3$, $a \in_{\bot} m1$, $a \in_{\bot} m2$.

Static Aspect Mining (1/3)

- L'algoritmo è una reaching definition data flow analysis

- $C \subseteq N^*$ l'insieme di nodi che sono chiamate a metodi e *c(n)* il metodo chiamato del nodo *n*
- Una chiamata a un metodo *p* al nodo *n* (*p=c(n)*) raggiunge un nodo *m* se un percorso *P=<n1,…nk>* in *G* esiste e tale che:

1. $k > 1$
2. $n_1 = n \wedge n_k = m$
3. $\forall 1 < i < k : n_i \notin C$

- Per rispettare le epsilon-relation i nodi START e EXIT sono assunti come nodi chiamata con

$$\forall p \in \mathcal{P} : n_p^s \in C \wedge n_p^e \in C \wedge c(n_p^s) = \epsilon \wedge c(n_p^e) = \epsilon.$$

- *RC(n)* è definita come insieme di chiamate che raggiungono il nodo *n*.
- La transfer function dell'insieme di chiamate *x* che raggiungono il nodo *n* è:

$$[\![n]\!](x) = \big(x - \mathsf{kill}(n)\big) \cup \mathsf{gen}(n)$$

  - Questa equazione dice che le chiamate che raggiungono l'entry del nodo *n* che non sono uccise da *n*, raggiungono l'uscita di *n* insieme alle chiamate generate da *n*

Static Aspect Mining (2/3)

- Una chiamata raggiunge un nodo *n*, se esiste un percorso dal nodo START a *n*

$$\mathsf{gen}(n) = \begin{cases} \{p \mid p = c(n)\} & \text{if } n \in C \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathsf{kill}(n) = \begin{cases} \mathcal{P} & \text{if } n \in C \\ \emptyset & \text{otherwise} \end{cases}$$

- Al nodo START solo *epsilon* è disponibile
- Se ci sono più percorsi viene fatta un'unione di tutti i percorsi istanza del problema MOP (Meet Over all Paths)

$$RC(n) = \bigcup_{p=\langle s,\ldots,n\rangle} [\![p]\!](\{\epsilon\})$$

- In presenza di loop (MOP non ha soluzioni) quindi viene calcolato solo il minimo punto fisso(MFP):

$$RC(n) = \begin{cases} \{\epsilon\} & n = n_p^s \\ [\![n]\!](\bigcup_{m \to n} RC(m)) \end{cases}$$

- Proprietà della transfer function garantiscono che MFP = MOP

Static Aspect Mining (3/3)

Dall'insieme delle reaching calls quattro relazioni di esecuzione possono essere definite

- $\forall n \in C \wedge m \in RC(n) : m \rightharpoonup n \wedge n \leftharpoonup m$ Qualsiasi chiamata che raggiunge un'altra chiamata genera un **outside-before execution** e **outside-after-execution** tra le due chiamate(*epsilon* relation automaticamente generate).

- $\forall n \in C \wedge n \in N_p \wedge \epsilon \in RC(n) : n \in_{\top} p$ Se una chiamata *epsilon* raggiunge un'altra chiamata allora esiste un percorso dal nodo START alla chiamata senza altre chiamate intermedie: si genera una **inside-first-execution**(la relazione *epsilon* viene generata se la chiamata ad *epsilon* raggiunge il nodo EXIT).

- $\forall n_p^e \in C \wedge m \in RC(n_p^e) : m \in_{\perp} p$ Qualsiasi chiamata che raggiunge il nodo EXIT genera una **inside-last-execution** (*epsilon* relation viene generata se la chiamata ad *epsilon* raggiunge il nodo EXIT)

Esperienze

- Implementazione del metodo statico al top di Soop [22]
- Come caso di studi si è preso JHotDraw
  - 18.000 linee di codice
  - 2.900 metodi
- Analisi delle relazioni inside-first e delle outside-before

size	candidates	size	candidates
2	53	8	1
3	19	9	1
4	4	11	1
5	6	12	1
6	3	13	1
7	2		

294 relations ($R^{\rightarrow}$) in 92 candidates

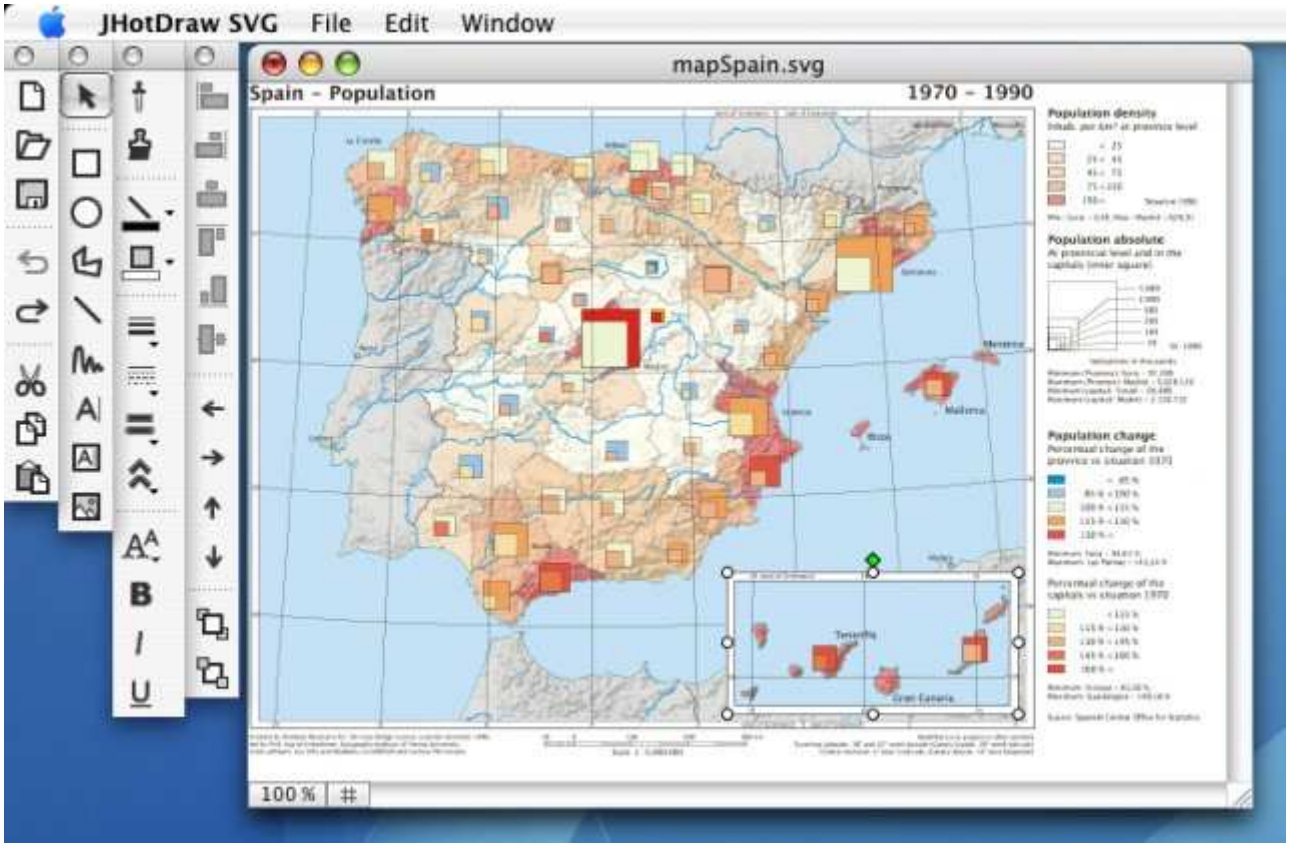
size	candidates	size	candidates
2	127	13	4
3	55	15	2
4	30	16	1
5	12	17	2
6	9	18	1
7	7	19	1
8	7	20	1
9	3	22	1
10	3	24	2
11	3	32	1
12	4	49	1

1236 relations ($R^{\in\top}$) in 277 candidates

- **Prima colonna**: dimensione dei crosscutting candidates misurata dal numero dei crosscutting method (numero di differenti metodi v per un unico metodo u)
- **Seconda colonna:** numero di candidati

- Ci sono molti candidati con un basso numero di crosscutting(127 candidati che hanno al massimo 2 metodi)
- Già si intuisce che:
  - molti metodi sono frutto di delegation e che non sono rifacibili in aspect
  - si necessita di un filtro

JHotDraw


JHotDraw SVG
File
Edit
Window
mapSpain.svg
Spain - Population
1970 - 1990
Population density
Population absolute
Population change
100 %

In dettaglio

- Inside first:
  - Il primo candidato ha 49 crosscutting il metodo invocato è "…*CollectionsFactory.Current*"
    - Metodo per accedere all'oggetto corrente
    - È un crosscutting, ma non reimplementabile in aspetti.
    - Utility method (classificazione di Marin)
  - Secondo metodo "…*DrawingView.view*" analogo al primo
    - Classificato da MArin come Acessor Method
  - Terzo e quarto Candidato
    - Metodi "*Decorator.Figure.getDecoratedFigure*" e "AbstractHandle.owner"
    - Classificazione di Marin Observer Crosscutting
    - Classificazione ideale Acessor Method
  - Quinto Candidato
    - Metodo "*UndoadableAdapter.undo*" con 22 crosscutting
    - Controlla quando l'oggetto rappresenta una azione anullabile
    - Marin classificato come crosscutting concerns
    - Refactoring molto complicato: classificazione idelae non reimplementabile in aspetti (con poca enfasi)
  - Sesto Candidato
    - "*AbstractFigure.WillChange*" con 20 crosscutting
    - Informa una figura che una operazione ha cambiato il suo contenuto visivo
    - Questo è il primo candidato che è reimplementabile in aspetti.
- Outside before:
  - La situazione è totalmente analoga
  - Solo il sesto candidato è reimplementabile in aspetti.

Filtraggio

- Solo i metodi che ritornano valore sono direttamente legati al chiamante e quindi sono delegazioni
  - Quindi i non-void-method (metodi con valore di ritorno) sono delegazione e quindi non reimplementabili

size	candidates	size	candidates
2	30	9	0
3	15	10	1
4	11	11	1
5	1	13	1
6	1	17	1
7	2	20	1
8	2		

261 relations ($R^{\in_T}$) in 67 candidates

size	candidates
2	11
3	3
4	2
5	1
7	1
12	1

62 relations ($R^{\rightarrow}$) in 19 candidates

- Alcuni dei metodi filtrati si possono rifare ad esempio:
  - *Abstract.figure.willChange*
    - (già discusso)
  - *Rettangle.add*
    - Nei 10 siti di chiamata un nuovo rettangolo viene creato dai due punti passati come argomento al metodo corrente
    - Possibile Crosscut da rifare in aspetti

- Altri metodi non possono comunque essere rifatti:
  - *ObjectInputStream.defaultReadObject*
    - Nonostante si possa rifare è parte di una libreria e quindi non conveniente
  - *AttributeFigure.write,AttributeFigure.read*
    - Le immagini possono essere composte e quindi la lettura e la scrittura sono delegazioni della composizione

Relzioni Inside-First-Execution estratte

7 **...figures.AttributeFigure.read** $\in_T$
...figures.EllipseFigure.read
...figures.RoundRectangleFigure.read
...figures.TextFigure.read
...figures.ImageFigure.read
...contrib.TextAreaFigure.read
...contrib.PolygonFigure.read
...figures.RectangleFigure.read
7 **...figures.AttributeFigure.write** $\in_T$
...figures.ImageFigure.write
...contrib.TextAreaFigure.write
...figures.TextFigure.write
...figures.EllipseFigure.write
...contrib.PolygonFigure.write
...figures.RectangleFigure.write
...figures.RoundRectangleFigure.write
9 **java.awt.Rectangle.translate** $\in_T$
...contrib.ComponentFigure.basicMoveBy
...figures.EllipseFigure.basicMoveBy
...contrib.TextAreaFigure.basicMoveBy
...figures.NullFigure.basicMoveBy
...figures.RectangleFigure.basicMoveBy
...samples.pert.PertFigure.basicMoveBy
...figures.ImageFigure.basicMoveBy
...figures.RoundRectangleFigure.basicMoveBy
9 **java.io.ObjectInputStream.defaultReadObject** $\in$
...figures.ImageFigure.readObject
...standard.StandardDrawingView.readObject
...contrib.TextAreaFigure.readObject
...figures.LineConnection.readObject
...standard.StandardDrawing.readObject
...figures.TextFigure.readObject
...standard.DecoratorFigure.readObject
...standard.CompositeFigure.readObject
10 **java.awt.Rectangle.add** $\in_T$
...figures.NullFigure.basicDisplayBox
...contrib.SimpleLayouter.calculateLayout
...samples.pert.PertFigure.basicDisplayBox
...contrib.zoom.AreaTracker.rubberBand
...standard.SelectAreaTracker.rubberBand
...figures.RectangleFigure.basicDisplayBox
...figures.EllipseFigure.basicDisplayBox
...figures.ImageFigure.basicDisplayBox
...figures.RoundRectangleFigure.basicDisplayBox
...contrib.ComponentFigure.basicDisplayBox

11 **...standard.AbstractTool.mouseDown** $\in_T$
...standard.ActionTool.mouseDown
...figures.ScribbleTool.mouseDown
...standard.DragTracker.mouseDown
...standard.HandleTracker.mouseDown
...samples.javadraw.URLTool.mouseDown
...contrib.zoom.ZoomTool.mouseDown
...standard.CreationTool.mouseDown
...standard.ConnectionTool.mouseDown
...contrib.dnc.DragNDropTool.mouseDown
...contrib.PolygonTool.mouseDown
...standard.SelectionTool.mouseDown
13 **...util.UndoableAdapter.setUndoable** $\in_T$
...standard.SendToBackCommand.UndoActivity.(init)
...figures.RadiusHandle.UndoActivity.(init)
...figures.BorderTool.UndoActivity.(init)
...figures.PolyLineHandle.UndoActivity.(init)
...standard.ResizeHandle.UndoActivity.(init)
...standard.ChangeConnectionHandle.UndoActivity.(init)
...figures.GroupCommand.UndoActivity.(init)
...figures.InsertImageCommand.UndoActivity.(init)
...standard.PasteCommand.UndoActivity.(init)
...figures.UngroupCommand.UndoActivity.(init)
...contrib.TriangleRotationHandle.UndoActivity.(init)
...contrib.PolygonScaleHandle.UndoActivity.(init)
...standard.SelectAllCommand.UndoActivity.(init)
17 **...standard.AbstractCommand.execute** $\in_T$
...util.UndoCommand.execute
...standard.BringToFrontCommand.execute
...standard.CutCommand.execute
...figures.InsertImageCommand.execute
...standard.ToggleGridCommand.execute
...standard.AlignCommand.execute
...contrib.zoom.ZoomCommand.execute
...standard.CopyCommand.execute
...standard.DuplicateCommand.execute
...standard.DeleteCommand.execute
...standard.SelectAllCommand.execute
...util.RedoCommand.execute
...standard.ChangeAttributeCommand.execute
...figures.UngroupCommand.execute
...figures.GroupCommand.execute
...standard.PasteCommand.execute
...standard.SendToBackCommand.execute

20 **...standard.AbstractFigure.willChange** $\in_T$
...contrib.TextAreaFigure.setFont
...contrib.GraphicalCompositeFigure.update
...contrib.TriangleFigure.rotate
...figures.TextFigure.moveBy
...contrib.PolygonFigure.scaleRotate
...contrib.PolygonFigure.setPointAt
...figures.LineConnection.startPoint
...contrib.PolygonFigure.removePointAt
...figures.ElbowConnection.updatePoints
...figures.TextFigure.setFont
...standard.AbstractFigure.moveBy
...figures.LineConnection.endPoint
...contrib.PolygonFigure.smoothPoints
...figures.PolyLineFigure.removePointAt
...figures.PolyLineFigure.setPointAt
...standard.AbstractFigure.displayBox
...contrib.html.HTMLTextAreaFigure.figureChanged
...contrib.TextAreaFigure.moveBy
...figures.RoundRectangleFigure.setArc
...contrib.PolygonFigure.insertPointAt

Related Work

- [14] Marin è l'approccio più simile a questo. In sostanza conta per ogni metodo il numero di luoghi di chiamata nel codice sorgente che chiamano il metodo.
- [9,10] Usano "unique method" simile al non void method. Un "unique method" è un metodo senza valori di ritorno che implementa qualcosa che non è implementato da altri metodi.
- [21] Usano la concept analysis. Estrazione di elementi e attributi da classi o da metodi al fine di raggruppare questi elementi in concept che possono essere visti come candidati.
- [6] Ceccato ha fatto una comparazione dei vari approcci.
- [16] è una combinazione di varie tecniche di aspect mining con il risultato di ottenere maggior precisione.
- [1] presenta un sistema semi-automatico per la reimplementazione dei tagli trasversali nei programmi object-oriented in aspect/oriented.

Conclusioni

- Si è visto che molti dei candidati sia filtrati che non filtrati non sono reimplementabili in aspect
- Ad ogni modo gli approcci(statico, dinamico) danno molte informazioni riguardo il comportamento dei crosscut
  - Potrebbero essere utilizzati per trovare le anomalie
- Un numero basso di candidati in un singolo test non può essere generalizzato
  - L'ipotesi è che il risultato non cambi
  - In linea con i risultati di Marin
  - Aspect mining fa fatica a identificare i candidati che possono realmente essere rifatti
- La reimplementazione di JHotDraw in un sistema che fa buon uso di aspect mostra che il rifacimento è un lavoro molto complesso
- In contrasto con molti autori:
  - La maggior parte dei tagli scoperti con qualsiasi tecnica di aspect mining rileva Delegation che non è reimplementabile
  - Delegation è considerata come semplice forma di taglio ma solo quella superimposed che è debolmente legata al resto del codice può essere rifatta.

Bibliografia

[1] D. Binkley, M. Ceccato, M. Harman, F. Ricca, and P. Tonella. Automated refactoring of object oriented code into aspects. In *21st IEEE International Conference on Software Maintenance (ICSM'05)*, pages 27–36, 2005.

[2] S. Breu and J. Krinke. Aspect mining using event traces. In *Proc. International Conference on Automated Software Engineering*, pages 310–315, 2004.

[3] M. Bruntink. Aspect mining using clone class metrics. In *Workshop on Aspect Reverse Engineering*, 2004.

[4] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwe. An evaluation of clone detection techniques for identifying cross-cutting concerns. In *Proc. International Conference on Software Maintenance*, 2004.

[5] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwe. On the use of clone detection for identifying crosscutting concern code. *IEEE Trans. Softw. Eng.*, 31(10):804–818, Oct. 2005.

[6] M. Ceccato, M. Marin, K. Mens, L. Moonen, P. Tonella, and T. Tourwe. A qualitative comparison of three aspect mining techniques. In *13th International Workshop on Program Comprehension (IWPC)*, 2005.

[7] D. Engler, D. Y. Chen, S. Hallem, A. Chou, and B. Chelf. Bugs as deviant behavior: a general approach to inferring errors in systems code. *SIGOPS Oper. Syst. Rev.*, 35(5):57–72, 2001.

[8] W. G. Griswold, Y. Kato, and J. J. Yuan. Aspect Browser: Tool Support for Managing Dispersed Aspects. Technical Report CS99-0640, Department of Computer Science and Engineering, UC, San Diego, 1999.

[9] K. Gybels and A. Kellens. An experiment in using inductive logic programming to uncover pointcuts. In *First European Interactive Workshop on Aspects in Software*, 2004.

[10] K. Gybels and A. Kellens. Experiences with identifying aspects in smalltalk using 'unique methods'. In *Workshop on Linking Aspect Technology and Evolution (LATE)*, 2005.

[11] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *European Conf. on Object-Oriented Programming (ECOOP)*, 1997.

[12] J. Krinke and S. Breu. Control-flow-graph-based aspect mining. In *Workshop on Aspect Reverse Engineering*, 2004.

[13] M. Marin. Refactoring jhotdraw's undo concern to aspectj. In *Workshop on Aspect Reverse Engineering (WARE)*, 2004.

[14] M. Marin, A. van Deursen, and L. Moonen. Identifying aspects using fan-in analysis. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004)*, 2004.

[15] D. Shepherd, E. Gibson, and L. Pollock. Design and evaluation of an automated aspect mining tool. In *International Conference on Software Engineering and Practice*, 2004.

[16] D. Shepherd, J. Palm, L. Pollock, and M. Chu-Carroll. Timna: A framework for combining aspect mining analyses. In *International Conference on Automated Software Engineering*, 2005.

[17] D. Shepherd and L. Pollock. Interfaces, aspects, and views. In *Workshop on Linking Aspect Technology and Evolution (LATE)*, 2005.

[18] D. Shepherd, T. Tourwe, and L. Pollock. Using language clues to discover crosscutting concerns. In *First International Workshop on the Modeling and Analysis of Concerns in Software (MACS)*, 2005.

[19] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton, Jr. N Degrees of Separation: Multi-Dimensional Separation of Concerns. In *21st Intl. Conf. on Software Engineering (ICSE)*, pages 107–119, 1999.

[20] P. Tonella and M. Ceccato. Aspect mining through the formal concept analysis of execution traces. In *11th IEEE Working Conference on Reverse Engineering (WCRE 2004)*, 2004.

[21] T. Tourwe and K. Mens. Mining aspectual views using formal concept analysis. In *Proc. IEEE International Workshop on Source Code Analysis and Manipulation*, 2004.

[22] R. Vallee-Rai, P. Co, E. Gagnon, L. Hendren, P. Lam, and V. Sundaresan. Soot – a java bytecode optimization framework. In *Proc. CASCON*, 1999.

[23] C. Zhang and H.-A. Jacobsen. Quantifying Aspects in Middleware Platforms. In *2nd Intl. Conf. on Aspect-Oriented Software Development (AOSD)*, pages 130–139, 2003.