쏘프트웨어의 재분해를 위한 코드결함의 식별방법

신춘옥, 김충혁

쏘프트웨어재분해를 효률적으로 진행하자면 코드에 내재되여있는 결함(Bad Smells)의 종류뿐아니라 그것의 원인으로 되는 쏘프트웨어의 실체들과 그 포괄범위들을 정확히 반영하여 코드결함식별방법을 정량적으로, 론리적으로 정식화하여야 한다.

선행연구[1, 3]에서는 코드결함을 측정량에 기초하여 식별하고 선행연구[2]에서는 오유예측을 위하여 코드결함식별원리를 론리적으로 정식화하였다.

그러나 쏘프트웨어재분해와 관련한 요구 그리고 코드결함으로 되는 코드부분을 정확하게 반영하지 않았다.

론문에서는 매 코드결함들을 쏘프트웨어의 실체들과 여러 측정량들과의 련관속에서 론리적으로 정식화하여 코드에 내재되여있는 결함들의 종류와 그 결함이 있는 코드부분을 정확히 식별하는 방법을 제안하였다.

1. 측정지표들과 1계술어론리식에 의한 코드결함들의 정의

코드결함을 정량적으로 론의하기 위하여 선행연구[2]의 측정지표들을 쏘프트웨어의 구성요소들과의 관계속에서 수정보충한다.

코드결함식별에서 리용되는 측정지표들과 그 정의를 표 1에 보여주었다.

표 1. 코드결함식별에서 리용되는 측정지표들과 그 정의

측정지표	정의
ATFD(c)	측정중인 클라스 c와 무관계한 클라스들에서 정의되고 클라스 c로부터 직접 접근되거나 혹은 접근자메쏘드호출에 의하여 접근되는 속성들의 모임이다. 즉 $\mathrm{ATFD}(c)=\{a \mid a \in c' \wedge (\mathrm{access}(c, a) \vee \mathrm{call}(\mathrm{accessor}, a))\}$ 이다.
ATLD(m)	현재의 클라스들에서 선언되였으며 측정중인 메쏘드 m에 의하여 직접 접근되거나 혹은 접근자메쏘드호출에 의하여 접근되는 속성들의 모임이다. 즉 $\mathrm{ATLD}(m)=\{a \mid a \in c \wedge m \in c \wedge (\mathrm{access}(m, a) \vee \mathrm{call}(\mathrm{accessor}, a))\}$ 이다.
CC(m)	측정하려는 메쏘드 m을 호출하는 메쏘드들이 정의된 클라스들의 모임이다. 즉 $\mathrm{CC}(m)=\{c' \mid m \in c \wedge m' \in c' \wedge c \neq c \wedge \mathrm{call}(m', m)\}$ 이다.
CDISP(m)	측정하려는 메쏘드 m으로부터 호출되는 메쏘드들이 정의된 클라스들의 수를 |CINT|로 나눈 값이다. 즉 $\lvert\{c' \mid m' \in c' \wedge \mathrm{call}(m, m')\}\rvert / \lvert\mathrm{CINT}\rvert$ 이다.
CINT(m)	측정하려는 메쏘드 m으로부터 호출되는 구별가능한 메쏘드들의 모임이다. 즉 $\mathrm{CINT}(m)=\{m' \mid \mathrm{call}(m, m')\}$ 이다.
CM(m)	측정중인 메쏘드 m을 호출하는 구별가능한 메쏘드들의 모임이다. 즉 $\mathrm{CM}(m)=\{m' \mid \mathrm{call}(m, m')\}$ 이다.
CYCLO(m)	메쏘드 m에서 선형인 독립경로의 최대수(대응하는 코드의 실행흐름에서 분기가 없으면 경로는 선형임.)이다.

표계속

측정지표	정 의
DBMC(m, c)	$\mathrm{DBMC}(m, c) = 1 - \frac{\lvert S_m \cap S_c \rvert}{\lvert S_m \cup S_c \rvert}$ 에 의하여 계산하는 메쏘드 m과 클라스 c사이의 거리이다. 여기서 S_m 은 메쏘드 m이 접근하는 실체들의 모임이고 S_c 는 c에 속하는 메쏘드들과 속성들을 포함하는 모임(m이 클라스 c에 속하면 m을 제외함.)이다.
FANOUT(m)	메쏘드 m에서 호출되는 클라스의 수이다. 즉 $\lvert\{c \mid \mathrm{call}(m, c)\}\rvert$ 이다.
LOC()	메쏘드 혹은 클라스의 코드행의 수(공백행과 주석을 포함.)이다.
LOCNAM(c)	접근자와 변이자메쏘드들과 그것에 해당한 주석들을 제외한 클라스 c에서 코드행의 수(공백행과 주석을 포함.)이다.
MaMCL(m)	메쏘드 m에서 사슬로 된 호출의 최대길이이다.
MAXNESTING(m)	메쏘드 m의 조종구조에서 최대의 중첩준위이다.
NOMP(c_1, c_2)	클라스 c_1과 클라스 c_2 사이 통보문넘기기의 수이다.
MeMCL(m)	메쏘드 m에서 사슬로 된 호출의 평균길이이다.
NMCS(m)	메쏘드 m에서 서로 다른 사슬화된 호출의 수이다.
NOAM(c)	클라스 c에 있는 접근자메쏘드들의 수이다. 즉 $\lvert\{\mathrm{accessor} \mid \mathrm{accessor} \in c\}\rvert$ 이다.
NOLV(m)	메쏘드 m에서 선언된 국부변수들의 수이다. 즉 $\lvert\{lv \mid lv \in m\}\rvert + \mathrm{NOR}(m)$ 이다.
NOMNAMM(c)	측정중인 클라스 c에서 국부적으로 정의되는 메쏘드들의 수이다. 즉 $\lvert\{m \mid m \in c \wedge m \neq \mathrm{accessor} \wedge m \neq \mathrm{mutator} \wedge m.d = \mathrm{public} \vee m.d = \mathrm{private}\}\rvert$ 이다.
NOPA(c)	클라스 c에 있는 공개속성의 수이다. 즉 $\lvert\{a \mid a \in c \wedge a.d = \mathrm{public}\}\rvert$ 이다.
NOPR(m)	메쏘드 m의 형식적파라메터목록의 길이이다.
TCC(c)	측정중인 클라스 c에서 실체변수들을 통하여 다른 메쏘드와 직접 련결된 메쏘드들의 수와 메쏘드들사이의 가능한 총 련결수사이의 비률이다. 즉 $\lvert\{(m, m')\} \mid \exists a \in c, m \in c, m' \in c : \mathrm{directconnected}(a, m, m')\rvert / \lvert\{(m, m') \mid \exists m \in c, m \in c : \mathrm{connected}(m, m')\}\rvert$ 이다. TCC는 [0, 1]범위의 값을 가진다.
WMCNAMM(c)	클라스 c에서 정의된 메쏘드들(접근자 혹은 변이자는 제외함.)의 복잡도 CYCLO의 합이다. 즉 $\sum_{m \in c \wedge m \neq \mathrm{accessor} \wedge m \neq \mathrm{mutator}} \mathrm{CYCLO}(m)$ 이다.
WOC(c)	클라스 c에서 비추상적이면서 접근자도 변이자도 아닌 어떤 기능을 수행하는 공개메쏘드들의 수를 공개메쏘드들의 총수로 나눈 값이다. 즉 $\lvert\{m \mid m \in c \wedge m.d = \mathrm{public} \wedge m \neq \mathrm{abstract} \wedge m \neq \mathrm{accessor} \wedge m \neq \mathrm{mutator}\}\rvert / \lvert\{m \mid m \in c \wedge m.d = \mathrm{public}\}\rvert$ 이다.

표 1에서 파라메터 c 는 클라스를, m 은 메쏘드를, a 는 속성을, $x.d$ 는 실체 x 의 호출지정자를 지적하며 술어 $\mathrm{access}(x, x')$ 는 x 가 x' 에 접근하는 관계를, $\mathrm{call}(x, x')$ 는 x 가 x' 를 호출하는 관계를 표현한다.

두 메쏘드 m 과 m' 가 같은 실체변수 a 에 메쏘드호출을 통하여 직접 혹은 간접적으로 접근하는 관계(이 경우 두 메쏘드는 직접 련결되였다고 함.)에 있을 때 술어 $\mathrm{directconnected}(a, m, m')$ 는 참이며 두 메쏘드 m 과 m' 사이의 가능한 련결을 술어

connected(m, m')로 표시한다. $x \in X$ 는 쏘프트웨어의 구성요소 x가 X를 정의하는 코드본체에서 선언되거나 정의될 때 참으로 되는 술어이다. 코드결함들의 종류를 코드결함이 있는 실체를 파라메터로 하는 단항술어(실례로 God Class(c)) 또는 코드결함을 일으키는 요소들사이의 관계술어(실례로 Feature Envy(m, c))로 표기한다.

각이한 코드결함을 1계론리식에 의하여 정식화한다. 그 론리식들에서 따름연산의 전제부를 이루는 부분론리식들은 해당 코드결함의 식별조건이고 결론부는 코드결함의 종류를 지적하는 술어이다. 결론부를 이루는 술어의 항들은 그 코드결함의 원인으로 되며 다음단계에서 재분해를 하여야 할 쏘프트웨어요소(메쏘드, 클라스, 클라스모임)들이다.

각이한 코드결함의 정의를 표 2에 보여주었다.

표 2. 각이한 코드결함의 정의

코드결함종류	코드결함식별을 위한 1계론리식
God Class	$((\text{LOCKNAMM}(c) \geq 176) \wedge (\text{WMCNAMM}(c) \geq 22) \wedge (\text{NOMNAMM}(c) \geq 18) \wedge (\text{TCC}(c) \leq 0.33) \wedge \lvert(\text{ATFD}(c)\rvert \geq 6)) \Rightarrow \text{God Class}(c)$
Feature Envy	$\exists m \in c : (c \neq c') \wedge ((\text{DBMC}(m, c) - \text{DBMC}(m, c')) > \beta) \Rightarrow \text{Feature Envy}(c, c')$
Inappropriate Iintimacy	$(\text{NOMP}(c_1, c_2) \geq 3) \Rightarrow \text{Inappropriate Intimacy}(c_1, c_2)$
Data Class	$(\text{WMCNAMM}(c) \leq 14) \wedge (\text{WOC}(c) \leq 0.33) \wedge (\text{NOAM}(c) \geq 4) \wedge (\text{NOPA}(c) \geq 3) \Rightarrow \text{Data class}(c)$
Brain Method	$(\text{LOC}(m) \geq 33) \wedge (\text{CYCLO}(m) \geq 7) \wedge (\text{MAXNESTING}(m) \geq 6) \wedge (\text{NOLV}(m) \geq 6) \wedge (\lvert\text{ATLD}(m)\rvert \geq 5) \Rightarrow \text{Brain Method}(m)$
Shotgun Surgery	$\exists m \in c : (\lvert\text{CC}(m)\rvert \geq 5) \wedge (\lvert\text{CM}(m)\rvert \geq 6) \wedge (\text{FANOUT}(m) \geq 3) \Rightarrow \text{Shotgun Surgery}(c, \text{CC}(m))$
Dispersed Coupling	$\exists m \in c : (\lvert\text{CINT}(m)\rvert \geq 8) \wedge (\text{CDISP}(m) \geq 0.66) \Rightarrow \text{Dispersed Coupling}(c)$
Message Chains	$((\text{MaMCL}(m) \geq 3) \wedge (\text{NMCS}(m) \geq 3) \wedge (\text{MeMCL}(m) \geq 2)) \Rightarrow \text{Message Chains}(m)$
Long Method	$\text{Loc}(m) \geq 50 \vee \text{CYCLO}(m) \geq 11 \Rightarrow \text{Long Method}(m)$
Large Class	$\text{Loc}(c) \geq 100 \vee \exists m \in c : \text{CYCLO}(m) \geq 21 \Rightarrow \text{Large Class}(c)$
Long Parameter List(m)	$\text{NOPR}(m) \geq 5 \Rightarrow \text{Long Parameter List}(m)$

2. 코드결함들의 식별방법

코드결함식별에서는 매 코드결함을 정식화한 론리식의 전제부를 이루는 부분론리식을 식별조건으로 한다. 그 식별조건이 성립하는가를 판별하여 코드결함의 종류와 그 결함을 가지는 또는 원인으로 되는 코드요소들을 식별한다.

매 코드결함의 식별방법은 다음과 같다.

① God Class의 식별

God Class의 식별조건을 이루는 다음의 매 관계식이 성립하는가를 판별한다. 여기서 getter와 setter메쏘드들을 가지는 클라스, getter와 setter메쏘드들을 가지지 않는 클라스가 God class로 검출될수 있는 확률이 같다는 사실을 고려한다.

· $\text{LOCKNAMM}(c) \geq 176$ 은 측정중인 클라스 c에 코드가 너무 많을 때 성립한다. getter와 setter메쏘드들은 IDE에서 생성되므로 LOC대신에 LOCKNAMM을 리용한다.

· $\text{WMCNAMM}(c) \geq 22$ 는 클라스 c가 수행하는 작업량이 많고 복잡할 때 성립한다. 매 메쏘드는 적어도 최소한의 순환복잡도를 가지므로 getter와 setter메쏘드들도 그 클라스

의 순환복잡도를 높인다. 따라서 그것들을 제외한 복잡도측정량을 리용한다.

· $NOMNAMM(c) \geq 18$ 은 클라스 c 에 많은 기능들이 들어있을 때 성립한다. 그 클라스의 기능성을 효과적으로 실현하는 메쏘드들만 론의하여야 하므로 getter와 setter메쏘드들은 제외시킨다.

· $TCC(c) \leq 0.33$ 은 클라스 c 의 함수들이 서로 다른 과제들을 수행하는것으로 하여 c 의 응집도가 낮아지지 않는가를 판정하기 위한것이다.

· $|ATFD(c)| \geq 6$ 은 측정중인 클라스 c 가 다른 클라스의 자료들을 정도이상으로 많이 리용할 때 참으로 된다.

매 관계식들이 성립하여 식별조건이 참으로 되면 측정중인 클라스 c 는 God Class로 판정되며 클라스추출 또는 메쏘드추출연산을 적용하여야 할 코드부분으로 된다.

② Feature Envy의 식별

Feature Envy의 식별조건인 $\exists m \in c : (c \neq c') \wedge (DBMC(m, c) - DBMC(m, c')) > \beta$ 는 측정중인 메쏘드 m 이 클라스 c 에서 정의되고 메쏘드 m 과 클라스 c 의 거리가 클라스 c' 와의 거리보다 더 클 때(즉 m 이 c 보다 c' 와 밀접한 관계가 있을 때) 성립한다. 식별조건이 성립하면 클라스 c 와 클라스 c' 는 Feature Envy관계에 있다. 즉 메쏘드 m 과 그것이 정의된 클라스 c, 관계가 보다 밀접한 c' 는 Feature Envy코드결함이 내재하고있는 코드부분이다.

③ Inappropriate Intimacy의 식별

Inappropriate Intimacy의 식별조건인 $NOMP(c_1, c_2) \geq 3$ 은 측정중인 두 클라스 c_1 과 c_2 가 있어 $NOMP(c_1, c_2)$ 가 턱값 3보다 크면 참으로 된다. 이때 클라스들의 쌍 (c_1, c_2) 는 Inappropriate Intimacy관계에 있으며 재분해의 대상으로 된다.

④ Data Class의 식별

Data Class의 식별조건에서 $WMCNAMM(c) \leq 14$ 는 측정중인 클라스 c의 메쏘드들이 복잡하지 않다는것을, $WOC(c) \leq 0.33$ 은 클라스 c 가 적은 기능들을 제공한다는것을 의미한다.

$NOAM(c) \geq 4$ 는 클라스 c 가 많은 접근자메쏘드를 가질 때 참으로 된다.

$NOPA(c) \geq 3$ 은 클라스 c 가 많은 공개속성을 가질 때 참으로 된다.

우의 관계식들이 다 참으로 되면 클라스 c 는 Data Class이다.

⑤ Brain Method의 식별

Brain Method의 식별조건에서 관계식 $LOC(m) \geq 33$ 은 측정중인 메쏘드 m 의 코드행이 매우 많다는것을, $CYCLO(m) \geq 7$ 은 메쏘드 m 의 순환복잡도가 높다는것을 의미한다.

$MAXNESTING(m) \geq 6$ 은 메쏘드 m 에 준위가 깊은 어떤 겹순환이 있어 리해하기가 어렵다는것을 의미한다.

측정중인 메쏘드 m 은 우의 세 관계식이 참일 때 Brain Method이다.

다음의 두 관계식이 참일 때에도 m 을 Brain Method로 식별한다.

$NOLV(m) \geq 6$ 은 메쏘드 m 에 국부변수의 수가 많아 리해하기가 어렵다는것을 의미한다.

$|ATLD(m)| \geq 5$ 는 메쏘드 m 이 클라스의 자료들을 많이 리용할 때 참으로 된다.

⑥ Shotgun Surgery의 식별

Shotgun Surgery는 쏘프트웨어의 여러 요소들사이에 존재하는 그중 복잡한 형태의 코

드결함이다.

Shotgun Surgery의 식별조건에서 $\exists m \in c : |\mathrm{CC}(m)| \geq 5$ 는 측정중인 클라스 c 에서 정의된 메쏘드 m 을 호출하는 메쏘드들이 서로 다른 적지 않은 클라스들의 모임 CC(m)에서 정의되였다는것을 의미한다.

$|\mathrm{CM}(m)| \geq 6$ 은 메쏘드 m 을 호출하는 메쏘드들의 수 즉 메쏘드 m 을 변화시킬 때 함께 변화시켜야 할 메쏘드의 수가 아주 많다는것을 의미한다.

$\mathrm{FANOUT}(m) \geq 3$ 은 그 메쏘드 m 이 변화되여야 할 대상이라는것을 의미한다.

관계식들이 참으로 될 때 쏘프트웨어요소 m 과 c, CC(m)의 매 클라스들은 Shotgun Surgery를 이룬다.

⑦ Dispersed Coupling의 식별

Dispersed Coupling의 식별조건에서 $\mathrm{CINT}(m) \geq 8$ 은 클라스 c 에 속하는 어떤 메쏘드 m 이 정도이상의 다른 메쏘드들을 호출할 때 참으로 된다.

$\mathrm{CDISP}(m) \geq 0.66$은 클라스 c에 속하는 어떤 메쏘드 m 에 대한 호출이 너무 많은 클라스들에 분산되여있을 때 참으로 된다.

관계식들이 모두 참일 때 클라스 c는 Dispersed Coupling코드결함을 가진다.

⑧ Message Chain의 식별

Message Chain의 식별조건에서 $\mathrm{MaMCL}(m) \geq 3$은 메쏘드 m 이 적어도 평균보다 큰 길이를 가지는 사슬로 된 호출을 가질 때 참으로 된다.

$\mathrm{NMCS}(m) \geq 3$은 통보문사슬명령문의 수를 의미한다. 보다 많은 통보문사슬명령문과 호출의 각이한 사슬들이 있을수 있다.

$\mathrm{MeMCL}(m) \geq 2$는 메쏘드 m 이 평균통보문사슬길이보다 큰 통보문사슬을 가질 때 참으로 된다.

관계식들이 참으로 되면 메쏘드 m은 Message Chain의 코드결함을 가지는것으로 판정한다.

⑨ Long Method의 식별

측정중인 메쏘드 m은 다음의 두 관계식들중의 어느 하나가 참으로 되면 Long Method이다.

$\mathrm{Loc}(m) \geq 50$

$\mathrm{CYCLO}(m) \geq 11$

⑩ Large Class의 식별

측정중인 클라스 c는 다음의 두 관계식들중의 어느 하나가 참으로 되면 Large Class이다.

$\mathrm{Loc}(c) \geq 100$

$\exists m \in c : \mathrm{CYCLO}(m) \geq 21$

⑪ Long Parameter List의 식별

측정중인 메쏘드 m은 $\mathrm{NOPR}(m) \geq 5$일 때 Long Parameter List의 코드결함을 가진다.

맺 는 말

매 코드결함들을 쏘프트웨어의 실체들과 여러 측정량들과의 관계속에서 론리적으로 정식화하여 코드에 내재하는 코드결함의 종류와 그 결함이 있는 코드부분을 정확히 식별할수 있게 하였다.

참 고 문 헌

[1] Santiago Vidal et al.; ACM Transactions of Software Engineering and Methodology, 27, 1, 2, 2018.
[2] Fabio Palomba et al.; IEEE Transactions on Software Engineering, 45, 2, 194, 2019.
[3] Hui Liu et al.; IEEE Transations on Software Engineering, 38, 1, 220, 2012.

주체110(2021)년 5월 5일 원고접수

An Identification Method of Code Smells for Software Refactoring

Sin Chun Ok, Kim Chung Hyok

In this paper, we have proposed an identification method of code smells that can correctly identify the kinds of bad smells and its entities for software refactoring.

Keywords: code smells, refactoring, predicate logic