무리짓기에 의한 쏘프트웨어재구축

전현철, 신춘옥

쏘프트웨어재구축은 코드의 동작을 변경시키지 않고 내부구조를 조정하여 쏘프트웨어의 리해성, 유지보수성, 재리용성, 유연성을 개선할수 있도록 한다.

《높은 응집도와 낮은 결합도》의 쏘프트웨어설계원리에 따라 메쏘드 혹은 마당들을 보다 많은 의존성을 가지는 클라스에 이동시키는 메쏘드, 마당이동재분해[2]가 연구되였다. 또한 강하게 련관되는 메쏘드들과 마당들을 원래의 클라스로부터 새 클라스에로 추출하는 클라스추출에 의한 재분해[1]가 연구되였다.

론문에서는 한 클라스로부터 다른 클라스에로 메쏘드 또는 마당들을 이동시키거나 클라스를 추출하는 기회를 식별하여 클라스들을 통합하거나 분할하는 클라스재분해방법과 메쏘드준위의 망들에 기초하여 체계의 실체들을 재그룹화하는 무게붙은 무리짓기알고리듬을 제안하였다.

1. 클라스준위의 재분해를 위한 전처리

정의 클라스준위의 다중의존관계방향망(CMDN: Class－level Multi－relation Dlrected Network) G_{cl} 은 $G_{cl}=(V_{cl}, E_{cl})$ 과 같이 표시된다. 여기서 정점 $clas \in V_{cl}$ 에는 클라스가 놓이며 E_{cl} 은 클라스들사이에 존재하는 계승, 관련, 집약과 같은 n개 형의 의존관계들의 모임이다. E_i 를 n개 형중에서 i번째 의존성관계를 가지는 릉들의 모임이라고 할 때

$$E_1 \cup E_2 \cup \cdots \cup E_n = E_{cl}$$

이고

$$E_1 \cap E_2 \cap \cdots \cap E_n = \phi$$

이다. 만일 $\exists E_i \in E_{cl}$ 이면 G_{cl} 은

$$G_1=(V, E_1), G_2=(V, E_2), \cdots, G_n=(V, E_n)$$

이다.

다중의존관계방향망 G_{cl} 은 다음과 같은 속성들을 가진다.

속성 1 E_t 가 정점들사이의 어느 한가지 형의 의존관계모임이라고 하자. 클라스 $clas_1, clas_2 \in V$ 이고 클라스 $clas_1$ 이 $clas_2$ 에 의존한다면(이것을 $clas_1 \xrightarrow{e_{12}} clas$ 라고 표기) $clas_1$ 로부터 $clas_2$ 에로 가는 릉 $e_{12} \in E_t$ 가 존재한다.

속성 2 계승관계와 비계승관계만을 론의할 때 다중의존관계방향망 G는 $G=G_1 \cup G_2$ 로 표시한다. 방향그라프 G_1 과 G_2 는 $G_1=(V, E_1), G_2=(V, E_2)$ 이며 E_1 은 계승을 제외한 의존관계를 표현하는 릉의 모임이고 E_2 는 계승관계를 표현하는 릉의 모임이다.

속성 3 클라스 $clas_i$ 에서 정의된 실체모임은 메쏘드들의 모임 $M_i=\{m_1, m_2, \cdots, m_n\}$ 과 속성들의 모임 $A_i=\{a_1, a_2, \cdots, a_p\}$ 를 포함한다. 여기서 $m_k \in M_i$ 는 클라스 $clas_i$ 에서

정의된 메쏘드를 표시하며 $a_t \in A_i$ 는 클라스 $clas_i$ 에서 정의된 속성을 나타낸다. 이로부터 $clas_i = M_i \cup A_i$ 이다.

1) 비계승계층구조에 대한 재분해전처리연산

응집도와 결합도문제를 해결하기 위하여 비계승계층구조와 계승계층구조의 잎정점들에 있는 매 클라스들의 실체들을 하나의 실체모임으로 통합한 다음 그것들을 재그룹화한다. 계승계층구조에서 잎이 아닌 정점들을 무리짓기과정에 분해하면 코드의 동작이 달라지므로 따로 론의하기로 한다.

재분해를 위한 전처리를 다음과 같이 진행한다.

① 원천코드의 추상문법나무를 분석하여 망 $G = G_1 \cup G_2$ 를 구축한다.

② G_2 의 정점 $clas_i \in V$ 가 계승계층구조에서 잎이 아닌 정점이면 $clas_i$ 를 제거해야 할 정점모임 V_D 에 추가한다.

③ V_D 에 속하는 모든 정점들과 그것과 련결된 릉들을 모두 삭제하여 나머지망 $\overline{G_1} = (\overline{V}, \overline{E_1})$ 를 구성한다. 여기서 E_D 를 V_D 의 정점들과 련결된 릉들의 모임이라고 하면 $\overline{V} = V \setminus V_D, \overline{E_1} = E_1 \setminus E_D$ 와 같이 표시된다.

④ 나머지망 $\overline{G_1}$ 에서 련결성분들의 모임 $CC = \{cc_1, cc_2, \cdots, cc_{|CC|}\}$ 를 찾는다.

⑤ 매 련결성분들에 있는 모든 클라스들을 실체모임 $Entities_i$ 로 통합하면 모든 $k \in [1, 2, \cdots, |CC|]$ 와 $m \in [1, 2, \cdots, |cc_k|]$, $clas_m \in cc_k$ 에 대하여 $Entities_i = \bigcup clas_m$ 이다. 여기서 $|cc_k|$ 는 련결성분 cc_k 에 들어있는 클라스의 수를 표시한다.

재분해전처리에 의하여 계승계층구조에서의 모든 상위클라스들을 체계로부터 추출한 다음 체계를 $Sys = \{Entities_1, Entities_2, \cdots, Entities_{|CC|}\}$ 로 표시할수 있다. Sys 에서의 모든 실체모임들이 첫 단계에서 재구조화되여야 할 대상들이다.

2) 계승계층구조에 대한 재분해전처리연산

비계승계층구조의 실체들을 재그룹화한 후 클라스들의 수와 그것들사이의 관계는 달라진다. 따라서 계승계층구조에 대한 재분해연산들을 수행하기 전에 다중의존관계방향망 $G'_{cl} = G'_1 \cup G'_2$ 를 첫 재분해단계에서 얻어진 체계구조에 기초하여 갱신하여야 한다.

계승나무들의 모임을 $TR = \{Tr_1, Tr_2, \cdots, Tr_{|TR|}\}$ 와 같이 표시하자. 계승관계망 G'_2 에서 출력차수가 령이고 입력차수가 령보다 큰 클라스를 계승나무의 뿌리정점으로 정의하고 $clas_k^{root}, k \in [1, 2, \cdots, |TR|]$ 로 표시한다. 뿌리정점으로부터 도달할수 있는 G'_2 에서의 정점들은 계승나무 Tr_k 를 구성한다. 하나의 뿌리정점 $clas_k^{root}$ 는 계승나무 Tr_k 에 대응한다.

계승나무로부터 Bad Smells를 제거할 때 코드의 동작을 보존하자면 계승계층구조에 있는 클라스들을 하나의 실체모임으로 통합한 후 재그룹화하지 말아야 한다. 이로부터 계승나무의 꼭대기로부터 밑에까지 내려가면서 클라스들을 분해한다.

계승계층구조에 대한 재분해전처리연산들은 다음과 같다.

① 다중의존관계방향망 $G'_{cl} = G'_1 \cup G'_2$ 를 비계승계층구조에 대한 재분해결과에 따라 갱신한다. 여기서 $G'_1 = (V', E'_1), G'_2 = (V', E'_2)$ 이다. 망 G'_2 에서 한 정점이 계승계층구조에 속하지 않는다면 그것과 련결된 릉은 존재하지 않는다. G_2 의 모임 V에 있는 클라스들을 하나하나 론의하면서 빈 메쏘드들을 포함하는 대면부들과 령차원수를 가지는 정점들을

모두 제거하여야 할 정점모임 V_D에 추가한다.

② V_D의 정점들을 삭제하여 나머지망 $\overline{G_2'} = (\overline{V'}, \overline{E_2'})$를 얻는다. 여기서 E_D를 V_D의 정점들과 련결된 릉모임이라고 하면 $\overline{V'} = V' \setminus V_D, \overline{E_2'} = \overline{E_2} \setminus E_D$이다.

③ $\overline{G_2'}$에서 련결성분들을 모두 찾으면 매 련결성분은 하나의 계승나무를 나타낸다.

2. 메쏘드들을 재그룹화하기 위한 무게붙은 무리짓기

1) 메쏘드준위의 무게붙은 망

매 실체모임 $Entities_i \in Sys$는 계승나무 $Tr_k \in TR$에서 임의의 클라스요소들사이의 관계를 메쏘드준위의 무게붙은 무방향망으로 서술한다. 재분해과정에 모든 속성들을 Getter 혹은 Setter메쏘드들로, 속성에 대한 접근을 Getter 혹은 Setter메쏘드들의 호출로 취급한다.

메쏘드들사이의 속성공유, 호출, 기능결합을 메쏘드준위의 무방향망을 구축하여 론의하며 관계의 세기에 따라 매 릉에 무게를 할당한다.

메쏘드 m_i와 m_j의 공유속성무게 saw를 다음과 같이 정의한다.

$$saw(m_i, m_j) = \begin{cases} \dfrac{|M_i \cap M_j|}{|M_i \cup M_j|}, & |M_i \cup M_j| \neq 0\text{일 때} \\ 0, & \text{기타 경우} \end{cases} \quad (1)$$

여기서 M_i와 M_j는 각각 m_i와 m_j에 의하여 호출된 속성들의 모임이다.

$I(m_i, m_j)$를 메쏘드 m_j에 대한 메쏘드 m_i의 호출수, n을 체계에 들어있는 메쏘드들의 총수라고 하자. 메쏘드 m_i와 m_j사이의 메쏘드호출무게 miw는 식 (2)에 의하여 계산된 miw_{ij}와 miw_{ji}의 최대값으로 정의한다.

$$miw_{ij} = \begin{cases} \dfrac{I(m_i, m_j)}{\sum_{k=1}^{n} I(m_k, m_j)}, & \sum_{k=1}^{n} I(m_k, m_j) \neq 0\text{일 때} \\ 0, & \text{기타 경우} \end{cases} \quad (2)$$

$$miw(m_i, m_j) = \max(miw_{ij}, miw_{ji}) \quad (3)$$

같은 기능정의역에서 자주 출현하는 메쏘드들은 긴밀한 기능결합도관계를 가지며 같은 클라스에 속할 확률이 높다. 따라서 기능결합도무게 fcw를 다음과 같이 계산한다.

$$fcw(m_i, m_j) = \begin{cases} \dfrac{ET_{ij}}{ET_i + ET_j}, & ET_i + ET_j \neq 0\text{일 때} \\ 0, & \text{기타 경우} \end{cases} \quad (4)$$

여기서 ET_{ij}는 메쏘드 m_i와 m_j가 함께 출현하는 기능정의역의 수, ET_i와 ET_j는 각각 m_i와 m_j가 호출되는 기능정의역의 수를 나타낸다.

의미론적류사성무게 ssw를 정의하기 위하여 메쏘드 m_i와 m_j는 각각 벡토르 $\vec{m_i}$, $\vec{m_j}$로 표시한다.

$\|\vec{m_k}\|$를 벡토르 $\vec{m_k}$의 유클리드거리라고 하면 $ssw(m_i, m_j)$를 다음과 같이 정의한다.

$$ssw(m_i, m_j) = \frac{\vec{m_i} \times \vec{m_j}}{\|\vec{m_i}\| \times \|\vec{m_j}\|} \tag{5}$$

그러므로 $ssw(m_i, m_j)$는 각이한 코드토막에서의 단어사용류사성에 의존한다. 메쏘드 m_i와 m_j가 개념적으로 관련된다면 $ssw(m_i, m_j)$의 값은 0보다 크다. 턱값 Th_1을 설정하여 그것보다 적은 무게를 가지는 가짜의미론적인 관계를 제거한다. 메쏘드준위의 망모형에서 릉무게를 식 (6)으로 정의한다. 여기서 $\alpha + \beta + \gamma + \eta = 1$이다.

$saw(m_i, m_j), miw(m_i, m_j), fcw(m_i, m_j), ssw(m_i, m_j) \in [0, 1]$이 주어지면 $w_e(m_i, m_j) \in [0, 1]$은 다음과 같이 계산된다.

$$\begin{aligned} w_e(m_i, m_j) = \alpha \times saw(m_i, m_j) + \beta \times miw(m_i, m_j) + \\ + \gamma \times fcw(m_i, m_j) + \eta \times ssw(m_i, m_j) \end{aligned} \tag{6}$$

2) 무게붙은 무리짓기알고리듬

실체들을 재그룹화하는데 공동체검출알고리듬을 리용한다.

알고리듬에서는 다음과 같이 정의된 무게붙은 모듈도 Q를 분할의 품질을 평가하는데 리용한다.

$$Q = \frac{1}{2W}\sum_{ij}\left(w_{ij} - \frac{s_i \times s_j}{2W}\right) \times \delta(Com_i, Com_j) = \sum_{k=1}^{n_c}\left[\frac{W_k}{W} - \left(\frac{S_k}{2W}\right)^2\right] \tag{7}$$

여기서 W는 망에서의 모든 릉들의 무게의 합이고 s_i는 정점 nd_i와 련결된 모든 릉들의 무게들의 합이며 w_{ij}는 정점 nd_i와 nd_j사이의 릉에 붙인 무게이다. 그리고 n_c는 망에서 공동체의 수이고 W_k는 공동체 Com_k내의 릉들의 무게의 합이며 S_k는 공동체 Com_k의 정점들에 련결된 릉들의 무게의 합이다.

식 (7)에 기초하여 전체 체계에 대한 무게붙은 모듈도의 증분을 공동체 Com_i와 Com_j를 통합한 후에 다음과 같이 계산한다.

$$\Delta Q_{ij} = \begin{cases} \dfrac{S_{in}(Com_i, Com_j) - S_i \times S_j}{2W}, & Com_i가\ \ Com_j와\ \ 련결되였을\ \ 때 \\ 0, & 기타\ \ 경우 \end{cases} \tag{8}$$

여기서 $S_{in}(Com_i, Com_j)$는 공동체 Com_i와 Com_j를 련결하는 릉들의 무게를 모두 합한것이다.

모듈도증분행렬 $A_Q = (\Delta Q_{ij})$를 망에서 매 공동체쌍을 통합하기 전에 모듈도증분을 계산하고 갱신하는데 리용한다.

코드동작을 보존하기 위하여 문법규칙과 전조건에 따라 분할하지 말아야 할 메쏘드들을 공동체검출을 시작하기 전에 하나의 공동체에 묶는다. 여기서 망에서의 다른 정점들은 모두 독립적인 공동체로 론의된다. 또한 모듈도증분행렬 A_Q의 매 원소 ΔQ_{ij}들을 계산한다. 만일 ΔQ_{mn}이 A_Q에서 최대원소라면 공동체 Com_m과 Com_n을 통합하며 동시에 A_Q를 갱신한다. 이와 같은 방법으로 공동체들을 A_Q의 최대원소가 령이 될 때까지 합친다.

이때 공동체구조는 최량인 분할로 되고 체계의 모듈도는 상승한다.

공동체검출알고리듬은 다음과 같다.

입력: 공유속성에 대한 린접행렬 $A_{n\times n}$

메쏘드호출에 대한 린접행렬 $B_{n\times n}$

동시실행에 대한 린접행렬 $C_{n\times n}$

의미론적련관의 린접행렬 $D_{n\times n}$

출력: 공동체모임 C, 무게화된 모듈도 Q

① 먼저 방정식 $A' = \alpha \times A_{n\times n} + \beta \times B_{n\times n} + \gamma \times C_{n\times n} + \eta \times D_{n\times n}$ 에 기초하여 메쏘드준위망 $G = (V, E)$ 의 린접행렬 A' 를 얻는다. 분할하지 말아야 할 메쏘드들은 하나의 공동체로 묶는다. 그외 매 정점 $nd_i \in V$ 를 공동체로 론의한다. 망 G가 $|C|$개의 공동체로 나누인다고 가정하면 공동체모임 C는 $C = \{Com_1, Com_2, \cdots, Com_{|C|}\}$ 이다. 이때 무게화된 모듈도는 0으로 놓는다.

② 식 (8)에 따라 초기 모듈도증분행렬 A_Q의 원소 ΔQ_{ij}를 다음과 같이 계산한다.

for(i = 1; i <= $|C|$; i ++) {
  for(j = 1; j <= $|C|$; j ++) {
   if $A'_{ij} \neq 0$

$$\Delta Q_{ij} = (S_{in}(Com_i, Com_j) - S_i \times S_j)/2W;$$

  else

$$\Delta Q_{ij} = 0;$$

  }

③ 모듈도증분을 고려하지 않고 행렬 A_Q에서 최대원소 $\Delta Q_{\min}$을 찾는다.

공동체 Com_m과 Com_n을 통합하고 통합된 공동체를 Com_n으로 표시한다. 행 m과 렬 m에 있는 원소를 없애고 행 n과 렬 n에 있는 원소들을 다음과 같이 갱신한다.

$$\Delta Q'_{nk} = \Delta Q'_{kn} = \begin{cases} \Delta Q_{mk} + \Delta Q_{nk}, & Com_k\text{가 } Com_m, Com_n\text{과 련결} \\ \Delta Q_{mk} - S_n \times S_k / 2W^2, & Com_k\text{가 } Com_m\text{에 련결,} \\ & Com_n\text{과 련결되지 않았을 때} \\ \Delta Q_{nk} - S_m \times S_k / 2W^2, & Com_k\text{가 } Com_n\text{에 련결,} \\ & Com_m\text{과 련결되지 않았을 때} \end{cases}$$

동시에 식

$$Q = Q + \Delta Q_{\min}$$

에 의하여 무게붙은 모듈도를 갱신한다. 같은 원래 클라스에 속하는 실체들로 이루어진 공동체들을 통합하여 생성된 최대모듈도증분이 0보다 낮아질 때까지 그 과정을 반복한다.

④ 서로 다른 클라스들의 실체들을 포함하고 해당한 $\Delta Q_{\min}$이 행렬 A_Q에서 가장 큰 공동체쌍을 계속 찾는다.

단계 ③의 방법으로 A_Q의 원소들을 계속 갱신한다. A_Q의 최대값원소가 0과 같거나

작을 때 무리짓기를 끝낸다.

3) 클라스추출재분해와 메쏘드, 마당이동재분해의 기회식별

메쏘드이동 및 마당이동재분해, 클라스추출재분해의 기회를 무리분할과 원래 클라스의 구조에서의 차이를 해석하여 동시에 식별한다. 같은 련결성분에서 매 원래의 클라스와 얻어진 매 공동체를 비교한다. C_t를 무리짓기분석에 의하여 얻어진 공동체모임이라고 하고 V_t를 원래의 클라스모임이라고 하면 C_t, $V_t \subseteq cc_t$, $t \in [1, 2, \cdots, |CC|]$이다.

그러면 모든

$$m, i, j, \in [1, 2, \cdots, |V_t|], n, p, q \in [1, 2, \cdots, |C_t|]$$

에 대하여 $i \neq j$이고 $p \neq q$이며

$$Com_p \cap Com_q = \phi$$

이다. 여기서 $|V_t|$와 $|C_t|$는 각각 련결성분 cc_t에 관한 원래 클라스와 새로운 클라스의 총 개수이다.

① 모든 $p, q \in [1, 2, \cdots, |V_t|]$, $p \neq q$, $k \in [1, 2, \cdots, N_p]$, $N_p \geq 2$에 대하여

$$Com_k \subset clas_p, Com_k \cap clas_q = \phi$$

이면 원래의 클라스 $clas_p$를 클라스추출연산대상으로 식별한다. Com_k는 $clas_p$로부터 추출되여야 할 새로운 클라스이며 N_p는 $clas_p$가 분해되여야 할 그룹의 수이다.

② 모든 $p, q \in [1, 2, \cdots, |V_t|]$, $p \neq q$, $k = 1$에 대하여

$$clas_p \subset Com_k, Com_k \cap clas_q = T \neq \phi$$

이면 메쏘드, 마당이동재분해의 후보를 원래 클라스 $clas_q$에서 식별한다. 실체모임 T가 클라스 $clas_q$로부터 $clas_p$에로 이행하여야 한다.

③ $clas_p = \bigcup_{i=1}^{Np} clas_{pi}, clas_q = \bigcup_{i=1}^{Nq} clas_{qj}$ 라고 가정하자.

$|Com_k \cap clas_p| > |Com_k \cap clas_q|$이면 실체모임 $Com_k \cap clas_p$를 원래의 클라스 $clas_p$로부터 추출하여 새로운 클라스 $clas_p^{new}$를 형성한다. 또한 메쏘드, 마당이동재분해를 $clas_q$에 적용하여야 하며 실체모임 $Com_k \cap clas_q$를 클라스 $clas_q$로부터 $clas_p^{new}$에로 이동시켜야 한다.

$|Com_k \cap clas_p| < |Com_k \cap clas_q|$이면 원래의 클라스 $clas_p$우에서 메쏘드 및 마당이동재분해를 진행하여야 하며 실체모임 $Com_k \cap clas_p$는 $clas_p$로부터 $clas_q$에로 이동시켜야 한다.

맺 는 말

론문에서는《높은 응집도와 낮은 결합도》의 쏘프트웨어설계원리에 따라 메쏘드준위의 망우에서 체계의 실체들을 재그룹화하는 무게붙은 무리짓기알고리듬에 기초하여 메쏘드, 마당이동, 클라스추출의 기회를 식별하며 클라스들을 통합하거나 분할하는 방법을 제안하였다.

참 고 문 헌

[1] G. Bavota, R. et al.; IEEE Trans. Softw. Eng., 40, 7, 671, 2014.

[2] L. M. Hakik et al.; Int. J. Comput. Sci. Netw. Secur., 14, 1, 11, 2014.

주체108(2019)년 11월 5일 원고접수

Study on Software Refactoring by Clustering

Jon Hyon Chol, *Sin Chun Ok*

In this paper we propose wehighted clustering algorithm to identify refactoring opportunities and a method that regroup system's components based on method-level network.

Keywords: refactoring, clustering, cohesion