분기점 기반 트리 분석을 통한 소프트웨어 클러스터링 결과 비교†

(Comparison of Software Clustering using Split Based Tree Analysis)

엄재철‡ **이찬근**¶

(Jaechul Um) **(Chan-gun Lee)**

요 약 본 연구에서는 소프트웨어 아키텍처 복원을 위한 계층적 클러스터링(Clustering) 결과를 정량적으로 평가 할 수 있는 비교 메트릭(Metric)을 제시한다. 소프트웨어 클러스터링의 정량적 평가는 소프트웨어의 구조적 변화에 대한 이해를 돕는 척도를 제시하는 연구이다. 이를 위해 생물정보학에서 상호 유전 형질 분석에 사용하는 분기점(Split) 개념을 소프트웨어 아키텍처 분석에 적용한다.

키워드 소프트웨어 아키텍처, 분기점, 클러스터링

Abstract We propose a novel metric for quantitatively comparing different clustered results generated from software clustering algorithms. A quantitative evaluation of software clustering helps understanding of architectural changes of software. The concept of split, which has been used for analysis of genetic characters in bio-informatics, is applied in the analysis of software architecture.

Key words Software Architecture, Split, Clustering

1. 서 론

소프트웨어 아키텍처는 소프트웨어 시스템 구조와 동작을 기술한 구성도로 소프트웨어 전반에 대한 이해, 개발, 유지보수 등에 도움을 준다. 일반적으로 소프트웨어 아키텍처는 소프트웨어가 진화함에 따라 변화하게 되는데, 이 때 설계 문서가 같이 수정되지 않는 문제가 발생할 수 있다. 이러한 문제는 소프트웨어 아키텍처 복원을 통해 해결될 수 있는데, 이것을 위한 주요한 방법이 소프트웨어 클러스터링이다. 본 논문에서는 소프트웨어 클러스터링 결과 비교 방법을 분기점 기반 트리 분석으로 제시한다. 이러한 소프트웨어 클러스터링 결과 비교는 소프트웨어 설계 분석, 역공학, 리팩토링 등을 돕는 방법으로 활용될 수 있다.

† 본 연구는 한국연구재단 기초연구사업(과제번호 2010-0015636, 2011-0013924)의 지원을 받았습니다.

‡ 학생회원 : 중앙대학교 컴퓨터공학부
steveum0105@gmail.com

¶ 종신회원 : 중앙대학교 컴퓨터공학부 교수. 교신저자
cglee@cau.ac.kr

2. 관련 연구

기존의 소프트웨어 클러스터링 결과 비교를 위한 대표적인 방법으로는 MoJo[1]가 있다. MoJo는 소프트웨어 아키텍처를 여러 개의 집합으로 나누고 Move와 Join 연산을 통해 비교한다. MoJo는 특히 소프트웨어 역공학 및 클러스터링 분야에서 상호 소프트웨어 아키텍처를 비교하는데 주로 사용된다. 그 밖에도 MoJo를 보완한 UpMoJo[2], MoJo를 수식화하여 재정의한 MoJoFM[3] 등이 소프트웨어 클러스터링 결과 비교를 수행하고 있다.

3. 소프트웨어 클러스터링 결과 비교

3.1 분기점 탐색

분기점은 발생학에서 어떤 두 유전 형질에 대한 공통 부모를 의미한다. 이러한 분기점은 여러 방법으로 분석되어 유전 형질의 파생 정보를 나타내는 요소로 사용된다. T. Mailund[4]와 Q. Zhang[5]은 분기점으로 트리 구조를 분석하는 방법을 설명하고 있다. T. Mailund는 트리에서 분기점을 찾고, 분기점 간의 거리를 측정하여 서로 다른 구조의 트리 집합을 비교할 수 있음을 설명하고 있다. Q. Zhang은 분기점을 이용한 트리 분석을 위한 새로운 메트릭을 제시하고 그것을 클러스터링에 응용할 수 있음을 보인다.

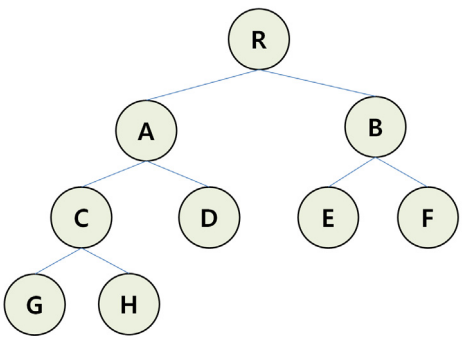

그림1. 유전 트리

그림1은 어떤 생물의 발생학적 유전 트리를 나타낸다. 그림1에서 개체 G와 D가 어떤 공통된 형질을 가지고 있다면, 그것은 공통된 부모인 A에서 유전되었다고 할 수 있다. 이 때 A는 G와 D의 분기점이며, A = Split(G, D)와 같이 표현된다. 다음 식은 분기점의 일반 정의를 나타낸다.

$$N_s = \mathrm{Split}(n_i, n_j) \qquad (1)$$

(1)은 두 노드 n_i, n_j에 대한 분기점 노드 N_s를 정의한다. 소프트웨어 아키텍처에서 분기점 역시 (1)의 정의를 따른다. 이 때 소프트웨어 아키텍처를 나타내는 트리는 클래스 단위 노드(Node)와 상속 관계의 선분(Edge), 파일 단위 노드와 패키지 구성 관계의 선분, 모듈 단위 노드와 의존성(Dependency) 관계의 선분 등 다양하게 표현될 수 있다.

본 논문에서는 소프트웨어 아키텍처의 계층 구조에서 분기점을 탐색한 후, 그것을 분석하여 소프트웨어 클러스터링 결과 비교를 수행한다.

3.2 분기점 순서(Split-Order)를 이용한 클러스터링 결과 비교

Q. Zhang은 일반적인 클러스터링 결과 비교를 위해 분기점 순서 방법을 제시한다. 분기점 순서는 각 클러스터링 결과에 대해 분기점을 탐색한 후, 그 분기점과 루트 간의 거리를 대소 관계로 비교한다.

세 노드 α, β, γ에 대한 분기점 순서는 SplitOrder(α, β, γ)와 같이 표현하며 다음과 같이 정의된다.

- SplitOrder(α, β, γ) = ' < ', Split(α, β)가 Split (α, γ)보다 상위 레벨 노드일 때
- SplitOrder(α, β, γ) = ' > ', Split(α, γ)가 Split (α, β)보다 상위 레벨 노드일 때
- SplitOrder(α, β, γ) = ' = ', Split(α, β)와 Split (α, γ)가 동일 레벨 노드일 때

그림2, 3은 계층적 클러스터링의 서로 다른 결과이다. 위의 그림2, 3에서 모든 노드 쌍에 대해 분기점 순서를 구하고 서로 같은 값을 갖는 노드 쌍의 개수가 유사도가 된다.

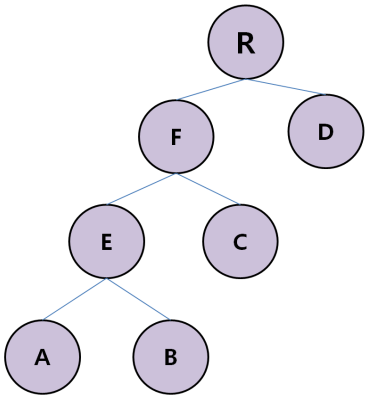

그림2. 계층적 클러스터링 결과 A

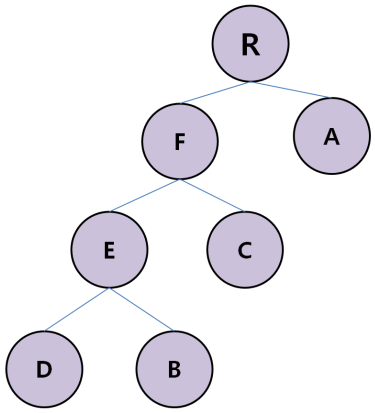

그림3. 계층적 클러스터링 결과 B

그러나 Q. Zhang의 분기점 순서 방법은 일부 경우에 직관적이지 않은 결과를 나타낼 수 있다. 가령 그림2와 3은 노드 A와 D의 위치가 바뀐 것을 제외하고는 서로 동일한 클러스터링 결과이다. 분기점 순서 방법을 이용한 리프노드 A~D에 대한 비교에서 총 12개의 조합쌍 중 서로 같은 기호의 개수는 0개이다. 즉, 그림2와 3은 서로 어느 정도 유사한 부분이 있음에도 유사도가 0으로 출력된다.

3.3절에서는 분기점 순서의 이러한 문제점을 개선한 계층적 클러스터링 결과 비교 방법을 제시하고, 소프트웨어를 대상으로 한 비교 분석을 논의한다.

3.3 분기점을 이용한 소프트웨어 클러스터링 결과 비교

3.1절에서 소개한 분기점의 정의를 이용하여 동일 소프트웨어의 클러스터링 결과를 비교한다. 이를 위해 다음 식을 사용한다.

$$D = \sum_{n_i, n_j \in N} \left| \begin{array}{l} \text{dist}(R_2, \text{Split}(n_i, n_j)) \\ - \text{dist}(R_1, \text{Split}(n_i, n_j)) \end{array} \right| \quad (2)$$

$$\text{MaxD} = \sum_{n_i, n_j \in N} \begin{array}{l} \max(\text{dist}(R_2, \text{Split}(n_i, n_j)) \\ , \text{dist}(R_1, \text{Split}(n_i, n_j))) \end{array} \quad (3)$$

$$d = \frac{\text{MaxD} - D}{\text{MaxD}} \quad (4)$$

(2)는 서로 다른 두 클러스터링 결과의 모든 노드 N에 대하여 분기점과 루트 R_1, 분기점과 루트 R_2 간의 거리 차의 합을 정의한다. 이 때 dist(α, β)는 두 노드 α와 β 간의 최단거리를 의미하고, D는 두 클러스터링 결과 간의 차이를 의미한다. (3)은 두 클러스터링 결과에서 같은 노드 쌍에 대해 두 분기점과 루트 거리 중 최댓값을 합한 것으로 D가 가질 수 있는 최댓값을 의미한다. (4)는 두 클러스터링 간의 유사도 d를 구하기 위해 MaxD에 대한 D의 비율을 정의한다.

그림 2와 3에 대해 D는 6이며, MaxD는 7이다. 따라서 두 클러스터링 결과의 유사도 d는 7분의 1로 약 14%가 된다. 이것은 앞선 3.2장에서 분기점 순서의 유사도가 12분의 0으로 0%가 나왔던 것과 다르게 클러스터링 결과가 일부 유사함을 반영하고 있음을 보인다.

4. 결 론

본 논문에서는 소프트웨어 클러스터링 결과 비교를 위해 분기점을 이용한 새로운 트리 분석 방법을 제시했다. 현재 본 논문에서 제시하는 메트릭을 소프트웨어 설계 분석, 역공학 등 다양한 분야에 활용하는 방법을 확장 연구하고 있으며, 기존의 다른 방법론과의 비교 분석 또한 연구를 계속하고 있다.

참 고 문 헌

[1] V. Tzerpos and R. C. Holt, "MoJo: A Distance Metric for Software Clustering," Proc. of Working Conference on Reverse Engineering, 1999.

[2] M. Shtern and V. Tzerpos, "Lossless Comparison of Nested Software Decompositions," Proc. of Working Conference on Reverse Engineering, 2007.

[3] Z. Wen and V. Tzerpos, "An effectiveness measure for software clustering algorithms," Proc. of the 12th IEEE International Workshop on Program Comprehension, 2004.

[4] T. Mailund, "SplitDist-Calculating Split-Distances for Sets of Trees", Proc. of BiRC, 2004.

[5] Q. Zhang, E. Y. Liu, and W. Wang, "Split-Order Distance for Clustering and Classification Hierarchies", Scientific and Statistical Database Management, Vol.5566, pp.517-534, 2009.

저자소개

엄 재 철

2012년 중앙대학교 컴퓨터공학부 학사.
2012년~현재 중앙대학교 컴퓨터공학부 석사과정.

<관심분야> 실시간 소프트웨어, 소프트웨어 클러스터링, 데이터 마이닝

이 찬 근

1996년 중앙대학교 전자계산학과 학사
1998년 KAIST 전산학과 석사.
2005년 Univ. of Texas at Austin 전산학과 박사.
2005년~2007년 미국 인텔 소프트웨어 엔지니어.
2007~현재 중앙대학교 컴퓨터공학부 조교수.

<관심분야> 실시간 소프트웨어, 수행시간 모니터링, 소프트웨어 테스팅