테스트 데이터 생성을 위한 개선된 이웃 선택 방법을 이용한 담금질 기법 기술†

(a improved neighborhood selection of simulated annealing technique for test data generation)

최현재‡ 이선열§ 채흥석¶
(Hyun Jae Choi) (Seon Yeol Lee) (Heung Seok Chae)

요 약 금질 기법을 이용한 테스트 데이터 자동 생성 기법은 오랫동안 연구되어 왔으며, 효율적인 테스트 데이터 생성 방법 중 하나이다. 그러나 인풋 도메인이 넓을 경우, 기존의 담금질 기법은 이웃 선택 기법의 한계 때문에 나쁜 성능을 보였다. 이러한 한계를 극복하기 위해, 우리는 새로운 이웃 선택 기법인 분기 거리 이웃 선택 방법을 제안한다. 제안된 기법의 성능을 검증하기 위해서 우리는 분기 거리 이웃 선택 기법, 기존의 이웃 선택 기법 그리고 랜덤 테스트 데이터 생성 기법을 비교하는 실험을 수행하였다. 실험 결과 제안된 기법이 2가지 기법에 비해 우수한 성능을 보임을 알 수 있었다.

키워드 테스트 데이터 자동 생성, 동적 테스트 데이터 생성, 담금질 기법, 분기 거리 이웃 선택, 패스 테스팅

Abstract Simulated annealing has been studied a long times. And it is one of the effective techniques for test data generation. But basic SA methods showed bad performance because of neighborhood selection strategies in the case of large input domain. To overcome this limitation, we propose new neighborhood selection approach, Branch Distance. We performs case studies based on the proposed approach to evaluate it's performance and to compare it whit basic SA and Random test generation. The results of the case studies appear that proposed approach show better performance than the other approach.

Key words Test data automation, Dynamic test data generation, Simulated annealing, Branch distance neighborhood selection, path testing

1. 서 론

소프트웨어 개발 산업에서 소프트웨어 테스팅은 전체 소프트웨어 개발 비용의 약 50%를 차지하는 많은 비용이 요구되는 작업이다[1]. 소프트웨어 테스팅에 요구되는 높은 비용은 소프트웨어 테스팅이 소프트웨어 개발 프로세스에서 매우 큰 비중을 차지하는 작업인 것을 나타낸다.

소프트웨어 테스팅은 테스트 데이터 생성, 테스트 실행, 테스트 결과 검증의 3단계로 이루어진다. 그 중에서 테스트 데이터 생성은 전체 소프트웨어 테스팅 비용의 약 40%를 차지한다[2][3]. 때문에 소프트웨어 테스팅 비용 절감은 전체 소프트웨어 개발 비용 절감을 위해 중요한 이슈이다.

테스트 데이터 생성 비용 절감을 위해 최근 테스트 데이터 자동 생성 방법의 하나인 탐색 기반

† 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.
‡ 학생회원 : 부산대학교 컴퓨터공학부
gonoki@pusan.ac.kr
§ 학생회원 : 부산대학교 컴퓨터공학부
sun302412@pusan.ac.kr
¶ 정 회 원 : 부산대학교 컴퓨터공학부 부교수
hschae@pusan.ac.kr
논문접수 : 2011년 05월 13일
심사완료 : 2011년 06월 10일

소프트웨어 테스팅이 많이 연구되고 있다. 탐색 기반 소프트웨어 테스팅은 메타 휴리스틱 알고리즘을 이용하여 테스트 요구사항을 만족하는 테스트 데이터를 자동적으로 생성하는 기법이다. 탐색 기반 테스트 데이터 생성 기법으로 랜덤, 담금질 기법[3][4][5][6][7], 유전 알고리즘[3][5][7][11][12] [13][14]을 사용하는 연구들이 수행되었다.

그 중 담금질 기법을 이용한 테스트 데이터 자동 생성 기법은 알고리즘이 단순하여 알고리즘 수행 속도가 빠르고 구현이 간단하기 때문에 많은 연구에서 적용되었고 몇몇 연구에서 우수한 성능을 보여주었다[3][4][5][6][7].

하지만 담금질 기법은 데이터 범위가 넓어지면 성능이 나빠지는 한계가 있었다[3]. 이러한 담금질 기법의 한계를 극복하기 위해, 본 논문은 담금질 기법의 새로운 이웃 선택 방법인 분기 거리 이웃 선택 방법을 제안한다. 또한 우리는 제안한 방법을 적용하기 위한 도구를 개발하였고 이 도구를 이용해 2개의 프로그램에 대해 실험을 수행하였다. : triangle3, triangle4

본 논문은 탐색 기반 테스트 데이터 생성 기법을 이용하여 패스 테스팅을 수행한다. 패스 테스팅은 생성한 테스트 케이스들이 목표로 하는 패스들을 커버하도록 하는 매우 복잡하고 강력한 기법이다. 탐색 기반 소프트웨어 테스팅의 테스트 적합성 기준을 높은 패스 커버리지로 설정하여 패스 테스팅을 수행한다. 패스 테스팅은 많은 탐색 기반 테스트 데이터 생성 기법에 사용되어 왔다[8][9][10].

이 논문의 기여는 다음과 같다. 첫째, 기존의 담금질 기법을 이용한 테스트 데이터 생성 기법의 한계를 극복하고자 새로운 이웃 선택 방법을 제안한다. 둘째, 제안한 기법의 성능을 평가하기 위해 사례 연구를 수행하였다. 셋째, 사례 연구의 결과를 통해 기존의 담금질 기법과 랜덤 테스트 데이터 생성 기법과 새로이 제안한 분기 거리 이웃 선택을 이용한 담금질 기법을 비교, 분석한다.

본 논문은 다음과 같이 구성된다. 2장에서는 배경 연구를 소개한다. 3장에서는 효과적인 테스트 데이터 생성을 위한 분기 거리 이웃 선택 방법을 제안한다. 4장에서는 제안한 방법을 성능을 알아보기 위해 사례 연구를 수행한다. 5장에서는 결론 및 향후 연구 방향을 기술한다.

2. 배경 연구

배경 연구에서는 탐색 기반 소프트웨어 테스팅, 담금질 기법 그리고 기존의 이웃 선택 방법에 대해서 설명한다.

2.1 탐색 기반 소프트웨어 테스팅

탐색 기반 소프트웨어 테스팅은 테스트 적합성 기준에 따라서 테스트 데이터를 자동적으로 생성하는 프로세스이다. 이 방법은 프로그램 수행을 통해 동적으로 정보를 수집하고 메타 휴리스틱 알고리즘을 사용하여 테스트 적합성 기준을 만족하는 테스트 데이터를 생성한다. 언덕 오르기 알고리즘, 담금질 기법, 탐욕 알고리즘, 유전 알고리즘의 메타 휴리스틱 알고리즘을 탐색에 사용한 탐색 기반 소프트웨어 테스팅 연구가 진행되었다.

테스트 적합성 기준을 만족하는 테스트 데이터를 생성하기 위해서 동적으로 정보를 수집하기 위해 적합도 함수를 사용한다. 적합도 함수는 테스트 데이터가 테스트 목적에 얼마나 적합한지를 표현하는 함수이다. 이것은 테스트 적합성 기준을 만족하는 테스트 데이터 탐색을 인도하는 데 사용된다.

2.2 담금질 기법

담금질 기법은 개체가 국소 최적점에서 벗어나 전역 최적점을 찾을 수 있도록 하는 알고리즘이다. 담금질 기법은 온도에 따른 원자의 에너지 변화를

모델로 한 것이다. 물체의 온도를 높여주면 열에 의해 원자의 위치가 초기 위치로부터 멀어진다. 그 후 물체의 온도를 서서히 내려주면 원자가 안정화되면서 초기 상태보다 내부 에너지가 한층 극소인 상태를 얻을 가능성이 많아진다. 담금질 기법은 이러한 물리 법칙을 모사한 것이다.

표 1. 담금질 기법을 이용한 테스트 데이터 생성 알고리즘

```
초기 재시작 횟수 r = 0
Start:
초기 상태 선택 s
초기 온도 선택 t
초기 상태의 Cost 평가    cost(s)
Loop while (종료 조건 == false)
   If (r ≥ 최대_재시작_횟수) Then
      GoTo Start:
   End If
   새로운 상태 선택 s' ∈ Neighborhood(s)
   새로운 상태의 Cost 평가 cost(s')
   δ = cost(s') - cost(s)
   If (δ ≤ 0) Then
      s = s'
   Else
      [0,1] 범위의 랜덤 값 생성    x
      If (x < exp(-δ/ t)) Then
         현재 상태 갱신    s = s'
      End If
   End If
   온도 냉각    t = coolingScheduling(t)
   재시작 횟수 증가 r = r + 1
End Loop
```

표 1은 담금질 기법을 이용한 테스트 데이터 생성 알고리즘을 의사 코드로 설명한 것이다. 초기 매개변수 설정을 하고 알고리즘을 시작한다. 현재 상태의 이웃을 선택하여 새로운 상태를 선택하고 적합도 함수를 이용하여 새롭게 선택된 상태의 cost를 평가한다. cost 값은 매개변수로 전달되는 상태가 얼마나 테스트 적합성 기준에 가까운가를 나타낸다. 이 값은 0에 가까울수록 적합성 기준에 더 가깝고 크면 클수록 적합성 기준에서 더 먼 것을 표현한다. 새로운 상태가 현재 상태보다 테스트 적합성 기준에 가까워 cost 값이 더 작을 경우 현재 상태를 갱신한다. 만약 새로운 상태가 현재 상태보다 테스트 적합성 기준에 더 멀어서 cost 값이 더 클 경우에도 온도를 통해 구해지는 확률(exp(-δ/t))로 현재 상태를 갱신한다. 그 후, 냉각 스케쥴링을 이용하여 온도를 감소시킨다. 종료 조건이 만족될 때까지 이러한 과정을 반복한다.

담금질 기법의 특징으로는 상태 전이 결정을 할 때 탐색 조건에 가까운, 좋은 상태로만 전이하는 것이 아니라 확률적으로 나쁜 상태로 전이할 수 있도록 하는 것이다. 확률은 일반적으로 exp(-δ/t) 라는 수식으로 결정된다. 확률에 영향을 주는 매개변수는 온도이다. 만약 온도가 높으면 확률이 높고 만약 온도가 낮으면 확률이 낮게 결정된다. 이와 같은 수행을 통해 현재의 상태가 국소 최적에 빠져 있더라도 이를 벗어나 전역 최적점을 찾을 수 있도록 한다. 알고리즘의 수행이 진행될수록 온도가 점차 낮아져서 시간이 지날수록 좋은 상태로만 전이할 확률이 점차 높아진다.

온도 냉각 알고리즘에 의해 나쁜 상태로 전이할 확률이 반복 횟수가 증가함에 따라 낮아진다. 이 때문에 반복 횟수가 늘어나면 상태가 지역 최적에서 벗어나지 못할 수 있다. 때문에 일정 횟수 이상 반복 수행하였을 경우 알고리즘을 재시작하는 것이 성능이 더 좋을 수 있다.

담금질 기법을 이용한 테스트 데이터 생성 기법의 주요 매개변수는 초기 상태 생성 방법, 데이터 범위, 적합성 평가 함수, 이웃 선택 방법, 이웃 크기, 최대 재시작 횟수, 초기 온도 설정, 냉각 알고리즘이 있다.

2.3 기존의 이웃 선택 방법

기존의 이웃 선택 방법은 기존의 테스트 데이터 자동 생성 연구들에서 이용한 기본적인 담금질 기법의 이웃 선택 알고리즘을 말한다.

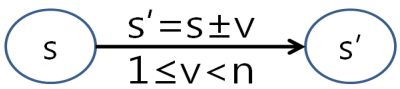

그림 1. 기존의 이웃 선택 방법

그림 1은 기존의 이웃 선택 방법을 그림으로 표현한 것이다. s' 는 새로운 상태, s는 현재 상태, v는 변화 값, n은 이웃 크기를 나타낸다. 이웃 크기는 현재 상태의 이웃을 결정하는 범위 값이다. 기존의 이웃 선택 방법은 현재 상태에서 이웃 크기 이내의 범위를 이웃이라고 판단한다. 따라서 새로운 상태는 이웃 크기 이내의 범위의 값이 선택된다.

표 2. 기존의 이웃 선택 알고리즘

```
현재 상태 s
현재 이웃 크기 n
변화 값 선택 v = {v| 1≤v<n}
새로운 상태 선택 s' = s ± v
If (isOutOfDomainRange(s')) Then
    다시 새로운 상태 선택 s' =
BasicNeighbor- hoodSelection (s)
End If
```

표 2는 기존의 이웃 선택 알고리즘을 의사 코드로 표현한 것이다. 만약 이웃 선택 알고리즘으로 생성된 새로운 상태가 데이터 범위를 초과한다면 다시 이웃 선택 알고리즘을 수행하여 새로운 상태를 구한다.

3. 분기 거리 이웃 선택

분기 거리 이웃 선택에서는 분기 거리에 대해서 설명하고 분기 거리 이웃 선택 방법에 대해 소개한다.

3.1 분기 거리

분기 거리는 테스트 데이터가 얼마나 목표에 가까운지를 평가하는 값이다. 이것은 탐색 기반 테스트 데이터 자동 생성기법에서 적합도 함수로 흔히 사용되는 Korel 의 방법[8]을 조금 수정한 것이다. 분기 거리의 값은 목표에 가까울수록 값이 작아진다.

표 3은 연산자 유형별 분기 거리 계산의 방법을 표로 나타낸 것이다.

표 3. 분기 거리

연산자 유형	조건 예	분기 거리 (K = 1000)
Equal	if (a == 1)	if (a == 1) 0 else abs(a - 1)
NotEqual	if (a != 1)	if (a != 1) 0 else K
And	if ((a==1) && (b==1))	sum((a == 1) ? 0 : abs(a - 1), ((b == 1) ? 0 : abs(b - 1))
Or	if ((a==1) ¦¦ (b==1))	min((a == 1) ? 0 : abs(a - 1), ((b == 1) ? 0 : abs(b - 1))
Grater	if (a > 1)	if (a > 1) 0 else 1 - a + 1
GraterEqual	if (a >= 1)	if (a >= 1) 0 else 1 - a
Less	if (a < 1)	if (a < 1) 0 else a - 1 + 1
LessEqual	if (a <= 1)	if (a <= 1) 0 else a - 1
Not	if (!(a == 1))	if (!(a == 1)) 0 else abs(a - 1)

(a == 1) 분기의 경우 만약 a와 1이 같다면 목표 조건을 달성하였기 때문에 분기 거리를 0으로 계산한다. 하지만 만약 a와 1이 다르다면 a와 1 사이의 차이 값이 (a == 1) 조건을 만족하기 위한 분기 거리가 된다. 예를 들어, a가 2일 경우 분기 거리는 abs(2-1) = 1이 된다.

(a != 1) 분기의 경우 목표 조건을 달성하였을 경우 분기 거리를 0으로 계산한다. 만약 a와 1이 같아 목표 조건을 달성하지 못하였을 경우 임의의 큰 값 K를 분기 거리로 나타낸다. 이것은 a가 어느 값으로 이동하여도 목표 분기의 조건을 달성할 수 있기 때문이다. 실험에서는 K 값을 1000으로 설정하였다.

And 조건의 경우 좌변과 우변의 두 조건을 모두를 만족해야 하기 때문에 좌변과 우변의 분기 거리를 더하여 조건의 분기 거리로 표현한다.

Or 조건의 경우 좌변과 우변의 두 조건 중 하나만을 만족하면 되기 때문에 좌변과 우변의 분기 거리 중 작은 값을 선택하여 조건의 분기 거리로 표현한다.

크거나 작은 비교 조건을 표현하는 분기문의 경우 비교 조건의 좌변과 우변 중 작은 값에서 큰 값을 뺀 후 1을 더하여 분기 거리를 표현한다.

크거나 같은 혹은 작거나 같은 비교 조건을 표현하는 분기문은 비교 조건의 좌변과 우변 중 작은 값에서 큰 값을 뺀 값을 분기 거리로 표현한다. 이 경우 양 변의 값이 같아지기만 해도 목표 조건을 만족하기 때문에 다른 값을 더하지 않는다.

3.2 분기 거리 이웃 선택 방법

분기 거리 이웃 선택 방법은 데이터 범위가 넓을 경우 성능이 나쁘다는 기존의 이웃 선택 방법의 한계를 극복하기 위한 것이다. 만약 분기 거리가 멀 경우, 즉 목표에서 먼 테스트 데이터는 이웃을 분기 거리만큼 떨어진 값으로 선택하여 다음 테스트 데이터를 분기 거리만큼 먼 거리에 위치한 값으로 선택한다. 분기 거리가 가까울 경우에는 기존의 이웃 선택 방법을 이용하여 기존의 담금질 기법의 장점을 그대로 활용하고자 하였다.

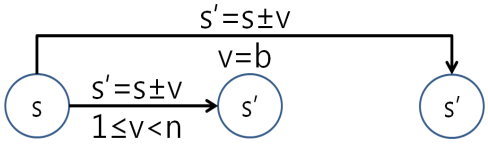

그림 2. 분기 거리 이웃 선택

분기 거리 이웃 선택 방법은 데이터 범위가 넓을 경우 성능이 나쁘다는 기존의 이웃 선택 방법의 한계를 극복하기 그림 2는 분기 거리 이웃 선택 기법을 그림으로 표현한 것이다. s’ 는 새로운 상태, s는 현재 상태, v는 변화 값, n은 이웃 크기, b는 분기 거리를 나타낸다.

표 4. 분기 거리 이웃 선택 알고리즘

```
분기 거리 b
현재 상태 s
이웃 크기 n
새로운 상태 선택 s’ = s ± b
If (b < n || isOutOfDomainRange(s’)) Then
    새로운 상태 선택 s’ = BasicNeighborhood
Selection(s)
End If
```

표 4는 분기 거리 이웃 선택 알고리즘을 의사 코드로 표현한 것이다. 만약 분기 거리가 이웃 크기보다 작거나 분기 거리 이웃 선택 알고리즘으로 생성된 새로운 상태가 데이터 범위를 초과한다면 기존의 이웃 선택 알고리즘을 이용하여 새로운 상태를 구한다.

4. 사례 연구

사례 연구에서는 테스트 대상 프로그램에 대해서 소개하고 실험에서 적용할 담금질 기법의 세부 매개변수에 대해서 설명한다. 그리고 실험 평가기준을 소개하고 실험 결과를 보여주고 이를 분석한다.

4.1 테스트 대상 프로그램

2가지의 C 프로그램에 대해서 실험을 수행하였다. triangle3과 triangle4는 [15][16] 에서 사례 연구로 사용되었던 프로그램이다. 두 프로그램 모두 3개의 매개변수를 받아 삼각형이 될 수 있는지, 삼각형이 될옷 수 있다면 어떤 삼각형이 될 수 있는지 유형을 분석하는 기능을 수행한다. triangle4가 triangle3 프로그램보다 분기문이 더 복잡하게 구성되어 있다는 차이가 있다.

표 5. 테스트 대상 프로그램

프로그램 이름	LOC	패스 개수	매개변수 개수
triangle3	69	9	3
triangle4	97	14	3

표 5는 테스트 대상 프로그램을 나타낸 것이다. 두 프로그램 모두 3개의 int 타입 매개변수를 입력으로 받는다. 실험은 Intel Core 2 Duo E6600, 400MHz 1G RAM의 성능을 가진 컴퓨터로 수행하였다. triangle3의 코드 라인 수는 69이고 도달할 수 있는 패스는 9개, 도달할 수 없는 패스는 0개

이다. triangle4의 코드 라인 수는 97이고 도달할 수 있는 패스는 14개, 도달할 수 없는 패스는 0개이다.

4.2 담금질 기법의 세부 매개변수

담금질 기법을 이용한 테스트 데이터 생성 기법에는 성능에 영향을 줄 수 있는 여러 가지 매개변수들이 존재한다. 이러한 매개변수들의 값 설정에 대해서 설명한다.

표 6. 세부 매개변수

매개변수 종류	세부 매개변수
냉각 알고리즘(b=0.001)	$t_{i+1}=t_i/(1+b*t_i)$
데이터 범위	[0..1000], [0..10000]
이웃 선택 방법	기존 이웃 선택, 분기 거리 이웃 선택
이웃 크기	50
적합성 평가 함수	분기 거리
초기 상태 생성 방법	랜덤 생성
초기 온도 설정	50
최대 재시작 횟수	200

표 6은 실험에서 사용한 담금질 기법의 세부 매개변수를 나타낸 것이다. 매개변수 종류로는 냉각 알고리즘, 데이터 범위, 이웃 선택 방법, 이웃 크기, 적합성 평가 함수, 초기 상태 생성 방법, 초기 온도 설정, 최대 재시작 횟수가 있다.

냉각 알고리즘은 [17] 에서 제안한 방법으로 반복 수행됨에 따라 온도가 서서히 낮아진다. 논문에서는 이 식의 b를 0에 가까운 값으로 설정하라고 언급하였고 0.001을 b 값으로 설정하여 실험에서는 수행하였다.

데이터 범위는 생성하는 테스트 데이터 값의 범위를 말한다. 담금질 기법이 데이터 범위가 넓어질 경우 성능이 떨어지는 한계를 극복하고자 제안한 방법이기 때문에 이를 실험하기 위해 0 이상 1000이하의 상대적으로 좁은 범위와 0이상 10000이하의 상대적으로 넓은 범위를 데이터 범위로 설정하였다.

이웃 선택 방법은 기존의 이웃 선택 방법과 새로이 제안한 분기 거리 이웃 선택방법 2가지를 사용하여 실험을 수행하였다

이웃 크기는 새로운 상태를 생성하기 위해 이웃 선택을 수행할 때 선택 기준이 되는 값이다. 이 값은 [3] 에서 사용된 바 있는 50을 적용하였다.

적합성 평가 함수는 테스트 데이터가 얼마나 목표에 가까운지를 나타내는 함수이다. 이것은 3.1. 에서 설명한 분기 거리 방법을 이용하였다.

초기 상태 생성 방법은 초기 테스트 데이터 생성 방법을 말한다. 랜덤 방법을 이용하여 초기 상태를 생성한다.

초기 온도 설정은 담금질 기법에서 목표와 가깝지 않은, 좋지 않은 상태로 전이할 확률을 결정하는 매개변수인 온도의 초기값을 설정하는 것이다. 이 값은 50으로 설정하였다.

최대 재시작 횟수는 담금질 기법이 지역 최적점에 빠지는 것을 방지하기 위해 알고리즘을 재시작하게 하는 수행 횟수를 말한다. 최대 재시작 횟수만큼 반복한 후 담금질 알고리즘을 재시작하게 된다. 이 값은 200으로 설정하였다.

4.3 평가기준

실험의 성능이 우수하다는 것을 보이기 위한 평가기준이 필요하다. 본 연구에서는 생성한 테스트 데이터의 평가를 위해 패스 커버리지를 이용한다.

본 논문은 기존의 담금질 기법, 새로이 제안한 분기 거리 이웃 선택을 이용한 담금질 기법, 랜덤을 이용하여 테스트 데이터를 생성하는 실험을 수행하였다. 알고리즘이 서로 차이가 있기 때문에 알고리즘의 차이 때문에 각 알고리즘의 반복 수행 횟수당 수행 시간이 차이가 날 수가 있다. 따라서 보다 공정한 실험을 위해 커버리지/반복횟수가 아닌 커버리지/시간(초) 차트를 이용해 실험 결과를 표현한다.

또한 탐색 기반 소프트웨어 테스팅 방법은 메타 휴리스틱 알고리즘을 사용하기 때문에 커버리지 달성 시간이 수행에 따라 다를 수 있다. 따라서 탐색 기반 소프트웨어 테스팅 방법의 우연성을 줄이기 위하여 각 알고리즘을 10번 반복 수행하여 커버리지 최대, 최소, 평균 값을 구하여 이를 실험 결과로 분석한다.

4.4 실험 결과

실험 결과는 분기 거리를 이용한 담금질 기법(BD), 기존의 담금질 기법(Basic), 랜덤(Random) 알고리즘을 이용하여 300초 동안 수행하였다. 탐색 기반 소프트웨어 테스팅 기법의 임의성을 줄이기 위해 실험을 10번 반복 수행한다. 반복 수행한 결과의 평균 패스 커버리지를 커버리지/시간(초) 차트로 표현하여 결과를 분석한다.

1) triangle3

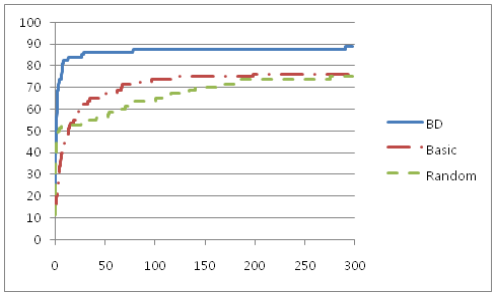

그림 3. triangle3[0..1000]

그림 3은 [0..1000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 모든 구간에서 BD의 성능이 우수하다. 그리고 Basic과 랜덤의 성능 차이가 크기 않은 것을 볼 수 있다.

표 7. triangle3[0..1000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	88.75
Basic	76.25
Random	75

표 7은 [0..1000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. BD가 Basic보다 16.39% 더 높은 최종 커버리지 결과를 보인다. 그리고 BD가 랜덤보다 18.33% 더 높은 최종 커버리지 결과를 보인다.

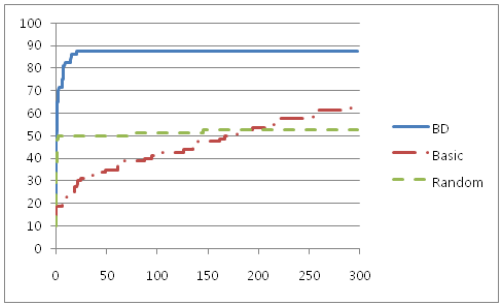

그림 4. triangle3[0..10000]

그림 4는 [0..10000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 모든 구간에서 BD의 성능이 우수하다. 그리고 Basic의 경우 시간의 경과에 따라 커버리지가 지속적으로 증가하여 193초 이후로는 랜덤보다 높은 커버리지를 보여준다. 랜덤은 50%의 커버리지는 매우 빠른 시간에 달성하지만 그 이후로 커버리지 증가 폭이 작아진다.

표 8. triangle3[0..10000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	87.5
Basic	62.5
Random	52.5

표 8은 [0..10000] 데이터 범위를 사용하여 triangle3 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. 분기 거리를 이용한 담금질 기법(BD)이 기존의 담금질 기법(Basic)보다 40% 더 높은 최종 커버리지 결과를 보인다. 그리고 분기 거리를 이용한 담금질 기법(BD)이 랜덤(Random) 알고리즘보다 66.67% 더 높은 최종 커버리지 결과를 보인다.

2) triangle4

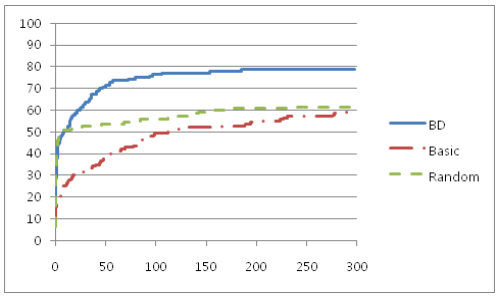

그림 5. triangle4[0..1000]

그림 5는 [0..1000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 대부분의 구간에서 BD의 성능이 우수하다. 그리고 Basic과 랜덤의 성능 차이가 크기 않은 것을 볼 수 있다.

표 9. triangle4[0..1000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	78.57
Basic	59.29
Random	61.43

표 9는 [0..1000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. BD가 Basic보다 32.53% 더 높은 최종 커버리지 결과를 보인다. 그리고 BD가 랜덤보다 27.91% 더 높은 최종 커버리지 결과를 보인다.

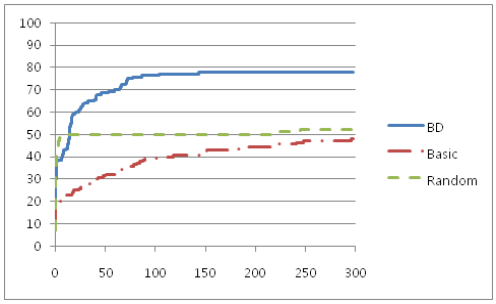

그림 6. triangle4[0..10000]

그림 6은 [0..10000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 결과를 커버리지/시간 차트로 표현한 것이다. 대부분의 구간에서 BD의 성능이 우수하다. 그리고 Basic의 경우 시간의 경과에 따라 커버리지가 지속적으로 증가하지만 랜덤보다 나쁜 성능을 보여준다. 랜덤은 50%의 커버리지는 매우 빠른 시간에 달성하지만 그 이후로 커버리지 증가폭이 작아진다.

표 10. triangle4[0..10000]의 최종 커버리지

알고리즘 종류	최종 패스 커버리지
BD	77.86
Basic	47.86
Random	52.14

표 10은 [0..10000] 데이터 범위를 사용하여 triangle4 프로그램의 테스트 데이터를 자동 생성한 최종 패스 커버리지 결과를 표로 나타낸 것이다. 분기 거리를 이용한 담금질 기법(BD)이 기존의 담금질 기법(Basic)보다 62.69% 더 높은 최종 커버리지 결과를 보인다. 그리고 분기 거리를 이용한 담금질 기법(BD)이 랜덤(Random) 알고리즘보다 49.32% 더 높은 최종 커버리지 결과를 보인다.

3) 실험 결과 분석

위의 실험들을 통해 데이터 범위가 좁을 경우 랜덤의 성능이 우수하고 Basic과 랜덤의 성능이

큰 차이가 나지 않는 것을 볼 수 있다. 그리고 데이터 범위가 넓어질 경우 Basic과 랜덤은 성능이 나빠지지만 BD 는 성능이 크게 변하지 않는 것을 볼 수 있다.

5. 결론 및 향후 연구

기존의 담금질 기법의 한계를 극복하기 위해 새로운 이웃 선택 방법을 제안하고 2가지 프로그램에 대해서 사례 연구를 수행하였다. 새로이 제안한 기법과 기존의 담금질 기법, 랜덤 알고리즘에 대해서 사례 연구를 수행하여 새로이 제안한 기법이 성능이 우수하며 기존의 담금질 기법의 한계로 지적되었던 데이터 범위가 넓어지는 경우에 대해서도 성능이 우수함을 보였다.

향후 연구에서는 좀더 다양한 프로그램에 대해 제안한 기법을 적용해보고 이를 다른 탐색 기반 소프트웨어 테스팅 기법과 비교해볼 계획이다. 또한 담금질 기법의 매개변수의 변화에 따른 성능의 비교를 통해 우수한 성능을 보이는 매개변수를 찾아낼 계획이다.

참 고 문 헌

[1] B. Beizer, Software testing techniques, Van Nostrand Reinhold Co. New York, NY, USA, 1990, .

[2] B. Beizer. Software Testing Techniques. International Thomson Computer Press, 1990.

[3] Man Xiao, Mohamed El-Attar, Marek Reformat, "Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques", Empir Software Eng (2007) 12:183–239

[4] N. Tracey, J. Clark, and K. Mander. Automated program flaw finding using 담금질 기법. In Software Engineering Notes, Issue 23, No.2, Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 1998), pages 73–81, 1998.

[5] C. Michael and G. McGraw. Automated software test data generation for complex programs. In 13th IEEE International Conferance on Automated Software Engineering, pages 136–146, October 1998.

[6] Y. Zhan and J. A. Clark, "The state problem for test generation in Simulink," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '06) Seattle, Washington, USA: ACM, 2006.

[7] N. Tracey. A Search-Based Automated Test-Data Generation Framework for Safety Critical Software. PhD thesis, University of York, 2000.

[8] Korel, B. 1990. Automated software test generation, IEEE Trans. on Software Engineering 16(8): 870–879.

[9] Lin, J-C. and Yeh, P-U. "Automatic Test Data Generation for Path Testing using GAs," Information Sciences, 131: 47–64, 2001.

[10] N. Mansour and M. Salame. Data generation for path testing. Software Quality Journal, 12(2):121–134, 2004.

[11] Michael CC, McGraw G, Schatz MA. "Generating software test data by evolution," IEEE Transactions On Software Engineering 2001; 27(12):1085–1110.

[12] B. F. Jones, H. H. Sthamer, and D. E. Eyres, "Automatic structural testing using 유전 알고리즘s," Software Engineering Journal, vol. 11, pp. 299–306, 1996

[13] P. M. S. Bueno and M. Jino, “Identification of potentially infeasible program paths by monitoring the search for test data,“ in Proceedings of the fifteenth IEEE international conference on Automated Software Engineering (ASE ’00) 2000, pp. 209-218.

[14] N. Tracey, J. Clark, K. Mander, and J. McDermid. Automated test data generation for exception conditions. Software - Practice and Experience, 30(1):61-79, 2000.

[15] R. Sagarna and J. A. Lozano, “Scatter search in software testing, comparison and collaboration with estimation of distribution algorithms,“ European Journal of Operational Research, vol.169, no.2, pp.392-412, 2006.

[16] Sagarna R., and Lozano J. A., “On the performance of Estimation of Distribution Algorithms applied to software testing,” Applied Artificial Intelligence, vol.19 no.5, pp.457-489, 2005.

[17] Lundy M, Mees AI (1986) Convergence of an annealing algorithm. Math Program 34(1):111-124

저 자 소 개

최 현 재

2009년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사)
2010년 현재 부산대학교 대학원 컴퓨터공학과 석사 과정.

<관심분야> 소프트웨어공학, 테스팅 자동화

이 선 열

2008년 부산대학교 전자전기정보컴퓨터공학과 졸업(학사)
2010년 부산대학교 대학원 컴퓨터공학과(공학석사)
2010년~현재 부산대학교 대학원 컴퓨터공학과 박사과정.

<관심분야> 소프트웨어공학, 소프트웨어 테스팅 자동화, 테스트 데이터 자동 생성

채 흥 석

1994년 서울대 원자핵공학 학사
1996년 한국과학기술원 전산학 석사
2000년 한국과학기술원 전산학 박사
2000년~2003년 ㈜ 동양시스템즈 기술연구소 선임연구원

2003년~2004년 한국과학기술원 전산학과 초빙교수
2004년~현재 부산대학교 컴퓨터 공학과 부교수.

〈관심분야〉 객체지향 방법론, 소프트웨어 테스팅, 소프트웨어 메트릭, 소프트웨어 유지보수, 미들웨어 설계, 프로덕트라인 공학

스마트폰 탑재용 사용자 컨텍스트 데이터 경량화 기법†

(A Method for Reducing Users Context Data on Smart Phone)

김지훈‡ 박창규§ 이정원¶
(Jihoon Kim) (Changgyu Bak) (Jungwon Lee)

요 약 최근 스마트폰이 점차 대중화되고 탑재되는 센서의 발달로 폰을 통해 획득할 수 있는 데이터의 종류가 다양해지고 있다. 이로 인해 스마트폰을 통해 사용자의 상황을 알아내는 상황인지 기술과 이를 활용하는 연구들이 활발히 진행되고 있다. 그러나 수집된 데이터를 통해 의미 있는 정보를 가공하기 위해서는 복잡한 처리 과정과 정보 분석 과정이 필요하며, 이러한 가공과정은 일부는 스마트폰에서, 또 일부는 서버에 대량의 데이터를 전송하여 처리하고 있는 실정이다. 본 논문에서는 모바일 플랫폼에서 생성 가능한 컨텍스트를 정의하고 수집하기 위한 컨텍스트 생성기를 설계한다. 이는 컨텍스트를 폰 안에서 생성함으로써 서버로 전송하는 데이터 용량을 감소시킬 수 있고 통신비용을 줄일 수 있다.

키워드 스마트폰, 모바일 컨텍스트, 상황인지 시스템, 컨텍스트 생성

Abstract Recently, smart phones are widely used and as the phone sensors are evolved, we can get more various kinds of data from them. Using the data, many researchers take an effort to aware user's situation and the context-awareness for smart phones are actively applied to the real-world applications. However, to make an advanced information from sensing data need complex processing and analyzing information. Some of these computing processes are fully handled on smart phone or some data are processed by transferring to server. In this paper, we proposed a method for reducing user's context data generated on smart phone and designed a context generator. As a result, we can reduce a transmission data size and save a communication cost.

Key words Software reuse, Software Reuse-related Knowledge, Web 2.o-based Software Engineering

1. 서 론

스마트폰 상황인지 기술은 스마트폰으로부터 얻을 수 있는 정보들을 활용하여 사용자의 상황을 인지하고 적절한 서비스를 제공하는 기술이다. 최근 스마트폰 사용의 급격한 증가와 더불어 상황인지 기술에 대한 연구도 활발해지고 있다. 이러한 스마트폰의 상황인지는 이를 최종적으로 이용하는 목적에 따라 다양한 방법으로 이루어질 수 있다[1]. 예를 들어 Sound-Sense는 마이크로폰을 통해 얻은 데이터를 인간이 인식할 수 있는 형태의 데이터로 분류하며, VTrack은 차량 운행시 운전자의 GPS 데이터를 이용하여 목적지까지의 추천 루트와 예상 시간을 제공해주기도 한다[2,3].

그러나 스마트폰의 저장 공간과 메모리, 프로세스의 성능이 대폭적으로 향상되고 있긴 하지만

† 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2011-0005305).
‡ 비 회 원 : 아주대학교 전자공학부
jihoony85@ajou.ac.kr
§ 학생회원 : 아주대학교 전자공학부
pck0912@ajou.ac.kr
¶ 종신회원 : 아주대학교 전자공학부 교수
jungwony@ajou.ac.kr
논문접수 : 2011년 05월 18일
심사완료 : 2011년 06월 17일

이렇게 사용자의 상황을 인지하기 위해서는 복잡한 분석 과정이 필요하고 이러한 과정을 처리하기에 PC에 비해 성능이 제한적이며 배터리 한계 또한 문제점이 되고 있다.

본 논문에서는 스마트폰상의 어플리케이션을 활용한 지능형 서비스 제공을 위하여 데이터 수집 과정에서 센서 데이터로부터 1차적인 컨텍스트를 생성하는 모듈을 폰 안에 탑재, 처리함으로써 데이터 용량과 전송량, 통신비용을 줄일 수 있는 컨텍스트 생성기를 제안한다.

본 논문은 다음과 같은 순서로 구성된다. 2장에서는 관련 연구를 살펴보고 3장에서는 제안하는 컨텍스트 생성기를 설계하며 폰 안에서 컨텍스트 생성시 데이터 용량을 줄일 수 있는 방법을 제안한다. 4장에서는 이를 기반으로 실제로 구현하고, 마지막으로 5장에서는 결론 및 향후 연구를 제시한다.

2. 관련 연구

상황인지 휴대폰 기술개발 동향을 살펴보면 모바일 디바이스에서의 추론 및 학습을 두 가지 형태로 구분하고 있다. 하나는 휴대폰에서 직접 정보를 수집하여 이를 서버에 원격으로 보내고 서버에서 추론과 학습을 통해 얻어낸 결과를 휴대폰으로 전송하는 Streaming Intelligence라 부르는 기술이다[4]. 또 다른 하나는 추론 및 학습과 관련된 모듈을 휴대폰 내부에 탑재하여 수집되는 데이터를 자체적으로 처리하는 방식이다.

원격 서버를 이용할 경우 폰 자체의 계산량과 메모리의 제한은 없지만 통신에 접속해 있어야 된다는 점과 실시간 처리가 불가능하다는 단점이 있다. 이와 관련된 연구로 상황인지 모바일폰 연구인SenSay는 가속도, 조도, 온도, 마이크로폰 센서로 입력 받은 데이터를 통하여 사용자의 상태를 uninterruptible, active, idle, normal로 구분하여 인지한다. 이렇게 입력 받은 데이터와 인지한 상태를 통해 모바일폰의 수신모드를 자동으로 변경한다. 이 연구에서는 센서로부터 받는 저수준의 데이터를 바로 사용자의 상태를 인지하는데 사용하여 처리 과정은 간단하지만 고수준의 정보를 알아내기에는 부적합한 구조이다. 실제로 사용자가 대화 중일 때 uninterruptible 상태로 인지가 되어야 하는데 마이크로폰으로부터 들어오는 값으로는 대화나 소음을 구분하기 어렵다. 또한 폰에서 고려될 수 있는 상황의 조건이 한정적인 것도 한계라고 할 수 있다[5].

모바일폰에서 자체적으로 처리하는 경우는 추가적인 통신비용은 발생하지 않지만 추론과 학습에 소요되는 계산량과 메모리 사용에 제약을 받을 수 있다. 서버에서 데이터를 처리하는 상황인지 기술의 한 예인 Ubifit Garden은 MSP (Mobile Sensing Platform) 장비를 통해 획득한 센서 데이터를 이용하여 사용자의 운동량 등을 파악한다. 이렇게 얻은 정보를 폰의 어플리케이션을 통해 사용자가 볼 수 있게 하고 어플리케이션 내의 정원이 존재하여 운동량에 따라 꾸며지게 함으로써 피드백을 줄 수 있는 시스템을 개발하였다. 이 연구는 사용자의 운동량을 MSP 를 이용하여 자세히 알아낼 수 있고 데이터의 처리까지 하므로 폰에서의 부담이 줄어들긴 하지만 추가적인 센싱장비가 필요할 뿐 아니라 MSP와 휴대폰이 블루투스 통신을 이용하여 계속적으로 연결이 되어 있어야 한다는 단점이 있다[6].

이처럼 상황인지 휴대폰 연구에서 센싱되는 데이터를 가공하는 과정에는 폰의 처리능력과 통신을 함으로써 발생하는 단점들을 고려하여 시스템을 설계하는 것이 필요하다.

3. 상황인지를 위한 컨텍스트 생성기

본 논문에서 제안하는 컨텍스트 생성기는 정보를 수집하고 1차적인 컨텍스트를 생성하는 과정이 스마트폰 안에서 이루어진다. 이러한 시스템을 구성하는데 고려해야 할 조건은 모바일이라는 특성상 프로세스와 저장 공간의 제약이 따르므로 데이터 처리 과정을 간소화 하는 것과 저장되는 데이터 용량을 줄이는 것이다.

컨텍스트 생성기를 설계하기 위해서는 생성할 컨텍스트의 목록을 정의하고 스마트폰에서 수집되는 정보를 가공하여 컨텍스트를 생성하는 과정을 거쳐야 한다

3.1 컨텍스트 모델 정의

사용자의 상황을 인지하기 위해서는 폰으로부터 얻는 저수준의 데이터를 상황정보로 가공하는 과정을 거쳐야 한다. 본 논문에서 제안하는 컨텍스트 모델은 이 과정을 그림 1과 같이 세가지 레벨로 분류하여 추론하는 과정을 거친다.

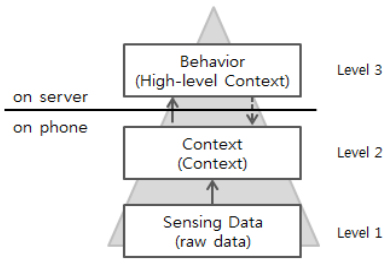

그림 1 모바일 컨텍스트 계층

- Level 1 : 센싱 데이터 레벨은 스마트폰에 존재하는 다양한 센서들을 통해 직접 받아들이는 저수준의 데이터를 의미하는 단계이다.
  예) GPS 위도 : 37 ° 16′ 55 ″ N,
  경도 : 127 ° 2′ 35 ″ W
- Level 2 : 두 번째 레벨은 센싱되는 데이터로부터 의미 변환 과정을 거쳐 컨텍스트가 생성되는 단계이며 이 과정까지 폰 안에서 처리가 된다.
  예) 집, 직장
- Level 3 : 세 번째 레벨은 behavior 컨텍스트로서 모바일에서 서버로 넘겨줄 컨텍스트 로그 데이터를 이용하여 추론 과정을 거쳐 알아낼 수 있는 컨텍스트이다.
  예) 수면, 출근

그림 1에서 구분한 선은 폰에서 처리되는 단계와 서버에서 처리되는 단계를 구분한 것이다. 기존의 연구에서는 behavior 컨텍스트의 추론이 센싱된 데이터를 이용하여 서버에서 이루어지는 것이 일반적이다. 본 논문에서 제안하는 구조에서는 센싱된 데이터로부터 의미 있는 컨텍스트를 생성하는 과정을 폰 안에서 처리하고 이를 서버로 보내 추론하는 과정을 거치게 된다. 실선으로 표시된 화살표는 데이터가 가공되는 흐름을 나타내며 점선으로 표시된 화살표는 서버에서 추론된 결과물들을 모바일에 적용시키기 위해 전송되는 데이터의 흐름을 나타낸다.

이렇게 중간 단계를 폰 안에 두게 되면 4장에서 제안하는 수집 기법을 통하여 폰에서 서버로 넘어가는 데이터의 용량을 줄일 수 있게 된다. 또한 실시간으로 생성되는 컨텍스트를 활용하여 이들의 조합이나 규칙 등을 통해 모바일 서비스를 제공하는 시스템에도 응용할 수 있다.

본 논문에서는 세가지 단계로 구분한 추론 과정 중 스마트폰 안에서 처리되는 2단계 레벨의 컨텍스트 생성 및 저장까지를 연구범위로 한다.

3.2 센싱 데이터

스마트폰을 통해 수집하는 데이터의 종류를 정하는 과정은 컨텍스트의 종류를 분류하고 컨텍스트 생성에 필요한 센서를 정하는 과정으로 이루어진다. 그림2와 같이 사용자의 상황을 나타내는 정보를 Behavior, Time, Space, Person, Environment, Service의 여섯가지 컨텍스트로 분류하였고 이 때 필요한 센서의 종류를 오른쪽에 표시하였다.

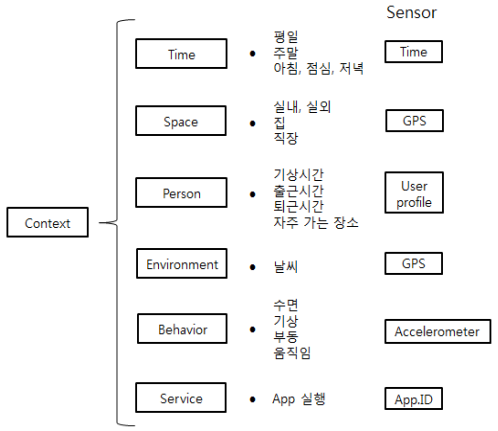

그림 2 모바일 컨텍스트 분류 및 컨텍스트 생성에 필요한 센서

- Time : 서비스 선택 시 고려해야 할 시간적 요소. 폰의 시스템의 시간정보로 생성
- Space : 사용자가 있는 공간에 대한 정보GPS 데이터를 이용해 생성
- Person : 서비스 선택 시 고려해야 할 사용자에 대한 개인적인 정보. User의 입력이나 추론과정을 거쳐 생성
- Environment : 환경정보. GPS 데이터를 이용하여 생성
- Behavor : 사용자의 행위 정보. 가속센서 및 추론 과정을 거쳐 생성
- Service : 어플리케이션 실행 정보. 폰의 시스템을 통해 생성

3.3 컨텍스트 생성

스마트폰의 센서를 통해 수집된 데이터는 가공 과정을 거치지 않으면 의미 없는 숫자 값에 불과하다. 스마트폰에 존재하는 각각의 센서에 따라 데이터의 범위와 기준이 다르므로 이를 고려하여 의미변환 과정을 거쳐야 한다. 표 1은 스마트폰 센서 종류와 데이터 값의 예시가 나타나있다.

표 1 센서 종류와 데이터 예시

센서	데이터 값	데이터 예시
가속도	X축 가속도	-0.2413m/s2
	Y축 가속도	1.6042m/s2
	Z축 가속도	10.3197m/s2
조도	광량	100lux
마이크로폰	소리 크기	30dB
GPS	Latitude	37 ° 16′ 55 ″ N
	Longitude	127 ° 2′ 35 ″ W
	Speed	23.28Km/h
	Accuracy	1800(수치)
스마트폰 정보	시간	yyyy-mm-dd HH:mm:ss
	Mode	0,1,2
	App. ID	String

표2는 표1의 데이터를 가공하여 사람이 인식할 수 있을만한 데이터 형태로 변환하는 과정을 나타낸다.

표 2 저수준 데이터로부터 컨텍스트 생성 과정

센서	의미변환	컨텍스트
가속도	A B Low ¦ Medium ¦ High	P_movement
조도		Illuminance
마이크로폰		Noise
GPS	Latitude Longitude → API → Address / Weather	Address
		Weather
	A Indoor ¦ Outdoor	Site
스마트폰 정보	변환 없이 그대로 사용	GPS_speed
		Time
		Mode
		Application

- 가속도 : 가속 센서를 통해 획득할 수 있는 데이터는 x, y, z축의 가속도 크기 값이다. 이 값의 변화량을 구하면 핸드폰 움직임의 크기를 알아낼 수 있는데 이 크기는 일정한 기준이 제시되어있지 않으므로 본 연구에서 임의로 기준 A, B를 설정하였다. 이 기준을 통해 데이터가 Low, Medium, High로 나뉘고 이 값이 P_movement(Phone movement)의 형태로 컨텍스트에 저장된다.

- 조도 : 조도센서는 데이터의 수치가 lux 형태로 획득된다. 이에 대한 구분은 국제 조도 기준(CIE)의 A(~200lux), B(300~2000lux), C(3000~ 20000lux)의 분류를 참고하였으며 세 단계로 나뉘어 Illuminance의 컨텍스트에 저장된다.
- 마이크로폰 : 마이크로폰을 통해서는 소리의 크기가 dB 형태로 측정된다. 이에 대한 구분은 환경운동연합에서 제시한 소음 기준표를 참고하여 30dB과 85dB로 기준을 정하였으며 Noise 컨텍스트에 저장된다.
- GPS : GPS로부터는 위치 정보를 GPS 위도와 경도의 수치 데이터로 알아낼 수 있으며 이동속도에 따라 변하는 GPS 속도 데이터와GPS가 얼마만큼의 정확도로 감지되는지의 여부에 따른 정확도 데이터를 수집할 수 있다. GPS의 위치정보를 구글 등에서 제공하는 API를 이용하여 주소와 날씨 정보로 받아올 수 있으며 이를 각각 Address와 Weather의 컨텍스트에 저장한다. GPS 정확도 수치에 따라 실내인지 실외인지를 구분하며 이 값 역시 임의적인 실험을 통하여 정하였다. GPS 속도는 실외에서 사용자의 이동 속도를 나타내는 수치로 이를 컨텍스트에 저장한다.
- 스마트폰 정보 : 폰의 시스템에서 획득할 수 있는 데이터로 데이터가 수집될 때의 시간과 폰의 모드, 현재 실행중인 애플리케이션의 이름을 수집하여 컨텍스트로 사용할 수 있다.

3.4 데이터 경량화 기법

데이터를 수집하고 가공하는 모든 과정이 폰 안에서 이루어지면 처리 과정이 간소해야 하며 수집되는 데이터의 용량을 줄이기 위한 노력들이 필요하다. 스마트폰 안에서 컨텍스트를 생성하면 다음에서 제안하는 이벤트 기반 컨텍스트 수집 기법과 moving average를 이용한 방법을 통해 데이터 용량의 크기를 줄일 수 있다.

3.4.1 이벤트 기반 컨텍스트 수집

기본적으로 스마트폰에서 센서를 통해 데이터를 수집하는 방법은 데이터가 측정 되는대로 수집되는 스트리밍 방식이다. 스마트폰 안에서 컨텍스트를 생성하면 컨텍스트의 변화에 따라 데이터를 수집하는 이벤트 기반 방법이 가능해진다.

표 3의 예시를 보면 그 차이를 확인할 수 있다. 표에서 Low, Medium, High를 구분하는 기준은 A, B의 수치이다. 기존의 방식이 센서에서 데이터가 측정되는 대로 세가지 상태를 구분하여 저장을 한다면 이벤트 기반 방식은 컨텍스트의 변화가 있을 때만 저장을 하는 방법이다. 이렇게 저장을 하면 컨텍스트의 변화가 있기 전까지 같은 컨텍스트로 판단할 수 있기 때문에 일정 시점에 대한 컨텍스트 정확도가 기존 방식과 동일한 상태로 저장 횟수를 줄일 수 있다. 표3은 총 10개의 입력 데이터 중 3번만 저장이 되어 70% 감소된 것을 알 수 있다. 일반적으로 일상생활에서 폰의 환경이 급작스럽게 바뀌는 경우는 많지 않기 때문에 컨텍스트가 저장되는 데이터 용량을 줄이는데 효과적이다.

표 4. 스트리밍 기반 컨텍스트와 이벤트 기반 컨텍스트의 저장 횟수 비교

A = 3, B = 7

순서	1	2	3	4	5	6	7	8	9	10
데이터	9	8	8	6	4	5	4	2	1	2
스트리밍	H	H	H	M	M	M	M	L	L	L
이벤트	H	•	•	M	•	•	•	L	•	•

3.4.2 Moving Average

4.1과 같은 방식에서 이벤트가 발생함에 따라 데이터를 저장하게 되면 경계부분에서 데이터 값이 트리거 되는대로 저장된다. 이를 moving average 기법을 사용하여 데이터 저장 횟수를 줄일 수 있다.

이동 평균은 추세의 변동을 알 수 있도록 구간을 옮겨 가면서 구하는 방법을 말한다. 주식

시장에서 거래액, 매매대금, 주가 등의 현상을 나타내는 지표의 하나로 사용되며 신호처리 분야에서도 moving average filter와 같은 형태로 사용되는 개념이다[7].

다음 식 (1)은 moving average 의 개념을 수식으로 나타낸 것이다. MA는 moving average를 뜻하며 p는 period, Dm은 현재의 데이터를 나타낸다. 즉, 현재 데이터에서 이전 p의 간격만큼의 데이터를 모두 더해 평균을 내어 그 시점의 데이터로 사용하는 방법이다.

$$MA = \frac{D_m + D_{m-1} + \cdots + D_{m-p+1}}{p} \quad (1)$$

이 개념을 컨텍스트 수집에 적용하면 p를 평균을 구할 데이터의 개수로, D를 센서에서 측정되는 각각의 데이터로 정할 수 있다.

이러한 moving average의 방법은 구간의 평균값을 사용하는 것이기 때문에 이전 데이터 값들이 현재 컨텍스트를 결정하는데 영향을 주게 되므로 데이터 하나의 크기변화에 둔감한 성질을 가지고 있다.

Moving avearage의 방법을 적용할 수 있는 컨텍스트는 일정한 기준을 통해 컨텍스트를 분류하였던 P_movement, Illuminance, Noise, Site 값이 있다.

4. 구현 및 실험결과

설계한 모듈로부터 실제 컨텍스트를 수집하기 위해 다음과 같은 환경을 구성하였다.

스마트폰은 삼성의 갤럭시S2(안드로이드 2.3, CPU 1.2Ghz, RAM 8Gb)를 사용하였으며 가속도, 조도, 마이크로폰, GPS 센서와 폰의 시스템 정보에서 데이터를 수집하였다.

그림 4와 같이 우선 컨텍스트 수집 목적을 일반 회사원의 Daily life를 로깅하는 목적으로 예상 시나리오를 작성하였으며 이를 토대로 시나리오 내용에 기반한 Context Set을 정하였다. 폰을 통해 저수준의 데이터를 수집하여 3.3장의 의미변환 과정을 거쳤으며 moving average의 방법은 실제 폰에서 가장 변화가 자주 일어나는 컨텍스트인 P_movement에 임의의 p값 5를 사용하였다. 이벤트 기반으로 컨텍스트를 수집하여 폰 내부의 SQlite 데이터베이스에 저장 하였다. 마지막으로 생성된 DB 파일은 한 시간마다 서버에 전송하여 저장되도록 구현하였다.

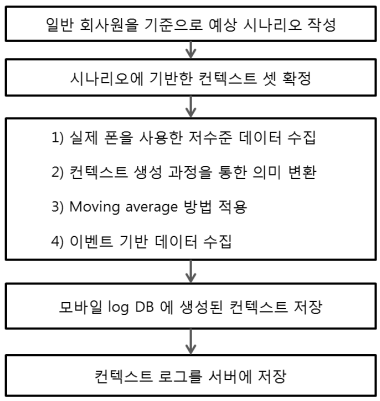

그림 3 구현 과정

그림 5는 컨텍스트 생성기를 알고리즘으로 나타낸 것이다.

```
procedure Context_generating    // input : sensor data and system information
return    Context_Set           // output : context set
begin
  while(D: each data among sensors or phone system) do                    // ①
    Last_context_set = Context_set;
    begin
      case 1 :                                                            // ②
        if(D < A)
          (# of output parameter of Context_set) = Low;
        else if(A < D < B)
          (# of output parameter of Context_set) = Medium;
        else
          (# of output parameter of Context_set) = High;
      case 2 :                                                            // ③
        if(D < A)
          (# of output parameter of Context_set) = Indoor;
        else
          (# of output parameter of Context_set) = Outdoor;
      case 3 :                                                            // ④
          (# of output parameter of Context_set) = API(D);
      case 4 :                                                            // ⑤
          (# of output parameter of Context_set) = phone_system;
      case 5 :                                                            // ⑥
          (# of output parameter of Context_set) = D;
    if(Last_context_set != Context_set)                                   // ⑦
    return Context_set;
  end
end
```

그림 6 컨텍스트 생성 알고리즘

우선 입력 받는 데이터(①) 처리방식에 의해 다섯 가지로 구분한다. case 1은 두 가지 기준으로 나누어 Low, Medium, High로 저장되는 컨텍스트 데이터 처리과정이며 (②), case 2는 GPS 정확도 값에 따라 Indoor, Outdoor로 나뉘는 과정을 나타낸다(③). case 3은 GPS 데이터를 이용하여 구글 API를 통해 정보를 받아오는 과정을 나타내며(④), case 4는 스마트폰 시스템을 통해 정보를 알아오는 과정을 나타낸다(⑤). 마지막으로 입력 받은 데이터를 그대로 Context Set에 저장하는 과정이 case 5이며(⑥) 이렇게 저장된 Context Set이 이전에 저장된 Context Set과 다를 경우 Output으로 출력된다(⑦).

이렇게 구현한 컨텍스트 생성기로 P_movement의 데이터 값을 이용하여 저장횟수가 얼마나 줄어드는지 확인하였다. 그림 6은 입력 받은 P_movement의 데이터를 그래프로 표시한 것이다. 가로축은 데이터가 측정되는 횟수를 의미하며 세로축은 P_movement 의 크기를 의미한다. P_movement의 측정 주기는 10Hz로 하였으며 moving average 방법을 적용한 데이터 값이다. 굵은 선을 기준으로 Low, Medium, High가 나뉘어지며 실제 저장되는 횟수는 event 기반으로 아래 표시된 화살표 부분에서 이루어진다. 총 70개의 데이터 중 13번이 저장되어 82% 감소된 것을 확인할 수 있다.

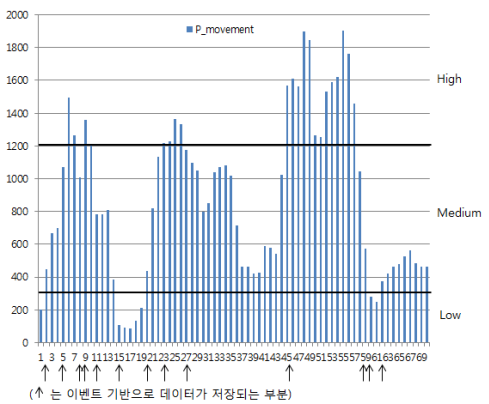

그림 6 P_movement 데이터 수집주기 비교

5. 결론 및 향후 연구

본 논문에서는 스마트폰으로부터 얻을 수 있는 데이터를 가공하여 1차적인 컨텍스트를 생성하고 저장되는 데이터 용량을 경량화 시키기 위한 컨텍스트 생성기에 대해 소개하였다.

본 논문에서 연구한 방법을 적용한 상황인지 시스템은 폰에서 수집되는 데이터의 용량을 효율적으로 줄일 수 있다는 점과 이로 인한 서버와의 통신 비용을 감소시킬 수 있다는 점에서 의의를 가질 수 있다.

본 논문에서 제안한 모듈을 포함한 연구가 현재 진행되고 있으며 안드로이드 기반 스마트폰에서 구현하고 있다. 추후 연구진행을 통해 로깅되는 컨텍스트를 활용하여 서비스를 제공하는 실제 구현된 시스템에서의 효용성에 대해 살펴보는 것은 향후 연구에서 진행할 계획이다.

참 고 문 헌

[1] Y.-S. Lee and S.-B. Cho, “Context-awareness support for mobile phones: Integrative method and applications,“ SK Telecom Telecommunications Review, vol. 19, no. 6, pp. 846-857, 2009. 12.

[2] H. Lu et al., “Sound-Sense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones,” Proc. 7th ACM MobiSys, 2009, pp. 165-78.

[3] Thiagarajan et al., “VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones,” Proc. 7th ACM SenSys, Berkeley, CA, Nov. 2009.

[4] 김경중, 조성배, “상황인지 휴대폰 기술개발 동향”, IITA주간기술동향, pp. 26-33, 2007.1.24.

[5] D.P. Siewiorek, A. Smailagic, J. Furakawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F.L. Wong, "SenSay: A context-aware mobile phone," Proc. 7th Int. Symp. of Wearable Computers, pp. 248-249, Oct. 2003.

[6] S. Consolvo et al., "Activity Sensing in the Wild: A Field Trial of Ubifit Garden," Proc. 26th Annual ACM SIGCHI Conf. Human Factors Comp. Sys., 2008, pp. 1797-1806.

[7] Rangarao, Kaluri Venkata, "Digital Signal Processing," John Wiley & Sons Inc., 2007

저 자 소 개

김 지 훈

2011년 아주대학교 전자공학부 졸업(학사).
2011년~현재 아주대학교 전자공학부 석사과정.

〈관심분야〉 유비쿼터스 컴퓨팅, 임베디드 소프트웨어, 모바일 컴퓨팅

박 창 규

2007년 아주대학교 전자공학부 졸업(학사).
2007년~현재 아주대학교 전자공학부 석·박사통합 과정.

〈관심분야〉 프로그래밍언어, SOA, 컴퓨터 아키텍쳐, 임베디드 소프트웨어 등

이 정 원

1993년 이화여자대학교 전자계산학과 졸업(학사).
1995년 이화여자대학교 전자계산학과 졸업(석사).
1995년~1997년 LG종합기술원 주임연구원.
2003년 이화여자대학교 컴퓨터학과 졸업(박사).
2003년~2006년 이화여자대학교 컴퓨터학과 BK 교수, 전임강사(대우).
2006년~현재 아주대학교 전자공학부 조교수.

〈관심분야〉 유비쿼터스 컴퓨팅, SOA, 임베디드 소프트웨어 등

저전력 임베디드 소프트웨어 개발을 위한 재사용 컴포넌트의 전력소모 특성 명세 방법†

(Energy Characteristic Specification Method of Reusable Component for Energy Efficient Embedded Software Development)

김두환‡ 이재욱§ 홍장의¶
(Doohwan Kim) (Jae-Wuk Lee) (Jang-Eui Hong)

요 약 컴포넌트 기반 개발 방법론은 개발 편의성, 개발에 소요되는 시간, 비용 등의 이점으로 인해 그 적용 영역이 확대되고 있다. 특히 일반적인 소프트웨어에 비해 플랫폼 의존성이 높고, 제품 군(product family) 개발이 가능한 임베디드 소프트웨어의 경우 컴포넌트 기반 개발은 그 유용성이 매우 높다고 할 수 있다. 그 중에서도 스마트 폰, 태블릿 PC 등과 같은 휴대형 임베디드 시스템의 경우, 전력 소모량이 적은 소프트웨어 개발의 중요성이 부각되고 있기 때문에 재사용 컴포넌트의 소모 전력에 대한 특성은 컴포넌트 기반 개발에서 고려되어야 할 중요한 품질요소 중의 하나가 되었다. 본 연구에서는 재사용 기반의 임베디드 소프트웨어 개발 시, 컴포넌트에 대한 소모전력 특성을 고려할 수 있도록 지원하기 위하여 소모전력 특성을 포함하는 컴포넌트 명세언어를 제안한다. 이는 추후 컴포넌트 저장소에서 소모전력 특성을 고려하는 컴포넌트 검색 및 선택 등과 같은 영역에서 활용할 수 있다.

키워드 컴포넌트, 컴포넌트 명세, 전력 분석

Abstract Component-based Software development(CBSD) is widely used in various area due to its efficiency of time, cost and effort. In the embedded software which has high dependency of platform and can be developed by product family, the efficiency of CBSD is maximized by reuse. These embedded software has various limitations of the resources. Specially, the effective energy consumption is very important in the portable embedded software such as smart phone and tablet PC, because they are operated with limited energy source like a battery. Therefore, energy efficient problem became very important issue in the CBSD. In this paper, we identified characteristics and environment that influence energy consumption of components. Afterward, we defined a component specification language which is consisted to describe energy characteristics of the components. This supposed specification language can be utilized to energy efficient component search and selection.

Key words Component. Component Specification, Energy analysis

1. 서 론

컴포넌트 기반 소프트웨어 개발 방법론은 기존에 개발된 재사용 컴포넌트들을 이용하여 소프트웨어를 개발하는 방법으로 개발 편의성 향상, 비용절감, 시간단축 등의 이점을 제공한다. 특히 임베디드 소프트웨어 개발에 있어서는 제품 간에 유사한 기능을 포함하는 경우가 많으며, 개발

† 이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2011 0010396)
‡ 학생회원 : 충북대학교 전자정보대학 컴퓨터과학과
dhkim@selab.cbnu.ac.kr
§ 학생회원 : 충북대학교 전자정보대학 컴퓨터과학과
jwlee@selab.cbnu.ac.kr
¶ 종신회원 : 충북대학교 전자정보대학 컴퓨터과학과 부교수
jehong@chungbuk.ac.kr
논문접수 : 2011년 06월 02일
심사완료 : 2004년 06월 24일

기간이 다소 짧다는 특성 등으로 인해 컴포넌트 기반 개발의 도입이 지속적으로 늘어나고 있다. 이러한 임베디드 소프트웨어는 일반적으로 가용 전력량, 메모리 등과 같은 자원에 대한 제약을 갖는다[1]. 이와 같은 특성들을 통해 임베디드 소프트웨어 개발 시에는 신속한 의사결정과 다양한 비기능적 요소들에 대한 고려가 필수적으로 요구된다. 특히 휴대형 임베디드 시스템은 배터리를 통해 제한된 전력을 공급받기 때문에 소모 전력량에 대한 비기능적 요구사항이 중요하다. 저전력 소모를 위한 소프트웨어 개발에서 우선적으로 수행해야 하는 것은 소프트웨어의 정량적 소모전력 분석이며, 일반적으로 소프트웨어의 소모 전력량 분석은 그것의 행위를 기술하는 모델을 이용하여 수행 가능하다. 이는 소프트웨어가 단독적으로 전력 소모를 발생시키는 것이 아니라, 그것이 실행 될 때 하드웨어의 동작을 유발시킴으로써 간접적인 전력 소모를 일으키기 때문이다[2]. 다시 말해 소프트웨어의 단위 행위는 그에 해당하는 일련의 하드웨어 동작을 나타내고, 이때 하드웨어의 동작을 통해 전력 소모가 발생된다.

소프트웨어 컴포넌트는 기능적 행위를 포함하는 구현된 코드이다. 컴포넌트 기반으로 임베디드 소프트웨어를 개발하기 위해서는 가급적 동일 기능을 갖지만 소모전력량이 적은 컴포넌트를 개발하거나 또는 재사용 컴포넌트를 선택할 필요성이 있다. 따라서 재사용 컴포넌트의 명세에 컴포넌트가 소모할 전력량을 포함하도록 명세하는 것이 필요하다.

컴포넌트 명세는 해당 컴포넌트의 정보를 제공하여 그것의 재사용을 지원하도록 하는 역할을 수행한다. 즉, 컴포넌트 명세는 컴포넌트의 이용, 검색 등에 있어서 아주 중요한 일종의 이력서와도 같다. 기존의 컴포넌트 명세에 있어서는 일반적으로 컴포넌트가 제공하는 기능과 이들 기능에 대한 정형적 인터페이스 명세만이 중요한 고려사항이 되었다[3]. 그러나 이제는 앞서 설명한 바와 같이 소모전력 특성을 컴포넌트 명세에 반영해야 한다. 소프트웨어의 소모 전력은 그것의 행위와 밀접한 관계를 갖는다. 따라서 컴포넌트 명세를 이용한 전력소모 특성 명세를 위해서는 각 컴포넌트에서 제공하는 인터페이스에 따라 관리하는 것이 합당하다.

본 연구는 컴포넌트 기반의 소모 전력량 분석을 지원하기 위해 컴포넌트 명세에서 각 컴포넌트에 대한 전력 소모 특성을 기술하기 위한 방법을 제시한다. 제안된 컴포넌트 명세 방법을 통해 얻을 수 있는 이점은 다음과 같다.

- 전력소모 특성을 고려한 컴포넌트의 체계적 관리 지원 가능
- 전력 효율적인 컴포넌트 선택을 지원하기 위한 가이드라인 제시
- 컴포넌트 기반 소프트웨어 아키텍처 상에서 전력 소모량 분석 지원 가능

본 논문은 다음과 같이 구성된다. 2장에서는 관련연구를 통해 기존의 컴포넌트 명세 기법들에 대해 소개하고, 3장에서는 컴포넌트 명세를 위해 요구되는 정보들을 살펴본 후 본 연구에서 제시하는 컴포넌트 명세 언어를 정의한다. 4장에서는 정의된 명세 언어의 적용을 살펴보고, 5장에서 이에 대한 활용 방안에 대해 논의한다. 마지막으로 6장에서는 결론 및 향후 연구에 대해 언급한다.

2. 관련연구

컴포넌트 기반 소프트웨어 개발을 수행하는 조직에서는 개발에 참여하는 모든 사람이 쉽게 컴포넌트에 접근할 수 있도록 전사적 차원에서 관리되는 컴포넌트 라이브러리가 필요하다.

이때, 라이브러리는 컴포넌트뿐만 아니라 그것의 사용을 지원할 수 있도록 해당 컴포넌트에 대한 정보들을 포함한다. 때문에 컴포넌트에 대한 정보를 효율적으로 관리할 수 있도록 하는 컴포넌트 명세의 구성에 대한 연구들이 진행되어 왔다[4-6]. 각 연구에서 제안하고 있는 컴포넌트 명세 기법은 일반적으로 요구되는 컴포넌트에 대한 정보들이 유사하게 포함되어 있는 반면 해당 개발 조직에서 주로 개발하는 소프트웨어의 도메인이나 특성, 정책 또는 가치에 따라 서로 상이한 정보를 포함하기도 한다. 이러한 연구들의 대표적인 사례는 RAS(Reusable Asset Specification)[4]와 X-ARM[5]이 있다.

RAS는 OMG에서 소프트웨어 컴포넌트뿐 아니라 개발주기 전반에서 발생하는 재사용 가능한 자산에 대한 명세를 지원하기 위해 발표하였다. RAS에서 정의한 컴포넌트 명세는 기본적으로 UML 기반의 소프트웨어 개발을 지원하기 위해 설계되어있으며, 이로 인해 일반적인 컴포넌트를 이용하기 위해 필수적으로 요구되는 정보 외에도 UML을 지원하기 위한 정보들이 다소 포함되어 있어, RAS를 이용한 컴포넌트 명세는 다소 번거로울 수 있다. 때문에 몇몇 컴포넌트 명세 기법들은 RAS를 참고하여 해당 조직에서 필요로 하지 않는 항목을 제거하고, 추가적으로 요구되는 정보들을 추가 정의하여 사용되기도 한다.

X-ARM은 RAS와 OSD(Open Software Description)[7]를 참조하여 컴포넌트와 그에 대한 정보를 효율적으로 저장소에 저장하기 위해 설계되었다. X-ARM에서도 RAS와 같이 MDD(Model Driven Development)를 지원하기 위해 컴포넌트 명세에 “ComponentModel”이라는 정보를 포함하고 있으나, 이는 굳이 UML을 사용하지 않더라도 기술할 수 있도록 하였다. 이와 같이 UML에 대한 사용을 배제하고, 유연하게 사용할 수 있도록 설계함으로써 X-ARM에서 나타내는 컴포넌트 명세는 RAS와 비교하여 확연히 단순한 구조를 갖는다. 다만, X-ARM은 컴포넌트를 저장소에 저장하고 배포하는 등의 이용을 지원하도록 설계하여, 이에 해당하는 정보들이 추가되어 있다.

위와 같은 명세기법들은 비기능적 품질 속성 명세 항목을 정의하고 있으나, 구체적으로 어떠한 속성을 어떻게 명세할 것인지에 대한 정의가 없어 실질적으로 이를 집적이용한 비기능적 송성 명세에는 어려움이 있다. 비기능적 품질 속성에 대한 기술을 수행하고, 컴포넌트 선택 등의 의사결정을 지원하기 위한 명세 기법으로는 L. Iribarne의 연구가 있다[6].

L. Iribarne의 연구는 컴포넌트 명세 기법들이 비기능적 요구사항에 대한 명세 및 이를 통한 의사결정 지원을 위해 XML기반의 비기능적 속성을 표기 하는 방법을 제안하였다. 이때, 비기능적 속성은 NFR[8]을 이용하여 기술하였으며, 컴포넌트에서 나타내는 다양한 비기능적 속성에 대해 기술하고 각각의 속성에 대한 상호 연관관계를 나타내어 컴포넌트 선택에 있어서 의사결정을 지원할 수 있도록 설계되어 있다. 하지만 제시된 기법에서 각각의 품질 속성이 절대치로 평가되기 때문에 컴포넌트간의 상대적 효율성 비교를 평가가 주로 이용되는 속성의 경우에는 제시된 방법으로 명세하기가 쉽지 않다는 단점을 갖는다.

이와 같이 기존의 컴포넌트 명세 기법들은 비기능적 품질 속성을 고려한 의사결정 지원에 적합하지 않거나, 그것을 지원하더라도 각 품질 속성에 대한 평가 등을 수행하기에 부족하다는 단점이 있다. 본 연구에서 제시하는 소모전력 특성 명세 방법은 컴포넌트의 전력소모량이 나타내는 특성에 따라, 컴포넌트가 제공하는 인터페이스별 전력소모량의 가변적 특성을 고려하여 해당 인터페이스에 대한 Macro-model의 형태로 기술한다. 이러한 방식의 명세는 호출된 인터페이스에 따라 달라질 수 있는 소모 전력량을 고려하여, 전력

효율성에 따른 컴포넌트 선택 시 이에 대한 가이드라인을 제시할 수 있다. 또한 컴포넌트 기반 아키텍처 상에서의 전력소모 분석에 대한 지원이 가능하다.

3. 전력소모 특성 명세 방법

3.1 컴포넌트 명세 정보

컴포넌트의 실질적 사용을 지원하기 위해서는 해당 컴포넌트에 대해 기술해야 하는 몇 가지 정보가 있다. 이러한 정보들은 컴포넌트를 식별하고, 기능을 파악하며, 이를 사용하기 위해서는 어떠한 인터페이스를 이용해야 하는지에 대한 정보를 참조할 수 있도록 한다. 이와 관련하여 Michael Gsterfer는 컴포넌트가 재사용되기 위해서는 다음과 같은 사항들이 명세 되어야 한다고 주장하였다[3].

- Operational specification
- Operation context
- Non-functional properties
- Required interface and resources

이러한 정보들은 해당 컴포넌트의 이름과 기능적 특성 등을 명세하고, 이를 이용하기 위해 요구되는 인터페이스 및 해당 컴포넌트에 대한 비기능적 속성을 기술함으로써, 개발자가 컴포넌트를 식별하고 사용할 수 있도록 지원한다. 여기서 비기능적 속성은 다양한 속성들을 포함할 수 있으며, 각각의 성격에 따라 표현 방법 등이 상이할 수 있다. 본 연구에서 명세하고자 하는 전력소모 특성 또한 이 범주에 속하며, 소프트웨어의 전력 소모량은 그것이 나타내고 있는 행위적 특성과 밀접한 관계가 있다는 점에서 일반적인 기술이 어려울 수 있다.

재사용 컴포넌트의 행위를 나타내는 구성요소는 오퍼레이션들이다. 즉, 컴포넌트의 전력소모 특성을 기술하기 위해서는 오퍼레이션들을 이용해야 한다. 이러한 오퍼레이션의 종류에는 해당 컴포넌트에 서비스 요청을 위해 외부로 노출된 오퍼레이션과 해당 서비스를 제공하기 위해 내부적으로 호출되는 오퍼레이션이 있다. 여기서 외부로 노출되는 오퍼레이션들이 인터페이스에 해당하며 각각의 인터페이스들은 서로 다른 행위를 수행한다. 내부적으로 호출되는 오퍼레이션에 대한 제어 흐름을 내포하고 있다. 이와 같은 이유로 컴포넌트의 소모전력 특성 명세는 각 인터페이스별로 수행되어야 한다.

3.2 전력 소모 요소

소프트웨어의 전력 소모는 그것의 운영 환경이 어떻게 구성되어 있는지, 또는 어떠한 컴포넌트가 사용되었는지에 따라 전력 소모량이 크게 달라질 수 있다. 이는 소프트웨어의 운영 환경에 따라 연산 수행에 소요되는 시간 등이 변하는 것과도 같은 이치이다. 그림 1은 컴포넌트 기반 소프트웨어의 전력 소모량에 영향을 미치는 요소들을 중심으로 도시화한 소프트웨어의 운영 환경을 나타낸다.

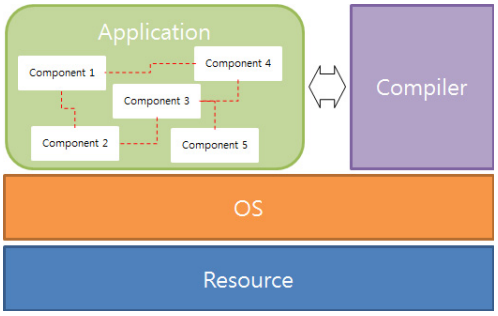

그림 1 컴포넌트 기반 소프트웨어의 운영 환경

3.2.1 Hardware Platform

소프트웨어의 실행을 통해 실질적인 전력소모의 주체는 하드웨어라는 점에서 그것의 구성에

따라서 소모전력 특성이 달라진다. 때문에 저전력 소모를 위한 초기 연구들은 하드웨어의 구성 소자 및 그들간의 구조를 전력 효율적으로 구성하는 것에 초점을 맞춘 연구들이 주로 진행되었다[9-12]. 그러므로 전력 분석의 대상이 하드웨어가 아닌 이를 제어하는 소프트웨어로 변경되었다 하더라도 하드웨어의 구성에 대한 정보가 없다면 이를 이용하기가 쉽지 않다. 특히 CPU의 구성 및 Clock speed에 대한 정보와 메모리의 크기가 중요한 역할을 한다[13, 14]. 따라서 재사용 컴포넌트가 추후 어떠한 하드웨어 플랫폼에서 실행할 것인가에 대한 정보가 요구된다.

3.2.2 운영체제

운영체제는 일반적인 응용 소프트웨어가 실행되기 위해 요구된다. 이는 응용 소프트웨어가 운영되기 위한 환경을 제공하고, 스케쥴링 또는 메모리 관리 등의 작업을 지원한다. 때문에 어떠한 운영체제를 사용하느냐에 따라 idle 상태의 소모 전력량 또는 운영되는 소프트웨어와의 상호작용에서 발생하는 전력량 등이 차이를 보일 수 있다[14]. 때문에 운영체제의 종류와 버전 또한 중요한 전력 소모 요소들 중 하나이다.

3.2.3 Compiler

소프트웨어가 같은 언어를 이용하여 구성되었다고 하더라도, 이에 사용된 Compiler의 종류나 버전에 따라 코드 최적화 정책이 서로 상이하게 적용될 수 있으며, 결과적으로는 동일한 코드를 컴파일 하더라도 컴파일러에 따라 해당 소프트웨어의 소모 전력량이 변화될 수 있다[15]. 때문에 소프트웨어 또는 컴포넌트의 소모 전력량을 명세하기 위해서는Compiler의 종류와 버전 등에 대한 정보가 요구된다.

3.2.4 Interface

그림 1을 살펴보면 “Component 5”를 제외한 모든 컴포넌트가 두 개 이상의 다른 컴포넌트와 상호작용하고 있다. 인터페이스는 이때 각 컴포넌트간의 상호작용이 가능하도록 외부로 노출된 오퍼레이션을 의미한다. 이때 컴포넌트는 최소한 하나 이상의 인터페이스를 제공해야만 한다. 만약 컴포넌트가 제공하는 인터페이스가 하나도 없다면 이는 소프트웨어에서 전혀 사용되지 않음을 의미하며, 필요 없는 코드가 포함되어 있는 것과도 같다. 3.1절에서도 설명한 바와 같이 컴포넌트가 제공하는 각 인터페이스는 서로 다른 행위를 나타낸다. 이는 곧 어떠한 인터페이스가 호출되었느냐에 따라 해당 컴포넌트에서 소모하는 전력량이 달라진다는 것을 의미하며, 이에 대한 정보는 컴포넌트 기반 소모 전력량 분석에서 매우 중요한 역할을 한다.

3.2.5 Input parameter

컴포넌트의 인터페이스는 별도의 파라미터 없이 호출될 수 있는 경우도 있지만, 입력으로 주어지는 파라미터가 요구되는 경우가 일반적이다. 파라미터를 요구하지 않는 인터페이스의 경우에는 항상 동일한 행위를 수행하므로 전력 소모량의 변화가 없지만, 파라미터가 요구되는 경우 주어진 파라미터의 특성에 따라 호출된 인터페이스의 분기, 반복 등의 제어 구조가 변경될 수 있다. 이러한 제어구조의 변화는 파라미터가 다음과 같은 특성을 나타낼 때 발생한다.

- 인터페이스의 분기를 결정하는 Flag로 사용되는 경우
- 파라미터가 가변적 길이를 갖고, 해당 인터페이스는 파라미터의 모든 요소에 어떠한 연산을 반복적으로 수행하는 경우

어떠한 컴포넌트에서 주어진 데이터의 정렬을 수행하는 인터페이스가 “List sort(List data, Boolean asc)” 로 정의되어 있다고 가정하자. 해당 인터페이스의 파라미터 중 “asc” 값이 “true”이면 오름차순 정렬을 수행하고, “false”이면 내림차순 정렬을 수행한다면, 해당 파라미터의 값에 따라 인터페이스의 분기가 어떻게 수행될 지 결정된다. 이러한 경우에는 하나의 인터페이스라도 “asc”와 같은 파라미터가 나타낼 수 있는 값의 종류에 따라 서로 독립된 인터페이스가 제공되는 것과 같이 전력소모 특성을 분류하는 것이 효율적이다. 또한 위의 인터페이스 정의는 “data”는 길이가 가변적이므로 이에 따라 수행해야 할 연산의 개수도 달라지게 된다. 결과적으로 “data”로 주어진 파라미터의 길이에 따라 소모되는 전력량 또한 변화된다.

이와 같이 입력으로 주어지는 파라미터가 해당 인터페이스의 분기, 반복 등의 제어 흐름을 결정하기 위한 요소가 될 때, 이것은 해당 인터페이스의 전력 소모량을 결정하는 중요한 요소가 된다. 그러므로 앞서 언급한 바와 같이 각 인터페이스의 소모 전력량은 일반적인 Scalar 값을 갖는 경우도 있지만(주어진 파라미터들이 단순한 연산에만 이용되며 제어 흐름에 영향을 미치지 않는 경우 또는 파라미터를 요구하지 않는 경우), 입력 파라미터에 따라 가변적인 전력소모량을 보일 경우에는 Macro-model을 이용하여 표현하는 것이 합당하다. 이와 같이 Macro-model을 통해 소프트웨어의 소모 전력량을 나타내는 방식은 T. K. Tan의 연구에서 이미 이용된바 있다[14].

3.3 전력소모 특성을 고려한 컴포넌트 명세 언어

본 절에서는 전력소모 특성을 고려하여 컴포넌트를 명세하기 위한 언어를 정의한다. 이를 위해서는 3.2절에서 식별한 소프트웨어의 소모 전력량에 영향을 미치는 요소들을 컴포넌트 명세 상에서 기술하는 방법에 대한 정의가 우선적으로 요구된다. 표 1은 하나의 컴포넌트를 기준으로 각 요소가 소프트웨어 컴포넌트의 전력 소모량에 대해 영향을 미치는 범위를 나타낸다.

표 1 전력 소모 요소의 영향 범위

구분	범위
Hardware	Component 전체
OS	Component 전체
Compiler	Component 전체
Interface	개별 Interface
Parameter	개별 Interface

3.3.1 전력 소모 요소에 따른 명세 방안

표 1을 살펴보면 전력 소모 요소들이 영향을 미치는 범위는 컴포넌트와 인터페이스로 나누어질 수 있다는 것을 알 수 있다. HW와 OS는 실제로 컴포넌트기반의 소프트웨어가 운영되기 위한 환경을 제공하는 역할을 하므로, 하나의 컴포넌트뿐 아니라 전체 소프트웨어에 동일하게 적용되는 사항이다. 물론 이러한 사항은 분산처리 환경에서 예외가 될 수 있으나, 컴포넌트 명세는 하나의 컴포넌트만을 명세하기 때문에 분산처리 환경에 대한 별도의 고려가 요구되지는 않는다. Compiler는 해당 소프트웨어 또는 Compiler가 실행될 수 있도록 기계어로 변환하는 역할을 수행한다. 여기서 Compile의 범위는 전체 소프트웨어가 될 수도 있으며, 컴포넌트 자체가 될 수 있다. 물론 두 가지 중 어떠한 경우라도, 하나의 컴포넌트가 둘 이상의 서로 다른 Complier로 나누어 Compile되지는 않으므로, 이 또한 전체 컴포넌트에 적용되는 요소이다.

이러한 세가지 항목들은 모두 컴포넌트 전체에 동일한 영향을 미치는 요소들 이였으나, 인터페이스와 입력 파라미터의 경우는 다수의 인터페이스 중 어떠한 인터페이스가 이용되었으며,

파라미터의 구성은 어떻게 되었는지에 따라 서로 다른 결과를 나타낸다. 때문에 이 두 가지 항목에 대해서는 3.2에서 식별한 것과 같이 각 인터페이스별로 독립적인 관리가 요구되거나, 파라미터의 구성에 따른 Macro-model을 이용해야만 한다.

3.3.2 CoSLanE

본 연구에서는 컴포넌트의 전력 소모 특성 정보를 고려한 컴포넌트 명세 언어인 CoSLanE (Component Specification Language for Energy)을 정의하였다. CoSLanE은 일반적으로 컴포넌트 명세에 이용되는 정보들을 바탕으로 하여, 전력 소모 특성을 명세하기 위한 정보들을 추가하여 정의되었다. 그림 2는 BNF[16]를 이용하여 정의한 CoSLanE의 구성을 나타낸다. CoSLanE은 컴포넌트를 명세하기 위해 <name>, <id>, <description>, <mngInf>, <commode>, <resource>, <interface>를 이용한다. 여기서는 전력소모 특성을 나타내기 위해 사용된 항목을 중심으로 설명한다.

(1) <resource>

<resource> 항목은 컴포넌트가 운용되는 환경을 명세하기 위해 사용된다. 본 연구에서는 이를 이용하여 해당 컴포넌트의 Hardware 구성과 OS 및 Compiler에 대한 정보를 나타내기 위해 사용하였다. <hardware>항목을 구성하는 <cpu>는 CPU의 제품 정보를 나타내며, <memory>는 운영환경 상에서 memory의 size를 나타내기 위해 사용된다. <OS>와 <compiler>는 각각 OS와 Compiler의 종류 및 버전을 나타내기 위해 사용된다. 이와 같이 <resource>를 통해 명세되는 정보들은 3.3.1에서도 정의한 바와 같이 해당 컴포넌트 전체에 적용되는 정보들로써 컴포넌트 명세 전체에서 한번만 정의함으로써 이용이 가능하다.

(2) <interface>

<interface>항목은 해당 컴포넌트에서 호출되는 Interface에 대한 전력 소모 특성을 실질적으로 명세한다. 이는 어떠한 인터페이스가 호출되었는지, 어떠한 파라미터를 통해 호출되었는지에 따라 가변적인 소모 전력량을 나타낼 수 있다. 때문에 컴포넌트가 제공하는 모든 인터페이스에 대해 독립적으로 정의해야 한다. <interface>항목은 <id>, <name>, <protoType>, <parameters>, <returnType>, <energyModel>로 구성되어 있다. <id>와 <name>, <protoType>의 항목은 각각 컴포넌트 라이브러리 상에서 관리 될 때를 고려하여 해당 인터페이스의 ID와 이름, 인터페이스의 Signature를 나타내기 위해 정의하였다.

```
<Component> ::= <name><id><description><mngtInf>
        <comModel><resource><interface>
<id> ::= <string>
<description> ::= <String>
<mngInf> ::= <location><version><author><status>
<location> ::= <string>
<version> ::= <version> | <number> | '.'<number>
<author> ::= <string>
<status> ::= <artifactType><ready>
<ready> ::= <bool>
<resource> ::= <hardware><OS><compiler>
<hardware> ::= <cpu><memory>
<cpu> ::= <string>
<memory> ::= <real_number>
<OS> ::= <name><version>
<compiler> ::= <name><version>
<interface> ::= <id><name><protoType><parameters>
            <returnType><energyModel>
<protoType> ::= <returnType>' '<string>
            '('<parameters>')'
<parameters> ::= <parameters> | <parameter>
<parameter> ::= <dtype>' '<string>
<returnType> ::= <void> | <dtype>
<energyModel> ::= <energyModel>
            | <opmode><expression>
            | <expression> |[<map>]
<opmode> ::= <string>
<expression > ::= <expression >+<term >
            | <expression >- <term > | <term >
<term > ::= <term >* <factor > | <term >/ <factor >
        | <factor >
<factor > ::= <real_number>| <variable>
<variable> ::= <name>
<map> ::= <parameter>'->'<factor>
<real_number> ::= <number>'.'<number>
<number> ::= <number> | [0|1|2| ... |9]
<name> ::= <string>
<string> ::= <string> |[a|b|c|...z|A|B|...Z]
        | [_| |-] | <number>
<bool> ::= [true|false]
<artifactType> ::= [model|code|binary]
<dtype> ::= [int|char|double|float|short] | <string>
```

그림 2 CoSLanE의 정의

<parameters>항목은 해당 인터페이스에 이용되는 파라미터들을 나타낸다. 본 명세 언어에서 가장 중요한 역할을 하는 <energyModel>은 주어진 파라미터 값에 따라 분기 구조가 변경되는 경우를 나타내기 위한 <opmode>와 해당 인터페이스의 Macro-Model을 나타내기 위해 사용되는 <expression>을 통해 명세 된다.

<expression>은 <term>을 통해 기술되거나, 재귀적으로 <expression>과 <term>의 산술적 합 또는 차를 통해 기술된다. 각 <term>은 <term>과 <factor>의 산술연산 곱셈 또는 나눗셈을 통한 조합 또는 <factor>만을 이용하여 구성된다. <factor>는 실수를 나타내는 <real_number>또는 해당 factor의 변수 명을 나타내기 위해 <name>을 이용하여 기술된다. <map>항목은 <parameters>에서 정의된 파라미터가 어떠한 <factor>에 해당하는지에 대한 매핑을 수행하기 위해 이용된다.

(3) <interface>를 이용한 전력소모 특성 명세

어떠한 interface A의 전력소모 특성을 표현할 때, 해당 인터페이스에 대한 전력소모 특성이 "2.4x + 314.5"와 같다면 앞서 정의한 CoSLanE의 <energyModel>을 통해 그림 3에서 나타내는 바와 같이 해석될 수 있다.

이와 같이 정의된 CoSLanE를 이용해 각 인터페이스에 대한 전력소모 특성을 기술함으로써 다음과 같은 효과를 얻을 수 있다.

- 해당 컴포넌트의 소모전력 특성을 효율적으로 표현할 수 있다.
- 전력 효율성을 고려한 컴포넌트 선택을 지원할 수 있다.
- 정의된 CoSLanE를 통해 소프트웨어 아키텍처 수준의 소모 전력량 분석을 수행할 수 있다.

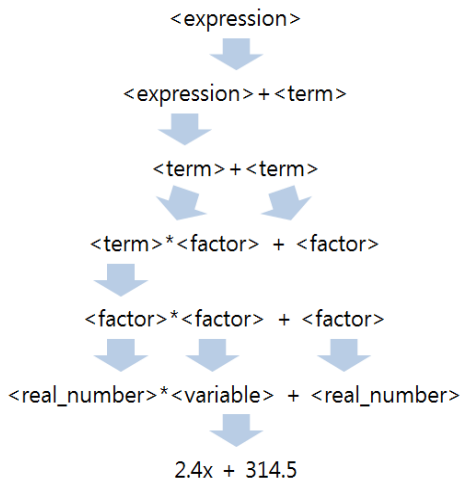

그림 3 CoSLanE을 이용한 전력소모 특성 표현

4. CoSLanE의 적용

3장에서는 컴포넌트의 소모 전력량에 영향을 미치는 요소들을 식별하고, 전력 소모 특성을 기술하기 위한 언어인 CoSLanE을 정의하였다. 본 장에서는 3장에서 정의한 소모전력 특성 명세 방법을 이용하여 전력소모 특성을 나타내는 컴포넌트 명세에 대해 논의하도록 한다. 그림 4와 그림 5는 이를 이용한 컴포넌트 명세의 예시이며 XML 형태로 표현하였다.

그림 4의 "Key"는 AES-SFT[17]의 구성 요소 중 암호화 키를 관리하는 컴포넌트로써, "expandKey()"와 "getRoundKey()"를 인터페이스로 갖는다. 주어진 예시에서 나타나는 두 개의 인터페이스가 모두 파라미터를 요구하지 않으므로, 그들의 전력 소모량은 Scalar 값으로써 각각이 "127681238.519417 (nJ)"과 "22659163.689320(nJ)"을 소모하게 된다. 그림 5는 Linux의 "ipc.h"를 통해 제공되는 인터페이스 중 Message send 등과 같은 함수들에 대한 명세 예시를 나타낸다. "msgctl()"의 경우 flag로 주어진 값에 따라서 어떠한 행위를 할 지에

대한 결정을 행하게 된다. 때문에 opmode 항목에 따라 달라지는 소모 전력량을 기술해준 것을 확인해 볼 수 있으다. “msgsnd()” 함수는 주어진 prarameter 중 “msgsz”의 크기에 따라 소모 전력량이 가변성을 나타내므로, 이에 대한 전력 소모 특성을 “4.0x + 4554.0”과 같이 macro- model로 표현해 준 후, “map”을 통해 어떠한 파라미터가 해당 macro-model에 영향을 주는지 정의 할 수 있다.

```
<Component ID="E001" Name="KEY" Description=" Key Managing Component of AES">
 < Interface ID="001" name="KeyExpansion">
  <Prototype type="void expandKey()"/>
  <EnergyModel model="127681238.519417"/>
 </Interface >
 <Interface ID="002" name="GetRoundKey">
   <Prototype type="void getRoundKey()"/>
   <EnergyModel model="22659163.689320"/>
 </Interface>
 <ManagementInformation Location="local library" Version="1.0" Author="D.H. Kim">
  <Status Ready="True" ArtifactType="Code"/>
</ManagementInformation>
 <Resource>
   <OS Type="Embedded Linux" Version="2.4"/>
   <Hardware CPU_Family="StrongARM(SA1100)"/>
   <Compiler name="gcc" Version="2.95.3"/>
 </Resource>
</Component>
```

그림 4 Key 컴포넌트의 명세 예시

5. 활용방안

5.1 전력소모 특성을 고려한 컴포넌트 관리

CoSLanE은 컴포넌트를 재사용하기 위한 정보들과 해당 컴포넌트의 전력 소모 특성을 기술하기 위한 정보들을 포함하여 설계되어 있다. 때문에 정의된 CoSLanE를 이용하여 컴포넌트 명세를 수행할 때, 컴포넌트의 전력소모 특성에 대한 정보를 포함하여 전사적 차원의 라이브러리를 통해 관리가 가능하다. 이러한 정보는 전력 소모량에 대한 요구사항의 중요도가 높은 소프트웨어 개발 시 유용하게 사용될 수 있다.

5.2 컴포넌트 선택 가이드라인

CoSLanE을 이용하여 컴포넌트를 명세할 경우 같은 기능을 제공하는 두 개 이상의 컴포넌트들 중 어떠한 컴포넌트가 전력 소모에 효율적인지를 판단할 수 있도록 가이드라인을 제시한다.

그림 6에서 Component A와 Component B는 각각 주어진 인터페이스를 통해 “2x+4854”와 “10x+4680”의 전력 소모를 한다. 이때 주어진 입력 값의 크기가 16byte 미만일 경우 Component B가 A보다 효율적으로 보이나, 이보다 클 경우에는 현저하게 많은 전력량을 소모한다는 사실을 알 수 있다. 이와 같이 컴포넌트에 대한 전력 소모량을 그래프 그렸을 때, 그래프간에 교차점이 발생한다면 어떠한 컴포넌트가 더 전력 소모 측면에서 효율적이라고 말 하기란 쉽지 않은 일이다. CoSLanE는 이러한 상황에서 소프트웨어 개발자가 해당 컴포넌트를 통해 처리되는 데이터양의 평균치를 분석하여 주어진 상황에서의 효율성을 고려한 선택을 수행할 수 있도록 지원한다.

```
<Component ID="Sys001" Name="ipc.h" Description="This is a library for IPC(Inter Process Communication ">
 < Interface ID="001" name="msgctl">
  <Prototype type="int msgctl(int msqid, int cmd, struct msqid_ds *buf)"/>
   <opmode=IPC_SET>
   <EnergyModel model="3334.7"/>
   </opmode>
   <opmode=IPC_STAT>
    <EnergyModel model="3835.7"/>
   </opmode>
   <opmode=IPC_RMID>
    <EnergyModel model="3603.6"/>
   </opmode>
 </Interface >
 <Interface ID="002" name="msgsnd">
  <Prototype type="int msgsnd(int msqid, struct msgbuf *msgp, int msgsz, int msgflg"/>
   <parameters>
    <parameter="int msqid"/>
    <parameter="struct msgbuf *msgp"/>
    <parameter="int msgsz"/>
    <parameter="int msgflg"/>
   </parameters>
   <returnType="int">
   <EnergyModel model="4.0x+4554.0"/>
   <map="int msgz->x">
 </Interface>
 <ManagementInformation Location="local library" Version="1.0" Author="GNU">
   <Status Ready="True" ArtifactType="Code"/>
 </ManagementInformation>
 <Resource>
    <OS Type="Embedded Linux" Version="2.4"/>
    <Hardware CPU_Family="StrongARM(SA1100)"/>
 </Resource>
</Component>
```

그림 5 ipc.h의 일부에 대한 명세

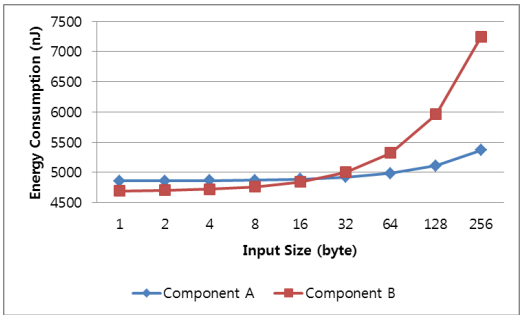

그림 6 같은 기능을 수행하는 컴포넌트간의 전력 소모량 비교 예시

5.3 Architecture를 이용한 전력 분석 지원

마지막으로, CoSLanE을 통해 명세한 컴포넌트를 이용하여 소프트웨어 아키텍처 수준에서의 전력 분석을 수행할 수 있다. 그림 7은 AES-SFT의 아키텍처를 나타낸다.

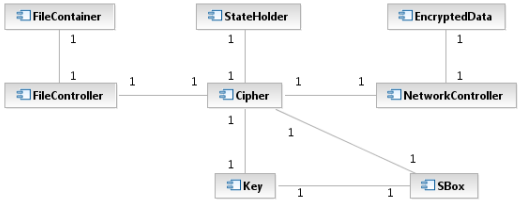

그림 7 AES-SFT의 Architecture

그림 4에서 명세한 "Key"는 그림 7에서 나타내는 아키텍처를 구성하는 하나의 컴포넌트다. 이때, 해당 아키텍처의 동적 구조를 통해 각 컴포넌트간의 상호작용을 도출하여 전력 분석을 수행한다면, 그림 4에 나타난 인터페이스의 소모 전력 특성을 통해 해당 컴포넌트의 소모 전력량을 도출 할 수 있다. 이와 같은 방법으로 소프트웨어를 구성하는 모든 컴포넌트들에 대한 명세가 존재 한다면, 이들을 합성하여 전체 소프트웨어가 소모하는 전력량을 도출 할 수 있다. 특히 이러한 경우 그림 5의 "ipc.h"와 같이 OS나 OS에 서비스를 요청하는 시스템 콜 등을 컴포넌트로 간주하여 이들에 대한 컴포넌트 명세를 수행한다면, 컴포넌트가 OS와 상호작용하는 경우에도 정확도가 높은 전력소모 분석을 수행할 수 있다.

6. 결론 및 향후 연구

본 연구에서는 소프트웨어 컴포넌트를 이용한 전력 소모 분석을 지원하기 위해 전력 소모 특성을 고려한 컴포넌트 명세 언어인 CoSLanE을 정의하였다. 정의된 CoSLanE을 통해 얻을 수 있는 효과는 다음과 같다.

- 전력 소모 특성을 고려한 컴포넌트 명세 및 이에 대한 효율적 관리
- 전력 효율적인 컴포넌트에 선택에 대한에 가이드라인 제시
- Software Architecture 수준에서의 소모 전력량 분석 지원

특히 위의 3)에서 나타내는 소프트웨어 아키텍처 수준에서의 컴포넌트 기반의 전력 분석은 소프트웨어 생명 주기의 앞 단계에서 분석을 수행할 수 있으며, 정확도 또한 높다는 장점을 기대해 볼 수 있다.

제안하는 명세언어를 이용하여 향후에는 CoSLanE에서 정의하는 내용을 기반으로 컴포넌트를 관리할 수 있는 컴포넌트 라이브러리 및 컴포넌트 기반 소프트웨어 아키텍처의 전력 분석 프레임워크를 구축할 예정이다.

참 고 문 헌

[1] Stephen S. Yau, "Embedded Software in Real-time Pervasive Computing Environments", in Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004.

[2] K. Naik, David S. L. Wei, "Software Implementation Strategies for Power-Conscious Systems", Mobile Networks and Applications, Vol. 6, Issue 3, pp.291-305, 2001.

[3] CJ Michael Geisterfer, Sudipto Chosh, "Software Component Specification: A Study in Perspective of Component Selection and Reuse", Proceedings of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Syatems (ICCBSS 2006), pp. 1-9, 2006

[4] OMG, "Reusable Asset Specification", version 2.2(formal/05-11-02), 2005 available from : http://www.omg.org

[5] G. Elias, et al., "X-ARM: An Asset Representation Model for Component Repository Systems", The 21st Annual ACM Symposium on Applied Computing Dijon, 2006

[6] L. Iribarne, et al., "A Non-Functional Approach for COTS Components Trading", in proc. Of the fourth workshop on requirements engineering (WER' 01), 2001

[7] A. V. Hoff, et al., "The Open Software Description Format (OSD)". Submission to the World Wide Web Consortium (W3C), 1997. available from : http://www.w3.org/TR/NOTE-OSD

[8] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., "Non-Functional Requirements in Software Engineering", Kluwer Academic Publisher, 2000.

[9] F. N. Najm, "Low-Power Design Methodology: Power Estimation and Optimization", Proceedings on the 40th Midwest Symposium on Circuits and Systems, Vol. 2, pp.1124-1129, 1997.

[10] A. Raghunathan, N. K. Jha, S. Dey, "High-level Power Analysis and Optimization", Kluwer Academic Publishers, Norwell, MA, 1998.

[11] J. Rabaey and M. Pedram, "Low Power Design Methodologies", Kluwer Academic Publishers, Norwell, MA, 1996.

[12] C. L. Su, C. Y. Tsui, and A. M. Despain, "Low Power architecture design and compilation techniques for high-performance processors", in Proceeding on IEEE COMPCON' 04, pp. 489-498, 1994.

[13] D. Sarta, D. Trifone, and G. Ascia, "A Data Dependent Approach to Instruction Level Power Estimation", IEEE Alessandro Volta Memorial Workshop on Low Power Design, pp. 182-190, 1999.

[14] T. K. Tan, A. Raghunathan, et al., "Energy Macromodeling of Embedded Operating Systems", ACM Transaction on Embedded Computing Systems, Vol. 4, Issue 1, pp. 231-254, 2005.

[15] J. W. Backus, M. Ibrahim, et al., "Compiler-based optimizations impact on embedded software power consumption", in Proceedings of the Jing Conference NEWCAS-TRRAISA, 2009.

[16] C. Backus, "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference", Proceedings of the International Conference on Information Processing, UNESCO, pp.125-132, 1959.

[17] NIST, "FIPS-PUB-197: Advanced Encryption Standard", 2001. Available from : http://www.aescrypt.com

저 자 소 개

김 두 환

2007년 충북대학교 컴퓨터공학과 졸업(학사)
2009년 충북대학교 전자계산학과 졸업(석사)
2010년~현재 충북대학교 컴퓨터과학과 박사과정.

〈관심분야〉 소프트웨어 아키텍처, 임베디드 소프트웨어 모델링, 임베디드 소프트웨어 품질공학, 저전력 소프트웨어

이 재 욱

2010년 충북대학교 컴퓨터공학과 졸업(학사)
2011년 충북대학교 컴퓨터과학과 수료(석사)
2011년~현재 충북대학교 컴퓨터과학과 박사과정.

〈관심분야〉 소프트웨어 아키텍처, 소프트웨어 모델링, 임베디드 소프트웨어 품질공학

홍 장 의

1988년 충북대학교 전산학과(학사)
1990년 중앙대학교 컴퓨터공학과(석사)
2001년 한국과학기술원 전산학(공학박사)
2002년 국방과학연구소 선임연구원
2002년~2004년 (주)솔루션링크 기술연구소장
2004년~현재 충북대학교 컴퓨터공학 부교수

〈관심분야〉 소프트웨어공학, 임베디드 소프트웨어, 소프트웨어 품질공학, 소프트웨어 프로세스 개선, 저전력 소프트웨어

AWT 기반 어플리케이션을 이클립스 플러그-인으로 리엔지니어링 할 때 재사용을 위해 고려할 사항들†

(Reengineering guidelines to develop eclipse plug-ins using AWT based legacy application)

양진석‡ 손동렬‡ 강교철¶

(Jin-Seok YANG) (Dong Ryul Dong) (Kyo. C. Kang)

요 약 우리는 소프트웨어 제품라인 개발 방법론인 FORM을 지원하기 위해 과거 AWT기반의 자바 어플리케이션으로 개발된 ASADAL의 동작 속도의 개선, 기능 추가, 그리고 좀 더 친숙한 사용자 인터페이스로의 개선이 필요했기 때문에 이클립스 플랫폼 기반의 플러그-인 어플리케이션으로 리엔지니어링을 진행해야 했다. 본 논문에서는 새로운 플러그-인 어플리케이션을 최소의 노력으로 개발하고 좀 더 빨리 배포하기 위해서 기 개발된 어플리케이션의 많은 부분을 재사용 할 수 있도록 리엔지니어링 과정에서 고려해야 할 세 가지 요소인 기 개발된 어플리케이션의 사용자 인터페이스 재사용 범위, 어플리케이션을 구성할 플러그-인들의 구성, 그리고 이후의 손쉬운 확장 및 기능 대체를 위한 플러그-인들 사이의 결합에 대해서 언급하고 각 요소에 대해서 사용한 방법 및 지침들을 소개한다. 그리고 ASADAL의 휘처모델 편집기와 행위모델 편집기의 리엔지니어링 사례를 들어 제시한 지침을 적용하여 개발한 플러그-인을 소개한다.

키워드 이클립스, 플러그-인 개발, AWT, 리엔지니어링

Abstract In order to improve working speed, add more functions and provide better user interface of ASADAL, the AWT based Java application to support FORM Software Product Line Development Methodology; we had to reengineer this application to be Eclipse based plug-in application. In order to improve working speed, add more functions and provide better user interface of ASADAL, the AWT based Java application to support FORM Software Product Line Development Methodology; we had to reengineer this application to be Eclipse based plug-in application. And then, this paper introduces the plug-in developed by applying guidelines presented in the reengineering cases of ASADAL's feature model editor and behavior model editor.

Key words Eclipse, Plug-in Development, AWT, 리엔지니어링

† 본 논문은 정보통신산업 진흥원의 소프트웨어 공학 요소기술 연구개발 사업에 의해 지원되었음을 밝힙니다.
The research was supported by WCU (World Class University) program throuth the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-2008-000-10100-0).

‡ 비 회 원 : 포항공대 정보통신 연구소
edeward@gmail.com
‡ 비 회 원 : 포항공대 정보통신 연구소
mymy29@gmail.com
¶ 종신회원 : Devision of ITCE, POSTECH
kck@postech.ac.kr

논문접수 : 2011년 09월 05일
심사완료 : 2011년 09월 25일

1. 연구배경

포항공대 소프트웨어공학 연구실에서 개발한 ASADAL [1]은 과거 10년 동안 소프트웨어 제품라인 공학의 개발 방법인 FORM (Feature-Oriented Reuse Method) [2]을 지원하기 위한 도구로써 개발되어 왔다. 자바 (Java) 진영에서 개발된 그래픽 라이브러리 AWT/Swing 기반으로 개발된 ASADAL은 두 가지 문제점에 대한 리엔지니어링이 필요로 하고 있다. 첫 번째는 오늘날

AWT의 근본적인 문제로 지적되고 있는 친숙하지 못한 사용자 인터페이스 룩 앤 필 (Look and Feel) 문제와 메모리 누수 문제로 부터 발생하는 성능적 문제이다.[3] 두 번째 개발환경에서 부터 기인하는 것으로 기 개발된 컴포넌트와 독립적으로 기능적인 확장이 용이한 개발환경을 개발자에게 제공하는 것이다. 과거의 독립형 자바 어플리케이션 개발환경에서는 기 개발된 컴포넌트와 새롭게 개발되거나 확장된 컴포넌트의 구분과 관리가 용이하지 못하기 때문이다. 그 결과 다양한 제품이 발생하게 되고 제품들 사이의 기능 및 데이터 모델의 일관성을 유지하지 못하는 문제가 발생하기도 하였다.

우리는 위의 리엔지니어링을 통해 FORM 제품라인 방법론을 지원하는 새로운 도구를 최대한 빠른 시간에 사용자들에게 배포하고 싶었다. 그러기 위해서는 기 개발된 ASADAL을 구성하는 사용자 인터페이스를 포함한 주요 컴포넌트를 재사용해야 했다. 그리고 장기 개발 로드맵에 따라서 재사용된 컴포넌트들을 점진적으로 새로운 기능으로 손쉽게 대체 할 수 있는 개발환경이 필요했다. 그 결과 우리는 ASADAL 개발 당시에 사용된 자바 플랫폼 보다 기술적으로 많이 발전한 이클립스 RCP (Rich Client Platform)를 기반으로 리엔지니어링을 진행하여 이클립스 플러그-인에서 재사용 할 수 있도록 ASADAL 컴포넌트들을 플러그-인으로 이송 (Migration) 하여 새로 개발된 플러그-인들과 함께 사용하고 점차적으로 플랫폼으로 이송된 컴포넌트를 신규 컴포넌트로 대체하기로 결정하였다. IBM에서 개발을 주도하고 있는 이클립스 플랫폼은 자바 어플리케이션을 위해 SWT/Jface기반의 사용자 인터페이스, 작업공간, 로깅 등의 다양한 편의 기능을 제공하여 AWT기반 자바 어플리케이션에서 지적되던 사용자 인터페이스와 성능 문제를 해결하고, 기능적 확장이 용이한 OSGi (Open Services Gateway Initiative) 플러그-인 아키텍처를 지원하고 있기 때문이다. NASA등의 기관에서 이클립스 플랫폼 기반의 어플리케이션을 개발하여 사용하고 있어 어플리케이션 개발 편의성과 안정성을 대변하고 있다. [4]

본 논문에서는 위와 같은 목적을 달성하기 위해 기 개발된 응용소프트웨어의 재사용 부분을 플러그-인으로 이송하는 내용을 포함한 리엔지니어링에서 고려할 사항과 이에 대한 개발방법 및 지침에 대해서 설명하고 있다. 2장에서는 과거의 플랫폼과 신규 플랫폼에 대해서 간략하게 비교하고, 리엔지니어링 목표에 대해서 설명한다. 3장에서는 2장에서 정의한 목표를 달성하기 위한 리엔지니어링에서 고려해야 할 요소와 지침에 대해서 설명한다. 그리고 리엔니지어링 과정을 통해 실제 개발된 ASADAL의 휘처 모델링 지원도구와 행위모델링 지원도구 플러그-인에 대해서 설명한다. 마지막 4장에서는 결론과 향후계획을 설명한다.

2. 연구배경

이 장에서는 과거 ASADAL의 운영 플랫폼과 새로운 개발 도구의 플랫폼 사이의 비교와 새로운 어플리케이션으로 이송될 ASADAL의 기능에 대해서 간략하게 설명한다.

아래의 표 1은 두 자바 어플리케이션 플랫폼들의 차이점을 간략하게 보여주고 있다. 플러그-인 어플리케이션들은 어플리케이션 본연의 기능들 이외의 로깅 (Logging), 그래픽 라이브러리, 그리고 문서 편집 등을 위한 기능을GEF나 EMF등을 사용함으로써 별도의 개발 없이 사용 할 수 있는 장점을 가지고 있다. 과거 어플리케이션에서 이러한 기본 기능을 직접 개발하던 것을 이클립스 플랫폼을 사용함으로서 해당 기능의 개발시간을 절약 할 수 있다.

표 1. 개발 플랫폼들의 비교

Spec	As-Is	To-Be
Application Type	Standalone	Plug-ins
JVM	1.4	1.5+
Graphic Library	AWT/Swing	SWT/JFace
Model 관리	직접개발	EMF
Framework	직접개발 (Gdt 등)	Eclipse Platform (Command, EMF,GEF,Log등)

우리는 ASADAL의 많은 기능들 가운데 휘처 모델링 지원도구와 행위 모델링 지원도구를 우선 플러그-인으로 새롭게 개발하기로 했다. 그리고 새로운 기능을 추가적으로 반영하여 개발하기로 했다.

FORM에서 핵심이 되는 휘처 모델링 지원도구는 기존 도구에 비해 아래와 같은 차이점을 가지고 있다.

FORM보고서에 명시된 휘처 모델링 요소의 완전한 호환성 및 추가 모델링 요소들의 반영에 따른 데이터 모델 구조

다른 도구들과의 호환을 위한 XML형식의 데이터 저장

다이어그램 이외의 휘처 모델의 트리 및 그래프를 이용한 다양한 뷰 (View) 형태 제공

행위 모델링 지원도구는 Statechart, DFD (Data Flow Diagram), 그리고 다이어그램에서 사용되는 이벤트 또는 데이터의 정의하기 위한 어트리뷰트 편집기로 구성이 되어 있다. 다이어그램 기반의 편집기능이 대부분을 차지하는 이 도구에서 휘처 모델의 경우와 같이 데이터 모델의 변경은 하지 않고 메뉴나 어트리뷰트 편집기에서의 사용자 인터페이스만을 새롭게 수정하기로 했다. 왜냐하면 휘처 모델 도구와 달리 다이어그램 기반의 편집기들은 신규 개발 시 많은 신간을 소요하기 때문이다.

3. 리엔지니어링에서 고려할 사항들

이 장에서는 우리는 기 개발된 어플리케이션을 최대한 재사용하여 앞 장에서 설명한 두 도구들을 플러그-인으로 새롭게 개발하기 위해 고려한 사항들과 개발을 위해서 마련한 지침에 대해서 소개한다.

리엔지니어링은 AWT기반 어플리케이션의 사용자 인터페이스의 재사용 범위 결정, 패키지 분석과 플러그-인 설계 그리고 신규 플러그-인과 이송된 플러그-인들 사이의 연결 순서로 진행이 되었다.

3.1. 사용자 인터페이스 재사용 범위 결정하기

현재 대부분의 윈도우 어플리케이션의 사용자 인터페이스는 고착화 되었다고 할 만큼 유사한 모양의 컴포지트 (Composite) 와 위젯 (Widget)으로 구성되어 있다. 그래서 기본 윈도우 인터페이스를 굳이 과거의 사용자 인터페이스 재사용할 필요는 없었다. 하지만 SWT 기반의 이클립스 플랫폼의 윈도우 사용자 인터페이스에서는 제공하지 않는 개념인 MDI (Multi Document Interface)와 일반적인 윈도우 어플리케이션 위젯으로는 구현할 수 없는 다이어그램 편집기에 대해서는 플러그-인 개발에 어떻게 반영을 할지 결정이 필요했다.

MDI 개념의 AWT 내부 프레임 (Internal Frame)을 ASADAL에서는 목적에 따라 여러 개의 기능들을 하나의 프레임으로 묶고 작업공간을 나누기 위한 용도로 사용을 하고 있었기 때문에 플러그-인으로 이송시 반드시 가져가야 할 요소는 아니었다. 그래서 이 역할을 이클립스 플랫폼의 뷰 파트 (View Part)와 에디터 파트 (Editor Part)로 대체하기로 했다. 이 두 요소는 어플리케이션을 구성하는 플러그-인들의 여러 기능을 작업공간상에 배치하는 것을 목적으로 설계된 이클립스의 요소들 이기 때문에 AWT 기반 어플리케이션의 MDI의 훌륭한 대안이 될 수 있다.

신규 플러그-인의 빠른 배포를 위해서는 개발에 많은 시간과 노력을 필요로 하는 다이어그램은 재활용 대상으로 아주 매력적인 요소이다. 이 매력적인 요소가 초기 배포에 도움을 주면서 이후 이클립스 플랫폼의 기본 사용자 인터페이스에 좀 더 적합한 GEF (Graphical Edit Framework) 또는 GMF (Graphical Modeling Framework) 기반의 새로운 다이어그램 편집기로 손쉽게 대체 될 수 있게 플러그-인으로 이송 될 수 있도록 해야 했다. 하지만 일반적으로 기 개발된 응용소프트웨어에서 다이어그램에 해당하는 편집 환경과 다이어그램 데이터 모델 등을 다른 모듈로 부터 정확하게 분리하여 나누는 것이 쉬운 작업이 아니기 때문에 우리는 재사용할 다이어그램을 위한 아래의 작업 지침들을 마련했다.

- 개발지침 1) 다이어그램을 위해서 필요한 기 개발된 응용소프트웨어의 패키지들을 모두 하나의 플러그-인으로 이송을 시킨다.
- 개발지침 2) 개발지침1에서 이송된 플러그-인에 신규로 개발할 플러그-인을 위한 인터페이스를 정의한다.
- 개발지침 3) 신규 개발 플러그-인에서는 이송된 플러그-인에서 제공하는 인터페이스를 통해서 다이어그램 사용자 인터페이스 인스턴스를 반환받고 받환 받은 인터페이스를 SWT-AWT 브릿지 기술을 [5]이용하여 SWT 안에 포함 (Embedded) 시킨다.

휘처 모델링 지원도구와 행위 모델링 지원 도구에서 재사용 하기로 결정한 사용자 인터페이스는 아래의 그림 1과 그림 2에서 붉은 사각형 영역의 다이어그램 부분이다. 그 외의 사용자 인터페이스는 새로 개발될 플러그-인에서 새롭게 개발된다.

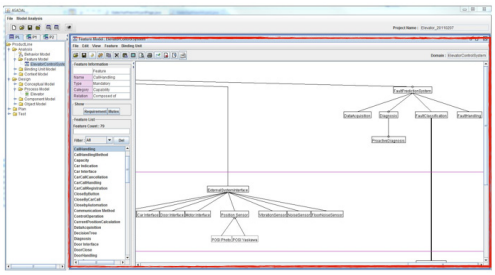

그림 1. ASADAL휘처모델링 도구 UI 재사용범위

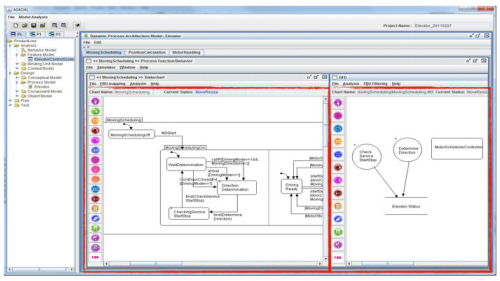

그림 2. ASADAL 행위모델링 도구 UI 재사용범위

3.2 패키지 분석과 플러그-인 구성

사용자 인터페이스에 대한 재사용 방향이 결정되었다면 어떻게 사용자 인터페이스를 포함하여 기 개발된 어플리케이션들의 패키지들을 플러그-인으로 이송하고 이송된 플러그-인들과 새로 개발해야 하는 플러그-인들 사이의 의존성 관계 구성에 대해서 고려해야 한다. 왜냐하면 이클립스 플랫폼 기반의 어플리케이션은 여러 개의 플러그-인들의 집합으로 구성되는 것이 일반적이고, 플러그-인들 사이의 의존성 관계가 복잡하면 어플리케이션을 구성하는 플러그-인을 대체하거나 새로운 플러그-인을 통한 기능 확장이 쉽지 않기 때문이다. 또한 AWT 기반의 어플리케이션 보다 빠른 응답성을 보장하기 위해서 이클립스 플랫폼에 제공하는 플러그-인의 늦은 바인딩 (Lazy Binding) [4] 기능을 십분 활용하기 위해서는 플러그-인들 사이의 의존성 관계 구성은 플러그-인 기반 어플리케이션 개발에서 상당히 중요한 문제이다.

우리는 어플리케이션을 구성하는 플러그-인들을 발굴하고 그들 사이의 의존성 관계를 구성하기 위한 아래의 개발 지침들을 마련하였다.

- 개발지침 4) 라이브러리 유형의 패키지는 독립적인 플러그-인으로 구성한다. 독립적인 플러그-인으로 구성함으로써 우리는 기능적 확장 측면에서 두 가지 장점을 얻을 수 있다. 첫 번째는 JAR형태로 라이브러리만을 별도로 배포할 수 있다는 점이다. 라이브러러 설정을 통해 개발자들은 관심이 없거나 작업 대상이 아닌 프로젝트를 작업공간에서 제거함으로써 작업 공간을 간결하게 유지할 수 있다. 두 번째는 차후 이 라이브러리만을 필요로 하는 새로운 플러그-인을 다른 플러그-인들과는 무관하게 개발 할 수 있다는 점이다.
- 개발지침 5) 휘처 모델링과 같이 대표 기능으로 분류될 수 있는 각각의 기능들은 패키지들 가운데 그 기능만을 담당하는 패키지들만 모아서 하나의 플러그-인으로 구성한다. 기존 패키지들을 최대한 구분하여 독립된 플러그-인으로 분리시킴으로써 패키지들 사이에 존재할 수 있는 다양한 의존관계를 정리 하고 이송 과정에서 인터페이스를 재정리 할 수 있는 장점을 가진다.
- 개발지침 6) 새롭게 개발할 플러그-인에는 기 개발된 어플리케이션의 패키지를 포함시키지 않는다. 신규 플러그-인과 기 개발된 어플리케이션에서 이송된 패키지를 포함하고 있는 플러그-인을 분리함으로써 차후 장기 개발 계획에 따라 기 개발된 어플리케이션에서 제공하는 기능의 대체를 용이하게 할 수 있다. 예를 들어 빠른 배포를 위해서 새로운 플러그-인에서 기 개발된 다이어그램 편집 환경을 사용하다가 새로운 다이어그램 편집환경의 사용 준비가 된 경우 기 개발된 다이어그램 편집환경의 기능을 제공하는 플러그-인의 의존관계를 끊음으로서 손쉽게 기능을 대체 할 수 있다.
- 개발지침 7) 앞에서 제시한 지침들에 따라 플러그-인을 발굴하고 의존 관계를 명시한 결과 플러그-인들 사이의 회전 의존성 (Cyclic Dependency) 이 발생하는 경우 회전 의존성을 야기하는 패키지 또는 클래스들을 독립된 플러그-인으로 구성하여 회전 의존성을 제거해야 한다. 이는 고품질의 이클립스 플러그-인 개발의 기본원칙 가운데 하나이다.

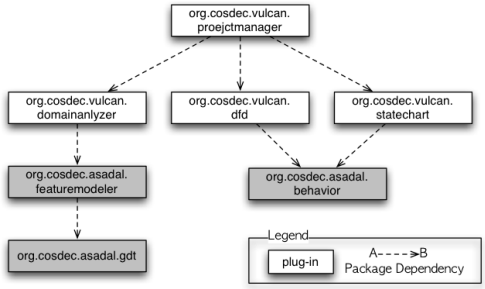

그림 3. 새로운 도구의 플러그-인들의 의존성 관계

우리는 위의 지침들을 바탕으로 새로운 어플리케이션의 플러그-인들을 위의 그림 3과 같이 구성을 할 수 있었다. 모두 7개의 플러그-인들 가운데 배경이 회색으로 표시된 3개의 플러그-인들은 재사용 범위에 따라 필요한 ASADAL의 패키지들을 이송한 플러그-인들이다. 흰색 바탕의 플러그-인들은 신규로 개발되는 플러그-인들이다. 이 가운데 "org.cosdec.vulcan.domainanalyzer" 는 휘처 모델링을 위한 새로운 사용 환경을 사용자에게 제공하고 휘처 모델의 뷰어들 가운데 다이어그램 뷰를 제공하기 위해서 ASADAL의 휘처 모델링 도구에서 제공하는 다이어그램을 사용한다. "*.vulcan.dfd"과 "*.vulcan.statechart" 플러그-인은 ASADAL의 Statechart 다이어그램과 Data Flow Diagram 편집 환경을 그대로 포함 (Embedding) 하고 있는 새로운 사용자 인터페이스를 사용자에게 제공한다.

3.3 이송된 플러그-인의 연결

어플리케이션을 구성하는 플러그-인들 사이의 관계가 정의가 되었다면 새로 개발될 플러그-인에서 이송된 플러그-인에 포함된 패키지의 내용의 수정을 최소화하여 연결하여 사용할 수 있는 방법에 대해서 고려해야 한다.

본 논문에서는 이송된 플러그-인을 사용하는 두 가지 경우에 대해서 다룬다. 첫 번째는 휘처 모델링 도구와 같이 새로 개발한 플러그-인에서 데이터 모델을 포함한 모든 부분을 새롭게 개발하고 이송된 플러그-인에서의 다이어그램 뷰(View) 만을 사용하기 원할 경우이다. 두 번째는 Statechart나 DFD의 경우와 같이 메뉴 등의 사용자 인터페이스 일부를 제외하고는 새로 개발된 플러그-인에서 이송된 플러그-인의 모델 및 뷰(View) 등의 대부분의 기능을 사용할 경우이다. 두 경우 모두 문제를 해결하기 위해서 적용된 소프트웨어 공학 기본 원칙은 정보은닉 (Information Hiding) 이다. [6]

첫 번째 경우 JFace의 프로바이더 (Provider) 개념을 차용하여 두 플러그-인들이 각각 사용하는 모델들 사이의 호환을 위한 모델 프로바이더를 제안하였다. (그림 4 참조) 모델 프로바이더는 신규 플러그-인의 모델을 이송된 플러그-인의 모델로 변환하여 이송된 플러그-인의 뷰에 모델을 공급하는 역할을 수행함으로써 이송된 플러그-인을 뷰로써 사용할 수 있게 만들었다. 이를 통해 차후 뷰어가 대체될 경우에도 손쉽게 대응 할 수 있게 되었다.

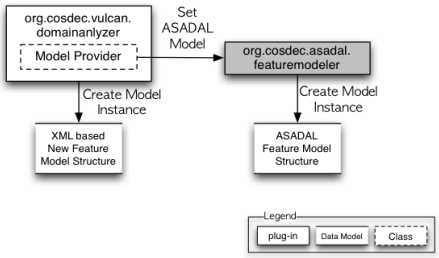

그림 4. 모델 프로바이더를 이용한 연결

아래의 [그림 5]와 [그림6]은 새로 개발된 휘처 모델링 지원도구가 모델 프로바이더를 통해 ASADAL의 휘처 다이어그램 뷰어를 SWT 사용자 인터페이스에 포함 (Embedding) 시켜 사용하고 있는 모습을 보여주고 있다.

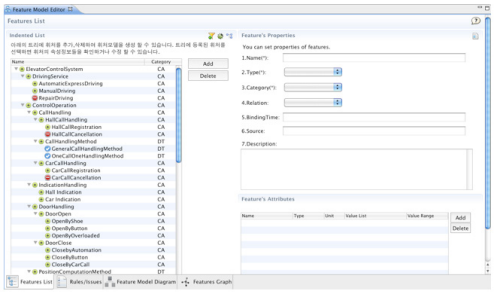

그림 6. 새로 개발된 사용자 인터페이스

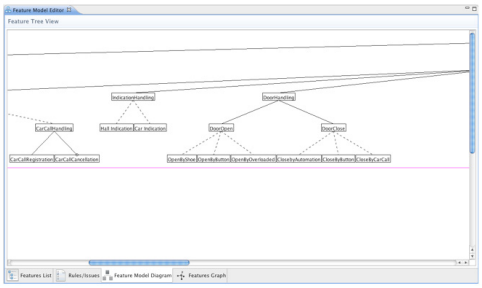

그림 7. 포함된 ASADAL 인터페이스

두 번째 경우를 위해서 이송된 플러그-인에서 제공하는 기능을 신규로 개발된 플러그-인에서 사용할 수 있는 어답터를 제공하였다. 이송된 플러그-인에서 제공하는 인터페이스를 직접 사용하지 않고 인터페이스로 동작하는 별도의 어답터를 이용하여 이송된 플러그-인으로 접근하게 함으로써 모델의 저장을 위해 미리 제공되는 이클립스 플랫폼의 사용자 인터페이스와의 연결을 위해서 이송된 플러그-인을 직접 수정하지 않고 개발을 진행 할 수 있었다.

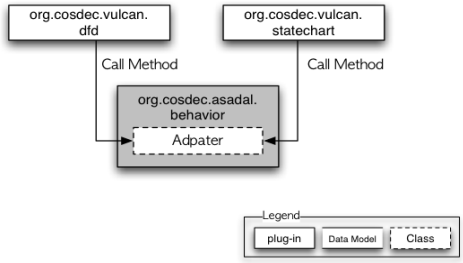

그림 8. 아답터를 이용한 플러그-인들 사이의 연결

아래의 그림 8과 그림 9는 기존 내부 프레임 사용자 인터페이스를 대체하도록 편집기 파트(Editor Part) 에 멀티 페이지 편집기로 구현된 Statechart와 Data Flow Diagram의 결과를 보여주고 있다. 이 두 개의 플러그-인들은 이클립스 플랫폼이라는 틀에서 사용자가 이송된 플러그-인의 기능을 사용할 수 있게 하는 통로로써의 역할만 수행한다. 이를 통해 사용자는 최소한의 이질감을 가지고 기 개발된 어플리케이션을 사용할 수 있게 된다.

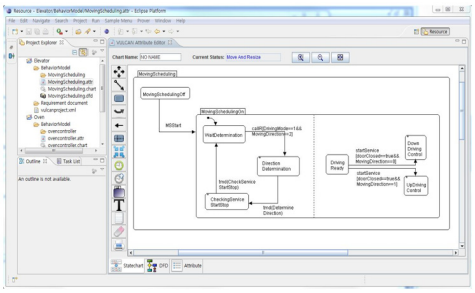

그림 9. 이송된 Statechart 편집기

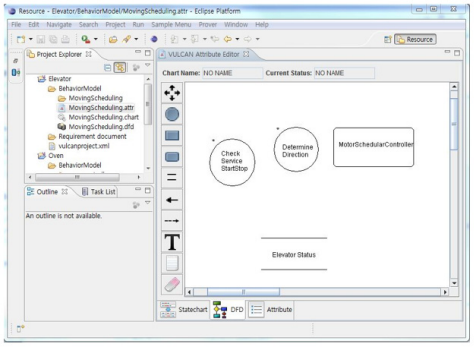

그림 10. 이송된 DFD 편집기

4. 결 론

본 논문에서는 신규로 개발된 플러그-인의 빠른 배포를 위해 재사용할 범위에 따라서 ASADAL의 일부 패키지를 플러그-인으로 이송하기 위한 지침들을 제시하고 활용 목적에 따라 이송된 플러그-인과 신규 플러그-인 사이를 모델 프로바이더와 아답터를 이용하여 플러그-인들 사이의 관계를 정의하고 연결하는 방법에 대해서 설명하였다.

휘처 모델링 지원도구와 행위 모델링 지원도구를 플러그-인으로 개발하는 사례를 통해 새로운 기능과 과거의 ASADAL 패키지를 활용하여 기 개발된 다이어그램을 연결하는 작업에 플러그-인당 1명의 개발자가 투여되어 약 1개월만 에 개발이 완료되는 것을 확인 할 수 있었다. 과거의 경험에 비춰 이클립스의 GEF와 EMF에 숙련된 플러그-인 개발자가 기본적인 다이어그램 편집 환경만을 개발할 때 1개월에서 2개월 정도의 개발시간을 필요로 하는데 비해 이 방법을 통해 훨씬 빠른 시간 안에 개발하고 배포를 할 수 있다는 것을 확인 할 수 있었다. 하지만 사용자 인터페이스 연결을 위해서 사용한 SWT-AWT 브릿지 기술이 어플리케이션의 운영 성능을 저하시키는 원인으로 작용하고 있어 목표로 삼은 어플리케이션의 성능 향상은 여전히 문제로 남아 있게 되었다. 그렇지만 이송된 플러그-인과 새로 개발된 플러그-인 사이를 모델 프로바이더와 아답터를 이용하여 느슨하게 연결함으로써 향후 성능 문제를 해결한 플러그-인으로 손쉽게 대체할 수 있게 개발을 함으로써 개발 계획에 따라 보완될 수 있을 것으로 예상된다.

우리는 본 논문에서 언급한 휘처 모델링 지원도구와 행위모델링 지원도구 이외의 ASADAL의 기능들을 점진적으로 이클립스 플러그-인으로 이송할 예정이며 개발에 참여하는 개발자가 이송을 일관되게 수행할 수 있도록 본 논문에서 제안한 개발 지침을 보완할 예정이다.

참 고 문 헌

[1] Kyungseok Kim, Jae Joon Lee, Miyoung Ahn, MinSeok Suh, Yeap Chang, Kyo Kang, ASADAL: A Tool System for Co-Development of Software and Test Environment Based on Product Line Engineering, The 28thInternational Conference on Software Engineering (ICSE2006), Shanghai, China, May 20-28, 2006, pp.780-786.

[2] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh, FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, 5, 1998, pp. 143-168

[3] Matthew Scarpino, Stephen Holder, Stanford Ng, Laurent Mihalkovic, "SWT/JFace In Action", Manning Publication, 2005

[4] Eric Clayberg, Dan Rubel, Eclipse: Building Commercial-Quality Plug-ins (2nd Edition), Addison-Wesley Professional, 2004

[5] Yannick Saillet, "ⓐMigrate your Swing application to SWT, Presented by developerWorks, your source for greate turorials", ibm.com/developer Works

[6] Hassan Gomma, "ÀDesigning Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures", Addison-Wesley, 2005

저 자 소 개

양 진 석

1999년 동국대학교 전산학과 졸업(학사)
2006년 고려대학교 컴퓨터학과 졸업(석사)
2006년~현재 포항공대 정보통신연구소 SE연구실 연구원.

<관심분야> 정형기법, 동기화 언어, 제품라인공학, 자동제어

손 동 렬

2003년 동국대학교 컴퓨터학과 (학사)
2009년~2010년 stx forcetec 근무
2010년~현재 포항공과대학교 정보통신연구소 연구원.

<관심분야> SW재사용, 객체지향 프로그래밍, 프레임워크

강 교 철

1973년 고려대학교 통계학과 졸업(학사)
1974년 콜로라도 대학교 산업공학 졸업(석사)
1982년~1984년 미시간대학교 Visiting Professor
1984년~1985년 (미국)벨 통신 연구소 Technical Staff
1985년~1987년 AT&T 벨 연구소 Technical Staff
1987년~1992년 카네기멜론 대학교 Project Leader
1992년~현재 포항공과대학교 교수
2000년~2001년 카네기멜론대학교 Visiting Scientist

〈관심분야〉 소프트웨어 공학, 실시간 내장형 시스템, 소프트웨어 재사용, 제품라인 공학

회원 가입 안내 및 회비 납부 요령

한국정보과학회 소프트웨어공학소사이어티는 회원 여러분에게 유익한 정보를 제공해 드리기 위하여 보다 충실한 내용의 논문지 발간 배포, 그리고 국제·국내 학술발표회 및 초청강연회와 단기강좌 등의 여러 가지 사업들을 추진하고 있습니다.

소프트웨어공학 소사이어티의 가입을 통해 정보 및 기술 교류, 그리고 인적 네트워크의 구성에 참여하시기를 기대합니다. 회원 가입을 위하여 아래의 회비 안내를 참고하시어 회비를 납부하시고, 다음 쪽의 입회원서를 작성하시어 아래 소프트웨어공학소사이어티 주소로 보내주시거나 팩스 또는 이메일을 통해 보내어 주시기 바랍니다.

한국정보과학회 소프트웨어공학소사이어티 연락처는 아래와 같습니다.

◈ 소프트웨어공학소사이어티

주 소 : (우) 121-742 서울시 마포구 신수동 1번지, 서강대학교 신과학관 202호
한국정보과학회 소프트웨어공학소사이어티 박수용

전 화 : (02) 705-8928

팩 스 : (02) 704-8273

전자우편 : sypark@sogang.ac.kr (박수용교수), bjlee@uos.ac.kr (이병정교수)

홈페이지 : http://www.sigse-kiss.or.kr/

◈ 회비 안내

회원구분	• 학생회원 : IT 분야 학과 또는 관심 있는 학생 • 정회원, 종신회원 : IT 분야 종사자
가입비	학생회원, 정회원 : 20,000원, 종신회원 : 200,000원
년회비	학생회원, 정회원 : 20,000원

- 회비납부 방법

(1) 무통장입금 또는 계좌이체 후 입회원서 발송
: 계좌 번호 : 제일은행 150-20-358028 (이병정)

(2) 소사이어티 주관 학술행사 개최시, 행사장 당일 가입 및 납부 가능

개인회원용 입회원서

회원구분	학생회원 ( ) 정회원( ) 종신회원( )				
성명	한글		생년월일		
	영문				
연락처	직장전화		휴대전화		
	e-mail				
주소	직장명/부서		직급		
	직장주소	(우)			
학력	학사	년 월 - 년 월	대학교	과	
	석사	년 월 - 년 월	대학원	과	
	박사	년 월 - 년 월	대학원	과	
관심분야					

본인은 한국정보과학회 소프트웨어공학소사이어티의 취지에 찬성하여 회원으로 가입하고자 이에 입회원서를 제출합니다.

년 월 일

신 청 인: (인)

한국정보과학회 소프트웨어공학 소사이어티 회장 귀하

논문지 논문 모집 (Call for Papers)

한국정보과학회 소프트웨어공학 소사이어티에서는 매년 4회에 걸쳐 '소프트웨어공학 소사이어티 논문지'를 발간하고 있습니다. 이 논문지에는 소프트웨어공학 전반에 걸친 연구논문과 산업계 논문을 게재해 오고 있습니다. 다음과 같은 소프트웨어공학 주제에 관련된 논문을 모집하고 있으니 학계와 산업계의 여러분들의 적극적인 논문투고를 바랍니다.

◈ 논문 주제
- ○ 소프트웨어 설계 및 아키텍처
- ○ 소프트웨어 재사용 및 프로덕트라인
- ○ 요구공학
- ○ 소프트웨어 품질 및 테스트
- ○ 관리 및 프로세스
- ○ 소프트웨어 정형 기법
- ○ 서비스기반 소프트웨어 개발
- ○ 임베디드, 모바일, 웹기반 소프트웨어 개발
- ○ 기타 소프트웨어 응용 (국방, 자동차, 조선 등의 분야)

◈ 논문심사
- ○ 투고된 논문은 편집위원회에서 심사 선정하며, 필요 시 외부 심사위원을 위촉하여 심사를 합니다. 제출된 논문은 반환하지 않습니다.
- ○ 심사료 및 게재료: 없음

◈ 논문 제출
- ○ 소프트웨어공학 소사이어티의 논문지 투고 양식(http://www.sigse-kiss.or.kr/)을 사용하며, 논문의 분량은 10장으로 제한합니다.
- ○ 논문지 투고규정에 따라 작성된 심사용 논문파일은 온라인투고시스템을 통하여 투고하시기 바랍니다.

◈ 문의처 (편집위원회)
- ○ 편집이사 : 고인영 교수 (KAIST, 042-350-3547, iko@kaist.ac.kr)
- ○ 편집이사 : 윤회진 교수 (협성대학교, 031-299-0841, hjyoon@uhs.ac.kr)
- ○ 편집위원 : 강성원 교수 (KAIST, 042-350-3512, sungwon.kang@kaist.ac.kr)
- ○ 편집위원 : 김문주 교수 (KAIST, 042-350-3543, moonzoo@cs.kaist.ac.kr)
- ○ 편집위원 : 김정아 교수 (관동대학교, 033-649-7801, clara@kd.ac.kr)
- ○ 편집위원 : 이우진 교수 (경북대학교, 053-950-6378, woojin@knu.ac.kr)
- ○ 편집위원 : 이찬근 교수 (중앙대학교, 02-820-5829, cglee@cau.ac.kr)
- ○ 편집위원 : 이 근 박사 (삼성전자, 031-277-7323, gskeun@gmail.com)
- ○ 편집위원 : 이세영 박사 (정보통신산업진흥원, 02-2132-1323, sarahlee230@gmail.com)

투 고 요 령

1. 소프트웨어공학소사이어티 논문지에 실리는 원고는 주제 논문, 일반 논문, 산업체 기고 등으로 구분하며 다음과 같은 분야에 대하여 모집한다.
   가. 소프트웨어공학 및 그 응용분야에 대한 연구결과
   나. 강좌 및 관련 교육사항 소개 (목적, 과정, 일정, 대상, 특징)
   다. 소프트웨어 도구 및 방법론 소개 (가격, 특징, 종류, 적용사례)
   라. 소프트웨어 산업에 대한 학계, 업계의 주요 관심사
   마. 기타 관련 사항

2. 투고자는 원칙적으로 본 소사이어티의 회원으로 한다. 다만 공동 또는 초청 기고자는 예외로 한다.

3. 논문은 원칙적으로 한글로 작성한다.

4. 원고는 한글(hwp), 워드(MS Word), PDF 형식 중 하나를 택하여 A4용지에 작성하며, 그림과 표를 포함하여 10쪽 이내로 한다.

5. 논문 내용에 직접 관련이 있는 문헌에 대해서는 이들 문헌에 관련이 있는 본문 중에 참고문헌 번호를 쓰고 그 문헌을 참고문헌 난에 인용 순서대로 기술한다. 참고문헌은 학술지의 경우 저자, 제목, 학술지명, 권, 호, 쪽수, 발행 연도의 순서로, 단행본은 저자, 서명, 쪽수, 발행처, 발행 연도의 순서로 기술한다.
   [1]Cole, R., "Parallel Merge Sort", SIAM Journal of Computing, vol.17, No.4, pp.770-785, 1988.
   [2]김수형, 강명호, 조형제, 송주석. "안전하고 효율적인 침입자 역추적 시스템", 정보과학회논문지, 제25권, 제10호, pp.1123-1131, 1998

6. 논문은 소프트웨어공학 소사이어티(http://www.sigse-kiss.or.kr/)의 온라인 투고 시스템을 통해 제출한다.

7. 논문투고신청서를 반드시 작성하여 이메일(hjyoon@uhs.ac.kr)로 제출한다.

7. 원고 접수는 수시로 하며, 접수일은 온라인 접수일로 한다.

8. 기타 자세한 사항은 한국정보과학회 논문지 투고 요령을 따른다.

한국정보과학회 소프트웨어공학소사이어티 임원명단

직책		성명	소속
회 장		박 수 용	서강대학교
부 회 장		권 기 현	경기대학교
부 회 장		민 경 호	LG전자
부 회 장		전 진 옥	비트컴퓨터
부 회 장		최 병 주	이화여자대학교
부 회 장		한 혁 수	상명대학교
감 사		이 긍 해	항공대학교
		차 성 덕	고려대학교
자 문 위 원 회		강 교 철	포항대학교
		권 용 래	KAIST
		배 두 환	KAIST
		성 기 수	KISTI 고문
		신 규 상	ETRI
		양 승 민	숭실대학교
		우 치 수	서울대학교
		이 경 환	중앙대학교
		이 단 형	KAIST
		정 기 원	숭실대학교
		황 선 명	대전대학교
협 력 기 관 위 원 회		권 경 룡	국방품질기술원
		박 찬 규	국방SW산학연합회
		신 석 규	SW시험인증센터
		유 법 민	지식경제부
		윤 태 권	SW기술진흥협회
		이 상 은	한국소프트웨어진흥원
운영위원회	**운 영 위 원 장**	권 기 현	경기대학교
	총 무 이 사	이 병 정	서울시립대학교
	기 획 이 사	백 종 문	KAIST
	조 직 이 사	이 관 우	한성대학교
	학 술 이 사	홍 장 의	충북대학교
	편 집 이 사	고 인 영	KAIST
	편 집 이 사	윤 회 진	협성대학교
	협 력 이 사	인 호	고려대학교
	홍 보 이 사	이 정 원	아주대학교
기술위원회	**기 술 위 원 장**	최 병 주	이화여자대학교
	요구공학분과장	박 수 진	서강대학교
	아키텍쳐분과장	조 은 숙	서일대학
	재사용분과장	이 관 우	한성대학교
	정형화분과장	김 문 주	KAIST
	테스팅분과장	김 영 철	홍익대학교
	프로세스분과장	이 세 영	NIPA

직책		성명	소속
	고신뢰성분과장	배현섭	슈어소프트테크
	재공학분과장	김진태	SEEG
	실시간분과장	이찬근	중앙대학교
산학위원회	**산학위원회장**	전진옥	비트컴퓨터
	자동차분과위원장	유영수	오토에버시스템㈜
	국방분과위원장	윤희병	국방대학원
	금융분과위원장	김정아	관동대학교
	전자분과위원장	이근	삼성전자
	의료분과위원장	최호진	KAIST
	전력분과위원장	이장수	한국원자력 연구소, MMIS팀 책임연구원
이사		강성원	KAIST
이사		계승교	삼성SDS
이사		권원일	STA컨설팅
이사		남영광	연세대학교
이사		김문주	KAIST
이사		김상기	현대자동차
이사		김영철	홍익대학교
이사		김정아	관동대학교
이사		민상윤	솔루션링크
이사		박복남	핸디피엠지
이사		백종문	KAIST
이사		손세창	인천공항공사
이사		손진규	삼성탈레스
이사		양상욱	KAI
이사		염근혁	부산대학교
이사		오재원	카톨릭대학교
이사		유준범	건국대학교
이사		윤형진	케피코
이사		윤희병	국방대학원
이사		이근	삼성전자
이사		이병걸	서울여자대학교
이사		이성남	방위사업청
이사		이우복	삼성전자
이사		이우진	경북대학교
이사		이은주	경북대학교
이사		이정원	아주대학교
이사		이해서	솔루션링크
이사		장주수	모아소프트
이사		정연대	N3SOFT
이사		정인상	한성대학교
이사		조병인	국방과학연구소
이사		조상윤	다한테크
이사		최승훈	덕성여자대학교
이사		최종무	단국대학교
이사		현창문	탐라대학교

2010-2011 소프트웨어공학소사이어티 논문지 편집위원회

편집위원장 윤회진 교수(협성대학교)

편 집 위 원 고인영 교수(KAIST)
강성원 교수(KAIST)
김문주 교수(KAIST)
김정아 교수(관동대학교)
이우진 교수(경북대학교)
이찬근 교수(중앙대학교)
이 근 박사(삼성전자)
이세영 박사(정보통신산업진흥원)

소프트웨어공학소사이어티 논문지 제24권 제2호 (통권 90호)

발 행 일 ॥ 2011년 6월 30일

발 행 인 ॥ 박수용
편 집 인 ॥ 윤회진

발 행 처 ॥ 사단법인 한국정보과학회 소프트웨어공학소사이어티
연 락 처 ॥ 서울특별시 마포구 신수동 1번지, 서강대학교 신과학관 202호
전 화 : 02-705-8928, 팩 스 : 02-704-8273
홈페이지 : http://www.sigse-kiss.or.kr/

인 쇄 처 ॥ (주)참기획 (전화 : 042-861-6380, 팩스 : 042-861-6381)

Copyright© 2011 한국정보과학회 소프트웨어공학소사이어티(비매품)