모바일 환경에서 재사용성을 향상시키기 위한 센서 프레임워크

최은영[O]
삼성전자
choiey@samsung.com

배두환
한국과학기술원
bae@se.kaist.ac.kr

Improving Reusability through Sensor Framework In Mobile Environment

Eun-Young Choi[O]
Samsung Electronics Co., Ltd.

Doo-Hwan Bae
Korea Advanced Institute of Science and Technology

요 약

편재형 컴퓨팅(pervasive computing)에서 컨텍스트에 대한 중요성은 점점 강조되고 있다. 편재형 컴퓨팅(pervasive computing) 환경에서 다양한 디바이스들은 사용자의 상황에 가장 적절한 서비스를 제공하기 위해 통신할 필요가 있는데, 특히 센서는 컨텍스트 데이터를 얻는데 효과적이다. 컨텍스트 정보를 얻기 위해 사용되는 센서들은 물리적인 센서들로부터 가상 센서까지 여러 가지가 있는데, 이러한 다양한 센서들을 관리하기 위해 센서 프레임워크가 필요하다. 본 논문에서는 모바일에서의 센서 어플리케이션 표준인 JSR256을 기반으로 한 센서 프레임워크를 어플리케이션 개발자의 시각이 아닌 센서 개발자의 시각에서 분석하였다. 그리고 관련 문제점을 발견하여 다양한 센서들을 효율적으로 관리하고 새로운 센서 모듈을 용이하게 개발할 수 있는 센서 프레임워크를 제안하고 제시된 센서 프레임워크를 통한 재사용성을 검증해 보았다.

1. 서 론

편재형 컴퓨팅(pervasive computing)에서 컨텍스트에 대한 중요성은 점점 강조되고 있다. 편재형 컴퓨팅(pervasive computing) 환경에서 다양한 디바이스들은 사용자의 상황에 가장 적절한 서비스를 제공하기 위해 통신할 필요가 있는데,[1] 컨텍스트는 정보 사회를 가져오기 위한 사회에 강력하게 영향을 줄 새로운 서비스 개발에서의 키이다.[2] 특히 모바일 디바이스에 장착된 센서는 사용자의 주변 정보를 실시간으로 직접 가져올 수 있다는 의미에서 컨텍스트 데이터를 얻는데 효과적이다.

1.1 다양한 센서들

센서는 물리적 센서와 가상 센서를 포함한다.[3] 물리적 센서는 자체적으로 데이터를 생성하는 센서를 말한다. 이러한 센서로는 카메라, 각속도 센서, 가속도 센서, 온도 센서, GPS나 바이오 센서들이 있다. 가상 센서는 전자 캘린더나 이메일, 문자 메시지 등을 브라우징해서 얻을 수 있는 소프트웨어와 데이터베이스로부터 나온 추가적인 정보와 함께 물리적 센서와 가상 센서의 조합으로 데이터를 얻는 센서이다.

1.2 센서 프레임워크

모바일 디바이스에서의 센서 프레임워크는 센서 검색, 센서 등록 및 삭제, 모니터링, 컨트롤 기능을 포함한다. 본 논문에서 논의되는 센서 프레임워크는 그림 1과 같이 JSR256[9]을 기반으로 한 것으로서, 센서 어플리케이션이 센서 프레임워크에 대해 동일한 인터페이스를 사용하고 필요한 센서를 컴포넌트로 추가 및 삭제할 수 있는 플러그인 구조를 가진다.

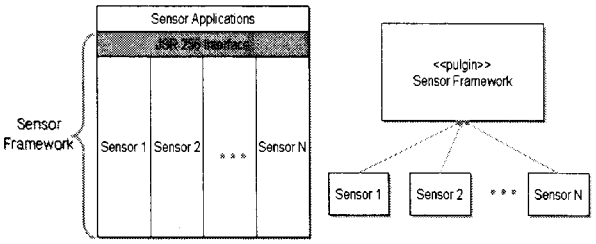


그림 1 현재 센서 프레임워크 구조

그러나 JSR256 인터페이스는 모바일 센서 어플리케이션 개발자를 위해 제안된 것이기 때문에[9] 이러한 센서 프레임워크는 센서 개발자를 위한 배려가 부족한 것이 사실이다. 최근 기존에 개발된 센서 모듈들 중 일부 인터페이스에서 비슷한 코드가 발견되면서 센서 프레임워크에서 중복 부분을 처리해 주면 새로운 센서 개발 시 드는 시간과 노력을 줄여 궁극적으로 생산성을 향상시킬 수 있다는 논의가 제기되었다. 본 논문에서는 센서 어플리케이션 개발자가 아닌 센서 개발자의 시각에서 센서 프레임워크를 바라보고 분석하여 문제점을 밝혀내고 해결 방안을 제시한다. 동시에 기존의 인터페이스를 최대한 활용하여

재사용성을 높이는 방안 또한 기술한다.

2. 현재 센서 프레임워크 분석

기존 센서 프레임워크에 문제점이 있는지 분석하기 위한 과정으로 다음과 같이 디자인과 소스 분석 면에서 살펴보았다.

2.1 디자인 분석

디자인 분석은 JSR256 클래스 구조와 인터페이스를 중심으로 진행되었다.

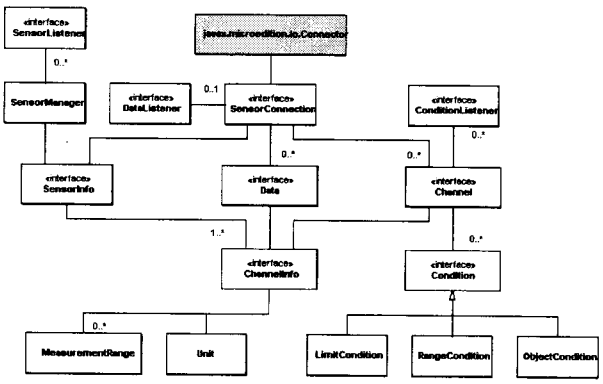


그림 2 JSR256 클래스 다이어그램

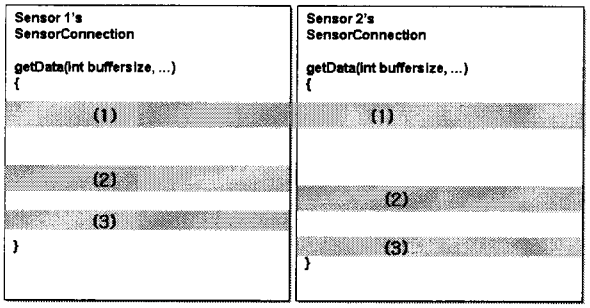
그림 2는 JSR256의 클래스 다이어그램이다. 특징적인 점은 SensorConnection 인터페이스에 가장 많은 클래스들이 집중적으로 연결되어 있다는 것이다. 또한 Condition이나 DataListener 같은 인터페이스는 동기 이벤트가 아닌 비동기 이벤트다. 이러한 이벤트 처리는 센서 개발자에게는 많은 고려가 필요한 부분이다. 하지만 현재의 센서 프레임워크에는 이벤트 처리에 대한 인터페이스가 따로 정의되어 있지 않다.

2.2 소스 분석

소스 분석은 지금까지 개발된 센서 모듈들의 소스 분석을 통해 진행되었다. 개발된 센서들을 모두 어플리케이션에 JSR256 인터페이스를 제공하도록 구현되어 같은 소스를 공유하는 인터페이스들도 있었지만 각 센서 별로 따로 구현해서 사용되는 다음과 같은 인터페이스들도 있었다.

- SensorInfo
- SensorConnection

센서마다 따로 인터페이스를 구현해야 하는 이유는 채널 수나 채널에 대한 정보 데이터 타입 등 각 센서 고유의 특성이 다르기 때문인데, 다른 센서의 경우 더 많은 인터페이스를 개별적으로 구현해야 할 수도 있다. 기존 소스를 바탕으로 분석된 위의 두 인터페이스를 좀 더 자세히 분석해 보면 인터페이스의 메소드에서 같은 패턴의 코드가 나타나고 있음을 알 수 있다. 예를 들어 SensorConnection의 getData() 메소드를 보면 그림 3과 같이 같은 액션을 취하는 코드들이 두 센서 다에서 각 센서의 특성에 맞게 조금씩 다르게 코딩 된 것을 발견할 수 있다.



(1) 각 센서에 데이터를 요청
(2) 가져온 데이터를 센서 프레임워크에서 정의해 놓은 데이터 타입에 맞게 변환
(3) 변환된 데이터를 getData()의 리턴 값으로 패칭

그림 3 비슷한 패턴의 코드가 보이는 SensorConnection의 getData() 메소드

이러한 패턴은 센서 데이터를 가져오기 위해 반드시 필요한 과정들이며 모든 센서에서 공통적으로 처리해 주어야 하는 부분이다. 이러한 부분들은 SensorInfo 인터페이스의 센서 파라미터를 세팅하기 위한 부분에서도 볼 수 있다.

또한 앞의 디자인 분석에서 나온 이벤트 관련 코드들을 함께 고려해 JSR256 인터페이스와 센서 모듈 간에 공통으로 처리해 줄 수 있는 부분이 존재한다는 것을 알 수 있는데 센서 고유의 특성 때문에 따로 구현된 인터페이스들에서 공통된 코드들을 합산해보면 전체 코드 길이의 20%가 나온다. 이는 앞으로도 개발될 센서들이 계속 있을 것임을 고려할 때 무시할 수 없는 수치이다.

2.3 분석 결과 정리

앞에서 언급된 사항을 종합하여 센서 간의 공통부분을 정리해 보면 표 1과 같다. 이로써 공통 부분을 각 센서 모듈이 공유할 부분이 존재하고 현재의 센서 프레임워크는 코드간의 재사용성을 높이기 위해 개선될 필요가 있다는 결론을 얻게 되었다.

표 1 센서간의 공통 부분

인터페이스 및 기능	세부 항목	설명
Event	Synchronous Event	어플리케이션이 센싱 된 데이터를 요청할 때 즉시 결과 값을 처리하는 경우.
	Asynchronous Event	어플리케이션이 센싱 된 데이터를 요청할 때 setDataListener() 또는 Condition 에 대한 센싱 데이터를 요청하는 경우.

Sensor Connect-ion	Data Convers-ion	각 센서는 고유의 데이터 특성을 가지고 있음. 센서로부터 가져온 데이터를 센서 프레임워크의 포맷에 맞도록 변환.
	Data Fetching	센서 프레임워크 포맷에 맞도록 Data로 변환한 후 getData()의 리턴 값으로 데이터를 패칭해주는 작업.
Sensor-Info	Sensor Setting	센서들은 각각의 고유한 특성을 갖는다. 채널 수를 비롯하여 각 채널의 이름, 정확도, 단위, 데이터 타입 등에 대한 세팅이 다르다. 이러한 부분에 대한 세팅하는 부분이 필요.

3. 접근 방법

접근 방법은 그림 4와 같다. 공통 부분을 추출하여 새롭게 아키텍처를 제시하고 RMM (Reuse Metrics and Modeling)[4]을 통해 개선 전후의 재사용율과 생산성 검증했다.

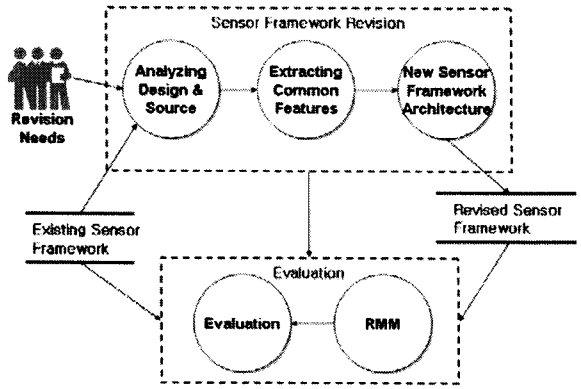


그림 4 접근 방법에 대한 플로우

또한 접근 기준은 다음과 같이 정의한다.

- 기존 센서들을 고려하여 어플리케이션과 바로 연결된 레이어에 있는 인터페이스들은 변경되지 않도록 하고 새롭게 개발하는 부분을 줄이기 위해 가능한 많이 기존 센서 프레임워크를 재사용하는 하는 것으로 한다.
- 센서 개발자들이 더 적은 노력과 시간으로 센서 모듈을 개발할 수 있도록 한다.

4. 센서 프레임워크 아키텍처 제안

그림 5는 위로는 기존 인터페이스를 유지하고 아래로는 센서의 각 특성을 살려야 하는 센서 프레임워크의 특성이 최대한 고려되어 앞에서 제시된 접근 방법에 따라 새롭게 제안된 센서 프레임워크 아키텍처이다. 각 부분은 센서 어플리케이션과 연동할 수 있는 JSR256 인터페이스가 있는 최상위 레이어와 Sensor Abstraction Layer가 있는 중간 부분 그리고 각 센서의 특성을 반영해 줄 수 있는 Sensor Specific Layer가 있는 최하위 레이어로 나뉜다.

4.1 최상위 레이어

최상위 레이어에서는 JSR256을 사용하는 기존에 개발된 센서들을 고려하여 기존에 사용되던 인터페이스와 메소드들이 그대로 사용된다.

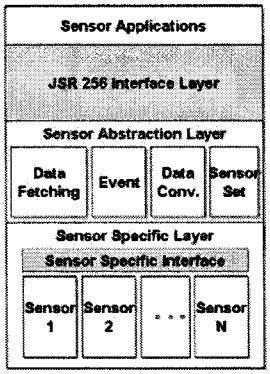


그림 5 제안된 센서 프레임워크 아키텍처

4.2 중간 레이어

중간 레이어는 앞에서 정리한 센서들간의 공통 부분을 처리하도록 그림 5와 같이 Data Fetching, Event, Data Conversion, Sensor Setting 모듈들을 두었다.

특히 이벤트 처리는 그림 6과 같이 하여 센서 개발자가 별도로 이벤트 처리를 해줄 필요가 없이 중간 레이어에서 데이터를 요청할 때마다 바로 주는 방식으로 변경 했다.

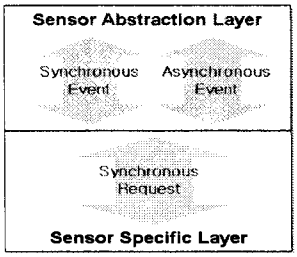


그림 6 Sensor Abstraction Layer에서의 이벤트 처리

4.3 최하위 레이어

Sensor Specific Layer는 중간 레이어와 연결되어 각 센서의 특성을 반영할 수 있도록 한 부분이다.

각 센서 개발자는 그림 7의 Sensor Specific interface를 구현하고 센서의 특성을 반영시켜줄 수 있는 추가 부분을 구현하면 된다. Specific Abstract Interface가 정의됨으로 이전에는 센서마다 따로 구현해야 하는 인터페이스들이 정해져 있지 않았지만 정해진 인터페이스를 구현하도록 함으로 센서 개발이

더 용이해지도록 변경되었다.

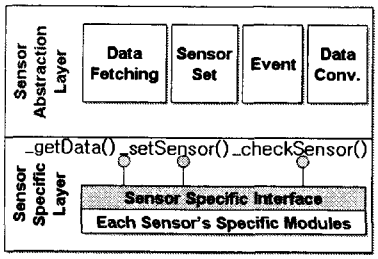


그림 7 Sensor Abstraction Layer와 Sensor Specific Layer 사이의 인터페이스

5. 재사용성 검증

새롭게 제시된 센서 프레임워크가 재사용성이 높은 생산성을 향상시킬 수 있는지에 대한 검증을 하기 위해 개선 전후의 전반적인 상황을 비교하고 RMM을 사용하여 실제 재사용율과 생산성에 대한 사항을 수치적으로 확인하면 다음과 같다.

5.1 전반적인 개선 전후 비교

표 2는 센서 프레임워크 개선 전후의 개발 관련 사항들을 전반적으로 비교한 표이다. 표로부터 각 센서 모듈이 바라보는 인터페이스를 동일하게 정의함으로 개발이 용이하게 되었고, 이벤트와 같이 공통적인 부분들이 중간 레이어에서 처리됨으로 이러한 사항들을 센서 개발자가 신경 쓰지 않아도 된다는 점에서 센서 프레임워크가 본 논문 초반에 언급되었던 데로 개선이 이루어졌음을 알 수 있다.

표 2 개선 전후 비교

비교항목	개선 전	개선 후
인터페이스 및 클래스	개선 전후를 비교해볼 때 개선 후 기존에 있던 인터페이스나 클래스는 그대로 사용하고 추가로 Abstraction Layer에 있는 인터페이스만 추가	
구현해야 하는 인터페이스	JSR256의 일부 인터페이스 중 개발하려는 센서와 맞아떨어지지 않는 부분을 찾아서 따로 구현	Specific Layer 관련 인터페이스만 구현하고 각 센서를 위한 특정 부분들만 구현
이벤트	동기와 비동기 이벤트 둘 다른 고려	Abstraction Layer에서 요청이 올 때 센서 데이터를 전달.
시간 및 노력	개선 후 개발에 드는 시간과 노력이 줄어들게 됨. (수치적 평가 자료 참조)	
개발자들이 느끼는 센서 프레임워크에 대한 접근성 정도	개선 후 이벤트에 대한 고려를 할 필요가 없고, Specific Layer에서 정의한 인터페이스만 구현하면 되기 때문에 접근성이 용이해 짐.	

5.2 RMM으로 본 정량적인 수치

표 3은 Hudson and Koltun의 Reuse Maturity Model이다. 이번에 개선된 정도는 두 번째인 Monitored에 해당하는데 이에 대한 메트릭은 코드수로 RMM의 Cost Benefic Analysis에 속하는 Cost Model[4]로 산정할 수 있다.

표 3 Hudson and Koltun Reuse Maturity Model

	1 Ad hoc/ Chaotic	2 Monitored	3 Coordinated	4 Planned	5 Ingrained
Motivation/ Culture	Reuse discouraged	Reuse encouraged	Reuse incentivized re-enforced rewarded	Reuse indoctrinated	Reuse is the way we do business
Planning for reuse	None	Grassroots activity	Targets of opportunity	Business imperative	Part of strategic plan
Breadth of reuse	Individual	Work group	Department	Division	Enterprise wide
Responsible for making reuse happen	Individual initiative	Shared initiative	Dedicated individual	Dedicated group	Corporate group with division liaisons
Process by which reuse is leveraged	Reuse process chaotic; unclear how reuse comes in.	Reuse questions raised at design reviews (after the fact)	Design emphasis placed on off the shelf parts	Focus on developing families of products	All software products are generalized for future reuse.
Reuse assets	Salvage yard (no apparent structure to collection)	Catalog identifies language and platform specific parts	Catalog organized along application specific lines	Catalog includes generic data processing functions	Planned activity to acquire or develop missing pieces in catalog
Classification activity	Informal, individualized	Multiple independent schemes for classifying parts	Single scheme catalog published periodically	Some domain analyses done to determine categories	Formal, complete, consistent, timely classification
Technology support	Personal tools, if any	Many tools, but not specialized for reuse	Classification aids and synthesis aids	Electronic library separate from development environment	Automated support integrated with development environment
Metrics	No metrics on reuse level, pay-off, or costs	Number of lines of code used in cost models	Manual tracking of reuse occurrences of catalog parts	Analyses done to identify expected payoffs from developing reusable parts	All system utilities, software tools and accounting mechanisms instrumented to track reuse
Legal, contractual, accounting considerations	Inhibitor to getting started	Internal accounting schemes for sharing costs and allocating benefits	Data rights and compensation issues resolved with customer	Royalty scheme for all suppliers and customers	Software treated as key capital asset

Cost Model에서 생산성은 아래와 같이 산정되는데, 이 과정에서 재사용성이 계산되어야 한다.

$$P = 1/C = 1/((b-1)R+1)$$

P : 생산성
C : 새로운 코드를 위한 소프트웨어 개발 비용
b : 새로운 코드를 통합시키는데 드는 비용. Gaffney and Durek[8]은 requirement, design, code가 재사용되었다면 b=0.85, test까지 재사용되었다면 b=0.8로 측정하였음. (b<1)
R : 재사용된 코드 비율 (R≤1)

재사용성은 다음과 같이 계산할 수 있는데[5]

$$\frac{\sum ReusedFunction'sActualLOC}{SWActualLOC}$$

두 개의 센서를 샘플로 보면 다음과 같다.

	Sensor 1	Sensor 2
재사용율	0.36	0.72

첫 번째 센서가 더 재사용율이 떨어지는 이유는 채널수가 많아서 각각의 채널을 세팅하기 위해 두 번째 센서보다 더 많은 새로운 코드가 들어갔기 때문이다.

이를 바탕으로 생산성을 계산해보면 아래와 같은데 1을 기준으로 잡았을 때 각각 5%와 12%가 향상된 것을 볼 수 있다.

	Sensor 1	Sensor 2
생산성	1.05	1.12

6. 결론

본 논문에서는 센서 어플리케이션 관점에서가 아닌 센서 개발자의 시각에서 현재 센서 프레임워크를 분석하였다. 그리고 새로운 센서 모듈을 개발하기 위한 시간과 노력을 줄이고 기존의 인터페이스 디자인 및 소스를 최대한 재사용하여 새로운 센서 개발 시 생산성을 높이기 위한 방안에 대해 살펴보았다. 또한 개선된 센서 프레임워크에 대한 전반적이고 수치적인 검증을 통해 재사용율과 향상된 생산성을 확인했다. 개선된 센서 프레임워크를 통해 향후 더 많은 센서 모듈들이 더 적은 노력과 시간을 가지고 개발되고 관리될 것이다.

7. 참고 문헌

[1] Jerome Euzenat, Jerome Pierson, Fano Ramparany, A Context Information Manager for Pervasive Computing Environments, INRIA Rhone-Alpes, France Telecom R&D

[2] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan, Context is KEY, Communication of ACM, 2005

[3] Matthias Baldauf and Schahram Dustdar, A Survey on Context -Aware Systems, Technical University of Vienna, 2006

[4] William Frakes, Carol Terry, Software Reuse: Metrics and Models, Virginia Tech, Incode Corporation, ACM Computing Surveys, June 1996

[5] Yoonkyu Jang, Sundeok Kim, Reuse Metrics for Embedded Software: An Empirical Sutudy, Samsung Electronics

[6] Will Tracz, Software Reuse Myths Revisited, Loral Federal Systems Company

[7] Fadi Shihadeh, A Look at Recent Trends in Software Metrics for Software Reuse, California State University San Bernardino, Dec 2005

[8] Gaffney, J. E.. and Durek, T. A.., Software reuse-key to enhanced productivity: some quantitative models, Inf. Softw. Technol., 1989

[9] Mobile Sensor API, JSR256 ver. 1.0, JSR256 Expert Group