DOI: 10.3745/KIPSTD.2011.18D.3.179

MVC+Prefetch Controller 패턴을 사용한 모바일 기기의 성능향상 기법

임 병 재[†] · 이 은 석[††]

요 약

모바일 기기는 유연한 이동성을 제공하는 대신에 제한된 자원을 가지고 있는 휴대용 장치로서, 최근에는 단순한 몇 가지의 기능만 제공하던 기존의 한계를 넘어서 많은 부가기능들을 추가적으로 제공하고 있다. 하지만 낮은 성능의 프로세서와 제한된 배터리용량으로 인해 사용자가 만족할 수 있는 성능을 제공하지 못하는 문제점이 발생하고 있다. 이러한 성능이슈는 하드웨어 교체를 통해 쉽게 해결될 수 있으나, 제품가격 상승이라는 치명적 단점을 가지고 있다. 본 논문에서는 성능이슈와 제품가격 상승 이라는 두 가지 문제를 동시에 해결하기 위하여 사용자 체감성능을 개선하는 방법을 제시한다.

사용자 체감성능은 사용자가 화면터치 또는 키패드를 통해 모바일 기기에 화면갱신 명령을 입력하고 모바일 기기의 출력장치인 LCD화면에 화면구성을 완료하는데 까지 걸리는 시간이 매우 중요한 요소이다. 모바일 기기는 물리적으로 작은 LCD화면을 사용하기 때문에 한번에 보여줄 수 있는 데이터의 양이 제한적이다. 화면을 구성하기 위해서 LCD화면에 보여줄 수 있는 양의 데이터만 사용한다면 빠른 화면구성을 할 수 있다. 이렇게 최소한의 데이터를 계산하고 DB에서 가져와 빠른 화면구성을 할 수 있도록 하는 Controller를 기존의 MVC 패턴에 추가한 MVC+Prefetch Controller 패턴을 제안한다. 제안한 패턴을 사용하면 사용자가 만족할 만한 체감성능을 보장할 수 있다. MVC+Prefetch Controller 패턴을 삼성전자 휴대폰 모델 S8500에 적용하여 사용자 체감성능 개선을 확인하였다.

키워드 : 임베디드 소프트웨어, 모바일 소프트웨어, MVC+Prefetch Controller Pattern, MVC

Performance improvement on mobile devices using MVC+Prefetch Controller Pattern

Byungjai Im[†] · Eunseok Lee[††]

ABSTRACT

Current mobile devices have surpassed its boundaries as a more communication tool to a smart device which provides additional features. These features have supported the smart life of its users, but have reached its limit from low-performance processors and short-battery time. These issues can be resolved b implementing higher performing hardware, but they come with a burden of high cost. This paper introduces a new way of managing computing resources in a mobile device by enhancing the quality of human-computer interaction. The real-speed felt by users are mainly influenced by the time it takes form a user's input to the device to display the completed result on the screen. Since the size of the screen for mobile devices are small, if the processor only fetch data to be used for displaying on screen, the time can be significantly reduced. MVC+Prefetch Controller pattern accomplished this goal by using the minimum amount of data from DB to fetch display and still manages to support high-speed data transfer to achieve seamless display. This idea has been realized by practice using Samsung mobile phone S8500, which demonstrated the superior performance on user's perspective.

Keywords : Embedded Software, Mobile Software, MVC+Prefetch Controller Pattern, MVC

1. 서 론

초창기의 모바일 기기들은 하나의 기능만을 수행하며, 단순한 입력 수단만을 제공했지만, 기술이 발전하고 모바일 기기의 판매경쟁이 심해지면서 제조사들의 모바일 기기 컨버전스 경향이 가속화되었다. 그 결과 모바일 기기들은 다

† 정 회 원 : 성균관대학교 정보통신공학부 석사과정
†† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2011년 3월 4일
수 정 일 : 1차 2011년 3월 14일
심사완료 : 2011년 3월 21일

양한 기능을 수행하게 되었고, 다양한 입력 방식이 개발되었으며 모바일 기기에 대한 기대감도 높아지고 있다[1][2].

대표적 모바일 기기인 핸드폰도 기본적인 전화 기능뿐 아니라, 문자 및 사진 전송, 음악/동영상 파일 재생, 일정관리, 사진/동영상 촬영 등 다양한 부가기능이 제공되며 이러한 부가기능들이 핸드폰의 특징이 되고 있다[3].

하지만, 사용자들이 원하는 성능을 제공하기에는 많은 문제점이 있다. 가장 큰 문제가 모바일 기기의 제한된 하드웨어 성능이다. 낮은 속도의 Processor와 제한된 용량의 배터리로 인해 전체적인 시스템 성능이 저하되고, 모바일 기기의 품질저하를 야기한다[4].

시스템의 성능저하를 해결하기 위하여 많은 개발자들은 하드웨어적인 방법을 선호한다. 고성능의 하드웨어와 대용량의 배터리를 사용하면 쉽게 문제를 해결할 수 있기 때문이다. 하지만, 제품가격상승을 유발하는 단점을 가지고 있다. 본 논문에서는 성능이슈와 제품가격상승 이라는 두 가지 문제를 동시에 해결하기 위하여 사용자 체감성능을 개선하는 방법을 제안한다. 사용자 체감성능은 사용자가 화면터치 또는 키패드를 통해 모바일 기기에 화면갱신 명령을 입력하고 모바일 기기의 출력장치인 LCD화면에 화면구성을 완료하는데 까지 걸리는 시간이 매우 중요한 요소이다[5]. 모바일 기기는 물리적으로 작은 LCD화면을 사용하기 때문에 한번에 보여줄 수 있는 데이터의 양이 적은 경우가 대부분이다. 따라서 소프트웨어의 변경을 통해서 제한된 크기의 LCD화면에 보일 수 있는 데이터만 DB에서 가져온다면 빠른 화면구성을 할 수 있고 성능이슈와 제품가격 상승 이라는 두 가지 문제를 동시에 해결할 수 있다. 본 논문에서는 최소한의 데이터를 계산하고 DB에서 가져와 빠른 화면구성을 할 수 있도록 하는 Controller를 모바일 어플리케이션에서 주로 사용하는 MVC 패턴에 추가하여 사용자 체감성능을 개선할 수 있는 기법인 MVC+Prefetch Controller패턴을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 모바일 어플리케이션 특징에 대한 설명과 사용자 체감성능에 대한 설명을 한다. 3장에서는 제안하는 “MVC+Prefetch controller 패턴”의 구조 및 동작에 대하여 소개를 한다. 그리고 4장 에서는 제안 기법에 대한 성능 평가를 위해 실제 모바일 기기인 삼성전자 모델 S8500에 제안한 패턴을 적용함으로써 얻어진 사용자 체험성능 개선 결과를 보여준다. 마지막으로 5장 에서 결론에 대해서 기술한다.

2. 관련 연구

2.1 모바일 어플리케이션의 특징

일반적인 모바일 기기 사용자들은 다운로드 비용이 들어가지 않는 네이티브 어플리케이션을 주로 사용한다. 네이티브 어플리케이션은 제품의 초기 생산부터 탑재되어, 초기 바이너리를 이루고 있으며 사용자들의 사용빈도가 높은 어플리케이션이다[6][7].

예를 들면, 핸드폰에서 사용하는 네이티브 어플리케이션으로는 음성/영상 통화, 문자메시지, MP3/비디오 플레이어 등등이 있으며 핸드폰 본래의 통화 기능 및 기본적인 부가기능을 담당하는 어플리케이션이다.

이러한 어플리케이션은 사용자의 사용빈도가 다른 어플리케이션에 비해 월등히 높아서 약간의 성능저하도 사용자에게 크게 느껴지게 되고, 경쟁사에게 좋은 비교대상이 되어 제품의 전체 이미지에 치명적인 나쁜 영향을 미치게 된다[7].

네이티브 어플리케이션은 전체코드가 모바일 기기에 생산 초기부터 탑재되어 모바일 기기가 폐기될 때까지 그 구조 및 기능이 동일하게 유지된다. 네이티브 어플리케이션의 특징은 다음과 같다.

(1) 사용자의 사용빈도가 높다.
(2) 사용자 체감성능에 큰 영향을 준다.
(3) 작은 성능저하도 제품 이미지에 치명적이다.

2.2 사용자 체감성능

과거의 제조사들은 모바일 기기를 제작할 때 하드웨어 성능에 초점을 맞춰서 제품개발을 하였다. 그 결과 소프트웨어 성능이 하드웨어를 따라가지 못하고 빠른 하드웨어를 적용하더라도 오히려 반응 속도는 이전보다 느린 경우가 발생하고 있다.

그 예로 핸드폰에 탑재된 MP3 Player가 있다. 하드웨어의 발전으로 핸드폰의 저장용량은 증가했지만 저장된 음악파일을 보이기 위해 UI를 구성하는 시간이 증가하였다.

저장메모리로 16G MMC(Multi Media Card)를 이용할 경우 3~4천 MP3 파일이 저장 가능하다. 이 파일들을 리스트로 구성한다면 10초 이상의 시간을 LCD 화면이 정지된 상태에서 리스트를 구성한다. 비록 최신의 하드웨어를 사용하더라도 이러한 화면구성 시간이 크게 단축되지 않는다. MMC의 용량이 증가 할 경우 화면정지 현상은 더욱 심하게 발생한다. 이러한 성능저하는 사용자에게 성능이 낮은 기기로 평가 받는 근거가 될 수 있다. 이처럼 하드웨어의 성능이 뛰어나더라도 실제 사용자 관점에서의 체감성능이 더욱 중요한 판단 근거가 되고 있다[8].

사용자의 관점에서 평가한 모바일 기기의 성능인 사용자 체감성능은 몇 가지 특징이 있다.

(1) 사용자의 입력이 종료한 시점부터 기기의 출력장치인 LCD에 원하는 데이터를 표시하는데 까지 소요된 시간이 중요하다.
(2) 기기의 출력장치인 LCD는 한번에 보여 보여줄 수 있는 데이터의 양은 적으며, LCD에 보이지 않는 준비상태의 데이터는 사용자 체감성능에 영향을 주지 못한다.

사용자 체감성능은 사용자가 입력을 하고 원하는 결과가 화면에 보이는데 걸리는 시간이 가장 큰 영향을 미친다.

출력 장치의 물리적 제한 때문에 사용자가 원하는 데이터(M개)를 모두 보이지 못하고, 실제 한번에 보일 수 있는 데이터(N개)는 제한적이다. 이는 아래의 식(1)과 같다.

$$M >= N \quad (1)$$

그 예로, 핸드폰에서 1000개의 MP3 파일을 리스트 형태로 구성할 경우, LCD 크기 제한 때문에 한번에 최대 5개 이하의 파일 리스트만 보여줄 수 있다. 더 많은 정보를 위해서는 스크롤을 이용해서 다음페이지로 이동해야 한다.

이 과정에서 사용자 체감성능은 파일리스트를 요청하고 5개의 MP3가 리스트형태로 보이는 시간이 된다. 하지만 개발자의 관점에서는 1000개의 파일을 리스트구조로 만들고 앞에 5개만 보이는 시간이 된다. 즉 같은 T ms이 걸렸지만, 사용자 체감성능은 5개의 MP3 파일을 보이는데 T ms이 걸린 것이고, 개발자는 1000개의 MP3 보이는데 T ms이 걸린 것이다. 이처럼 사용자 체감성능은 기존의 일반적인 성능측정이 아닌, 다른 관점에서 측정해야 한다. 그리고 사용자 체감성능을 개선시키는 것만으로도 시스템에 대한 사용자의 판단이 변화된다.

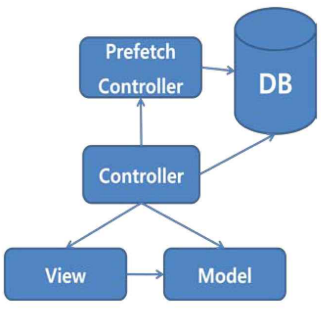
3. 제안 기법

LCD의 화면에서 한번에 보여줄 수 있는 제한된 양의 데이터를 우선적으로 가져와서 화면을 구성하는 Prefetch controller를 사용 할 경우 화면 구성시간을 보장함으로써 사용자 체감성능 개선이 가능하다. 여기서는 대표적 모바일 기기인 핸드폰을 가지고 MVC+Prefetch Controller 패턴을 설명하도록 한다.

3.1 MVC + Prefetch Controller 패턴

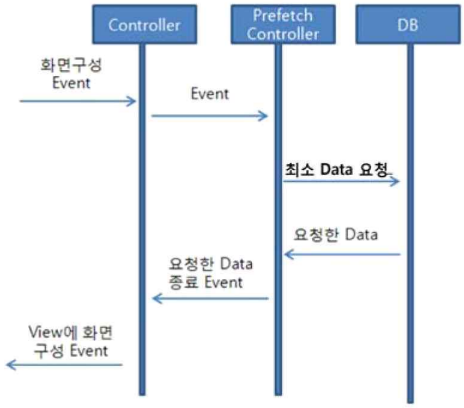
핸드폰 어플리케이션 구조는 MVC 패턴을 따르고 있다[9][10]. LCD 스크린에 보여줄 정보들을 Controller가 DB에서 가져와 View에 전달하고, View는 Controller로 부터 가져온 데이터와 Model에 미리 정의된 데이터들을 가져와 사용자에게 보이는 방식이다[11].

MVC+Prefetch Controller 패턴은 이러한 MVC 패턴에 Prefetch Controller가 Controller에 추가된 (그림 1) 과 같은 구조로 이뤄진다.



(그림 1) MVC+Prefetch Controller 패턴 구조

Prefetch Controller는 (그림 2) 와 같이 Controller가 DB에 데이터를 요청하는 단계에서 동작한다. 사용자의 버튼 선택 및 화면 터치로 인해 화면구성을 요청하는 Event가 발생하고 이 Event가 Controller에 전달되면 Controller는 Event를 Prefetch Controller에 전달한다. Event를 전달받은



(그림 2) MVC+Prefetch Controller 동작 Flow

Prefetch Controller는 출력장치인 LCD 화면을 한번 구성하기 위한 최소 데이터만을 DB에서 가져와 Controller에 전달하는 Flow 이다.

3.2 Prefetch Controller 동작

Prefetch Controller는 Controller에서 전달받은 이벤트를 이용해서 보여줘야 할 View Type및 데이터 개수를 파악한다.

핸드폰의 MP3 재생 목록을 예로 들면, 전달받은 Event로부터 Prefetch Controller는 View Type 은 Music List 이고, 데이터 개수가 5 라는 정보를 알아낸다. Music List 의 데이터개수는 MP3 파일의 제목과 가수, 앨범 이미지 등의 정보를 담고 있는 리스트 아이템을 화면에 최대로 보여줄 수 있는 개수가 된다. Music List 의 아이템은 (그림 3) 과 같이 구현된다.

(그림 3) Music List 의 아이템

Music List 의 경우 출력장치인 LCD 화면에 표시 가능한 아이템의 개수는 일반적으로 5로서, 5개의 음악 파일을 한 화면에 표기하고 있다.

이후 Prefetch Controller는 실제 DB에 접속하여 해당 값들을 Query를 통해 필요한 만큼 받아오고 이렇게 얻어온 값을 Controller에 전달하여 View가 화면 구성을 신속하게 할 수 있도록 한다. 이렇게 사용자의 입력이 들어와 화면 갱신이 발생할 때 최소한의 데이터만을 DB에서 가져와 화면구성을 하게 되므로 사용자에게 빠른 화면구성을 제공하고 사용자 체감성능 향상을 야기한다.

이처럼 MVC+Prefetch Controller 패턴을 적용하면 화면에 보일 수 없는 데이터를 가져오는 시간을 낭비 하지 않고 화면에 보여줄 수 있는 데이터만 가져와서 화면을 구성하기 때문에 화면구성 시간을 최소화할 수 있다. 그리고 총 List의 아이템 개수가 늘어나도 화면구성 시간은 항상 일정하게 유지된다. 사용자가 화면이 구성되기까지 기다리는 시간을

제거하여 사용자 체감성능이 보장된다면 사용자는 모바일 기기의 성능을 판단할 때 높은 점수를 준다. 이는 하드웨어 교체를 통해서 얻는 성능향상과 동일한 효과이다.

3.3 화면 구성 이후의 동작

Prefetch Controller가 사용자 입력을 받아 정보를 LCD에 표시한 이후, 사용자는 현재 화면에 원하는 데이터가 없다면 다음화면을 요청한다. 이 경우 새로운 데이터를 가져와야 되므로 이전 화면구성과 동일한 시간을 소비한 후 다음 화면을 구성한다. 반복적으로 화면구성 시간이 필요한 현상을 방지하기 위해서 첫 화면 구성이 끝난 다음 화면에 있는 데이터를 백그라운드 프로세스를 통해서 추가적으로 가져오도록 한다.

4. 성능 측정

본 논문에서는 일반 사용자가 많이 사용하는 4GB MMC (Multi Media Card) 를 이용하여 Prefetch Controller의 유무에 따른 성능변화를 수학적인 방법을 통해서 공식을 유도한다. 그리고 시뮬레이터를 통해 데이터를 측정하고 실제 핸드폰에서 성능을 측정하여 얻어진 결과를 분석 비교한다. 4G MMC의 경우 MP3 파일 한 개의 크기가 평균 4MB 이므로, 최대 1000개까지 MP3 파일이 저장가능 하다고 가정한다.

4.1 수학적 성능측정

수학적 성능 측정을 위해 사용되는 파라미터에 대한 정의는 다음의 <표 1>과 같다.

〈표 1〉 실험 분석 관련 파라미터.

Parameter	Description
C_RAN	무작위로 생성한 총 아이템 개수
C_MAX	하나의 화면에 보일 수 있는 아이템 개수
T_GET	하나의 아이템을 DB로 부터 가져오는데 걸리는 시간
T_SHOW	한 화면을 보이는데 걸리는 시간

먼저, Prefetch Controller가 존재 하지 않을 경우에는

$$T\_SHOW = T\_GET * C\_RAN \quad (2)$$

가 된다.
제안 기법인 Prefetch Controller가 존재 할 경우에는

$$T\_SHOW = T\_GET * C\_MAX \ (\text{if, } C\_RAN > C\_MAX)$$
$$T\_SHOW = T\_GET * C\_RAN \ (\text{if, } C\_RAN < C\_MAX) \quad (3)$$

로 나타낼 수 있다.

Prefetch Controller가 존재 하지 않을 경우는 화면구성을 위한 시간이 아이템 개수와 비례하여 증가한다. 하지만 Prefetch Controller가 존재하는 경우는 C_MAX가 고정된 상수 이므로 최대 T_GET*C_MAX 에서 더 이상 T_SHOW 값이 증가하지 않음을 알 수 있다.

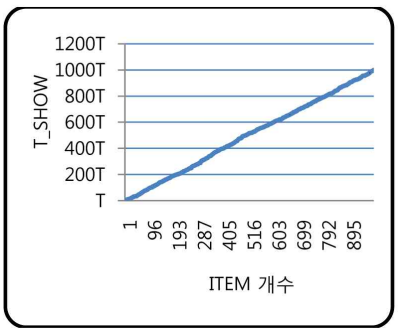
4.2 시뮬레이션 성능측정

4.2.1 시뮬레이션 성능측정 방법

시뮬레이션 성능측정은 1~1000의 정수를 무작위로 생성하여 이 값을 MP3 파일의 개수로 가정하여 Prefetch Controller의 유무에 따른 하나의 화면을 구성하기 위해 걸리는 시간 T_SHOW 를 측정 하도록 한다[12]. 여기서는 4.1절의 수학적 성능측정에서 제시한 공식(2)와 (3)을 이용하여 시뮬레이션을 진행하였다. 아이템 한 개를 구성해서 보여주는 시간을 T로 정의하고 C_MAX의 값은 5로 지정하였다.

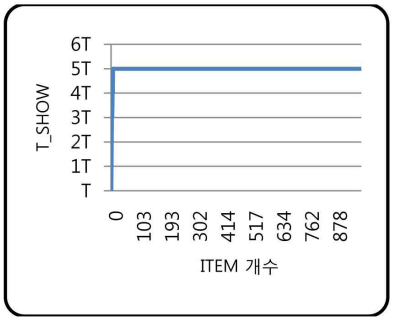
4.2.2 시뮬레이션 성능측정 결과

Prefetch Controller가 없는 경우는 (그림 4)와 같이 전체 아이템 개수(C_RAN) 이 증가할수록 화면을 구성하는 시간도 비례해서 증가하는 결과를 나타낸다.



(그림 4) Prefetch Controller 없는 경우

반면에, Prefetch Controller가 존재하는 경우는 (그림 5)에 보이는 결과처럼 아이템 개수가 증가함에 따라 화면 구성시간이 증가한다. 하지만 아이템 개수가 5 이상이면, T_SHOW가 아이템 개수와 관계없이 5T 로 일정하게 유지되는 것을 알 수 있다.



(그림 5) Prefetch Controller 있는 경우

이를 통해서 시뮬레이션 성능측정 결과 Prefetch Controller를 적용하지 않은 경우는 화면구성 시간이 전제 아이템 개수에 비례해서 증가/감소 하고, Prefetch Controller가 적용된 경우는 화면구성 시간이 아이템 개수가 C_MAX 이하에서는 비례관계를 유지하다가 아이템 개수가 C_MAX 이상으로 증가하게 되면 T_GET * C_MAX 로 유지됨을 확인했다.

4.3 실제 기기를 통한 성능측정

4.3.1 성능측정 환경

실험을 진행할 모바일 기기는 삼성전자의 핸드폰 모델 S8500이다. 화면 구성 속도와 관련된 기기의 실험 사양은 <표 2>와 같다.

〈표 2〉 모델 S8500 실험 사양

Samsung Wave(S8500)	
CPU	1 Ghz
OS	Bada OS
Memory 내장	2Gb / 8Gb
Memory 확장	32Gb
Display	3.3 Super AMOLED
해상도	800 * 400

4.3.2 성능측정 방법

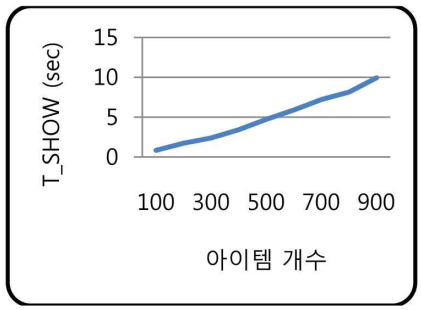
실험은 (그림 6)과 같은 S8500 의 메인 메뉴에서 우측상단의 Music Player 아이콘을 선택한 시점부터 (그림 7)과 같이 Music Player의 첫 화면인 Music list를 구성하여 LCD 화면갱신이 끝나는 시점까지의 시간측정을 통해서 이루어진다. 여기서 MMC는 SANDISK 4G 를 사용하고, C_MAX값은 5로 설정하였다.

(그림 6) S8500 메인메뉴 화면

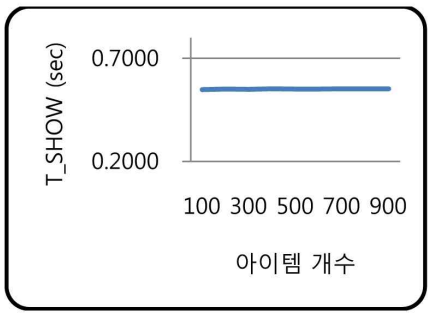
(그림 7) S8500 Music List 화면

4.3.3 성능측정 결과

Prefetch Controller가 적용되지 않은 경우는 (그림 8)과 같이 아이템 개수에 비례해서 지속적으로 T_SHOW가 증가하였다. 반면에, Prefetch Controller가 적용된 경우는 (그림 9)와 같이 아이템 개수가 5 이상 에서도 T_SHOW값은 0.56 sec로 일정하게 유지되는 것으로 나타났다.



(그림 8) Prefetch Controller 없는 경우



(그림 9) Prefetch Controller 있는 경우

5. 결 론

수학적 계산, 시뮬레이션 측정, 실제 기기측정 3가지의 결과에서 본 논문에서 제안하는 MVC+Prefetch Controller 패턴이 적용된 경우 화면을 구성하여 사용자에게 보여주는 시간이 기존방식에 비교하여 단축됨을 확인할 수 있었다. 이는 MVC+Prefetch Controller 패턴을 사용함으로써 추가적인 하드웨어 구입비용 없이 사용자 체험성능을 개선시킬 수 있고 이를 통해서 궁극적으로 시스템 성능개선 효과를 가져온다고 할 수 있다.

참 고 문 헌

[1] R. Benbunan-fich and A. Benbunan, “Understanding User Behavior with New Mobile Application.” , The journal of Strategic Information, vol.16, No.4, pp.393-412, 2007.

[2] König-Ries, B. and Jena, F., “Changllenge in Mobile Application Development”, it-Information Technology, Vol.52, No.2, 2009.

[3] G. Abowd, E, Mynatt, and T. Rodden. “The Human experience [of Ubiquitous computing].”, Pervasive Computing IEEE, 1(1): 48-57, 2002.

[4] I. Salmre, “Writing Mobile Code: Essential Software Engineering for Building Mobile Applications”, Addison-Wesley Professional, 2005, (chapter2)
[5] Ben Shneiderman, “Designing the User Interface: Strategies for Effective Human-Computer Interaction”, Addison Wesley, 2009.
[6] Y. Natchetoi, V. Kaufman, and A. Shapiro, “Service-oriented architecture for mobile applications. In proceedings of the 1st international”, workshop on Software architectures and mobility (SAM'08), pp.27-32, 2008.
[7] Heather Schneider, Valentino Lee and Robbie Schell, “Introduction to Mobile Application Architecture : Mobile Application Architectures” , Prentice Hall, 2004.
[8] Kim.h., Kim, M., Choi, J., and Ji, Y.G., “A study of Usability Evaluation for Tangible User Interface”, In proceeding of AE International 2008: 2nd International Conference on Applied Ergonomics, Las Vegas, Nevada, USA, Vol.14, No.17, 2008.
[9] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software” , Addison-Wesley, 1994.
[10] N. Medvice, et al., “Software Architecture and Embedded Systems.” , IEEE Software Vol.22(5), Sep., 2005.
[11] M. Pont and M. Banner, “Designing Embedded Systems Using Patterns: A Case Study”, Journal Systems and Software, Vol.71, 2004.
[12] Michael K. Molloy, Ph.D. “Fundamentals of Performance Modeling”, Prentice Hall; Facsimile edition, 1988.
[13] J. Gong and P.Tarasewich, “Guidelines for Handheld Mobile Device Interface Design”, Proceedings of the 2004 DSI Annual Meeting 2004.
[14] D. Schmodt, et. Al., “Pattern-Oriented Software Architecture”, Wiley, 2000.
[15] P. Braun, R. Eckhaus, “Experiences on model-driven software development for mobile application.” In proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based System, 2008.

임 병 재

e-mai : hansori99@hotmail.com
2006년 동국대학교 컴퓨터공학과(공학사)
2006년～현 재 삼성전자 무선사업부 선행개발팀 선임연구원
2009년～현 재 성균관대학교 정보통신공학부 석사과정
관심분야 : 소프트웨어 공학, 모바일 소프트웨어, 임베디드 소프트웨어 등

이 은 석

e-mail : leees@skku.edu
1985년 성균관대학교 전자공학과(공학사)
1988년 일본 Tohoku대학교 정보공학과 (공학석사)
1992년 일본 Tohoku대학교 정보공학과 (공학박사)
1992년～1993년 일본 미쯔비씨 정보전자연구소 특별연구원
1994년 일본 Tohoku대학교 Assistant Prof.
1995년～현 재 성균관대학교 정보통신공학부 교수
관심분야 : 소프트웨어공학, 오토노믹컴퓨팅, 에이전트지향지능형 시스템, CPS(Cyber Physical System) 등