영역 지식을 이용한 소프트웨어 이식

김동선[0], 박수용
서강대학교 컴퓨터학과
darkrsw@selab.sogang.ac.kr, sypark@ccs.sogang.ac.kr

Software Migration using Domain Knowledge

Dong-sun Kim[0], Soo-yong Park
Dept. of Computer Science, Sogang University

요 약

내장형 시스템에 장착되는 하드웨어의 성능이 향상되고, 네트워크에 연결이 되면서 고객의 요구는 더욱 복잡, 다양해졌으며 내장형 소프트웨어 개발에는 하드웨어 보다 높은 개발 비용과 시간이 필요하게 되었다. 이는 내장형 소프트웨어 개발에 기존 소프트웨어 개발 기술이 필요하게 됨을 의미하고 특히 비용 절감을 위해 재사용 기술의 적용이 요구된다. 기존의 컴퓨팅 환경(개인용 PC 등)에서 작동되는 소프트웨어를 내장형 소프트웨어로 이식하는 것이 가장 이상적인 방법이지만 내장형 시스템이 가지는 특징과 제약사항이 소프트웨어 이식의 장애로서 작용한다. 본 논문에서는 기존 소프트웨어를 내장형 시스템을 포함한 다양한 환경으로 이식하기 위해 영역 지식을 획득, 표현하고, 통합한 후 이식에 적용하는 절차를 제안한다.

1. 서 론

이동 환경에서의 정보처리에 대한 사용자의 요구와 유비쿼터스 컴퓨팅[2]에 대한 요구가 증가함에 따라 PDA, 핸드-셋(Hand-set), 냉장고, 자동차 등에서 작동되는 내장형 소프트웨어(Embedded Software)의 역할이 날로 증대되고 있다. 특히 내장형 소프트웨어가 탑재되는 내장형 시스템(Embedded System)의 하드웨어 성능이 급격히 향상되고 내장형 시스템이 독립적으로 작동하지 않고 네트워크를 통해 여러 기기들과 연결되면서 사용자가 요구하는 기능이 증가하고 있다. 그에 따라 사용자의 요구사항이 기존의 컴퓨팅 환경에서 요구하던 요구사항과 견줄 만큼 복잡, 다양화되고 있다. 게다가 지금까지 내장형 소프트웨어는 비교적 단순하고 개발비용이 내장형 시스템의 하드웨어보다 저렴하다고 간주되어 상대적으로 적은 연구만이 이루어 졌다. 그 결과 내장형 소프트웨어 개발 기술은 아직 초보적인 단계이며 특히 재사용에 대한 연구는 매우 미비하다[1]. 다행히 기존의 환경(개인용 컴퓨터 등)에서 개발된 소프트웨어가 가진 기능을 사용자가 다시 요구하기 때문에(네트워킹, 파일 처리, 멀티미디어 재생기능 등) 기존 개발 지식과 소프트웨어를 재사용할 기회가 있다. 하지만 내장형 시스템은 기존의 요구사항을 만족시킬 수 없는 여러 가지 특성과 제약사항들을 가지고 있다. 본 연구에서는 내장형 소프트웨어 개발은 물론 다양한 컴퓨팅 환경에 기존 개발 지식과 소프트웨어를 재사용하기 위해 영역 지식(Domain Knowledge)을 이용하여 소프트웨어를 다양한 환경으로 이식(Migration)하는 방법을 제안한다.

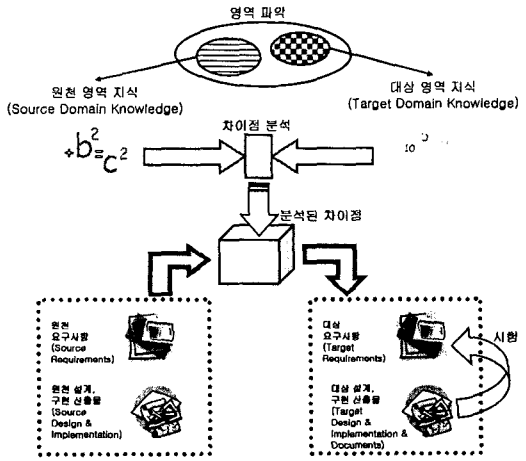
2장에서는 소프트웨어 이식에 대한 연구가 필요하게 된 연구배경을 소개한다. 3장에서는 영역 지식을 이용한 소프트웨어 이식 절차를 소개한다. 4장에서는 결론 및 소프트웨어 이식에 관한 향후 연구를 제시한다.

2. 연구배경

서로 다른 컴퓨팅 환경들 사이에서의 소프트웨어 이식은 최근에 PDA, Hand-set, 기타 여러 내장형 시스템 개발이 활발히 이루어 지면서 주목을 받기 시작했다. 초기의 내장형 시스템을 비롯한 여러 휴대 장치들에서의 소프트웨어는 장치 드라이버나 사용자 인터페이스 등을 제공하기 위한 단순한 형태만이 개발되었다. 그러나 최근에는 내장형 시스템이 하나 이상의 기능을 제공하거나 사용자가 설치하는 소프트웨어에 따라 다양한 기능을 제공할 수 있도록 설계 되어 내장형 소프트웨어의 역할이 크게 확대되었다. 내장형 시스템에 사용자들이 요구하는 소프트웨어의 기능은 점점 기존의 컴퓨팅 환경에서 제공되었던 기능을 그대로 요구하는 경우가 많아졌다[7]. 다시 말해 네트워킹, 멀티미디어 재생 기능 등 기존 환경(개인용 컴퓨터 등)에서 제공되던 복잡한 기능들과 게임, 정보 검색 등 기존의 응용 영역들을 요구하고 있다. 그러나 단순히 유사한 요구사항을 가진다는 이유로 PC에서 작동하는 소프트웨어가 PDA나 핸드-셋에서 작동될 수는 없다. 이질적인 환경들 사이에서 소프트웨어를 재사용 하기 위해서는 기존 소프트웨어가 작동되던 원천 영역(Source Domain)과 소프트웨어를 이식하려는 대상

영역(Target Domain) 사이의 차이점을 파악해야 한다. 두 개의 영역 사이에는 서로 다른 특성과 제약사항들이 있어 소프트웨어를 재사용함에 있어 장애가 될 수 있다. 즉, 이질적인 컴퓨팅 환경 사이의 소프트웨어 재사용은 단순히 소프트웨어 산출물을 다시 활용한다는 의미를 넘어서 소프트웨어의 작동 환경에 대한 이해 및 그 외 여러 가지 특성을 고려한 새로운 접근 방법이 필요하다. 이미 이질적인 환경 사이에서 소프트웨어를 재사용하기 위해 Horowitz는 기존 소프트웨어를 WWW(World Wide Web)으로 이식하는 연구[8], Cobb은 객체지향 데이터베이스를 WWW으로 이식하는 연구[9] 등을 진행하였다. 그러나 이 연구들은 WWW이라는 특정 환경만을 대상으로 삼았기 때문에 일반적인 이식에는 적용하기 어렵다. 즉, 특정 환경에 종속적이지 않은 소프트웨어 이식 절차가 필요하다.

3. 영역 지식 기반의 이식



[그림 1] 영역 지식을 이용한 이식 방안

소프트웨어를 다른 환경으로 이식하기 위해서는 여러 사항들을 고려해야 한다. [그림 1]은 영역 지식 기반의 소프트웨어 이식 방안을 보여준다. 첫째, 이식하려는 소프트웨어가 원래 작동되고 있는 원천 영역(source domain)과 소프트웨어를 이식하고자 하는 대상 영역(target domain)를 파악한다. 영역을 파악하기 위해 해당 영역의 전문가와의 인터뷰와 영역에 관련된 표준, 기술문서 등을 참조한다. 해당 영역내의 이식 대상 소프트웨어의 역할과 외부 시스템들을 파악하고 사이의 경계(boundary)를 명확히 파악한다[3]. 소프트웨어의 원천 영역과 대상 영역이 파악되면 두 영역의 영역 지식을 추출한다. 소프트웨어의 기능적인 면과 행위적인 측면을 분석하여 해당 영역의 기능적 제약사항, 기능적 불변 요구사항 등을 식별한다. 사용자의 요구사항 중 시장의 흐름이나 품질에 관련된 비기능적인 요구사항을 추출한 후 그것들과 기능적인 요구사항과의 연결점을 분석한다. 추출된 영역 지식은 소프트웨어를 이식하는 기준선이 된다.

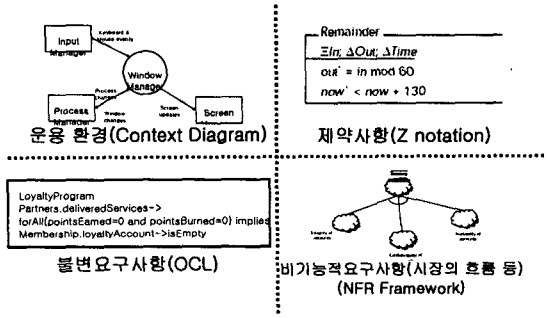
둘째, 파악된 영역 지식을 기반으로 이식할 소프트웨어를 원천 영역으로부터 대상 영역으로 이식한다. 원천 영역과 대상 영역 지식의 차이점을 파악한 후 원천 요구사항을 대상 영역에 맞게 변경한다. 설계 문서 및 코드를 대상 영역에 맞게 수정하는 작업을 수행한다.

마지막으로 이식된 소프트웨어가 변경된 대상 요구사항을 만족하는지 시험을 수행한다.

3.1 영역 파악 및 영역 지식 획득, 표현

영역 지식을 획득하기 위해서는 반드시 관심을 가지는 영역을 파악해야 한다. 영역 전문가(domain expert)를 이용하거나 해당 영역에서 개발된 소프트웨어에 관련된 표준, 기술문서 등 분석하여 영역을 파악한 후 영역 지식을 획득해야 한다. 영역을 파악하는 것은 크게 구조적인 측면에서의 파악과 데이터 흐름에 따른 파악으로 나눌 수 있다[3].

앞서 말했듯이 이식에 필요한 영역 지식은 매우 다양한 형태를 띠고 있다. 소프트웨어가 작동되는 운용 환경, 영역내의 제약사항, 불변 요구사항, 시장의 흐름 등이 그것이다. 운용 환경에 관련된 지식은 이식의 대상이 되는 소프트웨어와 소프트웨어 외부에 존재하는 개체들 사이의 관계를 묘사하는 구조적인 지식이다. 제약사항은 영역 내의 소프트웨어들이 반드시 준수해야 하는 조건이다. 제약사항은 하드웨어적인 특성 등에 의해 규정되는 기능적인 제약사항과 사용자의 요구사항과 경영 전략 등과 관련 있는 비기능적 제약사항으로 나누어진다. 불변 요구사항은 해당 영역의 사용자들이 해당 영역에서 작동하는 소프트웨어에 가장 기본적으로 요구하는 최소한의 요구사항으로서 제약사항과 관련이 있다. 제약사항과 불변 요구사항 사이의 차이점은 제약사항은 주로 소프트웨어 구현에 관련된 결정에 영향을 주고, 불변 요구사항은 결정의 대상이라는 것이다. 시장의 흐름은 해당 영역에 영향을 주는 동적인 요구사항이며 시간의 흐름에 따라 영역에 새로운 조건을 제시하기도 한다.



[그림 2] 영역 지식 표현 방식 예

영역 지식 기반의 소프트웨어의 이식에서 지식들을 충분히 획득하고 정확히 표현하는 것은 소프트웨어 이식 성공의 가장 핵심이 되는 절차이며 동시에 지식들을 하나로 통합하고 이식에 적용하는 것도 필수적인 절차이다. [그림 2]는 영역 지식과 그것을 표현할 수 있는 언어들의 예를 소개한다. 영역 지식을 표현하는 언어는 다음 조건들을 만족시켜야 한다.

기능적이고 행위적인 지식을 표현할 수 있어야 하며, 수치적인 지식을 표현할 수 있어야 한다. 시장의 흐름이나 사용자가 원하는 품질, 비기능적인 제약사항 등, 이식 대상 소프트웨어 외부의 비기능적인 지식도 표현할 수 있어야 한다[7]. 또한 지식들 사이의 포함, 배제, 확률적 연관성 등을 표현할 수 있어야 한다.

기존의 표현 방법들, 특히 정형적인 표현 방법들(Z[4], Petri Net, State Machine, OCL[6] 등)은 기능적이고, 행위적인 지식들을 표현하는 언어로서는 적합하지만, 비기능적인 지식들을 표현하는 것에는 부족한 점이 많다. 또한 그 언어들은 지식들 사이의 연관성을 명확히 표현하기에 표현력이 부족하다. 비기능적인 요구사항을 표현하기 위해 고안된 NFR Framework[5]은 지식들 사이의 상호의존성을 표현하고 있지만 기능적인 지식과의 연관성을 표현하는 것에 한계가 있다. 이를 해결하기 위해 영역 지식들을 일관된 하나의 지식으로 통합하는 방법 및 언어가 고안되어야 할 것이다.

3.2 요구사항 및 설계, 구현 산출물 이식

영역 지식을 획득하고 표현, 통합하는 것에 성공했다면 다음 단계는 원천 영역에 존재하는 소프트웨어의 요구사항을 대상 영역에 맞게 변경하는 것이다. 이 단계의 선행 작업은 원천 영역과 대상 영역의 지식을 서로 비교하여 차이점들을 찾아내는 것이다. 찾아낸 차이점들을 기반으로 소프트웨어의 요구사항을 대상 영역에 알맞도록 변형한다. 운용 환경의 차이점은 소프트웨어의 외부 환경에 존재하는 개체들을 변경하고 그들과 주고 받은 자료의 형태, 제어흐름의 변경을 위해 사용된다. 그리고 제약사항에 대한 차이점은 소프트웨어의 기능적이고 행위적인 모델과 품질에 관련된 요소에 영향을 준다. 불변 요구사항과 시장의 흐름에 대한 차이점은 비즈니스 규칙(business rule)과 같은 요구사항에 영향을 준다.

설계 및 구현에 관련 된 산출물은 이전 단계에서 변경된 요구사항과 양측 영역 지식의 차이점을 이용하여 변경된다. 아키텍처와 세부 설계에 관련된 산출물은 영역 지식의 차이점들을 기준으로 대상 영역에 적응하기 위한 형태로 변경되어야 한다. 코드와 같은 구현 산출물은 대상 영역에서 실행 될 수 있도록 대상 영역의 특성을 반영해야 하며, 이식된 요구사항을 반영하기 위해 새로운 부분은 첨가하거나 필요 없는 부분은 제거해야 한다.

이식 대상 소프트웨어에 관련된 문서는 두 영역 지식의 차이점을 반영하도록 수정되어야 하며 어느 지점이 어떤 지식을 근거로 변경되었는지 기록되어야 한다.

3.3 기타 산출물 이식 및 시험

마지막으로 이식된 소프트웨어가 이식된 요구사항을 정확히 반영하고 있는 시험한다. 이식된 요구사항은 원래의 요구사항에서 많은 부분이 수정/변경되었기 때문에 시험 방법도 변경될 수 있다. 만약 시험 방법의 변경이 필요하다면 대상 영역의 제약사항을 참고하며 시험 방법을 설계할 수 있다.

4. 결론 및 향후 연구

이동 환경에서의 정보처리에 대한 요구와 가전기기 등의 새로운 장치에서 소프트웨어의 역할이 증대되어 내장형 소프트웨어의 개발이 주목을 받고 있다. 사용자가 더욱 다양한 기능을 요구함에 따라 내장형 소프트웨어의 복잡도는 날로 높아지고 있으며, 시장의 경쟁이 치열해 짐에 따라 적은 비용과 짧은 시간에 소프트웨어를 개발해야 할 필요성이 부각되고 있다. 이것을 해결하기 위해서는 기존 환경에서 작동되고 있는 소프트웨어를 이식하는 것이 적절한 해결책이기는 하나 그에 대한 연구가 미비한 실정이다.

본 연구에서는 서로 다른 환경 사이의 소프트웨어 이식에 필요한 절차를 제안한다. 이식의 대상이 되는 소프트웨어가 개발된 원천 영역과 이식되는 대상 영역의 지식을 획득, 표현하고 차이점을 분석한다. 두 영역 지식의 차이점을 기반으로 소프트웨어의 요구사항, 설계, 구현 산출물을 이식하고, 이식된 소프트웨어는 변경된 요구사항을 기준으로 시험한다.

현재 영역 지식을 이용하여 SMIL Player를 Handset(BREW: Binary Run-time Environment for Wireless) 환경으로 이식하는 연구를 진행하고 있으며, 좀더 체계적인 이식을 위해서 영역 지식을 획득하는 절차, 표현하는 언어, 분석 및 통합 방법, 이식에 적용하는 방법 등이 추가적으로 연구하고 있다.

5. 참고 문헌

[1]Edward A. Lee, " What's ahead for embedded software?" , IEEE Computer, Vol. 33, Issue 9, pp. 18 -26, Sep 2000

[2]권수갑, " Ubiquitous Computing 개념과 동향" , 전자부품연구원 전자정보센터, Mar 2003

[3]K. Kang, et al., " Feature-Oriented Domain Analysis (FODA) Feasibility Study" , Software Engineering Institute, 1990.

[4]J. M. Spivey, The Z Notation: A Reference Manual, Prentice Hall, 1991

[5]L. Chung, et al., Non-functional Requirement in Software Engineering, Kluwer Academic Publishers, 2000

[6]Jos B. Warmer, Anneke G. Kleppe, The Object Constraint Language: Precise Modeling With Uml, Addison-Wesley, 1998

[7]Wayne Wolf, " What is Embedded Computing?" , IEEE Computer, Vol. 35, Issue 1, pp. 136-137, Jan 2002

[8]E. Horowitz, " Migrating Software To The World Wide Web" , IEEE Software, Vol. 15, Issue 3, pp. 18-21, May-June 1998

[9]M.A. Cobb, H. Foley et al, " An OO database migrates to the Web" , IEEE Software, Vol. 15, Issue 3 , pp. 22-30, May-June 1998