제품계열 방법을 응용한 동적 소프트웨어 구성 기법

황길승[0] 양영종
한국전자통신연구원 임베디드S/W연구단 S/W공학연구팀
{kshwang, yangyj}@etri.re.kr

The method of dynamic software adaptation by Product-Line approach

Kil-Seung Hwang[0] Young-Jong Yang
Software Engineering Research Team, Embedded S/W Research Division, ETRI(Electronic and Telecommunication Research Institute)

요 약

실행 환경 및 상황에 맞게 스스로 소프트웨어의 구성과 서비스를 변경시키기 위한 적응형 소프트웨어의 개발을 위해서는 변경 대상 및 변경 기법의 정의가 중요하다. 본 논문에서는 소프트웨어의 구성요소 및 기능을 실시간에 변경하기 위해 제품계열 방법에서 주로 사용되는 Information Hiding 기반의 가변성 관리 기법과 Parameterization 기반의 가변성 관리기법을 사용한다. 두 방법을 사용하면 실행 과정에 영향을 주지 않으면서 소프트웨어의 구성요소를 변경하거나 특성을 customizing할 수 있다.

1. 서 론

최근 Ubiquitous Computing이 중요한 컴퓨팅 패러다임으로 자리잡음으로써 하드웨어 뿐만 아니라 소프트웨어도 높은 적시성과 실시간성이 중요하게 요구되고 있다. 특히 임베디드 소프트웨어 또는 실시간 제어 소프트웨어 등은 상황변화에 종속적인 하드웨어의 특성에 따라 소프트웨어도 상황변화에 일부 또는 전체가 변경되는 동적 재구성(Self-reconfiguration) 기능이 필요하다.

소프트웨어의 상황에 따른 동적 재구성을 위해서는 소프트웨어의 가변성을 개발 단계에서 정의하고, 정의된 가변성을 소프트웨어가 실행되는 중에 실시간으로 제어하기 위한 기법이 중요한 기술이라 할 수 있다.

본 논문에서는 런타임 소프트웨어의 특성 변경을 위한 Parameterization 기반의 가변성 관리 기법과 능동적 재구성을 지원하기 위한 Information Hiding 기반의 가변성 관리 기법을 제안한다. 제안된 두 가지 가변성 관리 기법은 제품계열 기반 개발 방법에서 도입되어 사용되고 있는 가변성 관리 기법이다. 본 논문에서는 위의 두 가지 가변성 관리 기법을 이용한 소프트웨어 구성 방법과 동적 재구성을 위한 소프트웨어 시스템 구조를 제안한다. 제안된 소프트웨어 구조 및 가변성 관리기법을 사용하면 소프트웨어의 실행 과정에 영향을 주지 않으면서 소프트웨어의 구성요소를 변경하거나 특성을 재설정할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 서술하고 3장에서는 가변성 관리 기법에 대해 설명한다. 그리고 4장에서는 동적 시스템 재구성에 대해 설명하고 5장에서 결론을 맺는다.

2. 관련 연구

2.1 제품계열 기반 소프트웨어

80년대까지의 구조적인 S/W 개발방법은 S/W의 다양화, 대형화에 따른 요구의 증가로 90년대 초 객체지향 방법, 90년대 말 컴포넌트 기반 방법 등으로 진화해 왔고, 현재는 S/W 시스템 개발에 필요한 서비스 컴포넌트의 다양한 보급이 가능해지고, 컴포넌트 기반 기술이 성장함에 따라 시장성, 기술적 진보성이 보완된 제품계열 기반 개발방법, Model-Driven 개발방법, Service 기반 개발방법 등의 한단계 진화한 S/W 개발기술이 도입되고 발전하기 시작하고 있다.

S/W 제품계열은 공통의 유사한 기능을 가지고 있는 S/W 제품 또는 S/W 시스템의 집합을 의미하며[1], S/W 제품계열 기반 개발이란 특정 영역의 시장과 용도의 요구사항을 만족시키기 위해 S/W 제품을 미리 구축된 S/W 아키텍처 등의 S/W 핵심자산을 재사용하여 개발하는 방법이다. 이 방법은 미리 구축된 S/W 핵심자산을 재사용하므로, 처음부터 전체 시스템을 개발하는 방식보다 쉽고, 빠르게 S/W를 생산할 수 있는 장점이 있다.

2.2 가변성 관리

S/W 제품계열의 핵심은 S/W제품군에 공통적인 공통아키텍처를 기반으로 하여 각 제품별로 가지는 특징들에 대한 가변성을 제어하는 기술이다.

일반적으로 제품계열 기반의 방법들은 다음 4가지 정도의 가변성 관리 방법들을 사용한다.[2]

- Parameterization
- Inheritance
- Information Hiding
- Variation Point

Parameterization을 이용한 가변성 관리는 핵심 자산에 정의되어 있는 인수를 가변 요소로 보는 경우이다. Parameterization은 핵심자산의 attribute의 값을 변경시킴으로써 가변성을 결정하는 메커니즘을 가진다. 이 방법은 인수화된 attribute들의 값을 변경시키거나 초기화시키기 위한 컴포넌트의 인터페이스가 존재함으로써 가능하다. FODA(Feature Oriented Domain Analysis)[3], EDLC(Evolutionary Domain Life Cycle) Model[4], FAST[5] 등의 제품 계열 기반 개발 방법이나 도메인 분석방법에서 이 Parameterization 을 이용하고 있다.

Information Hiding을 이용한 가변성 관리는 동일한 인터페이스를 가진 여러 버전의 컴포넌트를 이용하여 가변성을 처리하는 방법이다. 각 버전의 컴포넌트 안에 가변성은 숨겨져 있는 형태이며, 이 경우 가변 요소는 각 버전의 컴포넌트 자체가 된다. 이 접근법은 변화되는 범위가 각 컴포넌트 단위 또는 그 이내로 제한되는 경우나 인터페이스가 변하지 않아야 하는 경우에 적용될 수 있다. 재사용의 관점에서 보면, 단지 선택가능한 컴포넌트의 집합에서 하나의 컴포넌트를 선택하여 소프트웨어에 삽입하는 것만으로 가변성을 결정하는 효과를 가져오게 된다. FODA, FAST, EDLC 등은 Parameterization 방법과 이 방법을 동시에 적용하고 있고, Defense Advanced Research Project Agency(DARPA) Software Technology for Adaptable, Reliable System(STARS)[6] 프로그램에서도 이 방법을 사용하고 있다.

Information Hiding 방법은 결정에 따라 적용되는 대상이 모두 동일한 인터페이스를 가지는 경우이고, Inheritance를 이용한 가변성 관리는 결정에 따라 가변적으로 제공될 대상이 동일한 인터페이스를 따르지 않는 경우에 적용될 수 있는 방법이다. 여기서는 Operation의 추가 및 변경 등을 통해 가변성이 제공되어 질 수 있다. KobrA[7,8], PuLSE[9] 등의 방법에서 이 방법이 사용된다.

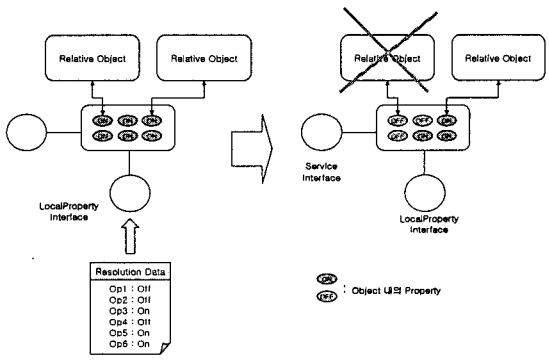
Variation Point를 이용한 가변성 관리는 Variation Point들의 집합으로 구성된 핵심 자산 컴포넌트를 이용하는 방법이다. 이 방법에서 개발자는 Variation Point들로부터 파생된 고유의 variants들을 선택함으로써 대상 시스템 컴포넌트를 구축할 수 있다. 이 방법은 개발자에게 고유의 variants를 생성하고 유지 관리하는데 있어 높은 유연성을 제공한다. 이 방법이 가지는 다른 방법들과의 차이점은 가능한 모든 variants들을 제공하고 개발자가 사용하고자 하는 방향대로 선택할 수 있도록 한다는 점이다. 하지만 예상된 variants 를 제외한 추가적인 경우의 발생에는 유연하게 대응하기 힘들다는 단점을 가진다

본 논문에서는 실시간으로 소프트웨어의 기능을 구성하기 위해 위의 4가지 가변성 관리 기법들 중 Information Hiding 방법과 Parameterization 방법을 사용한다. 소프트웨어가 환경 적응적으로 서비스를 제공하기 위해서는 다양하고 돌발적인 입력 데이터에 적절히 반응하는 것이 필요하다. Variation Point를 이용한 방법은 정해진 variants에 대해서만 유연성을 가지므로 적응형 소프트웨어에는 적용하기 힘들고, 소프트웨어의 실시간 변경을 위해서는 Inheritance를 이용한 방법도 적절하지 않다.

3. 소프트웨어의 동적 구성 기법

본 논문에서는 소프트웨어의 동적 구성을 두 가지 방법으로 구분하였다.

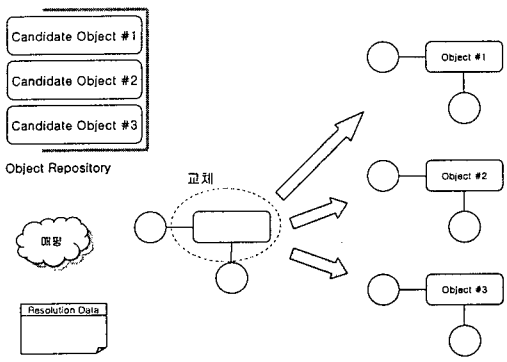
동적 구성의 첫 번째 방법은 소프트웨어의 구성 자체를 변경시키는 것이 아니고 소프트웨어 컴포넌트가 가지는 특성을 변경시키는 경우이다. 소프트웨어 컴포넌트가 가지는 특성을 실시간으로 변경시킴으로써 런타임 소프트웨어를 환경 및 상황에 적응시킬 수 있다. 이 레벨의 기법은 가변성 관리 모델의 Parameterization을 이용한 방법을 응용한 것으로써, 소프트웨어 컴포넌트가 개발될 때 특성을 변경하기 위한 메소드들을 외부에서 Parameterization으로 제어하기 위해 LocalProperty Interface를 정의하여야 한다. 환경 및 상황 데이터는 수집되어 적절한 형태로 가공된 후 LocalProperty Interface의 인수 형태로 제공되어 실행될 것이다.



<그림 1. Parameterization을 응용한 소프트웨어 구성>

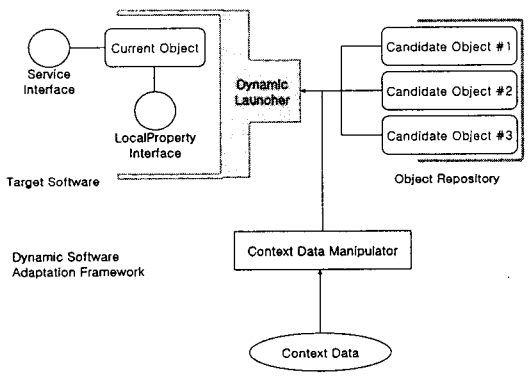
동적 구성의 두번째 방법은 전체 소프트웨어에서 특정 기능을 담당하는 소프트웨어 컴포넌트를 통째로 교환하는 경우이다. 이 경우는 서비스의 내용 자체가 변경되어야 할 경우에 유용하게 사용될 수 있다. 실제 상황과 맞는

객체를 특정 런타임 저장소에 저장해 놓고 상황이 일치하면 현재의 객체와 상황에 맞는 객체를 교환시키는 방법으로써, 가변성 관리 방법의 Information Hiding 방법을 응용한 방법이다. 이 방법을 사용할 때는 반드시 후보 객체들이 동일한 인터페이스를 가지도록 개발되어야 한다.



<그림 2. Information Hiding을 응용한 소프트웨어 구성>

설명한 소프트웨어 동적구성의 두 방법은 적절한 상황 데이터를 제공받을 수 있는 것을 가정하고 있으며, 상황 데이터를 가변성 결정 데이터로 매핑시키기 위한 별도의 장치를 필요로 한다. 이러한 모든 요구사항들을 고려하여 소프트웨어의 동적 구성 변경을 가능하도록 지원하는 동적 소프트웨어 적응 프레임워크는 다음과 같은 구조를 가질 수 있다.



<그림 3. 동적 소프트웨어 구성을 지원하는 프레임워크>

실제로 수집되어 가공된 상황데이터를 소프트웨어의 variants들과 매핑시켜 주는 역할을 담당하는 부분이 바로 Dynamic Launcher 이다. 여기서는 소프트웨어 내의 모든 가변 부분들에 대한 포인터를 가지고 있으며, 수집된 상황정보에 따라 적절하게 가변성을 결정하는 역할을 수행한다.

4. 결론

상황에 맞게 소프트웨어가 동적으로 기능 및 구조를 변경시키는 적응형 소프트웨어 분야는 유비쿼터스 컴퓨팅의 핵심 기술로써 중요성이 증대되고 있다. 본 논문에서는 기존의 소프트웨어 공학 기술을 응용하여 실시간으로 소프트웨어의 동적 구조변경을 가능케 하기 위한 하나의 기법을 설명하였다. 이 기법들은 실제로 런타임 상태의 소프트웨어를 상황이나 환경에 맞게 실시간 변경하는 기능이 필요할 때 유용하게 적용될 수 있는 기법들이라 생각된다. 향후 이 기법들의 실제 구현을 통해 실현 가능성을 시험해 나가는 과정이 필요할 것으로 본다.

5. 참고문헌

[1] Northrop L., " Framework for Software Product Line Practice" , SEI Report, 2002. (http://www.sei.cmu.edu/plp/framework.html)

[2] Hassan Gomaa, Diana L. Webber, " Modeling Adaptive and Evolvable Software Product Lines Using the Variation Point Model" , 37th Hawaii International Conference on System Sciences, 2004

[3] Kang, K., Feature-Oriented Domain Analysis, Technical Report No. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.

[4] H. Gomaa and G.A. Farrukh, Methods and Tools for the Automated Configuration of Distributed Applications from Reusable Software Architectures and Components, IEEE- Software, Vol. 146, No. 6, December 1999.

[5] D. Weiss and CTR Lai, Software Product Line Engineering, Addison Wesley, 1999.

[6] STARS. Software Technology for Adaptable Reliable Systems, The Product-Line Approach. Available online. (www.asset.com/stars/afdemo/prodline.htm)

[7] Atkinson, C., Bayer, J., Muthig, D., Component-Based Product Line Development: The KobrA Approach, Proceedings, 1st International Software Product Line Conference, 2000.

[8] Atkinson, C et al., Component-based Product Line Engineering with UML, Addison Wesley, Reading MA, 2002.

[9] Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J., PuLSE: A Methodology to Develop Software Product Lines, Proceedings of the Fifth Symposium on Software Reusability, 1999.