FCM환경에서 혼합클래스를 이용한 동적 위빙 프레임워크 설계에 관한 연구

박제연°, 송영재
{ jy_bak°, yjsong }@khu.ac.kr
경희대학교 컴퓨터공학과

A Study on Design of Dynamic weaving Framework using Mixin class in Fractal component model

Jea-Youn. Park, Young-Jae. Song
Dept of Computer Engineering, Kyung-Hee University

요 약

기존의 컴포넌트 모델에서는 컴포넌트 사용과 설정의 분리가 용이하지 않아 개발자가 제약사항(처리시간과 메모리)간의 트레이드 오프(trade-off)를 고려할 수 없을 뿐만 아니라 AOP를 적용하기 힘들다. 본 논문에서는 제어인터페이스를 임의적으로 추가할 수 있는 FCM(fractal component model)을 사용하여 aspect을 적용하였고, 런타임시 비기능적인 속성을 효율적으로 재구성하는 동적 위빙(dynamic weaving)을 지원하기 위해서, fractal component의 제어기능을 담당하는 membrane에 있는 구성요소 중 제어 객체(control object)와 인터셉터 객체(interceptor object)를 간단하게 선택하고 구성하기 위해 혼합클래스를 사용하는 프레임워크를 제안한다. 또한 aspect의 재사용성을 높이기 위해, AspectDataBase를 프레임워크에 설계하였다

1. 서 론

컴포넌트 기반 프로그래밍은 객체지향 프로그래밍이 갖는 바이너리 표준 문제와 버전 관리, 개발 언어의 비호환성 문제를 해결한 반면에 컴포넌트를 다양한 영역에 사용하기 위해 코드를 추가적으로 요구할 경우, 추가되는 코드가 흩어지거나 뒤엉켜져서 컴포넌트 재사용을 방해하게 된다. AOP(Aspect Oriented Programming)는 이러한 컴포넌트의 다양성 문제를 해결하기 위해 나온 방법이다.

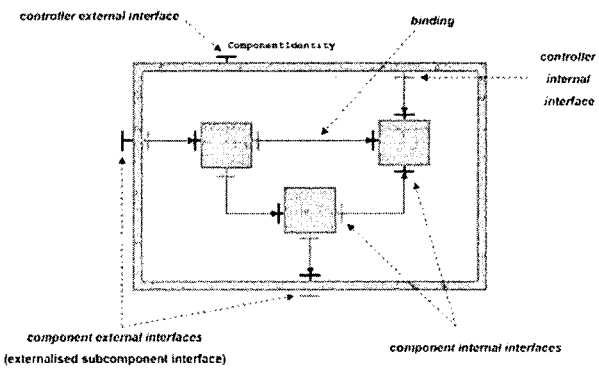
기존의 컴포넌트 모델에서 비기능적인 속성을 동적으로 추가하는데는 많은 어려움이 있으며, 따라서 어플리케이션 개발자가 제약사항(처리시간과 메모리)간의 트레이드 오프를 고려할 수 없었다. 제어 역행화(Inversion of Control), 즉 프레임워크에서 컴포넌트를 합성할 때는 컴포넌트의 설정을 사용에서 분리하는 방법으로 위의 문제를 해결할 수 있다[1]. FCM은 제어능력을 가진 open set을 컴포넌트에 추가하여 제어 역행화를 지원한다.

본 논문에서는 첫째, FCM에 aspect 개념을 추가하기 위한 joinpoint라는 제어 인터페이스(control interface)를 정의하고, 둘째, 런타임시 비기능적인 속성을 효율적으로 재구성하는 동적 위빙(dynamic weaving)을 지원하기 위해서, fractal 컴포넌트의 제어기능을 담당하는 membrane에 있는 구성요소 중 제어 객체(control object)와 인터셉터 객체(interceptor object)를 혼합클래스(mixin class)를 사용하여 간단하게 선택하고 구성할 수 있도록 하였다. 마지막으로는 aspect을 재사용하기 위해 AspectDataBase를 프레임워크에 적용하였다.

2. 관련 연구

2.1 FCM(fractal component model)

fractal component[2]는 복잡한 소프트웨어 시스템을 관리(모니터하고 동적으로 재조정)하고 배치하고 구현하기 위한 open set을 가진 컴포넌트 모델이다.



[그림 1] concrete fractal component model

컴포넌트의 내부특성에 대한 제어(intercession)와 내부·외부특성에 대한 인트로스펙션(introspection) 여부에 따라 타입이 분류된다.

fractal 컴포넌트 구조는 크게 두 부분으로 구성된다.

- content-실제 비지니스 로직에 해당하며, 서브 컴포넌트와 속성, 메소드들이 들어있다.
- membrane-컴포넌트 인터페이스가 들어 있으며,

오퍼레이션 호출을 중간에서 가로채어 컴포넌트 content를 모니터한다.

membrane에 크게 두 가지 컴포넌트 인터페이스가 있다. server interface, client interface로 구성된 기능적 인터페이스와, 컴포넌트를 재구성하고 인트로스펙션(introspection)하기 위해 비기능적인 특성을 담당하는 제어 인터페이스가 있다.

미리 정의된 제어 인터페이스로 life cycle 제어 인터페이스, attribute 제어 인터페이스, content 제어 인터페이스, binding 제어 인터페이스 등이 있으며 이러한 제어 인터페이스는 컴포넌트의 용도에 따라 임의대로 추가할 수 있다.

2.2 CBD-AOP 통합 및 위빙에 관한 연구

aspect and software component[3]에서는 aspect을 컴포넌트화하여 aspect에 대한 핵심 비즈니스는 advice component에 정의하고 point-cut과 위빙 규칙은 weaving component에 정의하였다. 그러나 런 타임시 동적위빙을 위해 클래스를 다 중상속하므로 코드의 길이가 길어진다.

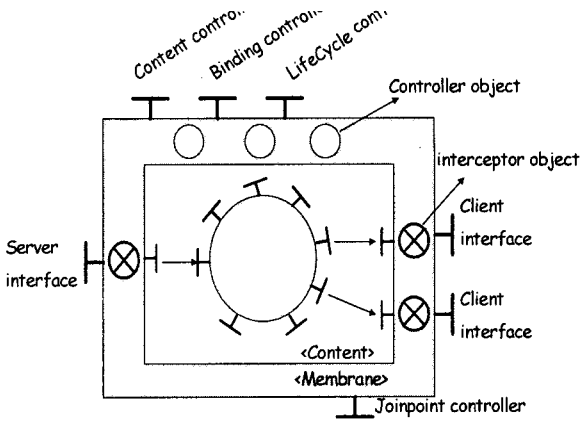
JAsCO[4]에서는 자바 빈즈 컴포넌트 모델에 AOP를 적용하였다. aspect bean에 pointcut의 종류와 처리시점(when-after, before), advice(what-확장 behavior)를 훅을 사용하여 명세하고, 커넥터에는 특정 컨텍스트내 어느 곳에 aspect bean을 배치할 것 인가를 명세하였다. 그러나, aspect bean과 커넥터를 분리하여 명세되어 있어 수정이 복잡한 문제점이 있다.

Jboss-Aop[5]에서는 jBoss(jeee based application server)를 사용하여 인터셉터를 통해 advice를 생성하고 poincut은 xml로 정의하였으나 aspect은 컴포넌트가 아닌 클래스여서 모듈화가 용이하지 않다.

Julia 프레임워크[6]에서는 FCM을 기반으로 컴포넌트 정적 구성과 동적 재구성을 지원하는 확장가능한 소프트웨어 프레임워크로, 분산시스템을 효율적으로 지원해준다. fractal 컴포넌트의 membrane을 프로그램하기 위한 프레임워크이며, 특히 제어 객체와 인터셉터 객체를 사용자가 쉽게 선택, 조립할 수 있도록 제어 객체에 대한 확장 open set을 제공한다. 반면에 모든 aspect을 재사용하지 않고 새로이 생성하므로 처리시간의 오버헤드가 발생할 수 있다.

3. 혼합클래스를 이용한 동적 위빙 FCM 프레임워크

본 논문에서는 프레임워크에서 컴포넌트를 합성할 때 컴포넌트의 설정을 사용에서 분리하기 위해 개방형 컴포넌트 모델(open component model)인 FCM에 cross-cuting되는 aspect을 관리하기 위해 제어능력을 가진 open set을 추가하였으며, 바이트코드 타입의 제어 객체와 인터셉터 객체를 사용자가 직접 선택하고 수정하여 컴포넌트간의 합성, 컴포넌트와 aspect간의 위빙(weaving)을 동적으로 지원하도록 설계하였다.



[그림 2] Fractal-AOP functional 컴포넌트 모델 구조

3.1 FCM-AOP 적용

Fractal-AOP의 기능적 컴포넌트 구조와 aspect에 해당하는 advice 컴포넌트 구조는 [그림 2]와 같으며, 두 joinpoint controller가 위빙(weaving)시 point cut 후보 중 하나가 된다.

content controller, Binding controller, LifeCycle controller, attribute controller등은 fractal component에서 미리 정의한 controller로서, attribute를 read/write, 컴포넌트의 라이프사이클 상태를 변경, 컴포넌트간의 연결/해제, 서브-컴포넌트의 추가/삭제를 결정할 수 있다.

join-point controller에는 미리 정해진 join point를 기술하고 동적 위빙을 지원하기 위한 인터페이스로 AspectDataBase에 저장될 수 있는 point cut에 대한 join point 후보들이다.

fractal-AOP 컴포넌트는 자바 객체로 표현하며, 크게 3그룹으로 나뉘어진다.

- 컴포넌트 인터페이스를 구현한 object로 [그림 2]에서 ㅜ 로 표현한 부분이며, 컴포넌트 인터페이스당 하나의 object가 있으며, 각 object에 대한 레퍼런스를 할당받아 object의 모든 메소드 호출을 위임받으며 client interface에 대한 레퍼런스는 null이다.
- component membrane을 구현한 object로 [그림 2]에서 ○⊗ 로 표현한 부분으로 비기능적인 특성을 제어하기 위한 controller interface를 구현한 controller object와, server/client interface에서 호출되는 메소드를 가로채는 interceptor object로 구성된다. 각 제어 객체와 인터셉터 객체는 다른 객체들과 통신하기 위한 해당 레퍼런스를 가질 수 있다.
- 컴포넌트 content를 구현한 객체로 [그림 2]에서 C에 해당하는 부분이며 실제 비니지스 로직이 들

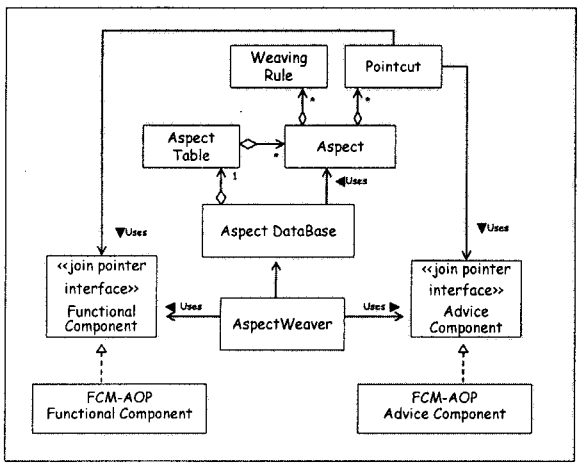
어있는 부분이다.

3.2 혼합클래스를 이용한 동적 위빙

FCM-AOP에서 기능적 컴포넌트에 cross-cutting되는 새로운 aspect을 동적으로 추가하기 위해 혼합 클래스를 사용하였다. 혼합클래스는 다중상속과 단일상속의 중간방법으로 속성은 제외하고 메소드만 다중 상속하여 복잡성을 줄이기 위한 방법이다. 여러 개의 제어 객체와 인터셉터 객체를 이용하여 join pointer controller를 생성하는 클래스를 혼합 클래스라 하며, 추상클래스를 슈퍼클래스로 가지고 있으며 추상클래스 멤버이름 중 _super_접두사가 붙은 멤버는 base class에서 상속하여 재정의할 수 있는 반면에 _this_접두사가 붙은 멤버는 base class에서 재정의할 수 없다.

동적위빙을 위해 FCM-AOP의 functional 컴포넌트와 advice 컴포넌트의 인터셉터 객체와 제어 객체를 혼합 클래스(mix-in class)를 사용하여 서브 클래스에 대한 바이트코드를 생성한다. 또한 혼합클래스를 사용하여 몇 개의 제어 객체를 단일 인터셉터 객체에서 관리할 수 있다. 이런 aspect을 동적으로 weaving하고 컴포넌트 간의 합성을 동적으로 재구성하기 위해, 다시 말해 바이트코드 생성기를 통해 미리 컴파일 된 functional component에 새로이 추가되는 join point를 위해 혼합 클래스를 사용한다.

3.3 제안하는 프레임워크를 위한 class-Diagram



[그림 3]프레임워크 클래스 다이어그램

[그림 3]은 fractal component에 동적위빙을 지원하기 위한 프레임워크의 클래스 다이어그램이다.

- AspectWeaver
  런 타임시 각 aspect과 컴포넌트를 동적으로 위빙한다. FCM-AOP의 object를 혼합클래스를 사용하여 join point controller interface에게 전달하며 이 정보를 AspectDataBase로 보낸다. 재사용할 수 있는 aspect이 없으면 advice component를 추가하여 직접 위빙한다.
  component와 system aspect간의 통신을 담당한다.
- AspectTable
  프레임워크에서 사용되는 모든 aspect을 포함하고 있으며, AspectDataBase에서 사용하며, 속성으로는 index와 functional 컴포넌트와 advice 컴포넌트의 join point controller 인터페이스 이름과 aspect 이름을 가진다.
- Aspect
  aspect object는 생성 시 Aspect Table의 column에 들어가고 해당 aspect에 대한 여러 point cut은 Aspect Table의 Row에 들어가서 해시테이블 형태의 Aspect table을 만든다.
- Point cut
  FCM-AOP의 functional 컴포넌트와 advice 컴포넌트의 point cut으로 설정할 join point에 관련된 정보를 저장한다.

4. 결론 및 향후 연구방향

본 논문에서는 제안하는 프레임워크는 컴포넌트의 설정과 기능을 분리하기 위해 fractal component model에 joinpoint controller를 추가하여 functional 컴포넌트와 advice 컴포넌트를 생성하였으며, 혼합 클래스를 이용하여 각각의 제어 객체와 인터셉터 객체를 선택적으로 합성하여 동적 위빙(Dynamic weaving)이 가능하도록 설계하였다. 또한 AspectDataBase에 미리 정의해둔 pointcut에 관련된 정보를 저장함으로 재사용을 높였다.

향후 연구과제로는 Aspect을 효율적으로 생성하고 관리하기 위해 AspectWeaver에 패턴을 적용하고, 제안한 프레임워크을 구현하여 JULIA, XWEAVER 등과 같은 다른 프레임워크과의 성능을 평가하고자 한다.

6.참고문헌

[1] Michael Mattsson, Jan Bosch, Mohamed E. Fayad, October 1999 Communications of the ACM

[2] http://fractal.objectweb.org

[3] Houssam Fakih, Noury Bouraquadi, International Workshop on Aspect-Oriented Software Development (WAOSD 2004)

[4]D Suv, W Vanderperren, V Jonckers. JAsCo: an aspectoriented approach tailored for component based software development. In Proceedings of the 2nd international conference on Aspect-oriented software development, pp 21-29. ACM Press, 2003.

[5]Bill Burke, Austin Chau, Marc Fleury, Adrian Brock, Andy Godwin, and Harald Gliebe. JBoss aspect oriented programming. http://www.jboss.org/,

[6]E Bruneton, T Coupaye,... An Open Component Model and Its Support in Java. In Proceedings of the International Symposium on CBSE, Scotland, May 2004.