2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

상호 작용 제약 조건을 기술할 수 있는 컴포넌트 결합 컨트랙트

백경원, 이정태, 류기열
아주대학교 정보 및 컴퓨터 공학부
e-mail : beeback@madang.ajou.ac.kr

Component Composition Contract Specifying Interaction Constraint

Kyung-Won Beak, Jung-Tae Lee, Ki-Yeol Ryu
College of Computer and Information, Ajou University

요 약

기존의 컴포넌트 기술 방법들이 특정 컴포넌트에 대한 기능적 기술에 상호 결합을 위한 문맥 조건을 추가하는 방식으로 되어 있으므로 발생하는 컨트랙트로서의 미흡한 기능 및 컴포넌트 간 재귀적 결합에 대한 지원 기능이 불충분하다. 본 논문에서는 이러한 문제점의 해결을 위하여 결합 컨트랙트라는 컴포넌트 컨트랙트 기술 방안을 제안하고 간단한 예를 들어 결합 컨트랙트의 구조를 보였다.

1. 서론

컴포넌트의 핵심 기술은 컴포넌트의 기술 부분(specification)과 구현 부분을 완전히 분리 함으로써 제 3 자가 구현의 세부를 고려하지 않고 잘 정의되어 있는 컴포넌트의 기술 부분을 활용하여 컴포넌트들을 결합 함으로서 응용 프로그램을 생성할 수 있는 환경을 제공하는데 있다.[4]

이를 위한 컴포넌트 기술 방법으로 기존의 객체지향 방식에서 사용되는 인터페이스에 컴포넌트의 결합 조건 기술을 위해 필요한 상호 문맥적인 내용 등을 precondition, postcondition 의 형태로 추가하는 등의 방법으로 컴포넌트를 기술하는 연구 등이 활발히 진행되고 있다.[2][4][5]

이렇게 기술된 컴포넌트의 기술 부분은 컴포넌트의 결합 시 상호 간 만족되어야 할 결합 조건을 기술하고 있다는 측면에서 컨트랙트로 표현되기도 한다.

그러나 현재 제시되어진 컨트랙트의 기술은 대부분 컴포넌트의 상호 결합 조건을 기술하기 보다는 특정 컴포넌트의 기능 및 이 기능의 사용 조건을 좀더 정확히 표현하는 방향에 대하여 주로 연구되어왔다. 최근 이러한 컴포넌트 기술 방법에 대하여 이는 컨트랙트로서의 기능을 완전히 표현하지 못하고 있을 뿐만 아니라, 컴포넌트 간 재귀적 결합을 지원하는데 충분하지 못하다는 점을 지적하는 연구결과 들이 발표되고 있다.[1][2][3]

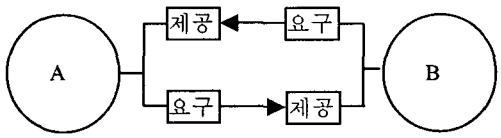
이에 본 논문에서는 이러한 문제의 해결 방안으로서 결합 컨트랙트라는 컨트랙트 기술 방안을 제시하고, 이 방법이 좀더 분명히 컴포넌트의 상호 결합 조건을 표현할 수 있고, 또한 재귀적 컴포넌트 결합을 지원할 수 있음을 보이고자 한다.

2. 결합 컨트랙트

컴포넌트의 기술 방법으로 기존의 객체 지향 방식에서 사용되는 인터페이스는 컴포넌트의 기능적 속성만을 기술하기 때문에 컴포넌트의 기술 방법으로 부족하다는 연구가 있다.[2] 따라서 컴포넌트의 문맥적 의미를 좀더 정확히 표현하기 위하여 인터페이스에 컴포넌트의 행위적 특징, 동기화 문제, 서비스의 질(QoS)등 컴포넌트의 특징을 추가하는 연구가 활발히 진행 되고 있다.[2]

또한 인터페이스는 사용자로부터 컴포넌트에 대한 한 방향 의존성을 기술하기 때문에 컴포넌트 간 상호 의존성을 표현하는데 적합하지 않다. 이에 컴포넌트 간 상호동작에 대한 문맥적 조건을 기술하기 위하여

컨트랙트를 기술하고 있다.[2] 컨트랙트의 조건 기술 방법은 컴포넌트의 각 서비스에 대하여 precondition, postcondition 의 형태로 추가하는 방법 등이 사용되어 컴포넌트의 결합 시 상호 간 만족되어야 할 결합 조건을 기술하고 있다. 컴포넌트 간 상호 동작 시 한 컴포넌트의 서비스는 이 서비스를 이용하는 컴포넌트와 서비스를 제공하는 측과 서비스를 요구하는 측으로 구분하여 생각할 수 있다. 따라서 컴포넌트의 인터페이스를 기술하는데 있어 각 기능들은 결합에 의한 상호 동작에 대하여 제공 서비스로 사용될 수 있으며 또한 요구 서비스로 사용될 수 있다.[그림 1]



[그림 1] 제공 서비스와 요구 서비스

[그림 1]에서 보는 바와 같이 두 컴포넌트가 결합하기 위하여 컴포넌트 A 는 제공 서비스를 구현하게 되고 이를 필요로 하는 컴포넌트 B 는 컴포넌트 A 의 요구 부분에 대한 충족을 위한 구현 부분이 있게 된다. 이와 같이 하나의 컴포넌트의 서비스는 경우에 따라 제공 부분으로 또는 요구 부분으로 구분되어 질 수 있고 이들 부분이 서로를 충족 시킴으로써 결합이 이루어 지게 된다.

컴포넌트 컨트랙트는 두 가지 구분될 수 있는데 컴포넌트 자체의 문맥적 의미를 정확히 표현하고자 하는 컴포넌트 컨트랙트와 결합에 참여하는 컴포넌트 간 상호 동작에 대한 상호동작 컨트랙트가 있다.[2] 그러나 컨트랙트는 상호 동작에 대한 의미 보다는 컴포넌트의 기능성을 보다 정확히 표현하는 기술로서 이해되고 있으며 결합에 대한 컨트랙트의 표현에 대한 연구가 아직 부족하다. 본 논문에서는 결합에 대한 컨트랙트의 표현 방법에 대한 방안을 제시하고 있다.

[그림 2]는 본 논문에서 제시하고 있는 결합 컨트랙트의 구조로서 크게 세 부분으로 구분되어 진다.

- 결합에 참여하는 컴포넌트 목록
- 결합에 참여하는 컴포넌트의 제공 또는 요구 서비스 목록과 결합 컴포넌트의 서비스로 등록
- 결합에 참여하는 컴포넌트의 상호 동작 조건

첫 번째 부분에서는 컴포넌트 결합 유형 중 단순 결합[2]에 참여하는 컴포넌트의 목록을 기술한다. 결합에 참여하는 컴포넌트는 컴포넌트 자체의 모습을 기술하는 자체 컨트랙트[2]를 가질 수 있으며 결합 컨트랙트에서는 결합에 참여하는 컴포넌트 목록에 기술되어 있는 모든 컴포넌트들의 컨트랙트를 계승 받게 된다. 따라서 결합 컴포넌트를 사용 시 결합에 참여하는 컴포넌트의 상호 동작 조건을 기술한 결합 컨트랙트를 사용할 뿐만 아니라 서비스의 요청이 목록에 기술되어 있는 각 컴포넌트에 이르게 되면 각 컴포넌트에 기술되어 있는 자체 컨트랙트를 사용하게 된다.

```
COMPOSITION CONTRACT component_name
IMPORT import_component_name_list
{ import_component_name
    (*REQUIRES*)
    service_name() : EXPORT
    service_name() : EXPORT
    ……
    (*PROVIDES*)
    service_name() : EXPORT
    service_name() :
    ……
}+
INTERACTION CONSTRAINT
    (*INIT*)
    service_name()
    (*CONSTRAINT*)
    { service_name() :
        interaction_constraint
    }+
```

[그림 2] 결합 컨트랙트 구조

두 번째 부분에서는 첫 번째 부분에서 기술된 결합에 참여하는 각 컴포넌트에 대하여 그들이 제공 또는 요구하는 서비스의 목록을 기술하게 된다. 결합에 참여하는 컴포넌트를 결합하여 새로운 결합 컴포넌트를 생성할 경우 이 결합 컴포넌트의 제공 또는 요구 서비스로서 결합에 참여하는 컴포넌트의 제공 또는 요구 서비스의 목록 중에서 선택되어 기술 된다 (**EXPORT**). 이로써 결합에 참여하는 컴포넌트를 이용하여 결합 컴포넌트를 표현함으로써 새로운 컴포넌트를 생성할 수 있고 생성된 컴포넌트는 다시 제 3 자에 의하여 다시 새로운 컴포넌트 결합에 참여 할 수 있다. 이는 컴포넌트를 결합하여 새로운 컴포넌트의 기술 부분에 대한 표현 방법을 제시 함으로서 재귀적 컴포넌트 결합(recursive component composition)을 지원하고 있다.

세 번째 부분에서는 다시 두 부분으로 세분되어 지는 데 먼저 결합 컴포넌트로서 동작하기 위한 초기화 부분이 있다(**INIT**). 초기화 부분은 결합 컴포넌트가 초기에 올바로 시동할 수 있도록 결합에 참여하는 컴포넌트의 서비스 중 일부를 이용하여 초기화 조건으로 기술 할 수 있다. 결합에 참여하는 각 컴포넌트의 컨트랙트에는 결합에 참여 했을 때 다른 컴포넌트와 상호 발생할 수 있는 모든 문맥적 의미를 기술하기 어렵다. 결합자의 입장에서 결합에 참여하는 모든 컴포넌트의 상호동작을 이해하고 조건지우는 방법이 더 유연하다고 본다. **INIT** 부분은 결합자의 입장에서 상호동작 중 먼저 발생해야 만 하는 문맥적 의미를 기술하는 방법을 제시하고 있다.

또한 두 번째 부분에서 기술한 결합 컴포넌트의 외부에 제공 또는 요구 되어지는 각 서비스에 대하여 결합에 참여하는 컴포넌트들의 상호 동작 조건을 기술 하는 부분이다(**CONSTRAINT**).

조건을 기술하는 방법과 내용에 대하여는 현재 많

은 연구가 진행되고 있지만 현재 제시되어진 컨트랙트의 기술은 컴포넌트 결합 시 결합에 참여하는 컴포넌트 간 상호 결합 조건을 기술하기 보다는 특정 컴포넌트의 기능 및 이 기능의 사용 조건을 좀더 정확히 표현하는 방향에 대하여 주로 연구되어 왔다. 그러나 특정 컴포넌트의 컨트랙트를 작성 시 결합에서 일어날 수 있는 문맥적 의미를 모두 표현하기 어려우며 결합 컨트랙트의 작성은 결합에 참여하는 각 컴포넌트의 모든 가능 문맥을 파악 한 후 제 3 자의 결합 의도에 부응하는 컨트랙트를 기술 하는 방법이 보다 유연하고 올바르다고 본다.

컴포넌트 결합 시 상호 동작 기술 방법에 대해서는 다양한 방법이 있을 수 있는데 본 논문에서 제시하는 방안에서는 상호 동작으로 발생하는 서비스 들의 실행 시퀀스(sequence)에 대한 조건을 기술 함으로써 결합 시 유지되어야 할 상호동작 조건을 표현하고자 한다. 상호 동작 조건을 표현하기 위하여 상호 동작 조건을 크게 두 가지로 분류하였다.

- must : 반드시 발생되어야 하는 시퀀스
- must not : 반드시 발생되지 말아야 하는 시퀀스

결합 컴포넌트의 서비스를 제공하기 위하여 결합에 참여하는 컴포넌트들 간의 상호 동작으로 발생하는 서비스의 실행 시퀀스 중 반드시 발생해야 하는 시퀀스(must)와 발생되지 말아야 하는 시퀀스(must not)가 존재하게 된다. 이를 컨트랙트 상에 명시함으로써 결합 컴포넌트의 문맥적 상호 결합 조건으로서 표현하고자 한다.

3. 결합 컨트랙트의 예

본 절에서는 파일 시스템의 서비스를 가지는 Directory 컴포넌트와 디렉토리를 화면에 보여주는 DirectoryDisplay 컴포넌트를 결합 시 사용될 수 있는 결합 컨트랙트의 예를 들어 설명하고 있다.

Directory 컴포넌트는 이 컴포넌트를 사용할 Notifier 컴포넌트의 등록과 해제에 대한 서비스와 파일 시스템의 삭제, 추가 변경 사항에 대하여 등록되어 있는 컴포넌트에 이 사실을 알려주는 서비스를 제공하며 각각의 서비스는 서비스를 사용하기 위한 조건으로 pre 와 post 조건을 가진다. [그림 3]은 Directory 컴포넌트의 인터페이스 컨트랙트의 모습을 보여주고 있다.

```
DEFINITION Directory;
  IMPORT Files;
  TYPE
    Name = ARRAY OF CHAR
    Notifier = PROCEDURE(IN name: Name);
  PROCEDURE AddEntry(n:Name;f:Files.File);
    (*pre and post condition*)
  PROCEDURE RemoveEntry(n: Name);
    (*pre and post condition*)
  PROCEDURE RegisterNotifier(n: notifier);
    (*pre and post condition*)
END Directory.
```

[그림 3] Directory 컴포넌트

[그림 4]는 Directory 컴포넌트에 자신을 등록 한 후 Directory 컴포넌트의 변화에 따라 변화된 내용을 전달해 오면 변화 된 내용을 화면에 보여주는 DirectoryDisplay 컴포넌트이다.

```
DEFINITION DirectoryDisplay;
  IMPORT Directory;
  PROCEDURE Notifier(IN n: Directory.Name);
    (*pre and post condition*)
```

[그림 4] DirectoryDisplay 컴포넌트

제 3 자는 위의 두 개의 컴포넌트를 결합하여 사용할 수 있으며 이때 두 컴포넌트 사이에 상호 동작이 이루어지며 이에 대하여 결합 컴포넌트 작성자의 작성 의도에 맞게 조건을 제시할 수 있는 컨트랙트가 필요하게 된다.

[그림 5]는 Directory 컴포넌트와 DirectoryDisplay 컴포넌트의 결합 컨트랙트의 한 예를 보여 주고 있다. 제 3 자는 두 컴포넌트를 **IMPORT** 하여 결합 하게 되는데 이때 결합에 참여하는 컴포넌트의 자체 컨트랙트는 그대로 계승 받게 된다. 즉 위의 두 개의 컴포넌트 컨트랙트 기술 부분의 pre 와 post 조건을 그대로 계승 받아 결합 컨트랙트의 조건이 동작할 뿐만 아니라 서비스의 요청이 결합에 참여 하는 컴포넌트에 이르게 되면 각 컴포넌트의 컨트랙트에 기술되어 있는 pre 와 post 조건을 수행하게 된다.

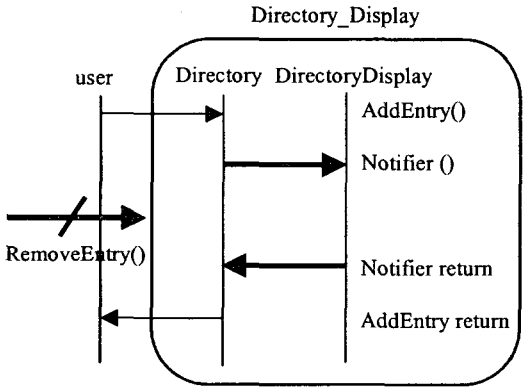
```
COMPOSITION CONTRACT Directory_Display
IMPORT Directory, DirectoryDisplay
Directory
    (*REQUIRES *)
        Notifier()
    (*PROVIDES *)
        AddEntry()                    : EXPORT
        RemoveEntry()                 : EXPORT
        RegisterNotifier()
DirectoryDisplay
    (*REQUIRES *)
        RegisterNotifier()            : EXPORT
    (*PROVIDES *)
        Notifier ()
INTERACTION CONSTRAINT
    (*INIT*)
    DirectoryDisplay .RegisterNotifier()
    (*CONSTRAINT*)
    Directory .AddEntry() :
      (*MUST*)        DirectoryDisplay. Notifier ()
      (*MUST NOT*)
          DirectoryDisplay. Notifier () ->
          Directory .RemoveEntry ()
    Directory .RemoveEntry() :
      (*MUST*)        DirectoryDisplay. Notifier ()
      (*MUST NOT*)
          DirectoryDisplay. Notifier () ->
          Directory .RemoveEntry ()
```

[그림 5] Directory_Display 결합 컨트랙트

다음으로 **IMPORT** 에 의해 결합에 참여하는 각 컴포넌트의 제공 또는 요구 서비스를 기술하게 되는 데 이중 제 3 자의 결합 컴포넌트의 작성 의도에 알맞은 서비스를 다시 결합 컴포넌트의 제공 또는 요구 서비스로 등록하여 사용하게 된다(**EXPORT**). 등록되어 사용되어 지는 서비스들은(AddEntry(), RemoveEntry(), RegisterNotifier()) 재귀적 컴포넌트 결합에 의하여 생성되는 결합 컴포넌트의 제공 또는 요구 서비스로서 사용되게 된다.

상호 동작 조건(INTERACTION CONSTRAINT)의 처음 부분에는 두 컴포넌트를 결합하여 시동시키기 위하여 DirectoryDisplay 컴포넌트가 Directory 컴포넌트에 등록되어 있어야 함을 의미한다. 결합 컴포넌트의 사용자는 우선 **INIT** 에 기술 되어있는 조건을 만족시켜야 한다. 즉 결합 컴포넌트 사용자는 DirectoryDisplay 컴포넌트를 등록하기 위하여 DirectoryDisplay 요구 서비스인 RegisterNotifier() 에 대응하는 제공 서비스인 Directory 컴포넌트의 RegisterNotifier() 서비스를 호출하여야 한다.

EXPORT 에 의하여 결합 컴포넌트의 제공 서비스로서 등록되어 있는 서비스에 대하여 결합에 참여하는 컴포넌트 간 상호 동작 조건을 기술 할 수 있다. 즉 결합 컴포넌트의 AddEntry() 서비스 사용 시 결합에 참여하는 컴포넌트의 상호 동작으로 발생할 수 있는 시퀀스 중 서비스가 시작 되면 DirectoryDisplay. Notifier() 서비스가 반드시 발생해야 하며 DirectoryDisplay. Notifier () -> Directory . RemoveEntry () 는 발생하지 말아야 함을 의미하고 있다[그림 6]. 이는 서비스가 시작되면 엔트리를 추가하는 동안에는 삭제할 수 없음을 의미한다.



[그림 6] AddEntry() 서비스에 대한 상호동작조건

이로서 결합에 참여하는 각 컴포넌트의 상호 동작에 대하여 문맥적 의미를 기술 할 수 있는 방법을 제시하고 있고 결합 컴포넌트 사용자는 결합 컴포넌트에서 제공하는 서비스인 AddEntry(), RemoveEntry()를 사용하여 다시 재귀적 결합 컴포넌트를 작성하는데 있어 보다 정확한 문맥을 이해 하며 작성하기 때문에 결합 컴포넌트의 작성 방법으로 보다 유연하고 올바르다고 본다.

4. 결론 및 향후 계획

본 논문에서는 기존의 컴포넌트 기술 방법들이 특정 컴포넌트에 대한 기능적 기술에 상호 결합을 위한 문맥 조건을 추가하는 방식으로 되어 있으므로 발생하는 컨트랙트로서의 미흡한 기능 및 컴포넌트 간 재귀적 결합에 대한 지원 기능이 불충분하다는 문제점 등의 해결을 위하여 결합 컨트랙트라는 컴포넌트 컨트랙트 기술 방안을 제안하였다.

본 논문에서 제안하는 결합 컨트랙트를 예에서 보인 바와 같이 기존의 컴포넌트 기술 방법에서 미흡하다고 간주되고 있는 컴포넌트 상호 간의 상호 연결 문맥을 좀더 명확히 기술할 수 있을 뿐만 아니라 재귀적인 컴포넌트의 결합을 가능하게 함으로서 다중티어(multi-tired)의 컴포넌트 시스템 아키텍처의 구축을 가능하게 할 수 있는 컴포넌트 프레임웍의 구현을 가능하게 할 수 있을 것으로 판단된다.

현재 본 연구진은 결합 컨트랙트의 상호 연결 문맥 부분을 좀더 명확히 기술될 다중티어 아키텍처를 위한 컴포넌트 프레임웍에 대한 연구를 계속 수행하고 있다.

5. 참고문헌

[1]Francisco Curbera, Sanfiva Weerawarana, Matthew J. Duftler, "On Component Composition Language", IBM T.J. Watson Research Center, 2000

[2]Felix Bachman, len Bass, Charles Buhman, Santiago Comella-Dorda, Fred Long, John Robert, Robert Seacord, Kurt Wallnau, "Volumn II: Technical Concepts of Component-Based Software Engineering", CMU/SEI-2000-RT-008, 2000

[3]A. Beugnard, J-M Jezequel, and D.Watkins. "Making Components Contract Aware. IEEE Computer, July 1999

[4] Clemens Szypaski, "Component Software Beyond Object-oriented Programming", Addison Wesley, 1998

[5] Koen De Hondt, Carine Lucas, Patrick Steyaert, "Reuse Contracts as Component Interface Descriptions", WCOP 1997