컴포넌트 조립을 위한 커넥터 설계 및 자동 생성

Connector Design and Automatic Creation for Components Assembly

채은주*, 한정수**
천안대학교 정보통신학부

Eun-Ju Chae(cobi80@cheonan.ac.kr)*, Jung-Soo Han(jshan@cheonan.ac.kr)**

요약

컴포넌트 기반 개발의 관점에서 시스템 개발은 코드를 작성하는 대신에 기존의 소프트웨어 컴포넌트들을 조립하는 것으로 대체되고 있으며 컴포넌트 조립에 관한 연구가 활발히 진행되고 있다. 본 논문에서는 컴포넌트 조립을 위하여 컴포넌트 사이의 관계를 명세화하여 정형화하였다. 커넥터를 통하여 컴포넌트를 조립하게 되면 메소드 호출이나 변경 없이 조립이 가능하다. 본 논문에서는 컴포넌트를 연결하는 커넥터 생성을 위한 명세와 제약조건을 정형화 하였으며, 작성된 커넥터 명세와 제약조건을 통하여 커넥터가 자동으로 생성되도록 하였다.

■ 중심어 : | 컴포넌트 | 커넥터 | 컴포넌트 조립 |

Abstract

In view of component based development, system development has replaced by assembling existent software components instead of coding and now component assembly research has progressed. In this paper, we formalized the specification using relationship between components for component assembly. If components are assembled through connector, the assembly is available without method call or modification. Therefore in this paper, we formalized the specification and constraint condition for creation of connector which connects components so that the connector is created automatically through the specification and constraint condition.

■ Keyword : | Component | Connector | Component Assembly |

I. 서 론

현재 소프트웨어 시스템은 새로운 기술로 전환함에 따라 높은 결함, 소프트웨어 품질 관리시의 어려움, 저 생산성, 높은 개발비용 그리고 제어상의 어려움이 야기되었다. 따라서 효과적인 비용으로 소프트웨어를 개발하는 기법을 찾는 것이 요구되어졌다. 가장 가능성 있는 솔루션 중 하나는 컴포넌트 기반 소프트웨어 개발 접근이다. 그래서 컴포넌트 기반 개발(CBD : Component Based Development)에 대한 연구가 활발히 진행되고 있다[1]. CBD는 객체지향 이후의 재사용을 목적으로 등장 하였으며, CBD는 소프트웨어의 재사용성과 빠른 생산성 그리고 품질을 높이기 위한 방법으로 확산되고 있으며, 새로운 애플리케이션 또는 시스템 개발은 컴포

접수번호 : #050322-002
접수일자 : 2005년 03월 22일
심사완료일 : 2005년 04월 08일
교신저자 : 채은주, e-mail : cobi80@cheonan.ac.kr

넌트 조립방법을 통하여 활발히 진행되고 있다. 즉 CBD 관점의 시스템 개발은 코드를 작성하는 대신에 기존의 소프트웨어 컴포넌트를 조립하는 것으로 대체되고 있다. 하지만 이질적인 환경과 언어로 인해 각각의 컴포넌트를 조립하여 사용하기란 매우 힘들다. 그래서 최근에는 이질적인 컴포넌트를 손쉽게 조립하는 연구가 국내·외로 활발히 진행되고 있다[2].

본 논문에서는 컴포넌트 조립을 위하여 컴포넌트 사이에 커넥터를 두어 컴포넌트들을 연결시켰다. 커넥터는 컴포넌트 사이에서 요구사항에 부분적으로 맞지 않은 부분을 조율하는 역할을 한다. 그래서 커넥터 설계를 통하여 컴포넌트들이 서로 정확하게 결합하였다. 또한 컴포넌트 조립을 위하여 커넥터 부분을 명세하였고 작성된 명세를 통하여 커넥터가 자동으로 생성되도록 설계 및 구현하였다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로 컴포넌트 조립한 논문들의 분석을 통하여 커넥터 자동 생성에 대한 필요성을 정의하였으며, 3장은 컴포넌트 통합 방법을 논의함으로써 커넥터를 통하여 컴포넌트의 효율적인 조립 방법을 채택하였다. 4장에서는 커넥터를 생성하기 위하여 커넥터에 대한 정의와 명세 그리고 아키텍처 제약조건을 통하여 커넥터를 설계하였다. 5장에서는 커넥터 설계를 통하여 커넥터가 자동으로 생성 할 수 있도록 구현을 하였다. 마지막으로 6장에서는 본 논문에 대한 결론과 향후과제를 논한다.

II. 관련 연구

컴포넌트 기반 개발은 애플리케이션 또는 시스템 개발할 때 개발비용을 줄이고, 컴포넌트의 재사용을 통하여 짧은 시간에 구축할 수 있다. 그래서 컴포넌트를 재사용하기 위한 컴포넌트의 생성, 조립 등에 관한 연구가 활발히 진행되고 있다. D.S.Rosenblum[3]은 C2 아키텍처 스타일에 따라 컴포넌트를 조립하는 ARABICA 모델링을 개발 하였다. 이 모델링은 C2 wrapping 방법을 통하여 기존 컴포넌트를 맵핑하거나 메시지 전달 방식의 인터페이스를 형성하여 컴포넌트를 조립한다. 이승연[4]은 플러그 앤 플레이 방식으로 유연하게 재구성 및 조립할 수 있도록 아키텍처 기반 설계로 개발된 COBALT Assembler의 컴포넌트 조립 도구를 개발하였다. C2 스타일을 따르는 아키텍처 기술 언어를 이용하여 정형화하며, 기술된 아키텍처 명세에 따라 생성된 컴포넌트 wrapper와 시스템 접속코드는 클라이언트 프로그램과 결합되어 수행될 수 있는 애플리케이션 시스템으로 구현되었다. Xiaodong Liu[5]은 애플리케이션 시스템을 구축할 때 요구사항에 맞지 않는 행동이 발생되지 않도록 하기 위하여 Adaptation 시나리오 기반으로 동적인 컴포넌트를 조립과 생성 기법을 이용하였다. XML을 이용하여 컴포넌트를 정의 명세를 통하여 컴포넌트의 변환과 확장이 용이하도록 설계 및 구축하였다. Tewfik Ziadi[6]는 컴포넌트 개념 (컴포넌트, 포트 그리고 커넥터)들을 정의하여 UML 확장을 통하여 플랫폼의 의존관계 모델을 정의하였다. OCL meta-level 제약조건과 맵핑 규칙에 의하여 명세 된 아키텍처 제약조건은 플랫폼 의존관계 모델을 이용하여 정의되며, 컴포넌트 포트를 조합하기 위하여 커넥터를 변환시켰다. 그리고 EJB와 CCM(CORBA Component Model)으로 맵핑한 컴포넌트 조립방법을 제시하고 있다.

모두 이질적인 컴포넌트 조립을 위하여 잘 정형화 된 어댑터를 이용하였다. 어댑터는 컴포넌트들을 연결 해주는 부분으로 컴포넌트의 요구사항(속성 및 매개변수)이 다를 경우 이를 조율 해주지는 않는다. 따라서 본 논문에서는 Tewfik Ziadi[6]이 정의한 컴포넌트간의 의존관계와 아키텍처 제약 조건을 이용하여 컴포넌트간의 요구사항을 조율해주는 커넥터를 자동 생성되도록 설계 및 구현하였다.

III. 컴포넌트 통합 방법

1. 컴포넌트 통합

1.1 포트 기반

상호작용과 정보의 흐름을 사용자가 보기 쉽게 표현한 컴포넌트를 조합하는 형태이다. 즉, 입력 포트와 출력 포트를 가진 컴포넌트를 조합하는 형태로써, 포트 고

유의 이름과 데이터 타입을 설정하고 동일한 타입의 입력 포트와 출력 포트를 상호 연결한다. 입력과 출력이 명확한 자료 처리 형태에 알맞으며, 컴포넌트간의 상호작용성 표현에 유용하다[2].

1.2 속성 기반

하나의 컴포넌트 속성들을 여러 구성요소로 분류하여 이들을 집합으로 구성하는 형태이다. 필요한 구성요소를 선택하여 조합하며, 사용자 관점의 참여에 따라 사용자 인터페이스 컴포넌트 조합에 유용할 수 있다. 구성된 속성 집합은 각각의 컴포넌트의 속성에 대응하여 서비스 가능하며, 정의된 속성에 따라 유동적인 정보 흐름을 제공한다.

1.3 아키텍처 기반

계층적으로 정의된 아키텍처에 속한 컴포넌트를 기능적으로 통합하는 것으로 비즈니스 컴포넌트의 통합에 이용될 수 있으며, 확장 가능하다. 통합은 특정한 도메인에서 서비스되는 컴포넌트를 선택하고, 그것에 정의된 컴포넌트를 지원할 수 있는 공통적이고 기술적인 컴포넌트를 공동 컴포넌트 계층에서 선택하고, 기본 컴포넌트를 지원하는 기본 컴포넌트 계층에서 선택하게 된다.

2. 컴포넌트 군 통합

2.1 점대점 연결

다른 컴포넌트와 간단하게 직접적으로 연결할 때 필요하다. 두 개의 컴포넌트 군을 고정하기 위한 것이 필요하며, 새로운 요구사항에 대해 통합된 컴포넌트의 재통합이 요구된다.

2.2 어댑터형 연결

어댑터는 이질적으로 개발된 각각의 컴포넌트에 대해 절충적인 역할을 함으로써, 통합 가능한 형태를 제공한다. 하나의 컴포넌트 군이 다른 컴포넌트 군에서 요구되는 서비스를 명세한 인터페이스를 제공한다. 그리고 서비스를 요구한 컴포넌트와의 통신이 불명확할 경우 동일한 인터페이스를 제공하며 인터페이스나 컴포넌트 자체에 의하여 정의된 프로토콜에 의하여 어댑터 형태로 서비스를 제공하게 된다. 또한 요구되는 서비스가 변경될 경우에는 새로운 어댑터를 적용함으로써 통합에 대한 융통성을 제공하게 된다.

2.3 허브를 통한 연결

컴포넌트 군이 서비스의 입출력을 담당하는 허브 중심으로 연결되는 형태이다. 허브는 컴포넌트간의 연결 타입, 내용의 명세 그리고 메시지 형태 등에 관하여 정의된 규칙을 포함하고 변경된 사항을 갱신하여 유연성을 제공할 수 있다.

2.4 커넥터를 통한 연결

컴포넌트 기반에서 컴포넌트들 사이를 연결하여 조립하는 방법이다. 커넥터는 포트들 사이의 제약 조건들이 존재하며 서로간의 연결 행위를 이용하여 상호작용으로 컴포넌트들이 조립된다.

따라서 본 논문에서는 커넥터에 대한 제약조건을 정의하고 이를 이용한 커넥터 자동생성을 목적으로 한다.

Ⅳ. 커넥터 설계

본 논문에서는 커넥터 생성을 위해서 도서관리 시스템을 이용하여 설명한다. [그림 1]은 도서관리 시스템에 관한 UML 컴포넌트 구성도이다. 도서관리 시스템은 Stock와 Library 두개의 컴포넌트로 구성되어있다. Stock 컴포넌트는 getTotalStock(), bookInput()와 bookOutput() 3개의 Operation이 있고 Library 컴포넌트는 bookReturn(), bookBorrow()와 totalResult() Operation이 있다. 두 컴포넌트 사이에 커넥터를 두어 연결할 것이다. 커넥터를 생성하기 위해서는 명세서가 필요하다. 명세서를 작성하기 위해서 몇·가지 규칙이 필요하며, 그 규칙은 다음과 같이 정의한다.

1. 정의(Definition)와 명세(Specification)

Components and Port. UML 클래스 스테레오타입 <<*component*>>에 의하여 컴포넌트가 명세 된다. 컴

포넌트는 포트들로 구성되며, 포트(Operation)는 카테고리를 기준으로 정의되는 연결부분이다. 포트의 기본 형태는 컴포넌트에 의하여 provided 또는 required로 동작하도록 정의된 operation 포트이다. Operation 포트는 UML 클래스 스테레오타입 <<operationPort>>로 명세 한다.

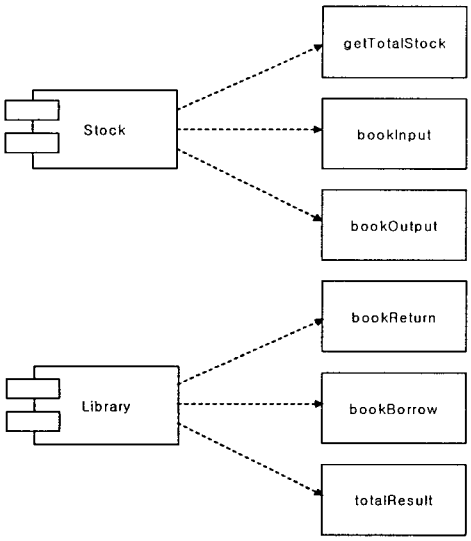

그림 1. 도서관리 시스템

그것은 포트 구조로 기술된 오직 하나의 UML operation을 포함하고 있다. 포트들은 UML 의존관계 스테레오타입 <<provided>> 또는 <<required>>에 의하여 컴포넌트들을 연결시킨다. 도서관리 시스템에서는 Stock 컴포넌트의 operation 모두 <<provided>>로 연결시키고 Library 컴포넌트는 bookReturn과 bookBorrow operation을 <<provided>>로 연결하고 totalResult operation을 <<required>>로 연결시킨다.

Connectors and Assembly. 컴포넌트 조립 명세는 컴포넌트 포트들 사이의 관계를 정의하는데 있다. *required*와 *provided* 포트의 개념은 두 포트의 사이를 연결시키는 것이다. 조립에 중요 요소로서 커넥터를 정의한다. 그것은 또한 호환성 없는 포트들로 적응하도록 사용될 수도 있다. UML 의존관계인 *provided* 또는 *required* 포트 중 하나를 스테레오타입 <<connect_type of ports>>으로 커넥터를 명세한다. Operation 포토들의 연결은 스테레오타입 <<connectOperation>>로 명세 한다. 도서관리 시스템에서는 Stock 컴포넌트의 bookInput operation과 Library 컴포넌트의 bookReturn operation을 연결하기 위한 두 operation 포트는 <<connectOperation>>으로 명세하고, Stock 컴포넌트의 bookOutput operation은 Library 컴포넌트의 bookBorrow operation과 연결하기 위하여 두 operation 포트는 <<conncetOperation>>으로 명세한다.

Ports Adaptation. 커넥터는 두개의 포트를 연결하기 위하여 사용된다. 어떤 컴포넌트의 operationPort들은 또 다른 컴포넌트의 operationPort들을 필요로 한다. 이때 이들 포트들 사이에 상호작용을 위한 조립 방법이 요구 된다. 따라서 포트들을 상호작용을 위하여 태그(tag)값 {adapt}를 붙여서 그 의미를 표시하며, 상호작용성은 맵핑 과정에서 처리된다[5]. 도서관리 시스템에서는 Stock 컴포넌트와 Library 컴포넌트를 조립하기 위하여 <<connectOperation>>으로 명세한 operation 포트에 {adapt} 태그를 붙였다. {adapt} 태그를 이용하여 커넥터 생성 명세서에서 조합할 operation 포트를 쉽게 정의할 수 있다.

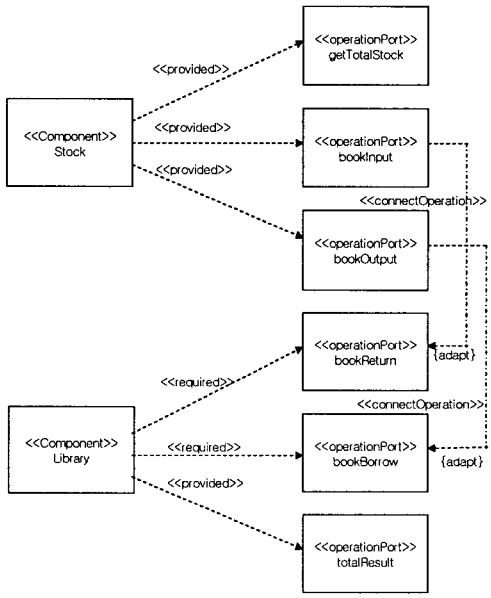

그림 2. 컴포넌트의 UML 클래스로 명세

표 1. 컴포넌트 명세

Component	operationPort	connect_type of ports	connectOperation
Stock	getTotalStock	provided	bookInput - bookReturn {adapt} bookOutput - bookBorrow {adapt}
	bookInput	provided	
	bookOutput	provided	
Library	bookReturn	required	
	bookBorrow	required	
	transfer	provided	

2. 제약조건의 정형화

확장된 UML 모델링에 대한 제한 조건들을 구조적인 규칙으로 정형화 시킬 수 있다. 이를 위하여 OCL(Object Constraint Language) 기반 제약조건의 정형화 구조는 [그림 3]과 같다. 의존관계의 스테레오타입 <<connectOperation>>은 스테레오타입 <<operationPort>>로 연결되어 있어야 한다. 즉, 의존관계 스테레오타입 <<connectOperation>>은 Stock 컴포넌트의 ModelElement나 <<operationPort>>로 표현된 포트를 제공받아 Library 컴포넌트의 ModelElement나 <<operationPort>>로 표현된 포트만을 연결되어야 한다. 따라서 의존관계 스테레오타입 <<connectOperation>>은 Stock 컴포넌트의 'bookInput' operation과 Library 컴포넌트의 'bookReturn' operation이 연결되고, Stock 컴포넌트의 'bookOutput' operation과 Library 컴포넌트의 'bookBorrow' operation이 연결되었다.

```
context Dependency

inv self.isStereoptyped("connectOperation") implies
(( self.supplier → forAll(S:ModelElement |
S.isStereotyped("operationPort")) and (self.client →
forAll(C:ModelElement | C:.isStereotyped("operationPort"))
```

그림 3. 제약조건에 대한 OCL 표현

V. 커넥터의 자동 생성

커넥터를 자동으로 생성하기 위하여 [그림 3]의 문맥의존관계를 통하여 [그림 4]의 커넥터 자동생성을 위한 규칙 명세서를 작성한다. 이 규칙 명세서를 통하여 [그림 5]의 커넥터 인터페이스를 이용한다. 사용자는 컴포넌트의 operationPort를 입력하면 [그림 6]과 같은 XML문서로 된 커넥터 명세서가 자동으로 생성된다. 이 명세서를 통하여 [그림 7]과 같이 커넥터가 자동으로 생성된다.

```
bookInput - bookReturn {adapt}
  supplier : Stock Stock.bookInput
  client : Library Library.bookReturn
bookOutput - bookBorrow {adapt}
  supplier : Stock Stock.bookOutput
  client : Library Library.bookBorrow
```

그림 4. 커넥터 규칙 명세서

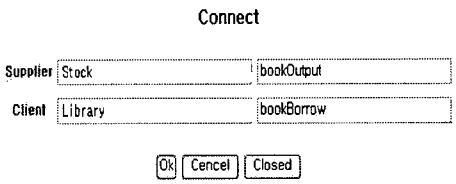

그림 5. 커넥터 인터페이스

```
<?xml version="1.0" encoding="euc-kr"?>
<connectOperations>
  <adapt>
    <supplier>
      <component>Stock</component>
      <operation>bookInput</operation>
    </supplier>
    <client>
      <component>Library</component>
      <operation>bookReturnt</operation>
    </client>
  </adapt>
  <adapt>
    <supplier>
      <component>Stock</component>
      <operation>bookOutput</operation>
    </supplier>
    <client>
      <component>Library</component>
      <operation>bookBorrowt</operation>
    </client>
  </adapt>
</connectOperations>
```

그림 6. 생성된 커넥터 명세서

[그림 4]의 명세를 이용하여 커넥터가 자동 생성되며, 생성과정은 {adapt} 태그를 찾는다. 그리고 스테레오타입 <<connectOperation>>으로 나타난 operation에서 supplier를 보고 의존하고 있는 컴포넌트를 찾아 커넥터에 그 컴포넌트의 객체를 생성한다. 생성된 객체의 스테레오타입 <<operationPort>>를 찾아 객체 참조를 한다. 그리고 그 객체를 사용할 클래스를 찾아 그 컴포넌트의 스테레오타입 <<operationPort>>를 찾아 의존상태의 컴포넌트를 사용하게 된다. 예를 들면, Library 시스템에서 스테레오타입 <<connetOperation>>의 의미는 Stock 컴포넌트의 bookInput()과 bookOutput() operation이 Library 컴포넌트의 bookReturn()과 bookBorrow()에 의존관계로 존재함을 뜻한다. 따라서 supplier의 스테레오타입 <<operationPort>>는 Stock 컴포넌트의 bookInput()과 bookOutput() 이기 때문에 Stock 컴포넌트의 객체를 생성하게 된다. 그리고 client에서 스테레오타입 <<operationPort>>로 표현된 Library 컴포넌트의 operation은 bookReturn()과 bookBorrow()이므로 커넥터가 이용할 operation이 된다. 따라서 [그림 7]과 같이 커넥터가 생성된다.

```
public class connector
{
  Stock stock = new Stock();
  public void bookReturn()
 {
    stock.bookInput();
  }
  public void bookBorrow()
 {
    stock.bookOutput();
  }
}
```

그림 7. 커넥터 생성

Ⅵ. 결 론

최근에 새로운 애플리케이션 또는 시스템 개발은 컴포넌트 조립방법을 통하여 활발히 진행되고 있다. 즉 컴포넌트 기반 소프트웨어 개발의 관점에서 시스템 개발은 코드를 작성하는 대신에 기존의 소프트웨어 컴포넌트를 조립하는 것으로 대체되고 있다. 컴포넌트 조립에 있어서 커넥터 기능은 컴포넌트 사이에서 요구사항에 부분적으로 맞지 않은 부분을 조율하는 역할을 한다. 따라서 본 논문에서는 커넥터 연결 방법을 사용하여 컴포넌트를 조립하였다. UML를 통하여 컴포넌트를 정의하였고 제약조건을 통하여 컴포넌트간의 의존관계를 정형화시켰다. 또한 정형화된 의존관계를 통하여 커넥터에 대한 XML 명세서를 설계하였다. 이를 이용하여 커넥터를 자동 생성하고, 컴포넌트간의 조립이 가능하도록 하였다. 그러나 컴포넌트 조립 시 각각의 컴포넌트의 매개 변수가 일치하지 않았을 경우가 발생한다. 이 부분을 어떻게 해결을 해야 할 것인가에 대한 연구가 부족하다. 민현기[7]의 논문에서는 데이터 변환 커넥터 패턴, 기능 변화 커넥터 패턴 그리고 인터페이스 어댑터 커넥터 패턴을 이용하여, 매개변수 불일치를 해결방법을 제시하였다. 따라서 매개변수의 불일치를 해결하기 위한 연구가 진행되고 있다.

참 고 문 헌

[1] E. V. Berard, Essays "Object-Oriented software engineering," Vol.1, Prentice Hall, 1993.

[2] 송영재, 김귀정, 변정우, 서영준, 최한용, 한정수, 객체지향 모델링과 CBD중심 소프트웨어공학, 이한출판사, 2004.

[3] D.S.Rosenblum and R.Natarajanr, "Supporting architectural concerns in component interopera-bility standards," Proceedings of IEE Software, Vol.147, No.6, pp.215~223, 2000

[4] 이승연, 권오천, 신규상, "아키텍처에 기반한 컴포넌트 조립 시스템의 설계 및 구현 방법과 지원 도구의 개발", 정보과학회, Vol.30, No.9, pp.812~820, 2003.

[5] X. Liu, B. Wang and J. Kerridge, "Achieving seamless component composition through scenario-based deep adaptation and

generation," Science of Computer Programming, Vol.56, No.1~2, pp.157~170, 2005.

[6] T. Ziadi, B. Traverson, and J. M. Jezequel, "From a UML Platform Independent Component Model to Platform Specific Component Models," Workshop in Software Model Engineering, 2002.

[7] 민현기, 김수동, "컴포넌트 프레임워크 설계를 위한 실용적인 커넥터 패턴", 정보과학회 논문지, Vol.31, No.1, pp.43~53, 2004.

저 자 소 개

채 은 주(Eun-Ju Chae) 준회원

- 2003년 2월 : 목원대학교 컴퓨터공학(공학사)
- 2003년~현재 : 천안대학교 정보기술대학원 재학중

<관심분야> : 소프트웨어 공학, 컴포넌트 조립 및 유지보수, 소프트웨어 아키텍처

한 정 수(Jung-Soo Han) 종신회원

- 1990년 : 경희대학교 전자계산공학과(공학사)
- 1992년 : 경희대학교 전자계산공학과(공학석사)
- 2000년 : 경희대학교 전자계산공학과(공학박사)
- 2001년~현재 : 천안대학교 정보통신학부 조교수

<관심분야> : 소프트웨어공학, CBD, OOP, S/W 개발 방법론