DOI: 10.3745/KIPSTD.2009.16D.5.715

오픈소스 기반의 JBI 컴포넌트배치도구의 설계 및 구현

박 제 원[†] · 최 재 현[†] · 이 재 성[††] · 이 남 용[†††]

요 약

JBI(Java Business Integration)기반의 ESB(Enterprise Service Bus)는 웹서비스표준으로 서비스의 유통경로를 구현한 SOA(Service Oriented Architecture)의 한 방법이다. 광범위한 벤더로 구성된 JBI기반의 ESB는 서비스를 플러그인 할 수 있는 확장성의 장점을 가지고 있지만 사용자가 서비스유닛을 사용하기 위해선 복잡한 절차의 패키징과정을 수동으로 연동해야 하고, 순차적인 절차를 따른 배포방법을 사용해야 하는 비효율적 유지관리의 문제점을 가지고 있다. 본 논문에서는 이러한 문제를 해결하기 위해 컴포넌트간의 연동을 위한 모델링에서 배포의 전 과정을 통합하여 관리할 수 있는 사용자 중심의 컴포넌트 배치도구를 설계 및 구현하였다. 배치도구의 설계를 위해 첫째, 기존 JBI를 지원하는 배치도구를 분석하여 문제점 및 개선사항을 도출하였다. 둘째, JBI 기반의 ESB에서 가지고 있는 문제점 등을 기반으로 요구사항을 도출해 내었다. 그리고 최종적으로 JBI를 지원하는 컴포넌트의 스키마를 분석을 통한 모델링의 속성 및 패키징, 배포, 검증 기능을 설계하고 구현하였다. 본 논문에서 제안하는 도구를 이용하면 사용자는 컴포넌트의 연동을 위한 모델링에서 배포의 전 과정을 별도의 도구와 수작업 없이 배치도구 내에서 통합적으로 수행하고 관리할 수 있으며 GUI기반의 컴포넌트 모델링과 관리를 통하여 사용자는 사용성과 편리성을 높일 수 있다. 또한 컴포넌트의 유효성검증 및 경로에 대한 유효성검증을 통하여 서비스유닛을 배포하는데 발생 할 수 있는 오류를 최소화 할 수 있을 것으로 기대된다.

키워드 : 자바비지니스통합, 엔터프라이즈 서비스 버스, 패키징, 서비스유닛

Design and Implementation of JBI Component Deployment tool Based on the open sources

Park Jae Won[†] · Choi Jae Hyun[†] · Lee Jae Sung[††] · Lee Nam Yong[†††]

ABSTRACT

The Enterprise Service Bus based on Java Business Integration is an web service standard and one of the methods for implementing distribution channels of Service Oriented Architecture. Consisting of open source group, extensive venders and users, the ESB based JBI has the problems of ineffectiveness as well as advantages of extensibility of service plug-in. That is, in case users need to use Service plug-in, manual connection of packaging process and sequential distribution method is required. This study, therefore, proposes as a way of trouble-shooting the user-oriented component deployment tool which can manage entire process for deploying The ESB middleware platform to Service unit. At the same time, this study elicited the requirements based on issues of JBI-based ESB and has developed the modeling property, packaging, distribution and evaluation thru Schema analysis of JBI-compatible component. Using the deployment tool this study proposed, users will be able to perform and manage the whole deploying process without additional manual work for connecting component. Not only it is expected that interface based on Graphic User Interface provide usability and convenience but they can also minimize the errors rate through component and route validity verification function provided in deployment tool.

Keywords : Java Business Integration(JBI), Enterprise Service Bus(ESB), Packaging, Service Unit

1. 서 론

JBI기반의 ESB(Enterprise Service Bus)는 SOA를 지원하며 웹서비스기술기반으로 서비스의 유통경로를 구현한 미들웨어플랫폼이며 JBI아키텍처를 기반으로 다양한 벤더와 사용자로 구성되어 있는 오픈소스그룹이다[8,9]. 오픈소스그룹의 다양한 벤더들은 JBI환경에 적합한 웹 표준기반의 컴포넌트들을 배포하고 있으며 사용자는 사용자가 사용하고자 하는 컴포넌트를 ESB에 서비스유닛화하여 플러그인하고 컴포넌트를 사용한다[7]. 이러한 JBI기반의 ESB는 확장성이라는 큰 이점을 가지고 있지만 그에 따른 몇 가지 단점을 가

※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 준 회 원 : 숭실대학교 컴퓨터학과 공학석사
††† 정 회 원 : 숭실대학교 컴퓨터학과 교수
논문접수 : 2009년 3월 3일
수 정 일 : 1차 2009년 4월 21일, 2차 2009년 6월 22일
심사완료 : 2009년 6월 22일

지고 있다. 첫째, 다양한 벤더들의 컴포넌트 배포로 인하여 사용자는 사용하려는 컴포넌트와 컴포넌트를 수동으로 연동하고 사용해야 한다[11]. 사용자가 기존의 서비스유닛[1]에 새로운 컴포넌트를 추가하거나 새로운 서비스유닛을 만들기 위해서는 별도의 수작업을 통하여 컴포넌트와 컴포넌트를 연동해야 한다.

둘째, 컴포넌트와 컴포넌트를 연동한 후 이를 서비스유닛으로 만들기 위한 컴포넌트 패키징과정을 거친다. 컴포넌트 패키징은 경량화와 스탠다드방식으로 분류되며 방식에 따라 패키징하는 과정도 다르다. 예를 들어 스탠다드방식은 컴포넌트단위에서 이루어지는 인자 및 변수를 JBI기반의 형식으로 변환하고, 컴포넌트 단위별로 패키징을 한 다음 최종적으로 연동되는 컴포넌트와 컴포넌트를 제어하기 위한 메타폴더를 생성하고 위 각 컴포넌트 단위별 패키징과 메타폴더를 패키징하기 위한 통합과정을 거쳐야 서비스유닛으로 사용할 수 있다. 사용자는 각 절차를 수작업으로 해야 하는 번거로움이 있다.

셋째, 사용자가 컴포넌트 및 서비스유닛은 순차적 절차로 이루어져 있어 이를 관리하고 편집하기위해선 폭포수모델처럼 처음단계에서부터 시작해야 하는 비효율적 유지보수관리의 문제를 지닌다. 패키징과정 도중 오류를 발견 하거나 패키징된 서비스유닛을 추가 또는 삭제해야 할 경우에 사용자는 모델링과정부터 다시 해야 하는 어려움이 있다. 또한 컴포넌트에서 사용되는 사용자파일이나 서비스운영에 필요한 라이브러리파일을 추가하거나 삭제해야 할 경우에는 별도의 수작업으로 패키징된 컴포넌트 및 서비스유닛에 삽입해야 하는 불편함이 있다.

컴포넌트를 ESB에 플러그인하여 사용하기위한 과정은 크게 컴포넌트 연동을 위한 모델링, 오류를 최소화 하기위한 유효성검증, 각 패키징방식에 따른 패키징, ESB에 플러그인하기 위한 배포의 4가지 과정으로 구분할 수 있다. 이 과정을 지원하기 위한 도구로는 오픈소스 프로그램인 Ant, Maven 등이 있으며 이를 통해 ESB에 플러그인 할 수 있다. 하지만 다양한 벤더들이 제공하는 컴포넌트에 비해 사용자가 사용하고자 하는 서비스의 유통경로를 모두 서비스화하기에는 너무 많은 경우의 수가 존재하기 때문에 이를 지원하는 컴포넌트의 배치도구의 개발이 시급한 실정이다. 본 논문에서는 오픈소스기반의 ESB이 지니고 있는 몇 가지 단점과 컴포넌트를 ESB에 배포하기까지의 과정에 사용되는 여러 종류의 자동화도구를 하나로 통합하여 이를 동적으로 수행하고자 JBI를 지원하 는 컴포넌트 배치도구를 제시한다. JBI를 지원하는 컴포넌트 배치도구는 크게 4가지 기능으로 구성되어있다. 첫째, 컴포넌트간 연동을 위한 모델링기능과 컴포넌트 편집기능, 둘째, 컴포넌트의 네임스페이스 및 속성 등에 대한 유효성 검증기능, 셋째, 연동된 컴포넌트를 JBI기반의 XML(eXtensible Markup Language)형식으로 변환하고 통합하기 위한 패키징기능, 넷째, 패키징된 서비스를 ESB에 플러그인하기 위한 배포기능의 범위를 가진다.

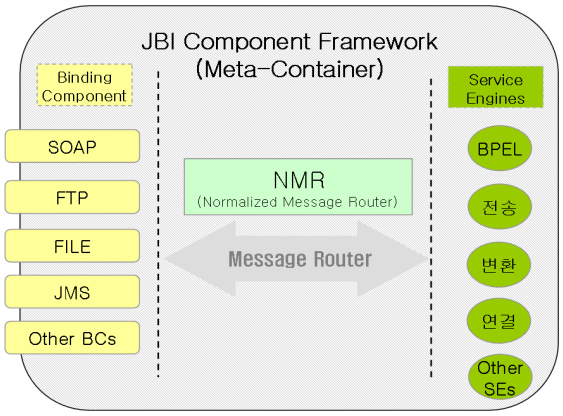
JBI를 지원하는 컴포넌트 배치도구는 컴포넌트모델링과 패키징단계에서 필요에 따라 라이브러리 및 컴포넌트를 추가, 삭제, 갱신할 수 있는 동적인 편집기능과 유효성검증에 대한 관리를 제공한다. 또한 배포기능을 통하여 ESB에 서비스유닛에 대한 배포 및 제거를 통합적으로 수행할 수 있어 정확하고 편리한 서비스를 제공할 수 있다. 컴포넌트의 모델링에서 배포까지의 과정에서 사용되는 여러 종류의 자동화도구를 하나로 통합하여 모든 작업을 한 번에 수행 할 수 있으며 그 과정에서 누락되고 발생할 수 있는 오류를 최소화 할 수 있을 것으로 생각된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 통해 JBI 컴포넌트 프레임워크와 현재 사용되고 있는 JBI를 지원하는 컴포넌트배치도구에 대해 알아본다. 3장에서는 본 논문에서 제안하는 배치도구의 모델에 대한 요구분석과 아키텍처에 대해 설명하고 각 기능에 대해 설명한다. 4장에서는 3장에서 제시하고 있는 아키텍처와 도출된 요구사항에 대한 상세설계를 제시한다. 5장에서는 구현된 컴포넌트배치도구와 세부적으로 모델링, 패키징 및 배포기능의 동작과정을 사례연구를 통해 이를 검증한다. 6장에서는 2장의 관련연구에서 제시하고 있는 현재 사용되고 있는 컴포넌트배치도구와의 비교평가를 통해 제시하는 도구의 우수성을 입증한다. 마지막으로 7장에서는 결론 및 향후 연구과제에 관하여 논의한다.

2. 관련 연구

2.1 JBI컴포넌트 프레임워크

JBI 컴포넌트 프레임워크는 (그림 1)의 컨테이너개념으로 바인딩컴포넌트, [2]서비스엔진컴포넌트로 구성되어 있으며 NMR(Normalized Message Router)을 포함하고 있다[10].



(그림 1) JBI 컴포넌트 프레임워크

1) 서비스유닛 (SU): 단위서비스로 정의되며, 서비스의 요청/응답과 서비스엔진간의 연결통로이다. 서비스 요청/응답은 서비스 유닛을 통해 서비스엔진으로 전달된다. 본 논문에서는 서비스 유닛은 메타폴더와 동일한 의미로 사용된다.

2) 서비스엔진(SE) : 서비스 요청을 받아 특정 기능을 수행하고 이에 대한 응답을 생성하는 컴포넌트로 서비스를 제공하는 주체이다.

서비스엔진컴포넌트는 BPEL(Business Process Execution Language), 메시지 전송, 메시지변환, 메시지연결 등의 비즈니스로직에 대한 구현부분으로 간단한 변환기능을 가지는 로직에서 BPEL같은 매우 복잡한 비즈니스프로세스가 처리되는 로직이 구현될 수 있다. 바인딩컴포넌트는 SOAP(Simple Object Access Protocol), FTP(File Transfer Protocol), FILE, JMS(Java Message Service)처럼 설치되어있는 서비스컴포넌트 사이에서 전송레벨의 바인딩을 담당하기 위해 제공되는 컴포넌트이다. 따라서 외부시스템과 표준전송프로토콜에서 실행중인 서비스들 간의 호출을 위해 주로 사용되며 WS-I(Web Services Interoperability Organization) 프로파일에서 사용하는 서비스들 간의 통신에도 전송레벨의 서비스를 제공한다. NMR은 이러한 바인딩컴포넌트와 서비스엔진컴포넌트사이에서 XML 메시지의 형태로 메시지 전송 포맷을 표준화하며 경로를 설정한다.

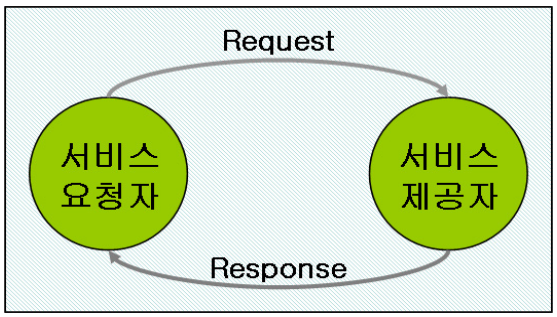
2.2 JBI 컴포넌트 패키징을 지원하는 Cimero

Cimero에디터는 현재 JBI를 지원하는 컴포넌트배치도구로 사용되고 있는 도구로 모델링, 패키징, 부분적인 배포기능을 제공하고 있다[11]. 7개의 바인딩컴포넌트와 7개의 서비스엔진컴포넌트를 제공하고 있으며 에디터의 아웃라인을 통하여 사용하고 있는 컴포넌트에 대한 식별을 쉽게 할 수 있는 기능을 제공하고 있다. 또한 GUI기반의 인터페이스로 사용자의 사용성 및 편리성을 높여 쉽게 컴포넌트 연동을 위한 모델링을 할 수 있으며 컴포넌트 유효성검증과 경량화 방식과 스탠다드방식의 패키징방법을 제공한다. Cimero에디터를 통해 분석하고 도출할 수 있는 개선사항 및 문제점은 세 가지로 볼 수 있다. 첫 번째, 현재 JBI를 지원하는 컴포넌트는 30여개이며 Cimero에디터에서 제공하고 있는 컴포넌트는 14개이다. 이는 사용자가 원하는 서비스를 제공하기에 컴포넌트사용의 제약이 있으며 원활하고 다양한 서비스를 제공하기 위해선 더 많은 컴포넌트를 추가로 지원해야 한다. 두 번째, Cimero에디터에서 패키징기능은 제공하지만 컴포넌트를 추가하거나 사용자 라이브러리 등의 리소스를 추가할 수 있는 기능이 없어 편집을 위한 과정을 요구할 시 별도의 도구를 사용하여 이원적으로 작업을 수행해야 하는 문제점을 가지고 있다. 세 번째, Cimero에디터에서는 현재 작업한 내용에 한에서만 배포/제거할 수 있으며 이미 운영되고 있는 서비스를 식별할 수 있는 기능은 제공하지 못하고 있다. 또한 배포/제거의 기능에 대한 제약사항이 많아 사용자가 통합적으로 사용하기에는 불편한 점이 많아 직접 ESB에 서비스유닛을 플러그인해야 하는 문제점을 가지고 있다.

3. 컴포넌트배치도구 모델

3.1 동작원리

JBI 컴포넌트 프레임워크에서 서비스 요청자는 서비스 제공자에게 서비스를 요청하고 제공하는 이에 대한 응답을 한다[8]. (그림 2)는 이러한 요청·응답사이에 대한 패턴이다. 서비스 요청자는 하나 또는 복수개로 이루어질 수 있으며 요청·응답에서 서비스요청은 NMR을 통해 표준화된 메시지로 변환되어 서비스 제공자에게 보내진다. 서비스 제공자는 요청받은 메시지에서 처리되어야 하는 비즈니스 로직에 따라 처리하며 요청에 대한 결과 값을 NMR을 통해 표준화된 메시지로 변환한 후 서비스 요청자에게 응답한다.



(그림 2) JBI 컴포넌트 프레임워크의 동작원리

이러한 서비스 요청자와 서비스 제공자 사이의 요청·응답에 대한 메시지변환패턴(MEP: Message Exchange Pattern)에는 One-Way, Reliable One-Way, Request-Response, Request Optional-Response의 4가지 방법이 있다. <표 1>과 같이 One-Way는 In-Only방식으로 서비스 요청자가 서비스요청을 위한 호출을 할 경우 호출이 실패해도 서비스 요청자는 이를 알 수가 없으며 메시지 입력만을 받고 호출이 성공했는지 실패했는지에 대한 응답은 하지 않는다. Reliable One-Way 역시 메시지 입력만을 받고 그에 대한 응답은 하지 않으나 호출이 실패할 경우에는 서비스 요청자에게 실패에 대한 메시지를 서비스 요청자에게 알린다. Request-Response는 메시지를 호출하고 응답을 받는 In-Out방식으로 서비스 요청자는 호출에 대한 응답결과를 받을 수 있다. Request Optional-Response 역시 In-Out방식으로 호출에 대한 응답결과를 받을 수 있다. 또한 서비스 요청자는 이 응답받은 메시지를 통해 연결되어 있는 다른 서비스를 요청할 수 있다.

〈표 1〉 메시지변환패턴

패턴	응답방식	설명
One-Way	In-Only	메시지 입력만 받음
Reliable One-Way	Robust In-Only	메시지를 입력받고 실패할 경우에만 응답받음
Request-Response	In-Out	메시지를 입력받고 결과 값을 응답받음
Request Optional-Response	In Optional-Out	응답받은 결과 값을 다른 서비스요청으로 활용

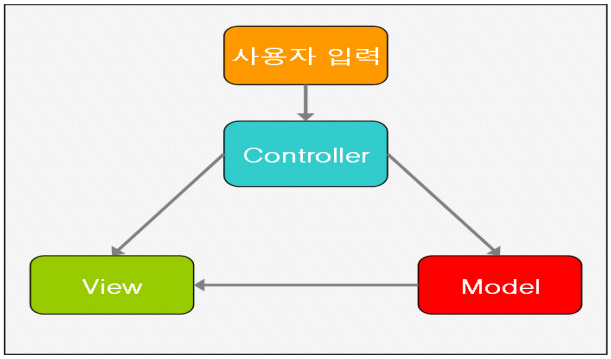
3.2 컴포넌트배치도구의 요구사항 도출

제안하는 JBI를 지원하는 컴포넌트배치도구의 요구사항 도출에는 크게 2가지의 전제조건이 있다. 첫 번째는 컴포넌트 모델링에서 배포의 전 과정이 본 논문에서 제안하는 배치도구 내에서 수행되어야 하며 두 번째는, 관련연구를 통해 알아본 Cimero에디터 배치도구의 문제점 및 개선사항을 보완하기 위한 요구사항 도출이다. 따라서 본 논문에서는 기존 Cimero에디터의 기본기능과 보완사항을 종합하여 컴포넌트연동을 위한 모델링, 모델링에 대한 유효성검증, 모델링 된 컴포넌트의 패키징, 패키징된 컴포넌트서비스의 배포 및 관리로 구분하여 분석하고 도출한다.

- 컴포넌트 모델링기능 : 컴포넌트모델링 기능은 JBI 컴포넌트 프레임워크에서 사용되고 있는 바인딩컴포넌트와 서비스엔진컴포넌트를 이용하여 서비스호출에서 응답 및 호출에 대한 처리완료까지의 과정을 설계하는 기능이다. 모델링기능은 한눈에 알아보기 쉽고 한 번에 이해할 수 있도록 가독성이 높아야 한다. 따라서 모델링기능은 GUI기반으로 바인딩컴포넌트와 서비스엔진컴포넌트를 이미지로 표현하며 연결선을 통하여 컴포넌트 간의 처리절차를 나타내야 할 필요가 있다. 기존 Cimero 에디터 배치도구는 바인딩컴포넌트 7개, 서비스엔진 컴포넌트 7개로 한정되어 있어 더 많은 컴포넌트를 표현하고 연동할 수 있어야 한다. 또한 사용자가 쉽게 컴포넌트의 이름을 명시하여 가독성을 높일 필요가 있다.
- 컴포넌트 유효성검증기능 : 컴포넌트의 유효성검증은 모델링작업에서 각 컴포넌트가 가지고 있는 속성에 대한 오류를 최소화하고 컴포넌트와 컴포넌트사이의 연결이 적절하게 이루어져 있는지 검증하는 기능이다. 컴포넌트가 가지고 있는 속성에는 네임스페이스와 엔드포인트 등의 공통된 속성이 있다. 컴포넌트사이의 중복된 엔드포인트 이름에 대한 유효성 검증과 오류가 발생한 컴포넌트의 식별의 기능이 필요하다. 또한 컴포넌트의 속성에는 요청·응답 패턴이 정의되어 있다. 예를 들어 In-Only만을 지원하는 컴포넌트에서 In-Out의 패턴을 요청한다면 재귀되는 응답결과에 대한 오류가 발생하게 된다. 또한 컴포넌트의 개별적인 속성에 대한 유효성 검증이 요구되는데 이는 컴포넌트 고유의 응답패턴속성에 대한 유효성 검증과 컴포넌트 간 연결에 대한 유효성 검증이 필요하다.
- 컴포넌트 패키징기능 : JBI컴포넌트패키징에는 경량화방식과 하나의 JBI환경에 플러그인하여 사용하는 스탠다드방식의 2가지 방법이 있다. 경량화방식은 하나의 JBI환경에서 운영되는 것이 아닌 복수개의 JBI환경을 운영하는 것이다. 이러한 경량화방식은 빈번하게 사용되는 서비스컴포넌트가 아닌 사용빈도가 낮은 컴포넌트에 유용하게 사용되며 여러 JBI환경으로 운영되어 필요한 기능을 집중적으로 관리할 수 있다. 또한 경량화방식은 서비스를 운영하는데 있어 사용하지 않는 불필요한 라이브러리나 컴포넌트를 제거하고 필요한 컴포넌트만을 플러그인하여 운영하기에 가벼우며 별다른 패키징 없이 XML형식으로 제공되어 JBI환경을 효율적이며 경량적으로 운영할 수 있다. 하나의 JBI환경에 플러그인 되어 사용하는 스탠다드방식은 사용빈도가 높은 서비스에 사용된다. 이러한 스탠다드방식의 패키징구조는 JBI 기반의 메타파일과 컴포넌트를 압축기술로 통합해놓은 압축파일로 되어 있으며 메타파일과 각 컴포넌트의 압축파일을 통합한 압축파일(서비스유닛)로 구성된다. 따라서 본 논문에서는 경량화방식과 스탠다드방식의 컴포넌트패키징 방법이 필요하다. 경량화방식은 기존의 Cimero 에디터 배치도구 방식과 일치하지만 스탠다드방식에는 개선해야 할 사항이 도출되었다. 첫 번째로 Cimero에디터 배치도구에서 스탠다드방식의 패키징은 일방적인 패키징방식으로 단위 컴포넌트별 압축과 메타파일과의 압축이 한 번에 이루어져 사용자 라이브러리 파일을 추가로 넣을 수 없어 별도의 도구로 수작업을 하고 있는 실정이다. Cimero에디터 배치도구는 설계단계에서부터 패키징에 대한 편집기능이 제외되어 하부디렉토리에 대한 depth가 2이상 될 경우 패키징 오류를 일으키게 된다. 예를 들어 Bean컴포넌트에서 사용하기 위한 클래스파일의 패키지가 org.apache.servicemix.junitCase.test일 경우 클래스파일의 위치는 depth 4의 하부디렉토리로 구성되어 패키징 오류가 일어난다. 이러한 문제점을 해결하기 위해서는 패키징과정을 2단계로 분리하여 동적인 편집을 가능하게 해야 하며 패키징을 위한 압축기능을 개선해야 할 필요가 있다.
- 서비스 배포 및 관리기능 : 경량화방법은 ESB 미들웨어플랫폼을 하나 더 운영하는 것으로 패키징된 서비스유닛의 배포 및 관리를 지원하지 않는다. 하지만 JBI 환경내에 플러그인하여 사용하는 패키징된 서비스유닛은 현재 운영되고 있는 ESB 미들웨어플랫폼에 추가적으로 배포하고 관리한다. 따라서 제안하는 에디터 내에서 ESB 미들웨어플랫폼에 서비스유닛을 쉽고 빠르게 배포하고 수정, 제거의 관리를 할 수 있는 기능이 요구된다. 기존 Cimero에디터 배치도구에서도 ESB미들웨어플랫폼에 배포하는 기능이 있으나 Eclipse도구에 별도로 ESB미들웨어플랫폼을 플러그인하여 사용하는 의존적인 형태이다. 또한 현재 Cimero에디터 배치도구에서 패키징한 서비스유닛에 한해서만 배포할 수 있으며 배포의 기능만을 가지고 있어 서비스유닛을 제거하지는 못하고 있다. 이에 ESB미들웨어플랫폼의 배포 및 제거 기능이 강화되고 ESB 미들웨어플랫폼의 배포와 제거에 대한 전반적인 관리가 필요하다.

3.3 MVC 패턴을 이용한 컴포넌트배치도구의 구조

본 논문에서 제안하는 JBI컴포넌트패키징 에디터의 구조는 MVC(Model, View, Controller)패턴을 사용하여 정확하게 설명할 수 있다. smalltalk에서 제안된 MVC의 구조는



(그림 3) MVC패턴

객체지향 구조에서 많은 연구 및 응용되고 있는 구조로서 입력, 처리, 출력을 모델(Model), 뷰(View), 컨트롤러(Controller)의 세 가지 구성요소로 분리하여 단순화시켰으며, 구성요소 간의 결합도를 줄임으로써 유지보수와 재사용성을 높혀준다[12]. (그림 3)은 이러한 기본적인 모델, 뷰, 컨트롤러간의 관계를 보여준다. MVC구조에 대한 각 컴포넌트의 역할은 다음과 같다.

모델은 컴포넌트의 중요 기능과 데이터를 가지고 있다. 모델은 각 컴포넌트의 요청처리 및 속성변경 등의 비즈니스 로직을 수행하는 부분으로 외부 변화가 발생하면 이벤트에 대한 결과를 처리하며, 컴포넌트에 대한 속성, 연결에 대한 논리적인 표현을 나타내는 부분으로 UI Events에서 데이터가 변경되었을 경우 이를 처리하여 뷰에게 통보를 해주는 역할을 담당한다.

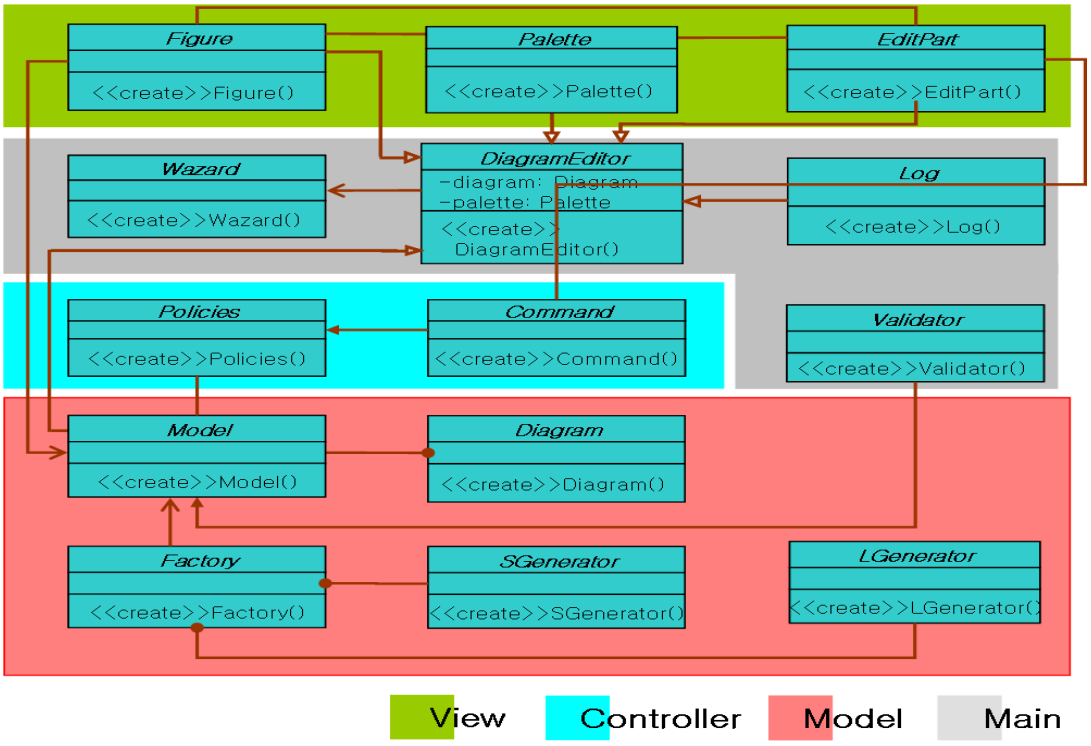
뷰는 모델에서 처리되고 변경된 컴포넌트의 정보를 시각적인 표현으로 제공하는 부분으로 모델에서 처리한 비즈니스로직을 표현한다. 따라서 뷰는 모델이 어떤 변화가 일어나고 있는지를 알아야한다.

컨트롤러는 UI 이벤트를 통하여 사용자 입력을 어떻게 다뤄야 하는지 정책, 정의 등의 명세를 다루는 부분으로 모델과 뷰의 중재자역할을 한다.

본 도구에서는 기존 MVC 모델을 모델에 기반하여 독립적인 계층을 프로세싱하면서 SOA의 기본 개념인 확장성의 개념에 맞춰 에디터 배치도구의 재사용 및 확장성 및 응집도를 높히고 결합도를 낮출 수 있는 구조로 배치도구를 설계 하였다. 다음 (그림 3)은 기본적인 모델, 뷰, 컨트롤러간의 관계를 보여준다.

(그림 4)는 본 논문에서 제안하고 있는 컴포넌트배치도구의 구조를 개략적 나타낸 클래스다이어그램으로 MVC패턴에 따라 구성하였다. 프리젠테이션계층인 뷰계층은 Figure, Palette, EditPart클래스로 구성되어 있으며 Controller계층은 Policies, Command클래스로 구성되어 있다. 특히 뷰계층을 구성하는 Pallete 구현에 있어서는 사용가능한 컴포넌트 정보 및 메뉴를 구성함에 있어 XML로 표현된 정보를 사용하여 동적으로 구성할 수 있도록 함으로써, 컴포넌트 도구에 새로운 컴포넌트를 추가할 경우 재컴파일이 필요하지 않도록 하여, 확장성을 보장할 수 있도록 하였다.

모델 계층은 컴포넌트의 속성에 대한 변경 및 처리를 위한 Model, Diagram클래스와 패키징을 위한 Factory, SGenarator, LGenerator클래스로 구성되어 있다. 본 배치도



(그림 4) 컴포넌트배치도구의 구조

구의 주요클래스는 DiagramEditor클래스로 각 Model, View, Controller의 클래스에서 상속하고 있으며 유효성검증을 위한 Validater, Log클래스와 배포 및 배치도구 파일관리를 위한 Wizard클래스를 포함하여 설계하였다. 여기서 뷰계층의 Figure클래스는 컴포넌트와 컴포넌트사이의 연결선을 표현하는 클래스로 하위클래스에 바인딩컴포넌트와 서비스엔진 컴포넌트의 각 이미지들과 연결선들을 포함하고 있다.

3.4 JBI를 지원하는 컴포넌트 정의

본 논문에서 제안한 에디터에서 제공하는 컴포넌트는 <표 2>와 같으며 바인딩컴포넌트 12개와 서비스엔진컴포넌트 9개로 총 21개의 컴포넌트로 구성되어 있다[11]. 바인딩컴포넌트와 서비스엔진컴포넌트로 분류하며 각 바인딩컴포넌트와 서비스엔진컴포넌트마다 패키징방법이 다르기 때문에 경량화방식과 스탠다드방식으로 다시 구분한다. 예를 들어 바인딩컴포넌트인 File, FTP , Mail, RSS, ScreenOutout, Timer컴포넌트는 경량화방식의 패키징을 사용하는 방법으로 ESB에 플러그인하지 않고 별도의 ESB를 운영하여 사용할 수 있다. 경량화방식을 갖는 바인딩컴포넌트의 특징은 주기적으로 발생하거나, 추적하거나, 특정한 경우에 많이 사용되는 컴포넌트라는 것이다. ESB내에 플러그인하여 주기적으로 파일을 추적하고 FTP서버를 폴링(Polling)할 경우 생기는 과부하를 방지하기 위해 경량화 방식으로 설계한다. 서비스엔진컴포넌트는 BPE Bean컴포넌트를 제외하곤 EIP(Enterprise Integration Pattern)패턴에 대한 컴포넌트로 바인딩컴포넌트의 경로설정을 담당하는 컴포넌트이다. 예를 들어 WireTap클래스는 바인딩컴포넌트와 바인딩컴포넌트를 연동해주면서 별도의 메시지추적기능을 포함하고 있다. 메시지추적기능으로 서비스요청 바인딩컴포넌트에서 요청한 메시지와 서비스제공 바인딩컴포넌트에서 처리하고 응답한 메시지를 비교하여 메시지의 변환이나 원본메시지의 무결성 및 검증을 체크할 수 있다.

〈표 2〉 바인딩컴포넌트와 서비스엔진컴포넌트 정의

컴포넌트종류 \ 패키징방법		경량화	스탠다드
바인딩 컴포넌트	File	O	
	FTP	O	
	Mail	O	
	RSS	O	
	ScreenOutput	O	
	Timer	O	
	Bean		O
	JMS		O
	Xslt		O
	Web		O
	Socket		O
	Xmpp		O
서비스엔진 컴포넌트	BPE Bean	O	
	Router	O	
	Transformer	O	
	Aggregator		O
	MessageFilter		O
	Pipeline		O
	RoutingSlip		O
	Splitter		O
	Wiretap		O

4. 컴포넌트 배치도구의 설계 및 구현

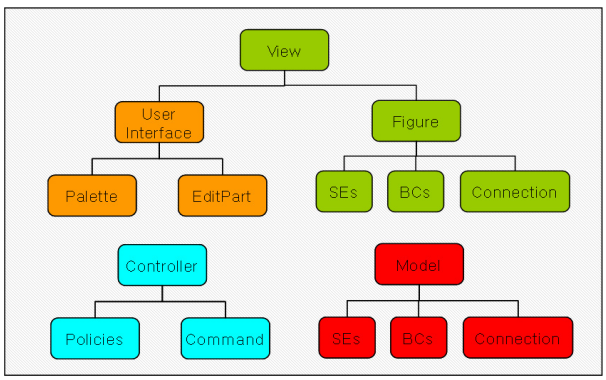
4.1 컴포넌트 배치도구 설계

컴포넌트 배치도구의 설계는 본 논문의 기능요구사항인 모델링기능, 패키징기능, 배포 및 컴포넌트관리기능으로 4가지의 기능요구사항을 기반으로 제시한다.

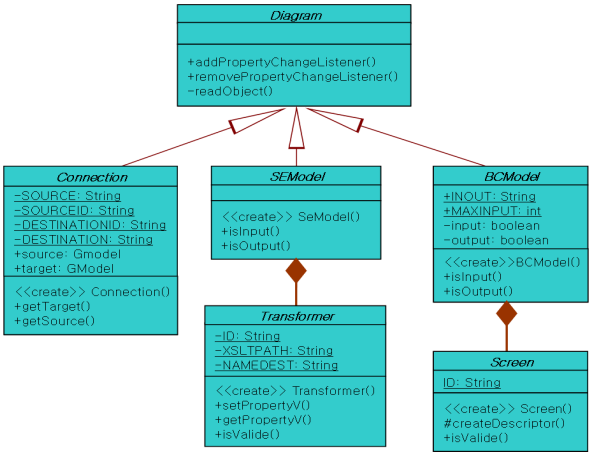
4.1.1 모델링기능 설계

(그림 5)는 컴포넌트의 모델링을 위한 편집도구에 대한 설계로서 MVC 패턴에 따른 구조를 나타낸 것이다[1]. 뷰계층은 User Interface 클래스 패키징과 Figure 클래스 패키징으로 구성되어 있으며 User Interface의 Palette는 편집도구의 Toolbox이며 EditPart는 다이어그램을 나타낼 수 있는 캔버스와 캔버스상의 편집 작업을 위한 부분이다. Model에는 서비스엔진컴포넌트, 바인딩컴포넌트, 경로설정을 위한 Connections로 구성되어 있으며 컴포넌트의 속성과 요청된 변경사항을 처리하는 비즈니스로직부분이다. View의 Figure 클래스 패키징과 Model클래스 패키징은 같은 구조로 되어 있으며 Model에서 변경된 사항을 업데이트하여 Figure클래스 패키징을 통해 화면에 보여준다. Controller의 Policies는 컴포넌트와 Connections의 선택, 추가, 삭제, 제거 등의 규칙 및 정의를 설정하는 클래스 패키징이며 Command클래스 패키징은 Policies에 따른 선택, 추가, 삭제, 제거의 수행 작업을 정의해 놓았다.

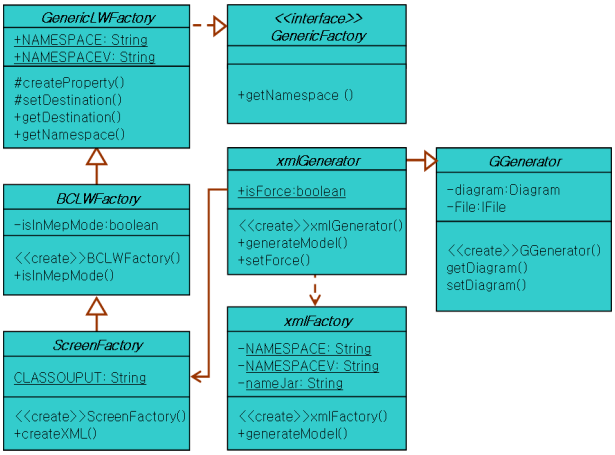
Diagram은 모든 컴포넌트(그림 6)과 (그림 5)의 Model부분을 세분화한 클래스다이어그램으로 바인딩컴포넌트인 Screen컴포넌트를 예로 나타낸 것이다. 와 연결선인 Connection클래스를 관리하는 클래스로 선택된 컴포넌트의 속성을 생성하거나 제거하는 역할을 가진다. Connection클래스는 대상 컴포넌트와 연결하려는 Source 컴포넌트와 Target 컴포넌트로 구성되어 있으며 컴포넌트의 In, Out 모



(그림 5) 모델링기능의 클래스패키징구조



(그림 6) 모델링 클래스다이어그램

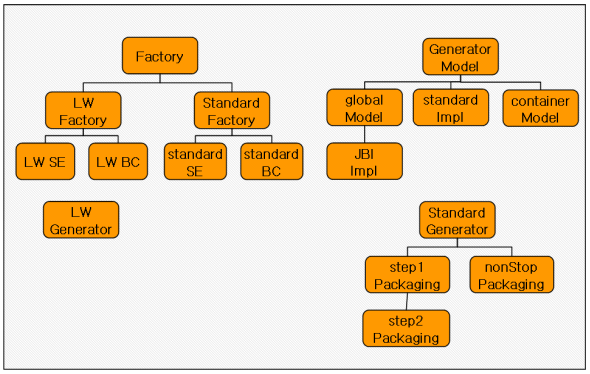


(그림 8) 패키징 클래스다이어그램

드에 따라 연결선을 설정한다. SEModel과 BCModel 클래스는 Model의 세분화된 클래스로 서비스엔진컴포넌트와 바인딩컴포넌트로 분류되어 관리된다. 본 논문에서 JBI를 지원하는 컴포넌트의 속성은 지원되는 컴포넌트의 스키마파일을 참조로 자주 사용하는 속성을 추출하여 설계하였다.

4.1.2 패키징기능 설계

컴포넌트모델링을 통해 설정 된 속성을 JBI기반의 XML 형식으로 변환하기 위한 방법으로 JET라이브러리를 사용하여 설계하였다. JET라이브러리는 Java와 JSP 등의 Java기반 언어를 컴파일 할 수 있는 라이브러리로 각 패키징방식과 컴포넌트의 종류에 따른 템플릿을 로드하여 Serialize로 직렬화된 컴포넌트의 속성을 JSP문법으로 삽입하여 XML형식의 파일로 생성한다[4]. (그림 7)은 컴포넌트의 패키징을 위한 클래스 패키징구조로 factory, GeneratorModel, LWGenerator, StandardGenerator의 클래스 패키징으로 구성하였다. Factory클래스패키징은 경량화방식과 스탠다드 방식의 패키징 방법으로 구분되며 각 방식에 따라 서비스엔진컴포넌트와 바인딩컴포넌트로 분류하여 관리한다. 경량화 방식의 바인딩컴포넌트 및 서비스엔진컴포넌트는 경량화 템플릿과 헤더를 포함하고 있는 LWGenaretor클래스를 통해 경량화방식으로 패키징된다.



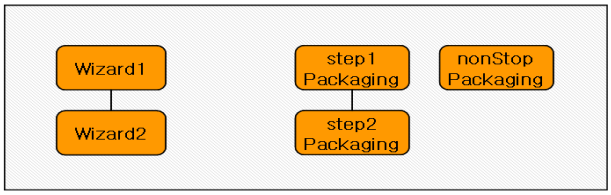
(그림 7) 패키징기능의 클래스패키징구조

GeneratorModel과 StandardGenerator클래스패키징은 스탠다드방식의 패키징을 위한 클래스패키징이며 스탠다드방식의 패키징은 1차 패키징과 2차 패키징의 2단계로 이루어진다. 이는 패키징과정에서 발생할 수 있는 편집을 위한것으로 사용자는 1차 패키징으로 생성된 각 컴포넌트의 폴더에 사용자파일과 같은 라이브러리를 드래그&드롭으로 간단하게 편집할 수 있다. 기존 Cimero에디터 배치도구는 1차 패키징과 2차 패키징이 통합된 형태로 사용자가 패키징과정 이후 편집을 원할 경우 Cimero에디터 배치도구를 벗어나 별도의 압축프로그램으로 사용자 라이브러리파일을 추가해야하는 불편함을 개선하기 위해 1차, 2차 패키징으로 분류하여 개선할 수 있도록 하였다. 1차 패키징은 각 컴포넌트의 속성과 컴포넌트별 메시지변환패턴(MEP)에 따른 Connection의 정보를 JBI기반의 형식으로 변환하여 각 메타디렉토리와 컴포넌트별 디렉토리에 생성하고 XML형식의 파일로 각 컴포넌트파일과 메타파일을 생성한다. 2차 패키징은 이러한 XML형식의 컴포넌트파일을 ZIP파일 형식으로 압축하여 통합하고 서비스에 대한 메타파일을 기준으로 다시 한번 각 컴포넌트와 메타파일을 통합하는 압축과정을 수행한다[3, 6].

(그림 8)은 바인딩컴포넌트인 Screen컴포넌트의 JBI패키징을 예로 나타낸 클래스다이어그램이다. GenericLWFactory은 컴포넌트단의 클래스인 ScreenFactory에서 컴포넌트의 속성값과 BCLWFactory에서 JBI기반의 XML헤더를 상속받는다. xmlGenerator와 xmlFactory에서는 GenericLWFactory에서 수합된 xml형식의 데이터를 기반으로 경량화방식의 템플릿을 로드하여 XML파일을 생성하고 패키징한다. 또한 이러한 패키징과정에서 동일한 이름의 XML파일이 존재하는지와 XML형식의 변환된 속성 값이 경량화방식의 템플릿에 적합한지의 유효성검증을 수행한다.

4.1.3 배포와 컴포넌트관리기능의 설계

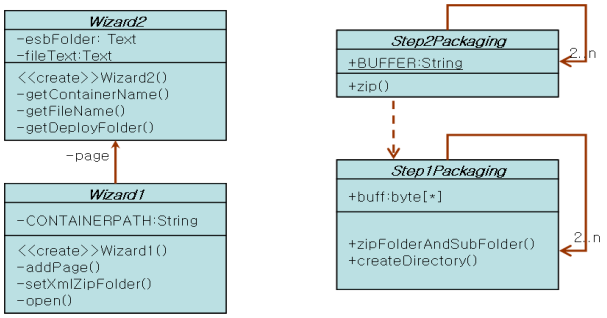
(그림 9)는 패키징된 컴포넌트인 서비스의 배포와 컴포넌트관리를 위한 클래스구조를 나타낸 것이다. 서비스를 배포하기 위한 설정은 본 배치도구를 설치하고 설정할 때나 에



(그림 9) 서비스배포와 컴포넌트관리의 클래스구조

디터파일을 새로 만들기 할 때 마법사화면으로 설정할 수 있다. 마법사화면을 통하여 ESB미들웨어플랫폼의 경로를 입력하면 ESB미들웨어플랫폼의 배포 디렉토리와 동기화되어 에디터에서 바로 배포할 수 있는 기능을 제공한다. 배포 디렉토리가 동기화되어 현재 운영되고 있는 서비스유닛의 목록이 배치도구의 배포폴더와 연동되어 본 배치도구 내에서 서비스를 배포·제거 할 수 있다. 1차 과정에서 사용자는 필요에 따라 XSLT파일, RULE파일, 자바클래스파일, 사용자 리소스파일 등의 사용자 라이브러리파일을 각 컴포넌트별 폴더에 드래그&드롭만으로 쉽게 추가하거나 제거할 수 있다. 또한 본 논문에서 제안하는 배치도구는 Eclipse의 플러그인방식으로 동작되어 사용자가 원하는 작업을 배치도구내에서 통합적으로 수행 할 수 있다. 예를 들어 Bean컴포넌트나 BPE Bean컴포넌트에서 사용되는 Java클래스파일은 본 배치도구내에 Java프로젝트를 통해 구현하고 컴파일한 후에 클래스경로까지 모두 추가할 수 있다.

(그림 10)은 서비스배포와 컴포넌트관리를 위한 클래스다이어그램으로 step1 Packaging클래스에서 JBI 메타파일과 컴포넌트의 수만큼 폴더와 서브폴더를 생성하고 XML파일을 생성한다. step2Packaging클래스는 step1Packaging클래스를 통해 생성된 컴포넌트 파일과 폴더 수만큼 반복하면서 ZIP파일형식으로 압축하고 최종적으로 압축된 컴포넌트들과 JBI메타폴더를 다시 압축하여 통합한다. 기존 Cimero 에디터 배치도구는 위 통합과정을 하나의 과정으로 통합하여 컴포넌트 재사용을 위한 기능을 제공하지 않는다. 이에 본 논문에서는 1차 압축을 위한 폴더생성 및 xml파일 생성과 압축을 분리하여(step2Packaging클래스를 통해) 서비스유닛화 되어도 1차 패키징에서 생성된 컴포넌트 파일과 폴더는 삭제되지 않는다. 이에 사용자는 원하는 만큼 컴포넌트를 편집하고 재사용할 수 있으며 동적으로 컴포넌트를 관리할 수 있다.



(그림 10) 서비스배포와 컴포넌트관리 클래스다이어그램

4.2 모델과 뷰사이의 통신에 적용을 위한 설계패턴

4.2.1 구현환경

제안하는 에디터는 x86기반의 윈도우XP에서 구현하였으며 <표 3>은 에디터의 구현환경을 나타낸다. Eclipse기반의 플러그인으로 개발하였으며 Eclipse에서 그래픽을 위한 필수 플러그인인 GEF와 EMF플러그인을 설치하였다. 개발언어로 Java의 JDK 1.5.0과 SDK 5.0.1을 사용하였으며 운영되는 ESB미들웨어플랫폼은 Servicemix 3.2를 대상으로 구현하였다.

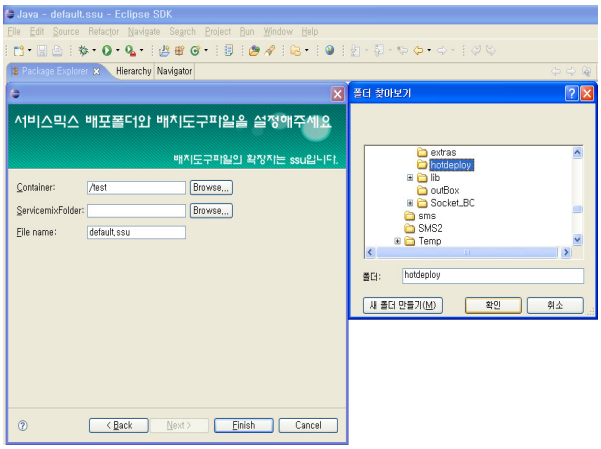
〈표 3〉 컴포넌트배치도구의 구현환경

항목	구현환경
에디터플랫폼	Eclipse 3.2.2
필수 플러그인	GEF, EMF Plugins
개발언어	JDK 1.5.0
	SDK 5.0.1
ESB미들웨어플랫폼	Servicemix 3.2
XML 버전	1.0(encoding = UTF-8)

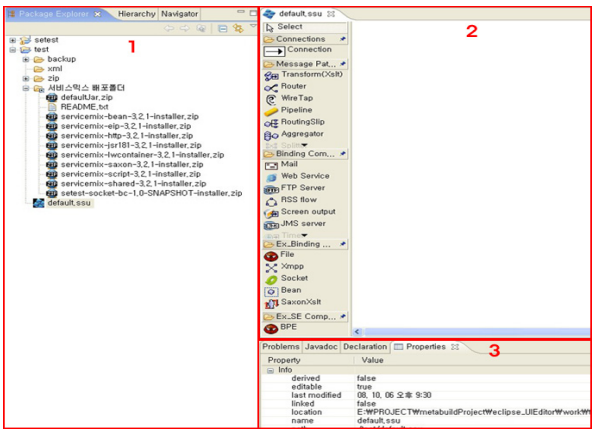
4.2.2 컴포넌트 배치도구 구현

구현된 컴포넌트배치도구를 Eclipse 3.2.2 플러그인폴더에 인스톨한 후 실행하면 모델링, 패키징, 배포, 관리서비스를 이용하여 JBI를 지원하는 컴포넌트배치도구를 Servicemix 3.2에서 사용할 수 있는지 검증한다. (그림 11)은 배치도구의 마법사화면으로 Servicemix의 배포폴더와 배치도구파일을 설정하는 화면이다.

(그림 12)는 배치도구의 메인화면으로 3개영역으로 구분되어 있다. 1번은 배포폴더와 1,2차 패키징, 라이브러리파일 등의 패키지익스플로러로 xml폴더에 경량화방식의 패키징이 저장되며 zip폴더에는 스탠다드방식의 1,2차 패키징이 저장된다. 서비스믹스배포폴더는 실제 서비스믹스의 배포폴더인 핫디플로이폴더와 연동되어 현재 운영 중인 서비스믹스에 배포되어 있는 컴포넌트를 보여주고 있다. 이 기능을 통



(그림 11) 컴포넌트배치도구 마법사화면



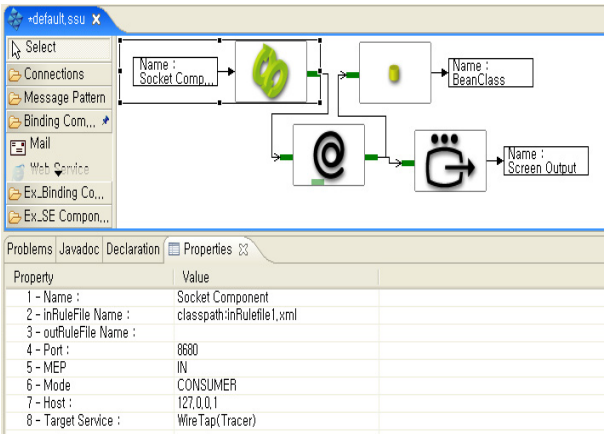
(그림 12) 배치도구의 메인화면

하여 본 배치도구내에서 운영되고 있는 모든 서비스를 관리할 수 있다. 2번은 모델링을 할 수 있는 에디트파트로 각 컴포넌트들을 모아놓은 툴박스와 흰바탕의 캔버스로 구성되어 있다. 사용자가 툴박스를 통하여 선택한 컴포넌트를 흰바탕의 캔버스에 클릭하면 컴포넌트가 생성된다. 3번은 컴포넌트의 속성을 나타내는 창으로 사용자입력으로 속성을 변경할 수 있으며 현재 (그림 12)에서 속성창은 배치도구 파일에 대한 속성정보를 나타내고 있다.

5. 컴포넌트 배치도구 검증

5.1 모델링 기능 검증

에디트파트의 툴박스에서 컴포넌트를 선택하고 connection으로 연결한다면 화면에서는 이를 그대로 보여줄 수 있어야 하며 (그림 13)은 소켓컴포넌트와 일반 자바프로젝트인 Bean컴포넌트를 추적이 가능한 WireTap컴포넌트와 연동하고 WireTap컴포넌트에 ScreenOut컴포넌트를 연동하여 추적한 원본메시지와 변환된 메시지, 메시지속성을 콘솔에 출력하기위한 모델링을 나타낸 것이다. (그림 13)은 하위 속성창을 보면 소켓컴포넌트의 이름은 SocketComponent이며



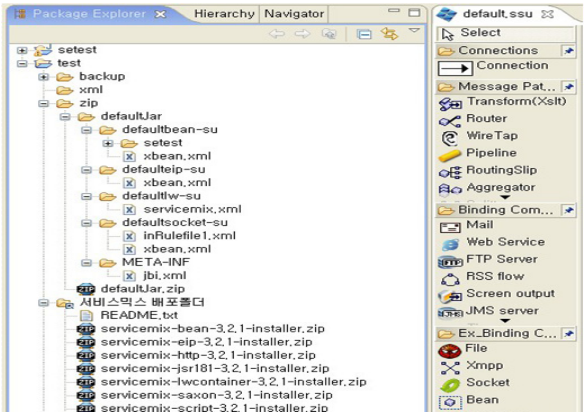
(그림 13) 모델링 검증

In-Only의 메시지변환패턴을 가지고 있다. 소켓통신을 위한 외부 포트는 8680으로 설정하였으며 소켓을 통해 들어온 메시지를 가공하기 위한 inRoleFile은 사용자파일인 rulefile1.xml파일을 사용하고 outRoleFile은 In-Only방식이므로 공백으로 설정하고 target컴포넌트로 WireTap컴포넌트를 지정하고 있다.

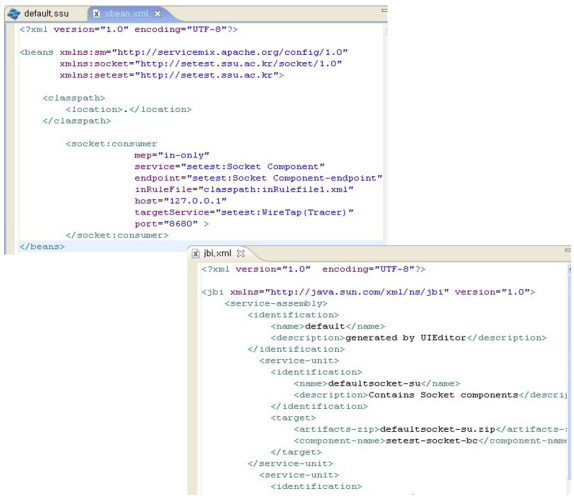
5.2 패키징 및 컴포넌트관리 검증

(그림 14)는 컴포넌트모델링 후 스탠다드방식으로 패키징하기 위한 과정이다. 1차 패키징기능으로 배치도구파일명을 기준으로 각 컴포넌트의 XML파일과 폴더, JBI메타파일과 폴더가 생성되었다. 또한 Bean컴포넌트의 Java클래스파일을 추가하기 위해 setest라는 Java프로젝트를 생성하여 배포도구 내에서 Java파일을 구현하였으며 이러한 방법으로 Socket 컴포넌트의 사용자 리소스파일을 구현하여 드래그를 통하여 추가하였다. 동적으로 편집된 컴포넌트를 2차 패키징 하면 1차 패키징된 폴더의 아래에 서비스믹스 배포폴더에 플러그인하여 사용할 수 있는 통합된 서비스인 defaultJar.zip파일이 생성된다.

(그림 15)는 1차 패키징을 통해 생성된 JBI메타파일과



(그림 14) 패키징 검증

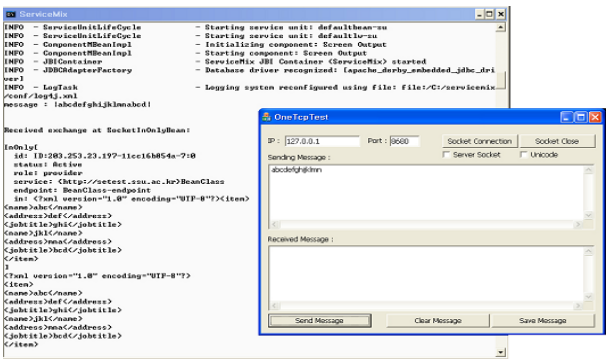


(그림 15) 패키징을 통한 컴포넌트XML파일 생성

Socket컴포넌트에 대한 XML파일이다. 모델링을 통해 설정된 Socket컴포넌트의 속성이 그대로 XML형식으로 변환되어 있으며 헤더파일과 통합되어 있으며 JBI메타파일에서 사용되고 있는 컴포넌트의 ID와 이름이 기록되어 있음을 알 수 있다.

5.3 배포 및 추적

(그림 16)은 2차 패키징 과정에서 통합된 서비스를 배포하고 제거하는 기능을 나타낸 것이다. zip폴더 아래의 defaultJar.zip파일을 서비스믹스배포폴더에 드래그를 통해 이동하면 배포가 되고 제자리로 이동하면 서비스가 제거된다. 본 논문에서 제안하는 컴포넌트 배치도구를 통해 모델링되고 패키징되어 배포된 서비스가 사용자의 요구대로 모델링 되었으며 정상적으로 JBI기반의 ESB미들웨어플랫폼에서 동작하는지에 대한 검증을 나타낸 것이다. 소켓프로그램



(그림 16) 서비스 검증

을 이용하여 모델링과정에서 설정된 Socket컴포넌트의 포트번호인 8680으로 접속하여 임의의 메시지를 보낸 결과 Bean컴포넌트에서 응답하여 메시지를 처리하였다. (그림 16)은 이 과정을 검증하기 위해 WireTap컴포넌트를 사용하여 메시지를 추적한 화면이다. 콘솔화면을 통해 입력된 메시지와 메시지의 MEP타입 및 메시지 속성, 메시지가 최종 변환된 컴포넌트의 이름, 변환된 메시지 내용이 콘솔에 출력되고 있음을 알 수 있다.

6. 실험 및 비교평가

본 논문에서 제안하는 배치도구는 기존의 서비스믹스를 지원하는 Cimero에디터 2.7.1의 개선사항 및 문제점을 분석하고 JBI기반의 ESB가 가지고 있는 문제점을 분석하여 JBI를 지원하는 컴포넌트배치도구를 구현하였다. 이번 절에서는 본 연구에서 제안한 컴포넌트배치도구의 성능을 검증하기 위해 수작업을 통한 배치파일, Cimero를 통한 배치파일 완성, 본 논문에서 제시한 컴포넌트배치도를 시험하고 비교평가 한다. 시험 및 비교평가는 두 가지로 구분하여 제시한다. 첫 번째는 패키징하기 위해 사용되는 Loc(Line of code)를 비교하여 논문에서 제안한 도구의 개선사항을 정량적으로 평가한다. 두 번째는 기능적인 평가를 통한 기능적 우위에 대한 비교로 본 도구의 우수성을 제시하였다.

6.1 비교평가

<표 4>는 서비스믹스의 배치파일과 Cimero에디터 2.7.1

〈표 4〉 서비스믹스를 지원하는 배치도구에 대한 비교평가

비교항목 \ 비교도구		서비스믹스 배치파일	Cimero에디터 2.7.1	본 논문에서 제안하는 배치도구
적용된 설계 패턴		▪ 적용된 설계패턴 없음	▪ MVC 기본구조로 설계 ▪ 이클립스 형태의 플러그인 형태로 제공	▪ 기존 MVC 구조를 기반으로 확장성을 높이는 구조로 설계 (XML 활용) ▪ 이클립스 형태의 플러그인 형태로 제공
인터페이스	주요 특성	▪ 화면영역 지원하지 않음	▪ 뷰어영역, 편집영역만 존재	▪ 크게 뷰어영역, 편집영역, 유효성 검증 영역, 배포/제거 영역 등으로 나뉘어 짐 ▪ 2단계 편집기능 제공(편집, 수정 및 편집)
	특징	▪ 수작업을 통한 패키징	▪ 자동화를 통한 패키징이 가능하지만 한 번에 한 개의 패키징만 가능하며, 최소 한의 수작업을 통해야 패키징 가능	▪ 완전 자동화가 가능하며, 한 번에 여러 개의 패키징이 한 번에 가능함
		▪ 지원하지 않음	▪ 편집 창에서 패키징만가능 함 ▪ 한 번에 한 개의 작업창만 가능함	▪ 편집 창에서 패키징과, 배포가 가능하고, 한 번 실행으로 동시에 여러 개의 작업창을 띄어서 다수의 패키징 가능
		▪ 지원하지 않음	▪ 2차수까지만 지원가능하며 2차수가 넘어갈 경우 패키징 오류 발생	▪ 사용자 편의를 위해 컴포넌트 관리항목의 수정 및 편집 기능 제공 ▪ 차수에 상관없이 폴더의 최하위 서브폴더의 차수까지 패키징 가능하며 사용자 리소스파일을 배치도구 내에서 추가할 수 있음
		▪ 수작업을 통한 배포/제거	▪ 배포/제거 기능을 지원하지 못함	▪ 배포/제거를 완전 자동화 할 수 있는 통합 지원 기능 제공
		▪ 추적 기능을 제공하지 않음	▪ 추적 기능을 제공하지 않음	▪ 컴포넌트의 유효성 검증기능 제공
지원되는 컴포넌트의 개수		▪ 지원하지 않음	▪ 14개 컴포넌트 지원	▪ 21개 컴포넌트 지원
주요기능		▪ 수작업을 통한 패키징 및 배포/제거 작업 수행	▪ UML 표기법을 완전히 지원하지 않음 ▪ 배포/제거, 검증 등 통합기능을 제공하지 못함	▪ UML 표기법 기반으로 구현됨 ▪ 클래스 다이어그램, 프레임워크, 다이어그램 등 UML에서 제공하는 주요다이어그램을 지원하며, 사용자편이의 기능과 UI 제공함

을 비교대상으로 평가한 표이다. 비교항목은 배치도구의 주된 기능인 컴포넌트 모델링, 패키징, 배포 및 컴포넌트관리로 구성되어 있으며 각 항목별로 세부항목이 포함되어 있다. 모델링항목은 GUI기반의 모델링지원 여부, 지원되는 컴포넌트의 수, 컴포넌트 및 모델링의 유효성검증 여부의 세부항목이 있으며 패키징항목에는 패키징의 자동화여부, 편집기능, 재사용 여부의 세부항목이 있다. 컴포넌트관리 항목에는 컴포넌트를 동적으로 관리할 수 있는 편집기능이 제공되는가에 대한 평가항목을 포함하고 있다.

서비스믹스 배치파일은 가장 기본적인 형태로 서비스믹스에서 기본적으로 사용하고 있는 Maven과 Ant를 사용하여 서비스하고 있으며 Cimero에디터는 본 논문에서 제안하는 배치도구처럼 Eclipse기반의 플러그인형태로 제공되는 방식이다. 서비스믹스 배치파일에서의 컴포넌트 모델링은 컴포넌트의 스키마를 일일이 분석하여 라인인터페이스기반으로 사용해야 하는 불편함이 있었으며 유효성검증은 지원되지 않고 있다. Cimero에디터는 GUI기반으로 사용자가 사용하기 편리하고 유효성검증으로 신뢰성을 높였으나 지원되는 컴포넌트의 수가 바인딩컴포넌트 7개, 서비스엔진컴포넌트 7개로 본 논문에서 제안하는 배치도구에서 지원되는 바인딩컴포넌트 12개, 서비스엔진컴포넌트 9개보다 7개가 적은 것으로 나타났다. 패키징항목에서 서비스믹스배치파일의 경우는 배치파일을 통합 수작업으로 수정기능과 재사용기능이 지원되지 않고 있으며 Cimero에디터의 경우 패키징 방법이 자동화가 되어 있으나 절차가 순차적으로 되어 있어 이를 수정하고 편집할 수 있는 기능이 지원되지 않고 있다. 재사용기능은 저장된 Cimero에디터파일을 로드하여 재사용 할 수 있다.

본 논문에서 제안하는 배치도구는 Cimero에디터처럼 자동화 된 패키징방법을 제공하고 있으며 패키징 절차를 2단계로 나누어 패키징과정에서 수정 및 편집할 수 있는 기능을 지원하고 있다. 또한 저장된 배치도구파일을 load하여 재사용 할 수 있는 기능을 지원하고 있다. 배포항목은 서비스믹스 배치파일의 경우 수작업으로 서비스믹스 배포폴더에서 배포하고 제거해야한다. Cimero에디터는 현재 에디터 내에서 모델링 된 서비스만을 배포하고 제거할 수 있으며 에디터 내에서 배포된 예전 서비스의 배포/제거는 할 수 없어 이에 대한 작업은 수작업을 통해 이루어진다. 하지만 본 논문에서 제안하는 배치도구는 현재 배치도구내에서 모델링된 서비스뿐 아니라 기존의 서비스, 현재 운영되고 있는 서비스에 대한 배포 및 제거기능을 지원하여 배치도구 내에서 통합적으로 관리되고 있다. 마지막으로 컴포넌트관리 항목의 편집기능은 서비스믹스와 Cimero에디터 모두 지원하지 못하고 있다. Cimero에디터의 경우 컴포넌트의 서브폴더가 2차수를 넘어가면 패키징 오류를 일으키며 패키징이 안되고 사용자 리소스파일을 추가해야 할 경우 별도의 수작업과 도구를 사용하여 추가해야 한다. 본 논문에서 제안하는 배치도구는 시스템이 지원하는 서브폴더의 차수까지 지원하며 사용자 리소스파일을 배치도구 내에서 추가할 수 있는 기능을 제공하고 있다.

6.2 실험

실험은 우선 기존 방식대비 효과를 알아보기 위해 데이타 구현 시스템을 사용하였다. 실험방법은 다음과 같은 네 가지 비교 실험을 통해 본 논문에서 제안한 도구의 성능을 시험・평가한다.

첫 번째 비교실험은, 기능적으로 복잡도가 다른 ESB 서비스를 선별하여 선별된 서비스를 패키징하는 방법을 사용하여 수작업을 통한 배치파일, Cimero, 제안하는 배치도구에서 사용되는 Loc를 비교·평가한다.

두 번째는 비교실험은 Cimero에서 지원하지 못하는 7가지 컴포넌트에 대해 패키징 할 경우에 대해 본 도구와 비교한다. ESB상에서 각 컴포넌트간 패키징이 일어날 때 Cimero가지원하지 못하는 7개의 컴포넌트와 다른 컴포넌트 간 패키징 할 경우 지원하지 못하는 일부분을 수작업 해야 하므로 모든 컴포넌트를 지원하는 본 도구와의 비교를 통해 Cimero에서 추가적으로 작성해야하는 Loc에 대한 부분을 비교·평가한다.

세 번째는 배포에 관련된 Loc의 비교를 통해 성능을 평가한다. Cimero에디터는 에디터 내에서 배포된 예전 서비스의 배포/제거는 할 수 없어 이에 대한 작업은 수작업을 통해 이루어진다. 따라서 예전 서비스 관련배포에 사용되는 Loc를 제안도구와 비교·평가하여 성능을 증명할 수 있다.

네 번째는 여러 개의 서비스를 패키징 할 때 사용되는 Loc와, 각각 서비스패키징을 하는 Loc의 비교를 통해 성능을 평가할 수 있다. 본 도구는 한 번에 여러 개의 서비스를 패키징 할 수 있지만, Cimero에서 한 번에 한 개의 서비스만 패키징이 가능하다. 따라서 각각의 서비스를 패키징한 Loc와 여러 서비스를 한꺼번에 패키징할 때 사용되는 Loc를 비교·평가하여 성능을 증명 할 수 있다. 하지만 본 논문에서는 여러 개의 패키징 서비스와 각각의서비스 패키징 비교 부분에 대해서는 생략한다. 다음 <표 5>는 위에서 제시한 각 배치도구 방법에 따른 패키징 효율성에 관한 실험을 나타내고 있다.

- 에디터 플랫폼: Eclipse 3.2.2
- 필수 플러그인: GEF, EMF Plugins
- ESB미들웨어 플랫폼: Servicemix 3.2
- XML버전 : 1.0
- 서비스믹스배치파일(Maven Pom파일), Cimero에디터 2.7.1

<표 5>는 각각의 16개의 서비스를 조건에 따라 패키징하고 그 결과를 작성한 표이다. 실험조건 1을 보면 서비스믹스 배치파일을 수작업을 통해 작성시하는 배치파일과, Cimero와 제안도구와의 비교를 보면, Cimero에 비해서도 10~15% 정도 효율이 높음을 알 수 있는데 이는 본 도구가 일반적인 패키징에서도 2단계를 통한 패키징을 통해 Cimero

〈표 5〉 배치도구에 간 패키징 효율성에 관한 실험

(단위 Loc)

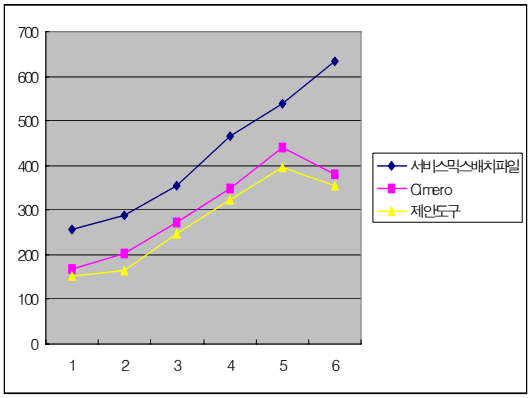
비고	실험조건		서비스믹스 배치파일	Cimero	제안 도구	Cimero 와 Loc 차이
서비스 1	조건 1	일반 서비스패키징	257	167	152	15
서비스 2		일반 서비스패키징	289	202	165	37
서비스 3		일반 서비스패키징	354	273	248	25
서비스 4		일반 서비스패키징	465	349	322	27
서비스 5		일반 서비스패키징	537	440	396	44
서비스 6		일반 서비스패키징	632	379	354	25
서비스 7	조건 2	Cimero 상에 없는 컴포넌트패키징 1	124	124	93	31
서비스 8		Cimero 상에 없는 컴포넌트패키징 2	269	245	221	48
서비스 9		Cimero 상에 없는 컴포넌트패키징 3	327	254	196	131
서비스 10		Cimero 상에 없는 컴포넌트패키징 3	389	389	292	97
서비스 11		Cimero 상에 없는 컴포넌트패키징 3	586	586	469	117
서비스 12	조건 3	배포 및 제거 서비스 패키징	625	625	406	219
서비스 13		배포 및 제거 서비스 패키징	698	698	489	209
서비스 14		배포 및 제거 서비스 패키징	785	785	604	181
서비스 15		배포 및 제거 서비스 패키징	854	854	709	145
서비스 16		배포 및 제거 서비스 패키징	957	957	775	182

※ 단 패키징에서 Loc는 다음과 같이 정의된다.(Loc = 소스코드 + xml 설정파일(xbean)+ 메타파일)

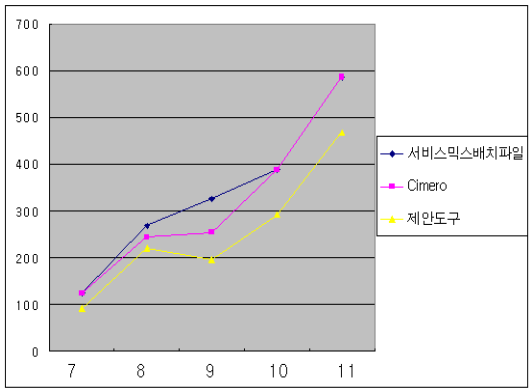
에 비해서도 보다 고효율의 자동화 과정이 이루어짐을 알 수 있다. 실험조건 2는 Cimero 상에 없는 7개의 컴포넌트를 기반으로 패키징 할 경우로 실험을 하였다. Cimero 상에 없는 7개의 컴포넌트를 패키징 할 경우 서비스믹스 배치파일과 같이 최소한의 수작업을 통해 패키징 작업이 이루어지게 된다. 만약 참조할 컴포넌트가 Cimero상에서 존재하는 컴포넌트와 존재하지 않는 컴포넌트가 패키징 된다면 일부분은 수작업을 통해 패키징을 해야 한다. 따라서 이 경우 참조할 패키징 할 컴포넌트의 개수에 따라 다르지만, 보통 최소 20~40%까지 작성하는 코드의 효율이 감소하는 걸 볼 수 있다.

조건 3을 보면 배포 및 제거를 위한 서비스 패키징에 관한 실험이다. 단 여기서 배포 및 제거 서비스는 예전 서비스의 배포/제거라는 단서가 붙는다. 하지만 예전 서비스도 서비스 대상목록에 포함되는 경우가 많기 때문에 이 역시 매우 필요한 기능이다. 실험을 보면 Cimero에서 이러한 기능을 지원하지 않게 때문에 역시 20~40%까지 작성하는 코드의 효율이 감소하는 걸 볼 수 있다.

다음 (그림 17), (그림 18)은 실험조건 1,2를 그래프로 나타내고 있다. (그림 17)을 보면 일반 서비스 패키지일 경우 제안도구가 다른 방법보다 낮은 Loc를 기록하여 보다 높은



(그림 17) 실험조건 1의 패키징 효율성 실험



(그림 18) 실험조건 2의 패키징 효율성 실험

효율을 나타냄을 볼 수 있다. 두 번째 (그림 18)도 역시 패키징을 하는 컴포넌트의 개수에 따라 차이가 있지만, 다른 방법보다 역시 낮은 Loc를 기록하고 있다.

7. 결 론

본 논문에서 제안하는 배치도구가 기존의 서비스믹스를 지원하는 Cimero에디터보다 향상된 기능으로 나온 이유는 Cimero에디터의 개선사항과 문제점을 분석하고 요구사항으로 도출하고 이를 반영하여 설계하였기 때문이다. 본 논문에서 제안된 배치도구는 사용자로 하여금 모델링에서 배포의 전 과정을 통합적으로 수행하고 관리할 수 있어 사용성을 높이고서비스를 효과적으로 지원할 수 있다. 또한 사용자의 수작업을 최소한으로 줄여 서비스의 Time-To-Market을 실현할 수 있으며 수작업과정에서 일어날 수 있는 오류를 최소한으로 줄일 수 있다. 따라서 본 논문에서 제안하는 JBI를 지원하는 컴포넌트배치도구는 사용자에게 효과적인 편리성을 제공 할 수 있을 것으로 기대된다. 향후 연구과제로는 계속적으로 추가되고 있는 컴포넌트의 추가구현과 역으로 압축파일로 패키징된 서비스를 분해하여 완전하게 이를 재사용 할 수 있는 기능의 구현의 확장개발에 대한 연구가 필요하다.

참 고 문 헌

[1] 김운미, 정창원, 성재석, 주수종, "웹 통합문서의 저작과 검색을 지원하는 자동링크지원 시스템의 구축", 정보처리학회, Vol.7, No.1, 2000.

[2] 김지연, 안창원, "MVC 프레임워크를 적용한 웹기반의 시스템 관리도구의 설계 및 구현", 한국정보과학회, Vol.30, No.2, 2003

[3] 박명제, 민준기, 정진완, "추론한 데이터타입을 이용한 질의가능 XML압축", 정보과학회, Vol.32, No.04, pp.441-451, 2005.

[4] 조성대, 박우전, "XML을 이용한 JAVA기반 메뉴 자동생성시스템", 정보과학회, Vol.27 No.2, 2000.

[5] David A. Chappell, "Enterprise Service Bus", O'Reilly Media, 2004.

[6] I. Kamel, C. Faloutsos, "On Packing R-tree", CIKM, 1993.

[7] J Lee, K Siau and S Hong, "Enterprise Integration with ERP and EAI, Communications of the ACM". Vol.46, No.2, pp 54-60, 2003.

[8] Keen, M., et al, "Patterns: Implementing an SOA using an Enterprise Service Bus", IBM Redbook, 2004.

[9] MT Schmidt, B Hutchison, P Lambros, R Phippen, "The Enterprise Service Bus: Making service-oriented architecture real", IBM Systems Journal, Vol.44, No.4, pp.781-797.

[10] Ron Ten-Hove, Peter Walker, "Java Business Integration (JBI) 1.0", Sun Microsystems, 2005.

[11] Servicemix, http://www.servicemix.org

[12] Erich gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Pattern, Addison-Wesley Pub.Co., 1994)

박 제 원

e-mail : kkkjw22@hotmail.com

2006년 숭실대학교 컴퓨터학과(공학석사)

2006년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 웹서비스, SOA, ESB, 유비쿼터스컴퓨팅 등

최 재 현

e-mail : uniker80@gmail.com

2006년 숭실대학교 컴퓨터학과(공학석사)

2006년~현 재 숭실대학교 컴퓨터학과 박사과정

관심분야 : 웹서비스, SOA, ESB, 유비쿼터스컴퓨팅

이 재 성

e-mail : 3935430@hanafos.com

2009년 숭실대학교 컴퓨터학과 공학석사

관심분야 : 소프트웨어아키텍처, SOA 등

이 남 용

e-mail : nylee@ssu.ac.kr

1983년 고려대학교 경영정보학과(석사)

1993년 미시시피주립대학 경영정보학과 (경영학박사)

1979년~1983년 국군정보사령부 정보처 정보시스템분석 장교

1983년~1999년 한국국방연구원 군수체계 및 정보체계연구부장

2000년 한국전자거래학회 논문편집위원장

2004년 한국정보통신기술사협회 회장

1999년~현 재 숭실대학교 컴퓨터학과 교수

관심분야 : 웹서비스, SOA, ESB 소프트웨어테스팅, 시스템엔지니어링 등