한국 CAD/CAM 학회 논문집
제 13권 제 3호 2008년 6월 pp. 217-226

학술논문

병렬 웹 서비스를 이용한 조립체 모델 데이터의 획득

김병철*, 한순흥**

Retrieval of Assembly Model Data Using Parallel Web Services

Byungchul Kim* and Soonhung Han

ABSTRACT

Web Services for CAD (WSC) aims at interoperation with CAD systems based on Web Services. This paper introduces one part of WSC which enables remote users to retrieve assembly model data using Web Services. However, retrieving assembly model data takes long time. To resolve this problem, this paper proposes using parallel Web Services. As assembly models comprise a set of part models, it is easy to separate the problem domain into smaller problems. In addition, Web Services inherently supports distributed computing. This characteristic makes the parallel processing of Web Services easy. Firstly, the implementation of WSC which retrieves assembly model data based parallel Web Services is shown. And then, for the comparison, the experiments on the retrieval of assembly model data based on single Web Services and parallel Web Services are shown.

Key words : Web Services for CAD, Data sharing, Parallel Web Services

1. 연구 배경

협업 설계는 다양한 분야의 이해 관계자들이 설계 결정에 참여하고 인터넷 기반의 분산 환경에서 기업 간의 경계를 가로질러 제품 정보를 공유하는 새로운 설계 프로세스이다[1]. 협업 설계는 분산되어 있는 설계 자원을 통합시키고, 제품 설계 시간 및 제품 생산 비용을 단축시켜서, 글로벌 경쟁 속에서 살아남을 수 있는 경쟁력을 가질 수 있게 해준다. 특히 분산된 환경에서 CAD 데이터를 쉽고 빠르게 공유하는 것은 협업 설계를 실현하기 위하여 우선 해결해야 할 이슈 중 한 가지이다.

소프트웨어 시스템들이 제품 설계 프로세스에 참여할 때, CAD 데이터를 얻기 위한 방법은 2가지이다[2]. 첫 번째 방법은 CAD 데이터 교환으로써, 일반적으로 STEP(STandard for the Exchange of Product model data, 제품 모델 데이터 교환에 관한 표준)[2]이나 IGES(Initial Graphics Exchange Specification)와 같은 표준을 이용한 파일 기반 교환 방법이다. 두 번째 방법은 실시간 데이터 공유 방법으로써, CAD 시스템이 제공하는 API(Application Programming Interface)를 사용하는 방법이다.

그러나 CAD 시스템의 API를 사용하는 방법은 대부분 분산 환경을 지원하지 않는다. 이를 해결하고자 소켓(socket)이나 TCP/IP를 기반으로 한 방법들[3]이 시도되었다. 그러나 소켓이나 TCP/IP를 이용하는 방법은 체계화된 방법을 제공하지 않기 때문에, 컨소시업이나 대기업이 주도하는 CORBA(Common Object Request Broker Architecture)[4]나 DCOM(Distributed Component Object Model)[5]과 같은 분산 객체 기술을 이용한 접근 방법들이 시도되었다. 그러나 분산 객체 기술을 사용한 방법은 동기식(synchronous) 통신과 작은 양의 데이터를 빈번하게 교환하는 것을 주목적으로 하기 때문에, 비동기식(asynchronous) 통신과 대용량 데이터 전송을 보통 필요로 하는 공학적 설계 환경에는 적합하지 않다[6]. 더구나 CORBA나 DCOM은 구현의 복잡성으로 인하여 실제로 정착되지도 않았다.

분산 객체 기술을 사용하는 것에 대한 대안으로써, XML(eXtensible Markup Language)과 인터넷에 기

*학생회원, 한국과학기술원 기계공학과
**정회원, 한국과학기술원 기계공학과
- 논문투고일: 2007. 09. 04
- 심사완료일: 2008. 04. 03

반을 둔 웹 서비스(Web Services)[7]가 출현하였다. 웹 서비스는 다양한 기술들로 구성되어 있지만, 가장 핵심적인 기술은 웹 서비스의 인터페이스를 기술하는 WSDL(Web Services Description Language)[8], 데이터 형식을 정의하기 위한 XML 스키마(XML Schema)[9], 인터넷을 통하여 응용 프로그램들이 정보를 교환할 수 있게 하는 전송 프로토콜인 SOAP (Simple Object Access Protocol)[10]이다.

XML 및 이와 관련된 기술을 분산 설계 환경에서 사용하면, 분산 객체 기술의 사용과 관련된 문제점들을 극복할 수 있다. XML은 비동기식 통신 및 대용량 데이터 교환에 더 적합하다. 뿐만 아니라, SOAP가 HTTP(HyperText Transfer Protocol)를 기반 전송 프로토콜로써 사용할 수 있기 때문에, XML 통신은 방화벽을 쉽게 통과할 수 있다. 또한, 분산 객체 기술에 비하여 더 플랫폼 독립적이다. 또한, 더 중요한 것은 XML과 이와 관련된 기술들은, 활용 가능한 도구들을 많이 제공한다는 것이다[6]. 이러한 장점 때문에 엔지니어링 환경에 웹 서비스를 적용하는 연구[11]도 있었다.

그러나, 웹 서비스는 대용량 데이터를 포함하고 있는 XML 문서를 기반으로 통신을 한다. 이는 웹 서비스를 이용할 경우, 많은 양의 데이터 전송이 필요하다는 것을 의미한다. 더구나 XML 문서는 많은 수의 태그(tag)를 포함하고 있다. 데이터 분량 문제는 데이터 전송뿐만 아니라, 데이터를 생성하는 곳에서도 발생한다. CAD 조립체 모델의 경우 일반적으로 큰 크기를 가지기 때문에, 이를 웹 서비스를 위하여 XML 형태로 변환을 하는 데 많은 시간이 걸리게 된다. 물론, 미리 CAD 데이터를 변환을 시켜놓고 이를 이용하는 방법을 생각할 수도 있다. 그러나 이는 웹 서비스 제공자가 서비스 요구자의 상태 정보를 관리해야 하기 때문에, 서비스 제공자와 요구자 간에 강한 결합(tight coupling) 관계가 요구된다. 이는 분산 환경에서 시스템 통합을 어렵게 만드는 한 원인이다. 더구나 강한 결합 관계 및 상태 정보 유지는, 웹 서비스의 궁극적인 목적인 서비스 지향 아키텍처(Service-Oriented Architecture, SOA)[12]의 설계 원리에 맞지 않는다.

본 연구에서는 웹 서비스를 이용하여 CAD 조립체 모델의 데이터를 가져오는데 초점을 맞춘다. 그러나 웹 서비스를 통하여 CAD 조립체 모델의 데이터를 가져오는 데 많은 시간이 걸린다는 단점을 극복하기 위하여, 본 연구에서는 이를 해결하기 위하여 병렬 처리 방식으로 웹 서비스를 적용하는 것을 제안한다.

본 연구의 상세한 설명에 앞서, 본 연구와 비교할 수 있는 세 개의 기존 연구에 대하여 살펴본다. 비교 분석할 연구는 CAD를 위한 웹 서비스[13], 문서 구동 설계[1], OMG의 CAD Services[14]이다. Table 1은 본 연구와 이들 세 연구의 비교표이다.

본 저자의 이전 연구[13]에서는 CAD 시스템의 기능을 웹 서비스로 구현하는 'CAD를 위한 웹 서비스 (Web Services for CAD, WSC)'를 제안하였고, 이를 기반으로 CAD 모델 데이터를 가져오는 연구를 수행하였다. 이 연구에서는 웹 서비스를 위한 아키텍처 및 인터페이스, 데이터 모델을 제안하였고, 이를 Pro/Engineer에 적용하는 실험을 하였다.

Wang의 연구[1]에서는 웹 서비스를 이용하여 CAD 모델을 생성하는 연구를 수행하였다. 본 연구의 이전 연구[13]가 CAD 모델 데이터를 가져오는 데 초점을 맞춘 것과는 대조적이다. Wang의 연구에서 구현한 시스템에서는 입력으로 RDF(Resource Description Framework)[15]로 기술된 특징 형상 기반의 모델 생성 정보를 받고, 결과로 생성된 CAD 모델을 반환한다. 또한, 온톨로지(ontology)를 사용하여 중립 특징 형상을 표현하고, 이를 상업용 CAD 시스템의 특징 형상과 매핑을 시도한 것이 특징이다.

위의 두 연구는 웹 서비스에 기반을 두고 있기 때문에, 인터페이스가 단순하다. 대신, 전달되는 데이터가 XML 문서 형태로 구조화되어 있어 복잡하다. 이런 이유로 Wang의 연구에서는 문서 구동 설계 (document-driven design)라는 용어를 사용하기도 하였다.

Table 1. Comparison of the related works

	This study	Web Services for CAD[13]	Document-Driven Design[1]	OMG CAD Services[14]
Focus	Assembly & part data retrieval	Part data retrieval	Part model creation	Assembly & part data retrieval
Base technology	Web Services	Web Services	Web Services	CORBA
Data format	XML Schema	XML Schema	RDF	Raw data
Asynchronous or synchronous	Asynchronous	Asynchronous	Asynchronous	Synchronous
Process	Parallel process	Single process	Single process	Single process

OMG(Object Management Group)의 CAD Services[14]는, 웹 서비스가 아닌 분산 객체 기술 중의 한 가지인 CORBA를 기반으로 하는, CAD 데이터 공유를 위한 인터페이스이다. 비록, CORBA에 기반을 두고 있지만 그 목적이 WSC와 유사하여 비교할 가치가 있다. CAD Services는 CORBA에 기반을 두고 있기 때문에, 인터페이스가 복잡하다. 인터페이스들은 서로 연관되어 있고, 자신이 원하는 기능을 찾기 위해서는 많은 단계를 통하여 함수를 호출해야 한다. 대신 매 단계에서 전달되는 데이터 형식은 단순하다. 이는 CORBA가 동기식 통신과 작은 양의 데이터 교환에 적합하다는 특징을 잘 보여준다.

앞에서 살펴본 세 연구 중에서, 본 연구는 WSC 연구에 기반을 두고 있다. 따라서, 유사한 아키텍처와 인터페이스 및 데이터 모델을 사용한다. 그러나 WSC 연구에서는 CAD 모델 데이터를 가져오는 데 많은 시간이 걸린다는 문제점을 가지고 있다는 점이 파악되어, 본 연구에서는 이를 해결하기 위하여 병렬 처리를 사용한다.

본 논문은 이러한 내용을 위하여 다음과 같이 구성되어 있다. 2절에서는 CAD 조립체 데이터를 가져오기 위한, 병렬 처리 절차 및 인터페이스, 데이터 모델에 대하여 설명한다. 3절에서는 본 연구 내용을 구현하고 실험한 내용을 보여준다. 마지막으로 4절에서는 결론 및 향후 연구 방향에 대해 소개한다.

2. 조립체 모델 데이터 획득을 위한 WSC

2.1 기본 개념

본 연구에서는 WSC(Web Services for CAD, CAD를 위한 웹 서비스)의 한 기능인, 웹 서비스를 통하여 CAD 조립체 모델 데이터를 멀리 떨어진 컴퓨터에서 가져오는 것을 목적으로 한다. 이 기능은 입력으로 이진(binary) 형태의 CAD 조립체 모델을 받고, 결과로 XML 형태의 CAD 데이터를 생성한다. 이를 위한 WSC 구성 요소 및 구성 요소가 사용하는 데이터 형식은 Fig. 1과 같다.

Fig. 1에서 'CAD 시스템(CAD system)'은 제품을 설계하기 위하여 우리가 사용하는 소프트웨어 시스템을 의미한다. 'CAD 어댑터(CAD adaptor)'는 서비스 요구자에게 반환할, 결과 XML 문서를 가공하는 역할을 한다. 여기서 생성되는 XML 문서는 '웹 서비스 레이어(Web service layer)'를 통하여 서비스 요구자에게 전달된다. 웹 서비스 레이어는 웹 서비스로

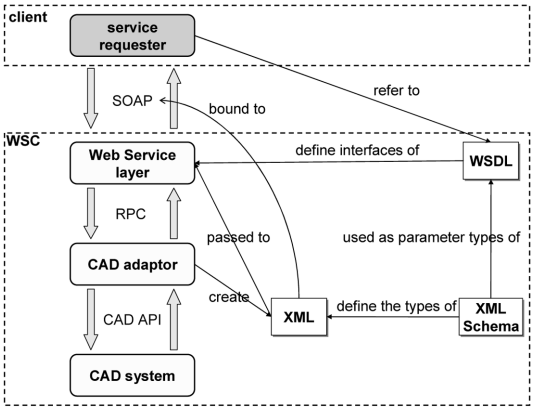


Fig. 1. Configuration components and data types of WSC[13].

CAD 어댑터의 기능을 제공하는 역할을 한다. 웹 서비스 레이어는 서비스 요구자가 전달한 CAD 모델 파일을 CAD 어댑터로 전달하고, CAD 어댑터가 보내오는 결과를 다시 서비스 요구자에게 전달한다. 이 때 서비스 요구자와 웹 서비스 레이어 사이의 통신을 위하여 SOAP가 사용된다.

CAD 어댑터는 CAD 모델에 관한 XML 문서를 생성한다. 이 때, XML 문서는 XML 스키마로 정의된다. XML 스키마 문서에는 CAD 모델 데이터를 정의하는 데이터 형식뿐만 아니라, 웹 서비스 요구자가 서비스를 요청할 때 사용하는 입력 인자를 위한 데이터 형식도 정의되어 있다. CAD 모델 데이터를 정의하는 데이터 형식은 WSDL 내에서 서비스의 반환 형식으로 사용된다. WSDL 문서는 XML 스키마 형태의 서비스 입력 및 출력 데이터 형식뿐만 아니라, 서비스가 제공하는 서비스 연산자를 정의한다. 또한 WSDL 문서는 서비스 요구자가 서비스를 요청할 때 참조된다. WSC가 CAD 어댑터로부터 얻는 XML 데이터를, 서비스 요구자에게 최종적으로 전달하기 위해서 SOAP가 사용된다. SOAP에는 앞서 정의한 WSDL을 수용하는 XML 문서와 웹 서비스에 필요한 부가적인 정보들이 포함된다.

2.2 문제점 및 해결 방안

웹 서비스를 통하여 데이터를 받기 위해서는 XML 문서 형태로 가공을 해야 한다. 그러나 조립체(assembly) 모델의 경우 일반적으로 크기가 크기 때문에 XML 문서 형태로 가공을 하는 데 많은 시간이 걸린다. 이는 WSC와의 원활한 의사 소통을 방해한다.

이를 해결하기 위해서 본 연구에서는 병렬 처리 방

식을 적용한다. 병렬 처리를 적용하기 위해서는 문제가 비슷한 형태의 더 작은 문제로 분리될 수 있어야 한다. 조립체 모델은 다수의 부품 모델들로 구성이 되어 있기 때문에 병렬 처리를 적용하기에 적합하다. 또한, 웹 서비스는 본질적으로 분산 시스템의 특성을 가지고 있기 때문에, 적은 노력으로 병렬 처리를 적용할 수 있다.

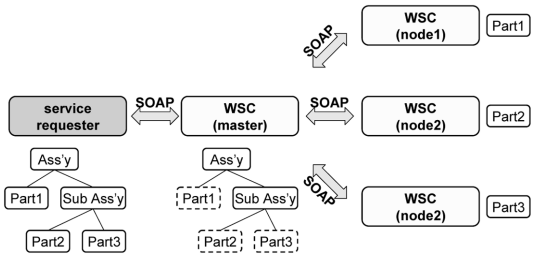


Fig. 2. Parallel WSC for the retrieval of assembly model data.

Fig. 2는 조립체 모델 데이터를 가져오기 위한 병렬 WSC의 구조를 보여준다. 여기서 사용되는 WSC 컴포넌트들은 모두 동일하다. 즉, 모두 같은 아키텍처와 인터페이스, 데이터 모델을 사용한다. 또한, 각 WSC는 동일한 CAD 시스템을 필요로 한다. 그러나, 상황에 따라서 다른 역할을 한다. 서비스 요구자가 WSC에 서비스를 요청하면 서비스 요청을 받은 WSC는 마스터(master) WSC가 된다. 이 때, 마스터 WSC는 조립체 모델의 구조를 분석하여 구성하고 있는 파트 모델을 알아낸다. 그리고, 구성 파트 모델을, 노드 역할을 하는 WSC로 보내어, 파트 모델 데이터를 가져오도록 요청한다. 각 노드로부터 반환 받은 파트 모델 데이터를, 마스터 WSC가 조립체 모델의 구조에 맞게 구성을 하면, 우리가 원하는 조립체 모델 데이터가 완성된다.

2.3 인터페이스 및 데이터 모델

서비스 요구자가 WSC를 사용하고 결과를 받기 위해서는, WSC와 상호작용하기 위한 인터페이스가 필요하다. 본 연구에서 사용되는 WSC의 인터페이스는 Fig. 3에 UML 클래스 다이어그램으로 표현되었다.

Fig. 3. UML class diagram for describing the interface of WSC.

Fig. 3의 WSCInterface 인터페이스는 3개의 연산자를 정의한다. 'RetrievePartData' 연산자는 이진 형태의 파트 모델을 입력으로 받아서, 이에 대한 데이터를 XML 형태로 반환한다. 이 때, 반환되는 XML 형태는 'Part' 클래스로 정의가 되는데, 이에 대한 구조는 이전 연구[13]에 자세히 설명되어 있다. 'RetrieveAssemblyComponentsList' 연산자는 이진 형태의 조립체 모델을 입력으로 받고, 조립체 모델을 구성하는 부조립체 및 파트 파일에 대한 목록을 반환한다. 'RetrieveAssemblyData' 연산자는 이진 형태의 조립체 모델과 조립체를 구성하는 부조립체 및 파트 모델을 입력으로 받고, 이에 대한 XML 데이터를 반환한다. 이 때 반환되는 XML 데이터는 'SubassemblyComponent' 클래스로 정의가 되는데, 이는 Fig. 4에 나와 있다.

Fig. 4에서 SubassemblyComponent 클래스는 부조

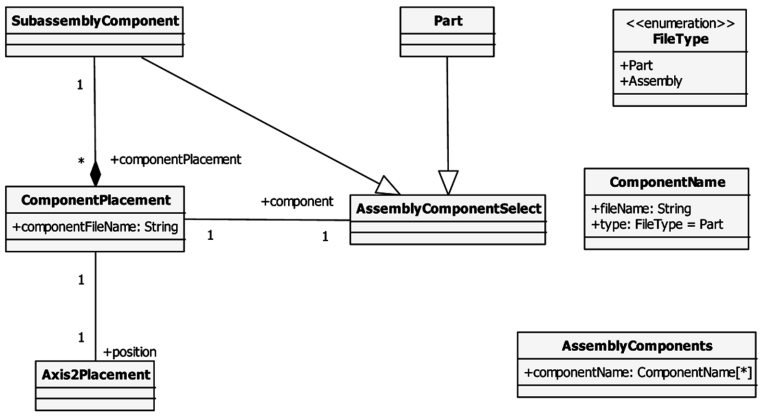


Fig. 4. UML class diagram for describing the data model of an assembly model.

립체 및 파트의 위치를 표현하는 다수의 ComponentPlacement 클래스로 구성된다. ComponentPlacement 클래스는 AssemblyComponentSelect 클래스를 참조하는데, 이 클래스는 실제로는 부조립체를 표현하는 SubassemblyComponent 클래스 또는 파트를 표현하는 Part 클래스이다.

최종적으로 UML 클래스 다이어그램으로 정의된 인터페이스와 데이터 모델은, WSDL과 XML 스키마로 표현된다.

2.4 병렬 처리 절차

WSC가 조립체 데이터를 처리하기 위해서는 Fig. 5와 같은 절차를 따른다. Fig. 5는 UML 시퀀스 다이어그램이다.

서비스 요구자가 WSC에게 조립체 모델 데이터를 알려달라고 요청을 하면(1: RetrieveAssemblyData), 서비스 요청을 받은 WSC는 마스터 WSC가 된다. 마스터 WSC는 서비스 요구자로부터 받은 조립체 모델의 구조 정보를 알아내고(2: GetAssemblyStructure), 조립체 모델 정보로부터 조립체를 구성하는 파트 모델을 찾는다(3: FindParts). 그리고, WSC 마스터는 다른 WSC 노드들을 찾는다(4: FindNodes). WSC 노드를 찾고 나면, 루프에 진입하게 되는데, 이 루프는 조립체 모델의 모든 파트 모델 데이터를 가져오게 되면 종료된다. 루프 내에서는 파트 모델 데이터를 가져오기 위한 작업이 진행되는데, 우선 처리해야 할 파트 모델을 찾는다(5: FindUnprocessedPart). 또한 이 파트를 처리할 유휴 WSC 노드를 찾는다(6: FindLazyNode). 그리고 이 노드에게 파트 데이터를 가져오는 작업을 할당한다(7: RetrievePartData). 이 때, WSC 노드는 비동기식으로 호출이 되기 때문에, 마스터 WSC와 WSC 노드들은 동시에 작업을 진행하게 된다. WSC 노드가 작업을 마치면, 마스터 WSC에게 작업 완료를 알리고(8: RetrievePartDataCompleted), 마스터 WSC는 노드로부터 넘겨받은 파트 데이터를 조립체 데이터와 결합을 시킨다(9: AttachPartToAssembly). 그리고 작업을 마친 노드는 다시 유휴 상태가 되고, 다음 작업을 기다린다. 모든 파트 모델에 대한 작업이 끝나면 루프를 종료하게 되고, 파트 데이터와 결합된 조립체 데이터는 서비스 요구자에게 반환된다.

2번 단계(GetAssemblyStructure)에서는 CAD adaptor를 통하여 조립체 모델의 구조 정보를 알아낸다. CAD adaptor는 일반적으로 CAD 시스템의 API를 사용하여 조립체 모델의 구조를 알 수 있다. 2번 단계(GetAssemblyStructure)에서 조립체 모델 정보를 가져오면, Fig. 4의 구조를 가지게 된다. 이를 XML 문서로 표현하면 Fig. 6(a)와 같다. 그러나 여기서 Part 요소에 대한 내용은 사실 비어있게 되고, 실제 내용은 WSC 노드에게 다시 요청을 한다. WSC 노드가 생성한 파트 데이터는 Fig. 6(b)의 형태를 갖는다. Fig. 6(a)의 Part 요소에 Fig. 6(b)의 Part 요소를 결합시키

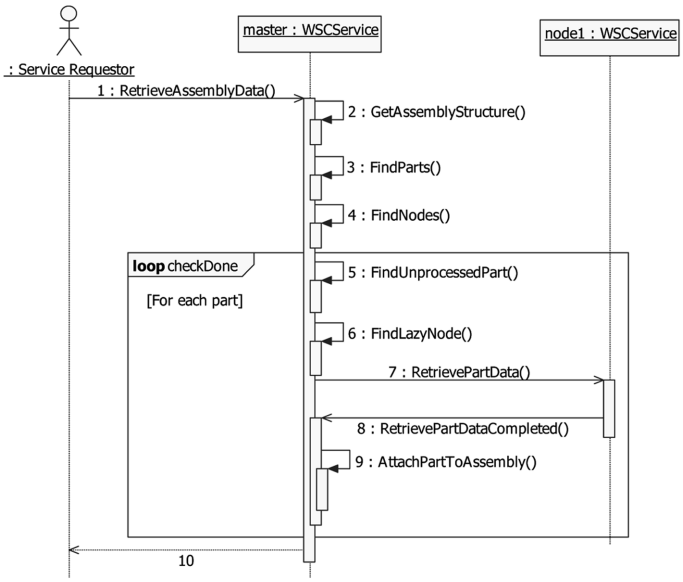


Fig. 5. UML sequence diagram for describing the parallel processing of an assembly model data.

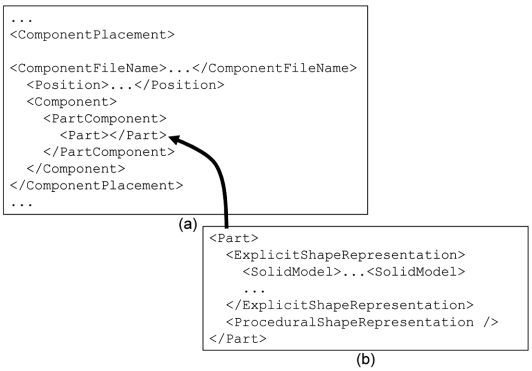


Fig. 6. XML representation of an assembly model and a part model.

면 최종적으로 조립체 데이터가 완성된다.

4번 단계(FindNodes)에서 WSC 노드를 찾을 때, 네트워크 상에 있는 모든 WSC들이 다른 WSC에 대한 위치를 미리 알고 있다고 가정을 하고 진행하였다. 그러나, 다른 WSC 노드를 찾기 위해 웹 서비스 기술 중 한 가지인 UDDI(Universal Description Discovery and Integration)[16]를 이용할 수 있다.

6번 단계(FindLazyNode)에서 유휴 노드에 파트 모델을 할당할 때에 다양한 전략을 생각할 수 있다. 주요 변수로 WSC의 처리 속도, WSC 노드들 간의 네트워크 속도, 파트 모델의 크기 등이 될 수 있다. 그러나 이러한 변수들은 측정하기가 어렵기 때문에, 본 연구에서는 가장 단순한 전략인, 유휴 노드에 임의적으로 파트 모델을 할당하는 방법을 적용하였다.

Fig. 5에서 병렬 처리가 일어나는 구간은 5~9번 단계를 포함하는 루프 구간이다. Fig. 7은 이 부분의 내

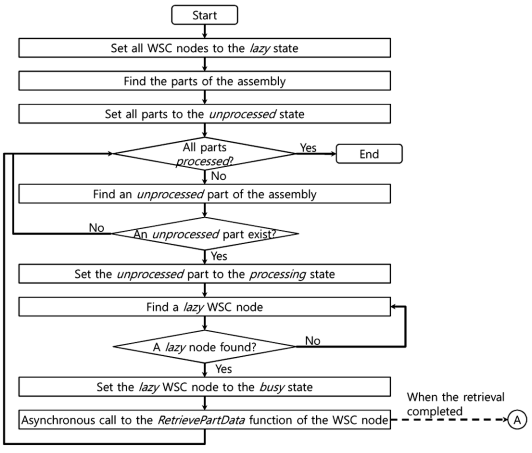


Fig. 7. Flowchart for processing the parts data of the assembly.

용을 플로우 차트로 표현한 것이다.

각각의 파트는 *unprocessed*, *processing*, *processed*의 세 가지 상태를 가진다. 이는 WSC 노드에서의 파트 처리 상태를 나타낸다. *Unprocessed* 상태는 그 파트에서 아직 XML 데이터를 생성하지 않았다는 것을 의미한다. *Processing* 상태는 어떤 WSC 노드에서 그 파트에 대한 XML 데이터를 생성하고 있다는 것을 의미한다. *Processed* 상태는 그 파트에 대한 XML 데이터를 획득해서 조립체 정보에 추가한 것을 의미한다. 초기에 파트 모델 정보를 가져올 때, 모든 파트의 상태는 *unprocessed*이고, 모든 파트의 상태가 *processed*가 되어야 전체 작업이 종료된다. 전체 파트의 상태가 *unprocessed*가 아니더라도, 상태가 *processing*인 파트가 있다면 프로세스는 종료되지 않는다.

각각의 노드는 *lazy*와 *busy*의 두 가지 상태를 가진다. *lazy* 상태는 그 WSC 노드가 현재 아무 작업도 하지 않고 있다는 것을 의미한다. *busy* 상태는 그 WSC 노드가 한 개의 파트로부터 XML 데이터를 생성하는 중이라는 의미이다. 모든 WSC 노드가 *busy* 상태라면, 그 중 한 개의 WSC 노드가 작업을 끝낼 때까지, 새로운 파트를 처리할 수 없다. 이 때에는 WSC 노드 중 한 개의 작업이 종료하기를 기다려야 한다.

Fig. 7의 내용을 기술하면, 초기에는 모든 WSC 상태가 *lazy*가 된다. 그리고 조립체를 구성하는 모든 파트의 상태가 *unprocessed*가 된다. 그 다음 모든 파트의 상태가 *processed*가 된다면 전체 프로세스를 종료한다. 그렇지 않다면 *unprocessed* 상태의 파트를 찾는다. 만약 *unprocessed* 상태의 파트도 없다면, 모든 파트의 *processing* 상태가 *processed* 상태가 될 때까지 기다려야 전체 프로세스가 종료된다. *unprocessed* 상태의 파트가 존재한다면, 그 파트의 상태를 *processing*으로 변경하고, 파트 데이터 획득 작업을 시작한다. 파트 데이터 획득 작업을 하기 위해서 우선 *lazy* 상태의 WSC 노드를 찾는다. 현재 *lazy* 노드가 없으면, *lazy* 노드가 생길 때까지 기다려야 한다. *Lazy* 상태의 노드를 찾았으면, 그 노드의 상태를 *busy*로 바꾼다. 그리고 처리해야 할 파트를 찾은 WSC 노드에게 전달하여 파트 데이터 획득 작업을 수행하게 한다. 이 때, 이 작업은 비동기 모드로 수행한다. 이는 마스터 WSC에서는 노드 WSC가 작업을 마칠 때까지 기다리지 않고, 계속 자신의 작업을 진행한다는 것을 의미한다. 비동기 모드 작업이 실제로 병렬 처리 효과를 만들어 낸다. 노드 WSC의 작업이 끝나면 Fig. 8

의 A로 프로세스를 분기시킨다.

Fig. 8은 노드 WSC가 파트 데이터 획득 작업을 끝냈을 때 수행되는 작업을 보여준다.

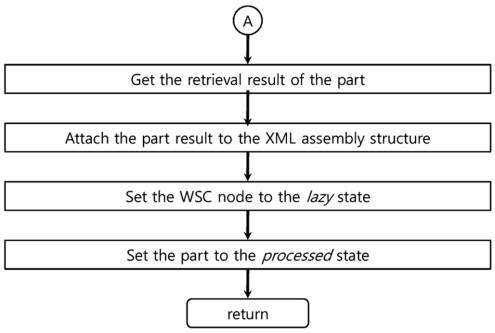


Fig. 8. Flowchart for post-processing the parts data of the assembly.

노드 WSC가 작업을 마치면, Fig. 8의 A로 프로세스를 분기시킨다. 그리고 노드 WSC가 생성한 XML 형태의 파트 데이터를 가져온다. 그리고 파트의 XML 데이터를 Fig. 6에서처럼 XML 형태의 조립체 정보에 붙인다. 또한, 노드 WSC의 상태를 *lazy*로 만들어 새로운 작업을 할 수 있도록 한다. 파트의 상태도 *processed*로 바꾼다.

이러한 과정을 모든 파트의 상태가 *processed*가 될 때까지 진행하면 최종적으로 XML 형식의 조립체 모델 데이터를 얻을 수 있다.

3. 구현 및 실험

3.1 구현 환경

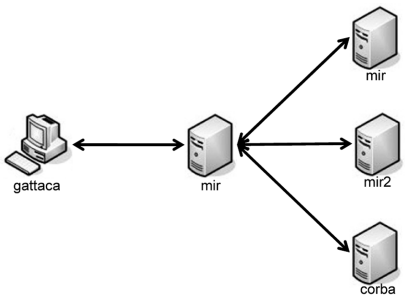
2절의 내용을 다음과 같은 환경에서 구현을 하였다.

- 운영체제: Windows XP
- CAD 시스템 (CAD system): Pro/ENGINEER Wildfire 3.0
- CAD 어댑터 (CAD adaptor)
  - Visual C++ 2003
  - WTL (Windows Template Library)
  - ATL (Active Template Library)
  - Pro/TOOLKIT
- 웹 서비스 레이어 (Web Service layer)
  - Visual C# 2005
  - .NET Framework 2.0
  - WSE (Web Services Enhancements) 3.0
  - IIS (Internet Information Services)
- WSCClient
  - Visual C# 2005
  - .NET Framework 2.0
  - WSE 3.0
- WSCViewer
  - Visual C++ 2003
  - MFC (Microsoft Foundation Classes)
  - ACIS R15
  - HOOPS 12.0

CAD 어댑터와 웹 서비스 레이어는 WSC(Web Services for CAD, CAD를 위한 웹 서비스)의 구성 컴포넌트이다. WSCClient는 실험을 위하여 구현된 WSC 서비스 요구자이다. WSCViewer는 WSC 서비스 요구자가 WSC로부터 받은 결과를 사용자에게 가시화 시켜준다.

3.2 실험 환경

실험은 Fig. 9와 같은 환경에서 수행되었다.



Name	Role	CPU	RAM
gattaca	WSCClient & WSCViewer	Intel P4 3.0 GHz	2 GB
mir	Master & node1	Intel Core 2 1.86 GHz	2 GB
mir2	Node2	Intel Pentium M 1.3	512 MB
corba	Node3	Intel P4 2.60 GHz	512 MB

Fig. 9. Configuration of experimental environment.

실험에는 3대의 PC와 1대의 노트북이 사용되었는데, 1대(gattaca)는 서비스 요구자 역할(WSCClient 및 WSCViewer)을 하고, 3대는 WSC 노드의 역할을 한다. 3대의 노드 중 1대(mir)는 마스터 WSC의 역할을 동시에 한다. 4대의 컴퓨터는 서로 100Mbps LAN으로 연결되어 있다.

실험에 사용된 테스트 모델은 Fig. 10과 같다. 이 모델은 연필 깎기 모델의 일부분이다. 테스트 모델은 Pro/Engineer 조립체 모델로써, 8개의 파트 모델로 구성되어 있다. 각 파트 모델에 관한 정보가 Table 2에 나와 있다.

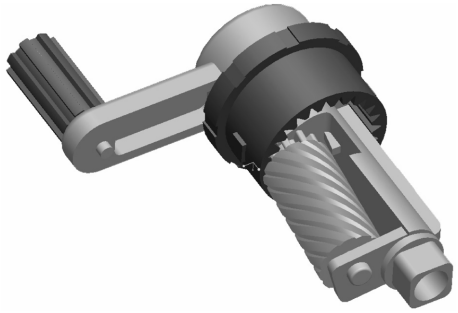


Fig. 10. Test model used in the experiment.

Table 2. Data of each part model composing the test model

Part no.	Name	No. of faces	No. of edges	File size (KB)
1	Blade guard	124	438	501
2	Blade holder	73	208	405
3	Handle	40	133	309
4	Handle holder	32	108	137
5	Handle shaft	12	22	69
6	Helical gear	30	117	310
7	Gear	68	252	277
8	Pin	12	26	84
Total		391	1304	2092

3.3 실험 결과

실험은 두 단계로 진행되었다. 각 단계의 실험은 5번씩 수행이 되었고, 실험 결과치는 5번의 실험 결과의 평균치이다. 첫 번째 실험에서는 병렬 처리를 사용하지 않고, 한 개의 WSC를 이용하여 조립체 모델 데이터를 가져왔다. 이 실험에서는 Fig. 9의 컴퓨터 중 WSCClient와 마스터 WSC만이 사용되었다.

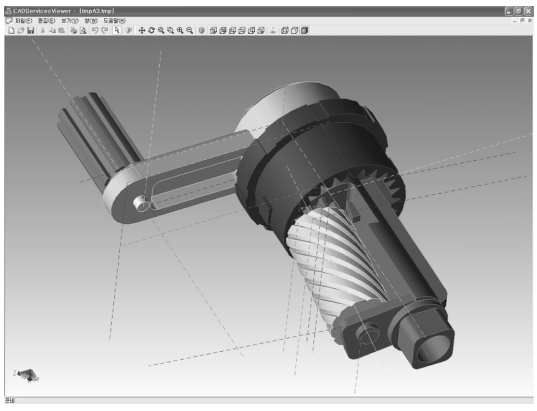


Fig. 11. Visualization of the assembly data retrieved from WSC.

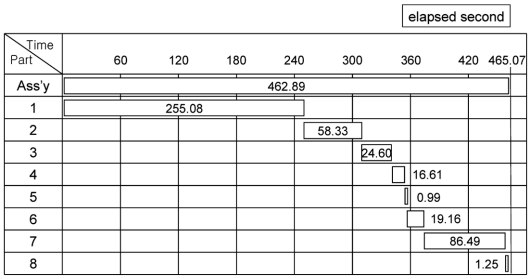


Fig. 12. Result of the time measurement in the first experiment.

WSC로부터 획득한 조립체 데이터를 WSCViewer로 가시화 한 결과가 Fig. 11에 나와 있다. Fig. 11의 각 파트의 색상은, 파트를 구분하기 위하여 임의로 부여된 것이다. Fig. 10의 테스트 모델과 Fig. 11의 결과가 동일한 것을 확인할 수 있다.

Fig. 11의 결과를 얻는데 걸린 시간이 Fig. 12에 정리되어 있다.

WSCClient가 서비스를 요청하고 응답 받을 때까지 걸린 시간은 평균 465.07초였다. 가장 복잡한 1번 파트를 처리하는 데 가장 많은 시간이 걸렸다. 반면 5번 파트와 8번 파트와 같이 간단한 경우에는 약 1초의 처리 시간이 걸렸다.

두 번째 실험에서는 Fig. 9의 4대의 PC를 모두 사용하여, 병렬 처리 방식으로 조립체 모델 데이터를 가져왔다. WSC로부터 획득한 조립체 데이터를 가시화 한 결과는 Fig. 11과 동일하다.

두 번째 실험에서 조립체 데이터를 획득하기 위하여 걸린 시간이 Fig. 13에 정리되어 있다. 두 번째 실험에서는 WSCClient가 서비스를 요청하고 응답 받을 때까지 걸린 시간이 평균 284.06초였다. 노드 1에서 1번 파트를 처리하는데 걸린 시간이 전체 시간에 가장 큰 영향을 주었다. 노드 1에서 1번 파트를 처리하는 동안 노드 2와 노드 3은 다른 파트를 모두 처리하고 유휴 상태가 되어 있었다. 만약 조립체를 구성파는

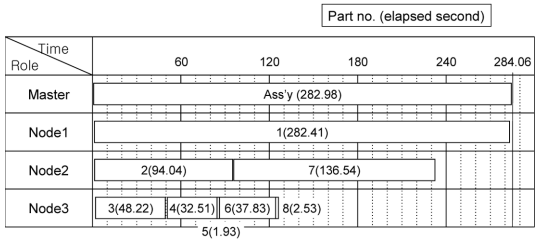


Fig. 13. Result of the time measurement in the second experiment.

파트들의 복잡도가 균일하다면 좀 더 좋은 성능 향상을 기대할 수 있을 것이라고 예상된다.

동일한 파트를 처리할 때, 첫 번째 실험에서 걸린 시간보다 두 번째 실험에서 처리하는 데 걸린 시간이 더 길었다. 이는 노드 2와 노드 3의 성능이 노드 1보다 떨어지기 때문이다. 그럼에도 불구하고, 전체적으로 걸린 시간은 두 번째 실험이 더 짧았다.

그러나 본 실험에서 사용된 컴퓨터들은 서로 가까운 거리에 위치해 있고, 빠른 속도의 LAN으로 연결되어 있기 때문에, 컴퓨터 간의 데이터 전송 시간이 파트 처리 시간에 비하여 짧았다. 만약, 먼 거리에 위치해 있는 컴퓨터들을 사용하여 실험을 한다면, 데이터 전송 시간이 조립체 데이터를 획득하는 데 걸리는 시간에 영향을 줄 것이라고 예상된다.

4. 결론 및 향후 연구

본 연구에서는 웹 서비스를 통하여 조립체 데이터를 획득하는 데 걸리는 시간을 단축하기 위하여, 병렬 웹 서비스를 적용하는 것을 제안하였다. 또한, 이를 위한 인터페이스 및 데이터 모델, 병렬 처리 방법 등을 제시하고 구현하였다. 그리고, 실험을 통하여 병렬 처리 방법을 사용하여 구현된 시스템이 더 좋은 성능을 보여준다는 것을 확인하였다.

그러나 본 연구에서는 WSC들이 서로에 대해서 알고 있다는 가정을 하고 연구를 진행하였다. 여기에 UDDI를 적용하면 인터넷상에 있는 WSC들이 동적으로 서로를 찾아서 병렬 처리를 수행할 수 있을 것이다. 또한, 본 연구에서는 단순한 로드 밸런싱(load balancing) 전략을 적용하였기 때문에, 조립체를 구성하는 파트의 크기가 균일하지 않을 경우에는 한 개의 WSC 노드에 많은 로드가 걸리는 현상이 발생한다. 따라서 다양한 로드 밸런싱 전략을 적용할 필요도 있다. 이 외에도, 먼 거리에 떨어져 있는 WSC들 간에도 성능 실험을 해 볼 필요가 있다.

본 연구의 구현에서 CAD 모델의 데이터를 가져오는 데 시간이 오래 걸리는 이유는, CAD 어댑터가 CAD 시스템과 상호 작용을 할 때, 분산 객체 기술과 유사한 방법을 이용하기 때문이다. 이는 비단 Pro/Engineer뿐만 아니라, 다른 상업용 CAD 시스템에도 해당되는 일이다. 작은 양의 데이터를 빈번하게 교환하는 것이 CAD 데이터를 가져오는 데 많은 시간이 걸리게 한다. 이를 해결하기 위해서는 궁극적으로 CAD 시스템 벤더들이 자신의 시스템에 WSC의 기능을 직접 구현하는 것이 필요하다. 또는 XML 형태의 데이터를 추출해 주는 기능이 지원될 필요가 있다. 현재 Pro/Engineer가 이러한 기능을 일부 지원하고 있지만, 아직 미약한 실정이다.

감사의 글

본 연구의 실험을 위하여 테스트 모델을 제공해 주신 (주)부품디비의 장광섭 과장님과 실험용 컴퓨터를 제공해 주신 한국과학기술원 기계공학과의 송일환 군에게 감사 드립니다.

참고문헌

1. Wang, Y. and Nnaji, B. O., “Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture”, *Journal of Computing and Information Science in Engineering*, Vol. 6, No. 2, pp. 127-138, Jun. 2006.
2. Kemmerer, S. J., *STEP: The Grand Experience*, National Institute of Standards and Technology Special Publication 939, U. S. Government Printing Office, Jul. 1999.
3. He, F. and Han, S., “A Method and Tool for Human-Human Interaction and Instant Collaboration in CSCW-based CAD”, *Computers in Industry*, Vol. 57, No. 8, pp. 740-751, Dec. 2006.
4. Siegel, J., *CORBA: Fundamentals and Programming*, John Wiley & Sons, Inc., Apr. 1996.
5. Eddon, G. and Eddon, H., *Inside Distributed COM*, Microsoft Press, Feb. 1998.
6. Bakis, N., Aouad, G. and Kagioglou, M., “Towards Distributed Product Data Sharing Environments — Progress So Far and Future Challenges”, *Automation in Construction*, Vol. 16, No. 5, pp. 586-595, Aug. 2007.
7. Newcomer, E., *Understanding Web Services: XML, WSDL, SOAP, and UDDI*, Addison-Wesley, Sep. 2002.
8. W3C, Web Services Description Working Group, http://www.w3.org/2002/ws/desc/.
9. W3C, XML Schema, http://www.w3.org/XML/Schema.
10. W3C, XML Protocol Working Group, http://www.w3.org/2000/xp/Group/.
11. 이재열, 윤장혁, 이순재, 김현, 김광수, “프로세스 중심의 동적 엔지니어링 웹서비스 지원 방법에 대한 연구”, 한국 CAD/CAM학회 논문집, 제9권, 제4호, pp. 361-372, 2004년 12월.
12. Erl, T., *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall, Aug. 2005.
13. 김병철, 한순흥, “웹 서비스를 이용한 CAD 모델 정보의 획득”, 한국 CAD/CAM학회 논문집, 제12권,

제4호, pp. 233-244, 2007년 8월.
14. *Computer Aided Design Services Specification V1.2*, OMG, Jan. 2005, http://www.omg.org/technology/documents/formal/cad.htm.
15. W3C, Resource Description Framework, http://www.w3.org/RDF/.
16. OASIS, UDDI.org, http://www.uddi.org/.

김 병 철

2001년 고려대학교 기계공학과 학사
2003년 한국과학기술원 기계공학과 석사
2008년 한국과학기술원 기계공학과 박사
2007년~2008년 한국기술교육대학교 기계정보공학부 대우교수
2008년~현재 한국기술교육대학교 기계정보공학부 겸임교수
2008년~현재 (주)부품디비 책임연구원

관심분야: Feature-based and Parametric Design, CAD Data Exchange, Intelligent CAD, Collaborative CAD

한 순 흥

KAIST 기계공학과 정교수이며, 2004년까지 International Journal of CAD/CAM(www.ijcc.org)의 편집장으로 활동함. 2003년까지 STEP센터(www.kstep.or.kr) 회장과 전자거래학회(www.calsec.or.kr) 회장 역임. 관심분야는 STEP, 가상현실 응용, 지능형 CAD이며, 연락처는 shhan@kaist.ac.kr, 홈페이지는 http://icad.kaist.ac.kr 임. 1990년 미국 미시건 대학에서 박사학위 수여