

CPFR 구현을 위한 다중 에이전트 시스템 구조 설계 * Designing a Multi-agent System Architecture for Implementing CPFR

최진성(연세대학교 컴퓨터·산업공학부 석사과정)
 한재준(연세대학교 컴퓨터·산업공학부 석사과정)
 김창욱(연세대학교 컴퓨터·산업공학부 교수)
 kimco@yonsei.ac.kr

Abstract

현재 인터넷 기술의 발달은 공급망상의 참여 기업들이 서로 독립적으로 생산관리를 실행하는 기존의 생산관리 방식에 많은 변화를 가져오고 있다. 대표적인 예가 참여 기업간의 정보 공유를 기반으로 하는 협업적 생산관리 방식인 CPFR (Collaborative Planning, Forecasting and Replenishment)이다. 이는 공급망상의 참여자들이 예외적 생산관리 상황이 발생하였을 때 이를 협업적으로 해결할 수 있는 생산관리 프레임워크를 의미한다. 본 논문에서는 CPFR 방식을 효과적으로 지원하며 이를 현실에 적용하기 위해서 데이터 뷰 (Data View) 에이전트, 협업 (Collaboration) 에이전트 및 비즈니스 규칙 (Business Rule) 에이전트로 구성된 멀티 에이전트 시스템 구조를 제시한다. 또한 이를 분산 컴퓨팅 기술인 EJB (Enterprise Java Bean)를 이용하여 웹기반 CPFR 어플리케이션을 구현한다.

1. Introduction

인터넷 기술의 발달은 공급망상의 기업들부터 소비시장에 이르기까지 전반적으로 많은 변화를 가져오고 있다. 기존 생산 정보 시스템은 기업 정보의 보안이라는 이유로 최소한의 정보 공유로 이루어져 있었고, 실제적인 생산관리는 독자적인 시스템에 의존하여 수행되었다. 하지만 기존 시스템으로는 현재 인터넷망을 통해 빠르게 전달되는 소비 시장 정보의 흐름을 더 이상 수용하기 힘들게 되었다. 따라서 공급라인상의 참여기업들은 전반적이고 실제적으로 협업할 수 있는 시스템을 필요로 하게 되었다. 이러한 요구로 연구된 많은 방법 중 VICS(Voluntary Inter-industry Commerce Standards)가 제시한 CPFR(Collaborative

Planning, Forecasting and Replenishment)[13]은 공급망상의 참여기업들이 효율적인 정보 공유와 예외적인 생산관리 상황을 협업을 통하여 해결할 수 있는 프레임워크이다. 현재 이 프레임워크는 많은 기업체들이 관심을 보이고 학계에서도 시스템적으로 구현하기 위해 연구를 진행하고 있다. 본 논문에서는 협업 시스템의 자동화를 지원하기 위하여 멀티에이전트 시스템을 응용한 CPFR 아키텍처를 설계하였다. 또한 이를 협업중의 보안성, 여러산업에서의 유연성, 공급망에서의 확장성, 다양한 시스템에서의 호환성 등의 장점을 살릴 수 있는 분산 컴퓨팅 기술인 EJB를 이용하여 구현하고자 한다.

2. CPFR (Collaborative Planning, Forecasting and Replenishment)

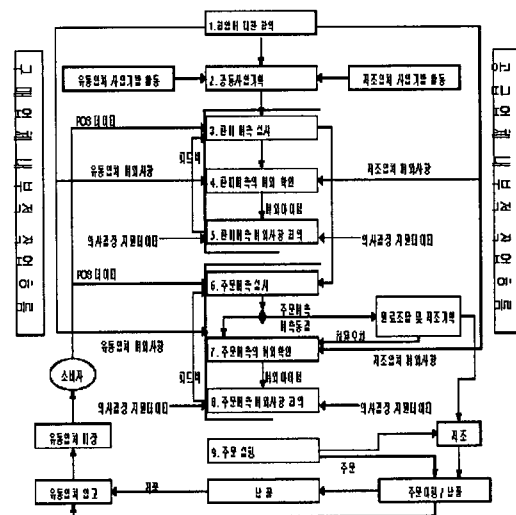


그림. 1 기본적인 CPFR 모델

CPFR 모델[그림. 1]은 기본적으로 9단계의 프로세스들로 이루어지며, 각 프로세스들은 거

* “이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음.”
 (KRF-2002-003-D00448)

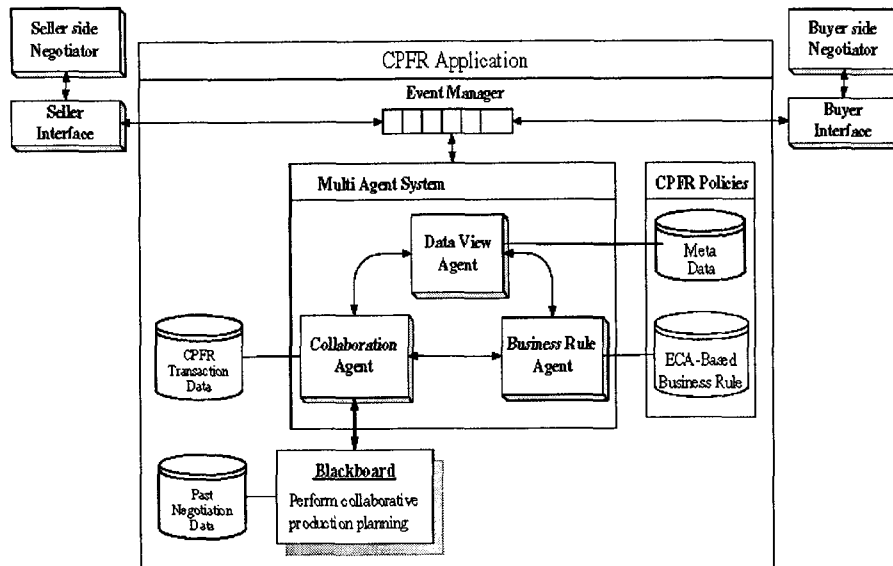


그림. 2 CPFR 구현을 위한 MAS 모델

래업체들간의 정보공유와 협상에 의하여 진행된다. CPFR은 기존의 비효율적이고 불평등한 관리방식을 타파하고 정보공유를 기반으로 하는 협업시스템을 이용하여 양사가 협의를 통하여 서로 만족할 수 있는 win-win 개념을 제공하고 있다.

- 1단계: 협업에 대한 합의서 작성
 - 두 회사는 각사의 사업 목표를 가지고 협업 합의서를 구체화 한다.
- 2단계: 공동 사업 계획
 - 두 회사는 사업 목표에 따른 협업의 전략이나 기술등의 범위를 결정한다.
- 3단계: 판매 예측 작성
 - 양사는 기본 공유 데이터를 가지고 판매 예측을 작성한다.
- 4단계: 판매 예측 예외 확인
 - 변경된 판매예측 데이터가 발생하면 작성된 판매 예측과 비교하며 예외사항이 발생하는지 분석한다.
- 5단계: 판매예측 예외사항 협의
 - 발생한 판매예측 예외사항들에 대한 각사의 데이터들을 가지고 예외사항을 해결한다.
- 6단계: 주문 예측 작성
 - 판매예측 데이터와 양사의 기본공유 데이터를 가지고 주문 예측을 작성한다.
- 7단계: 주문 예측 예외 확인
 - 변경된 주문예측 데이터가 발생하면 작성된 주문 예측과 비교하며 예외사항이 발생하는지 분석한다.
- 8단계: 판매예측 예외사항 협의
 - 발생한 주문예측 예외사항들에 대한 각사의 데이터들을 가지고 예외사항을 해결한다.
- 9단계: 주문 실행
 - 주문예측에 따른 주문을 시행하고 주문정보는 각사의 생산 및 판매 시스템으로 전송된다.

3. MAS (Multi-Agent System) architecture for implementing CPFR

멀티 에이전트 시스템은 에이전트의 자율성과 지능적인 요소를 가지고 특정 목적에 대하여 사용자가 원하는 작업을 자동적으로 해결해주는 시스템을 말한다. 일반적으로 사용자의 복잡한 요구는 하나의 에이전트에서 처리하기 어려운 경우가 많기 때문에 여러 에이전트간 정보의 공유를 통해서 문제를 해결하려는 협업적 멀티 에이전트 시스템에 대한 연구가 많이 진행되었다. 또한 SCM분야에 이를 적용하기 위한 자동화 시스템 연구가 활발히 진행중이다 [6]. 본 논문에서는 복잡한 CPFR의 개념을 포괄하고 이를 구현 할 수 있는 멀티 에이전트 시스템 아키텍처 모델[그림. 2]을 제시하고자 한다.

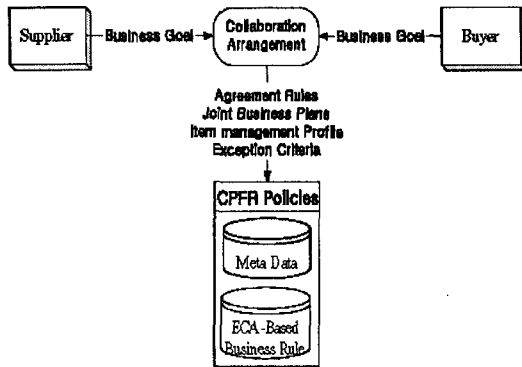


그림. 3 CPFR 정책 수립 모델

CPFR 9단계 프로세스중 1, 2단계에서 합의된 내용들은 [그림. 3]의 단계를 거쳐 CPFR 정책으로 저장되게 되며, 여러 개의 모듈들로 구성된 멀티에이전트 시스템[그림. 2]의 각 부분은 다음과 같은 역할을 한다.

- **Event manager:** 공급업체와 구매업체에서 요청하는 정보조회, 판매예측(CPFR 3단계 프로세스), 주문예측(CPFR 6단계 프로세스), 예측갱신, 주문(CPFR 9단계 프로세스) 등의 이벤트를 관리한다. 각 이벤트를 따라 해당 에이전트에 전달하여 준다.
- **Data view agent:** 참여업체는 이 에이전트를 통하여 다른 참여업체들의 정보를 조회할 수 있다. 이 에이전트는 Meta database를 참조하여 허용된 공유 범위 안에서 상대방의 정보를 검색할 수 있게 해준다.
- **Meta data base:** 업체들이 맺은 정보의 공유 범위를 기록하고 있는 데이터베이스이다. 이 범위에 어긋나는 검색에 대해서는 Data view agent를 통하여 정보열람 승인 거부 결과값을 전달한다.
- **Business rule agent:** 사용자가 발생시킨 이벤트와 ECA-based business rule database[12]에 입력되어 있는 정보와 비교하여 협의되지 않은 예외사항이 발생하는지 여부를 조사한다(CPFR 3 또는 7단계 프로세스). Business rule에 어긋나는 협의되지 않은 이벤트가 발생하면 Collaboration agent로 하여금 Black Board를 생성하고, 협의된 예외사항이 발생하면 Collaboration agent로 하여금 이벤트가 CPFR 정책에 의하여 처리되었음을 양 업체측에 전달하게 한다. 판매 예측,

주문 예측 데이터가 필요한 예측값 갱신, 주문 실행등의 이벤트가 발생하면 CPFR transaction DB에 저장되어 있는 바로 전 예측값을 참조한다.

- **ECA-based business rule database:** 협의된 예외사항의 범위, 아이템의 배송 관리 명세등을 Business rule로 저장하고 있으며, Business rule agent가 조회하는 이벤트에 따라 협의되지 않은 예외사항이나 협의된 예외사항등에 따른 결과값을 전달한다.
- **Collaboration agent:** 양사의 협업내용을 CPFR transaction database에 기록하고, 협상을 위한 Blackboard[11]를 제어하는 에이전트이다. Business rule agent로부터 협의되지 않은 예외사항이 넘어오면 Blackboard를 생성하고, CPFR transaction database로부터 협의되지 않은 예외사항과 유사한 최근 협상 결과들을 자동으로 검색하여 Blackboard상에 제공한다. 또한 협의된 예외사항의 처리, Blackboard의 생성 및 소멸, 협상에서 최종 합의된 결과등을 협상업체들에게 전송한다.
- **CPFR transaction database:** Collaboration agent에서 처리한 협상 결과 데이터를 저장하고 있으며, 이 데이터베이스는 CPFR 시스템을 위한 Knowledge source가 된다. 인증 받은 사용자들은 협상을 위해 Collaboration agent를 통하여 데이터베이스를 조회할 수 있다.
- **Blackboard:** 일종의 협상데이브로 Collaboration agent에 의해 생성되고 소멸되며, 협상 권한을 가진 사용자들이 예외 사항들에 대한 그들의 의견과 해결 방안등을 보드상에 기록하며 대화하는 방식으로 협상을 진행하게 된다.
- **Past negotiation database:** 블랙보드상에서 진행된 모든 내용들에 대한 기록을 저장한다. Blackboard의 Knowledge source가 된다. 인증 받은 유저들은 협상을 위해 Collaboration agent를 통하여 데이터베이스를 조회할 수 있다.
- **Negotiator:** 협의되지 않은 예외사항에 대한 해결방안을 Blackboard상에서 협상하는 양 업체의 협상자이다.

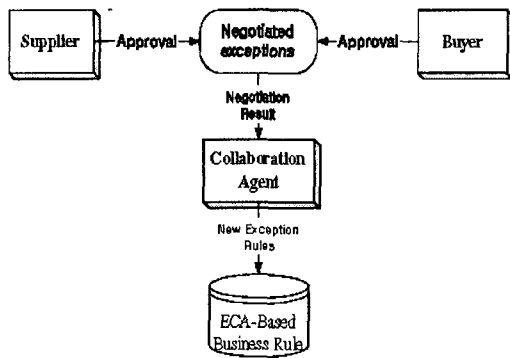


그림. 4 Business Rule 갱신 모델

Blackboard상에서 협상한 새로운 예외사항들에 대한 합의 결과들은 [그림. 4]의 단계를 거쳐 선택적으로 Business rule로 갱신되거나 추가 된다.

4. Case study

본 연구에서 제시한 CPFRR 멀티 에이전트 시스템을 구현단계로 접근하기 위해 Use case 들을 작성하였다. 그중 한 예로 주문 실행 Use Case로 표현한 것이 [그림. 5]이다. Use Case 명세는 시스템의 목적, 역할과 요구사항을 획득하는데 매우 유용하며 시스템에 대한 분석, 디자인, 구현 전반에 걸쳐 중요한 역할을 담당한다. 따라서 CPFRR의 복잡한 프로세스는 Use Case들을 통해 쉽게 이해될 수 있으며, 이를 바탕으로 구조화 하여 소프트웨어 컴포넌트(Component)들로 설계할 수 있다[8]. 또한 지금까지 피상적으로 다룬 내용들을 시스템과 사용자간의 Contract을 통해 서로의 역할을 기술함으로써 전체 시스템의 모습을 파악할 수 있다[9].

<p>Use Case 이름: 주문 프로세스 Primary Actor: 구매 업체 목적:</p> <ul style="list-style-type: none"> - 구매업체: 필요한 수량을 공급업체에게 요청한다. 협의에 벗어난 긴급 주문시 블랙보드를 통해 공급 업체와 상의 할 수 있다. - 공급업체: 구매업체의 주문을 바탕으로 생산을 통제할 수 있으며, 구매업체의 상황을 파악하여 재고를 최적화 할 수 있다. <p>선행조건: 구매 업체가 시스템에 접속하여 인증 받고 주문 실행 권한을</p>

<p>받는다. 후행조건: 주문에 대한 기록이 저장된다.</p> <p>주요 성공 시나리오</p> <ol style="list-style-type: none"> 1. 구매 업체가 주문 생성을 요청한다. 2. 시스템은 주문을 위한 입력창을 제공한다. 3. 구매업체는 주문 정보를 시스템에 전송한다. 4. 구매업체가 전송한 주문정보를 Event manager가 받아들인다. 5. Event manager는 주문정보를 Business rule agent에게 전달한다. 6. Business rule agent는 정보를 ECA-based-rule 데이터베이스에 예외사항 유무를 조회한다 7. Business rule agent를 거쳐 이상이 없는 주문정보를 Collaboration Agent로 전달한다. 8. Collaboration agent는 주문정보를 CPFRR 처리 데이터베이스에 저장한다. 9. 구매업체와 공급업체측에 주문에 대한 결과를 전달한다. 10. 결과를 확인한다.

확장

- 6-7a Business Rule에 벗어나는 정보가 입력됐다.
1. 해당 정보를 Collaboration agent로 전달한다.
 2. Collaboration agent는 전달 받은 항목에 대해 CPFRR 처리 데이터베이스에서 유사한 항목의 최근 처리자료를 검색한다.
 3. Collaboration agent는 블랙보드를 생성한다.
 4. Collaboration agent는 검색된 데이터와 협의되지 않은 예외 항목, 주문정보를 블랙보드에 전달한다.
 5. 구매업체와 공급업체측에 알린다.
 6. Event manager는 ID에 부여된 협상 권한 인터페이스를 가져다 준다.
 7. 구매업체와 공급업체는 Data view agent를 통하여 CPFRR 처리 데이터베이스와 Blackboard 협상 데이터베이스를 조회하고 Blackboard 상에서 협상을 진행한다.
 8. 협상이 완료되면 Blackboard는 닫히고 협상 자료들은 Blackboard 협상 데이터베이스에 저장된다.

그림. 5 주문실행의 Use case 명세서

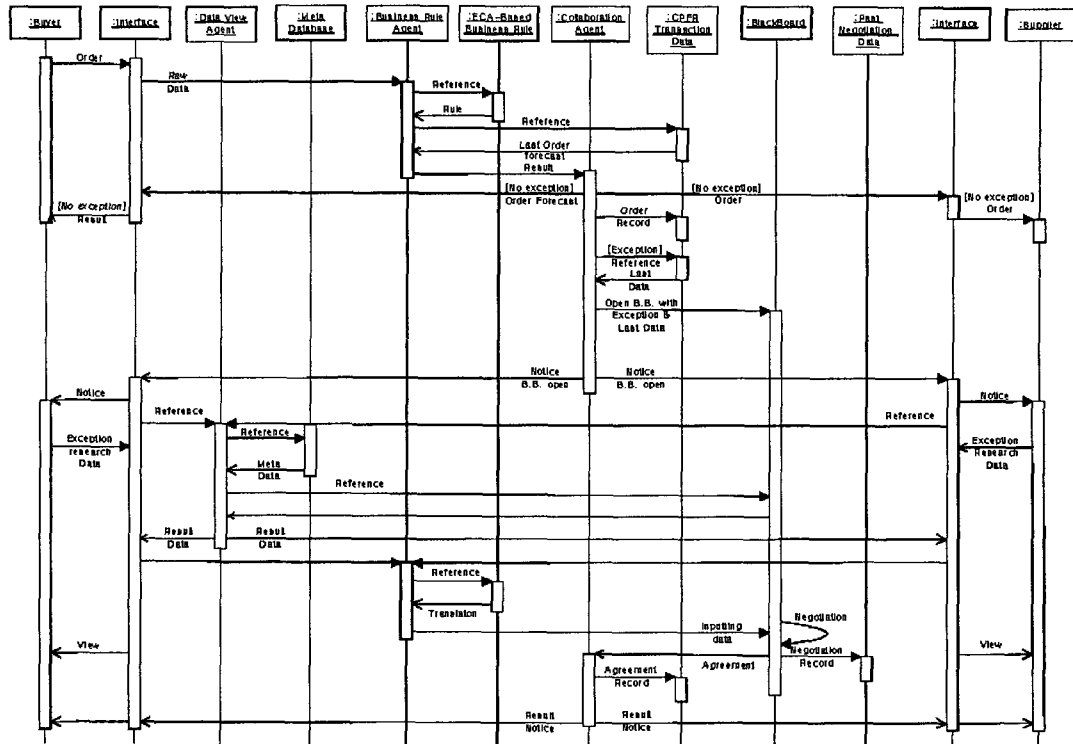


그림 6. 주문 시행의 Sequence diagram

[그림. 5]은 주문 실행을 명세한 것으로 “주요 성공 시나리오”는 구매 업체의 주문 실행 이벤트가 Business rule에 어긋나지 않았을 때 발생하는 흐름을 기술한 것이다. “확장”은 Business rule agent가 입력된 정보를 ECA-based rule DB를 조회한 후 예외사항들이 발생했을 때 실행되는 대안을 나타낸다. [그림. 5]의 Use Case 명세를 시간의 경과에 따라 메시지의 흐름을 파악하기 위해 시퀀스 다이어그램으로 나타낸 것이 [그림. 6]이다.

5. Application model with EJB

본 논문에서는 위에 제시한 MAS 아키텍처를 EJB(Enterprise Java Bean)를 이용하여 구현하였다. EJB를 이용하면 3-Tier 혹은 n-Tier를 지원하는 엔터프라이즈급의 분산 객체 시스템을 구현할 수 있다. CPFR 가이드 라인은 공유(Coexistence), 중앙서버(Centralized Server), 분산서버(Distributed Servers) 등의 모델을 제시하고 있다[1]. EJB는 세 가지 모델중 분산 서버형태에 적합한 방식이다. EJB로 구현한 본 논문의 MAS 시스템은 다음과 같은 특징을 가지게 된다.

- CPFR정책을 따르는 유사한 도메인에는 거의 변경없이 기존 컴포넌트를 적용하여 원하는 서비스를 제공할 수 있다.
- 연관이 적은 도메인일지라도 본 연구에서 제안한 구조를 따른다면 다시 시스템을 개발하는 것이 아니라 필요한 컴포넌트를 새로 추가하거나 컴포넌트의 일부분을 수정함으로써, 핵심적인 뼈대는 유지하며 개발 기간, 자금, 인력을 절감할 수 있다.
- 웹을 지원하므로 시스템 구축시 추가 비용이 적으며 ERP, CORBA 등의 시스템과 연결하여 사용할 수 있다.
- SUN(Sun Microsystems, Inc.)에서 제공하는 풍부한 API(Application Programming Interface)를 사용할 수 있으며 보안, 트랜잭션, 객체간 통신등의 기술이 내부적으로 지원되므로 업무규칙을 집중적으로 구현할 수 있다.
- EJB의 n-Tier 구조는 계층마다 상호 연관성이 줄어 유지 보수비용이 줄어들고 계층간 통신이 가능하며 동일한 코드를 재사용함으로써 업무 표준화가 이루어진다.

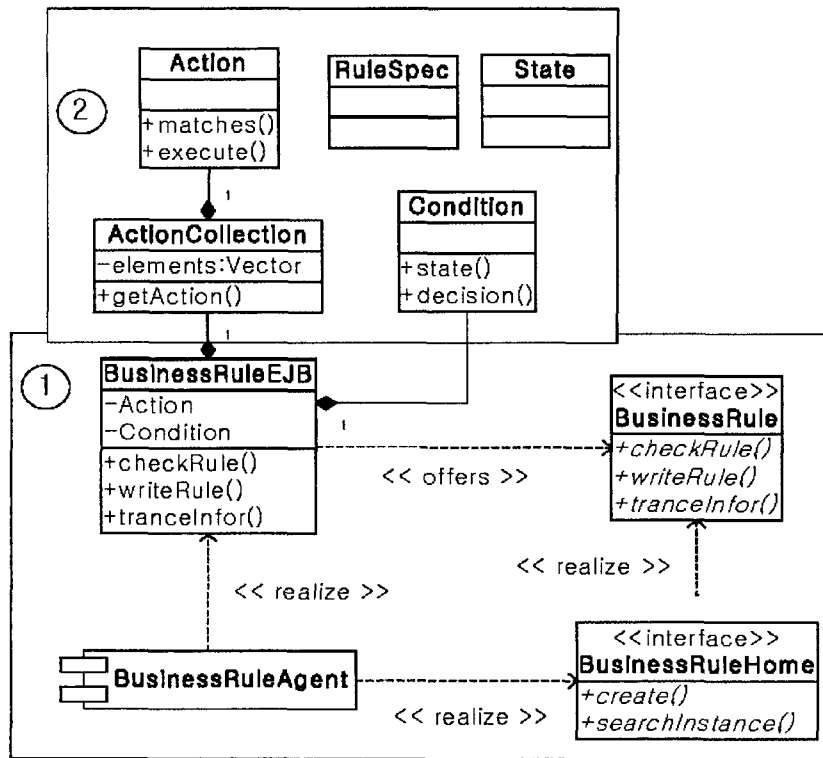


그림. 7 비즈니스 규칙 에이전트 내부 모습

이러한 시스템의 높은 유연성 때문에 참여업체들은 쉽게 비용적인 부담이나 시스템 개발 기간에 따르는 부담 없이 CPFR MAS 시스템으로 전환할 수 있게 된다.

[그림. 7]은 MAS 구조에서 여러 컴포넌트들 중 하나인 Business rule agent의 내부 구조를 보여준다. 위의 컴포넌트는 열거, 대행동의 디자인 패턴[4]을 충족시키도록 설계하여 재사용이 가능하도록 하였다. EJB명세에 적합한 컴포넌트들은 기본적으로 ①처럼 두개의 인터페이스와 하나의 구현 클래스가 필요하다[3].

BusinessRuleHome 인터페이스는 BusinessRuleEJB 클래스의 객체 생성과 참조를 담당한다. ②의 클래스들을 포함하고 있는 BusinessRuleEJB 클래스는 내부적으로 실행되며 요청한 일을 수행할 수 있는 메소드를 가지고 있다.

②는 State, Action, Condition, ActionCollection, RuleSpec. 클래스를 가지고 있다. RuleSpec 클래스는 ECA-Base Business rule 데이터베이스의 내용을 처리하며, State 클래스는 사용자 정보, 제품 정보, 예측 정보 등의 데이터를 포함한다. Condition 클래스는 사용자의 이벤트를 판단하고 Action 클래스는 비즈니스 규칙에 따라 수행될 작업을 나타낸다.

ActionCollection은 Action 클래스를 담고 있으며 Condition과 Action을 대응시키는 역할을 한다.

Business rule agent는 사용자가 Business rule 인터페이스의 메소드를 호출함으로써 작업이 시작된다. 메소드가 호출되면 BusinessRuleEJB 클래스의 실제 구현된 메소드 안에서 ActionCollection 클래스와 Condition 클래스가 호출된다. Condition 클래스는 state(), decision() 메소드를 가지며 RuleSpec, State 클래스를 참조하여 어떤 일을 수행해야 하는지 판단한다. ActionCollection 클래스는 Condition 클래스의 판단 결과를 가지고 해당 하는 Action을 실행한다.

[그림. 8]은 본 논문의 MAS 구조를 컴포넌트를 이용하여 구현한 컴포넌트 다이어그램이다. Web/Client Component는 사용자가 시스템에 접속했을 때 보여지는 부분이며 이를 통해 작업이 시작된다. Login 컴포넌트는 사용자에 따른 권한을 할당하는 컴포넌트이다.

①의 SaleForecasting(판매예측), OrderForecasting(주문예측) Forecasting Update(예측갱신), Order(주문실행) 컴포넌트는 MAS에 이벤트를 전달하는 화면 컴포넌트이다.

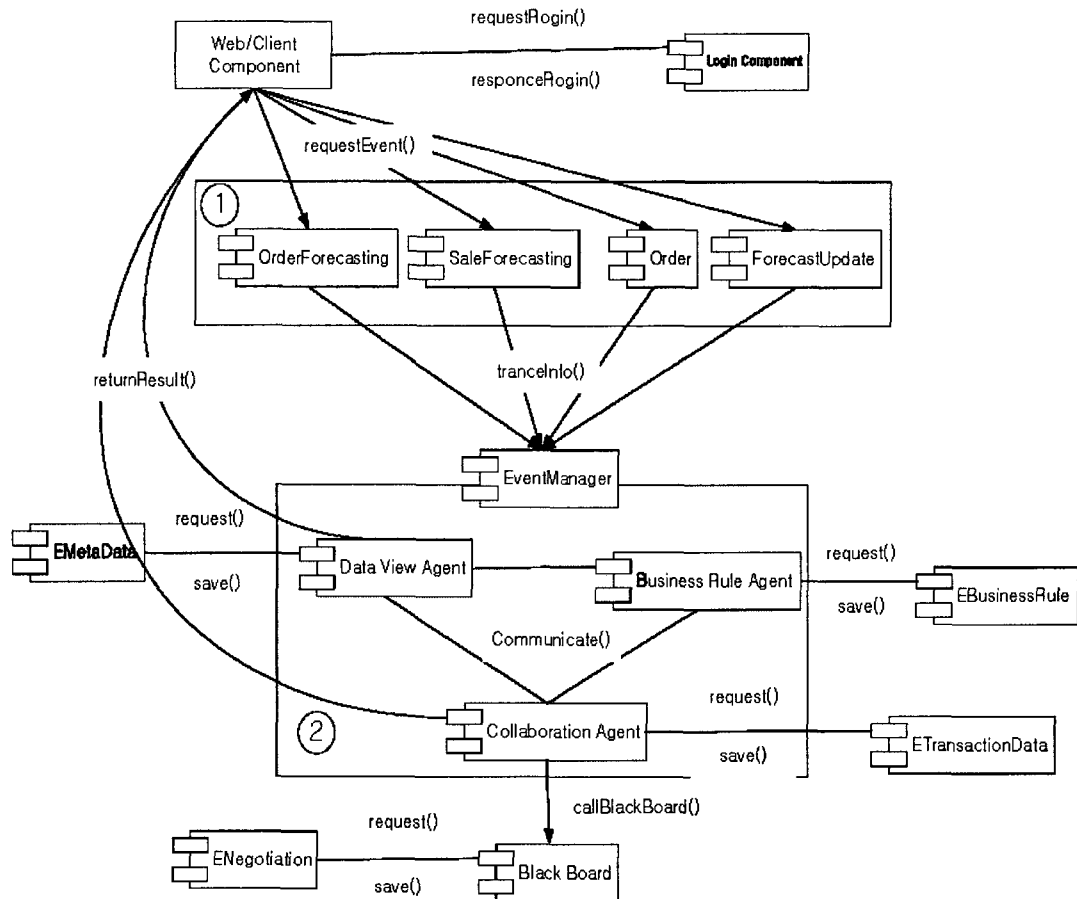


그림. 8 MAS를 위한 UML 컴포넌트 다이어그램

②는 Event manager, Data view agent, Business rule agent, Collaboration agent 컴포넌트로 구성되어 있다. Event manager 컴포넌트는 ①에서 전달된 이벤트를 정의된 진행순서에 따라 각 에이전트 컴포넌트로 전달한다.

Data view agent는 다른 참여업체들의 정보를 조회할 수 있게 해주며 Meta database를 참조하는 EmetaData컴포넌트를 통해 허용된 공유 범위 안에서 상대방의 정보를 검색 할 수 있으며 다른 컴포넌트에게 이 정보를 전달할 수 있다.

Business rule agent는 ECA-based business rule database를 참조하는 EtransactionData를 이용하여 적합한 업무 규칙을 선택하여 Collaboration agent에게 전달한다.

Collaboration agent 컴포넌트는 양사의 협업내용을 CPFR transaction 데이터베이스를 관리하는 ETransactionData 컴포넌트를 통하

여 기록하고, 협상 결과를 Web/Client 컴포넌트에게 전달한다.

6. Conclusion

본 연구에서는 기업간 협업 시스템의 대표적 사례인 CPFR을 Multi-agent system을 적용하여 아키텍처를 설계하였고, EJB를 이용한 웹 기반 어플리케이션으로 구현하였다.

차후 연구과제로는 본 논문의 CPFR MAS을 다자간 협상이 가능한 CPFR MAS 아키텍처로 확장하는 것이다. 또한 이에 따라 기업들의 기존 시스템이 가지고 있는 다양한 형태의 자료들을 실용적으로 공유하기 위한 XML 기반 ontology transfer system[15]에 대한 연구가 필요하다. 이밖에 개발된 소프트웨어를 많이 수정 하지 않고도 기존의 다양한 시스템에서 사용할 수 있는 재사용성이나 호환성에 대한 연구도 함께 진행 되어야 한다.

7. References

- [1] Johnson, M., "Collaboration data modeling: CPFR implementation guidelines", *White Paper*, Available at <http://www.cpfr.org>.
- [2] RosettaNet., Available at <http://www.rosettanet.org>.
- [3] Enterprise Blue Prints, Available at <http://java.sun.com/blueprints>.
- [4] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.
- [5] Chappell, D.A. and Chopra, V. (2001), *Professional ebXML Foundations*, Wrox Press.
- [6] Yonghui, F.R.P., Robert, D. S. and Jingru W., (2000), "Multi-agent enabled modeling and simulation towards collaborative inventory management supply chains", *Proc. of the 2000 Winter Simulation Conference*
- [7] Jose, M.V., Paul, A.B., and Michael, N.H. (2001), "Inside an agent", *IEEE Internet computing*, **5**(1), 82-86.
- [8] John, C. and John D. (2000), *UML Components: A Simple Process for Specifying Component-based Software*, Addison-Wesley.
- [9] Use Case Format, Available at <http://www.usecases.org>
- [10] Craig L. (2002), *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, Prentice Hall.
- [11] Corkill, D. D. (1991), "Blackboard systems", *AI Expert*, **6** (9), 40-47.
- [12] Goh, A., Koh, Y.-K., and Domazet, D.S.(2001), "ECA-based support for workflows", *Artificial intelligence in engineering*, **15**(1), 37-46.
- [13] CPFR Guideline Ver2.0, Available at <http://www.cpfr.org>
- [14] ebXML., Available at <http://www.ebxml.org>
- [15] Riza, C. E., and Oguz, D. (2002), "A multi-agent system infrastructure for software component market-place: An ontological perspective", *SIGMOD record*, **31**(1), 55-60