

# 에이전트 기반 협동설계 시스템에서의 사례기반 의사 충돌 해결

이경호\* · 이규열\*\*

## Case-Based Conflict Resolution in Agent-Based Collaborative Design System

Kyung-Ho Lee\* · Kyu-Yeul Lee\*\*

### 요 약

선박설계는 그 과정이 매우 복잡하고 많은 양의 데이터를 다루고 있는 작업으로서 그 환경이 분산화, 이질화 됨에 따라 최근들어 CSCW(Computer Supported Collaborative Work)의 요구가 증대되고 있다. 본 논문에서는 분산된 선박설계 환경에서의 협동작업을 지원하기 위한 에이전트 기반 선박설계 시스템을 개발하였다. 특히 여기서는 설계 에이전트간의 정보교환 및 지식공유를 통한 선박설계의 의사결정 과정에서 발생하는 의사충돌 문제를 해결하기 위하여 사례기반 추론 기법을 이용하여 이의 해결을 시도하였다. 설계 에이전트, 이들을 중재하는 퍼실리테이터, 충돌 처리기, 그리고 사례기반 시스템의 유기적인 도움을 받아 설계자는 설계과정에서 발생하는 설계 시스템간의 의사충돌 문제에 대한 의사결정을 과거의 유사한 문제해결 사례로부터 효과적으로 처리할 수 있다.

Key Words: Conflict Resolution, Agent-Based System, Case-Based Reasoning,  
Concurrent Engineering, Collaborative Work, Ship Design

\* 한국해양연구소(Korea Research Institute of Ships & Ocean Engineering)

\*\* 서울대학교(Seoul National University)

## 1. 서 론

분산된 환경에서의 동시공학적 설계 방법론이 날로 그 중요성이 부각되면서 어떻게 하면 분산된 이기종, 상이한 소프트웨어 환경간의 설계정보를 서로 교환하고 공유할 수 있을 것인가 하는 문제를 해결하기 위한 많은 시도가 이루어지고 있다. 최근 급속히 부각되고 있는 에이전트(Agent) 기반 시스템의 채용은 이러한 문제를 효율적으로 해결할 수 있는 방안의 하나가 될 수 있다. 이러한 에이전트 기반의 동시공학적 설계 시스템을 구현하기 위해서는 시스템간의 원활한 정보교환(Communication)과 더불어 에이전트간의 정보교환 과정에서 발생하는 의사충돌 해결(Conflict Resolution) 및 의사결정 중의 협의(Negotiation) 등의 방법론이 우선적으로 해결해야 할 과제라고 할 수 있다.

본 논문에서는 동시공학적 선박설계 에이전트 시스템 환경에서 이들의 정보교환 과정에서 발생하는 설계 에이전트간의 의사 충돌 문제를 해결하기 위하여 과거의 의사 충돌 해결 사례를 바탕으로 현재의 충돌 문제를 해결하고자 하는 사례기반 충돌 해결 전략을 수립하였다. 이를 위하여 먼저 선박 초기설계 과정에서 발생할 수 있는 설계 시스템간의 의사 충돌 문제를 추출하여 이를 정리하였고, 이것을 바탕으로 선박 초기설계 의사 충돌 해결을 위한 사례베이스를 구축하였다. 또한 에이전트간의 지식 공유 및 정보 교환을 통한 협동설계를 수행할 때 일어나는 의사 충돌 문제를 효과적으로 처리하기 위하여 충돌 해결 처리기(Conflict Resolution Handler)를 개발하여 이를 에이전트간의 정보를 중재하는 퍼실리테이터내에 두어 구축된 사례베이스의 추론을 통하여 충돌 문제에 대한 해를 제시하도록

하였다. 이를 바탕으로 선박 초기설계, 특히 기장설계, 기본계획, 선형설계, 구조설계 등에서 일어나는 의사 충돌 문제를 대상으로 각 에이전트, 퍼실리테이터, 충돌 해결 처리기 및 사례기반 시스템에서의 문제 해결을 위한 정보 교환 과정과 시스템 구현 사례를 통하여 사례기반 추론 기법에 의한 설계 에이전트간의 충돌 해결 전략에 대한 유효성을 검증하였다.

## 2. 에이전트 기반 설계 시스템

### 2.1 에이전트의 정의

에이전트는 그 종류와 특성에 따라 여러 가지로 정의할 수 있지만 가장 쉽고 간단하게 “외부 프로그램과 정보교환을 위해 정해진 통신 규약에 따라서 통신을 수행할 수 있는 독립적인 컴퓨터 프로그램”이라고 정의할 수 있다. 다시 말해서 에이전트는 지식베이스와 추론기능을 가지며 사용자, 자원(Resource) 또는 다른 에이전트와의 정보교환과 통신을 통해 문제해결을 도모하는 소프트웨어로서 이들이 가지는 고유의 통신 언어와 규약을 통해 분산된 이질의 시스템들간의 지식공유를 통한 CSCW(Computer Supported Collaborative Work)를 구현하기 위한 것이다.

에이전트 기반 소프트웨어 공학(Agent-Based Software Engineering)은 이런 이질적인 환경 하에서 보다 쉽게 협동적인 프로그램을 개발하기 위해 연구되었으며 그 핵심이 에이전트이다. 에이전트 기반 소프트웨어 공학의 관점에서 에이전트는 에이전트 통신 언어(Agent Communication Language : ACL)로 다른 에이전트와 통신하는 소프트웨어 요소이다. 즉, 에이전트는 상황에 따라

적당한 ACL을 주고받으며 다른 에이전트와 정보 및 서비스를 교환하고 결국 협동적으로 일을 수행한다. 이때 ACL의 표현력은 에이전트가 수행할 수 있는 일의 범위를 결정짓는 중요한 요소로, 여기서는 미국 Defense Advanced Research Project Agency(DARPA) 및 여러 기관이 후원하는 Knowledge Sharing Effort(KSE) 과제에서 개발된 Knowledge Query And Manipulation Language(KQML) 과 Knowledge Interchange Format(KIF)을 각각 외부 언어와 내부 언어로 사용한다[Finin & Weber et al., 1994; Genesereth & Fikes et al., 1992; Patil et al., 1992; Labrou & Finin, 1997; Finin & Labrou et al., 1995; Finin & Fritzon et al., 1995].

ACL을 통해 주고받는 정보와 서비스 및 이들 정보를 다른 형태로 재 가공하거나 추론 등을 통해서 새로운 정보를 도출해 내는 에이전트의 능력에 따라 에이전트 기반 소프트웨어 공학은 이질적인 환경을 효과적으로 통합하는 시스템 통합 기법으로서의 가능성을 보여준다.

## 2.2 에이전트화(Agentification)를 통한 정보 교환 및 공유

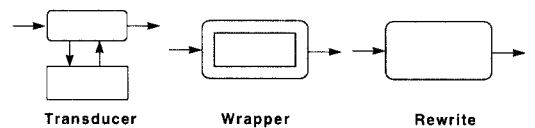
서로 이질적인 분산된 설계 환경 하에서 설계 정보의 교환, 설계지식의 공유를 통한 협동설계 시스템을 구현하기 위해서는 먼저 각각의 프로그램이 정보를 교환하고 서로의 지식을 공유할 수 있도록 이들의 에이전트화 작업이 선행되어야 한다. 다음으로 에이전트간의 정보의 교환과 지식의 공유를 위해서는 고유의 언어가 필요하다. ACL은 에이전트간의 통신에 쓰이는 언어이다. 이 ACL의 표준이 존재한다면 에이전트의 구현과 에이전트의 메시지 인터페이스를 분리시킬 수 있으므로 효과적으로 에이전트 시스템을 구

성할 수 있다. 하지만 그렇다고 해서 ACL을 위해서 아무 언어나 사용할 수는 없다. 그것은 에이전트가 다양한 상황에서 다양한 정보를 교환하며 여러 가지 일을 수행하므로 ACL의 표현력에 따라서 얼마나 다양한 종류의 에이전트가 만들어 질 수 있는가를 가늠하는 중요한 요소가 되기 때문이다.

### 2.2.1 Agentification

일반적으로 기존의 프로그램을 에이전트화 하는 방법에는 [그림 1]과 같이 Transducer, Wrapper의 이용 및 프로그램 재작성 등이 있다[Genesereth & Ketchpel, 1994]. 하지만 어느 방법을 사용하든 에이전트는 KQML 수행자(Performative)와 Content인 KIF를 처리하여야 하며, 이를 체계적이고 효과적으로 지원하기 위해서는 에이전트 기본 아키텍처가 확립되어 있는 것이 바람직하다.

앞에서 언급하였듯이 ACL을 이용하여 에이전트는 데이터, 사실, 지식 등의 정보를 전달할 수 있다. 실제로 이러한 ACL로 표현된 정보가 에이전트간에 전달되기 위해서는 에이전트 시스템은 다음과 같은 기능이 있어야 한다.



[그림 1] 에이전트화를 위한 3가지 접근법

- 1) 에이전트간의 메시지를 전달하기 위하여 통신을 할 수 있는 기본적인 기능
- 2) Contents 내용을 해석하는 기능 (KIF 인터프리터)
- 3) KQML의 지시 사항(Performative)을 수행하는 기능

### 2.2.2 에이전트 통신언어 (ACL)

현재 DARPA의 KSE 프로젝트에서 시스템간 (특히 Knowledge Based System) 지식의 공유를 위한 연구를 진행하고 있다. 이는 새로운 시스템 개발 시 처음부터 시작하는 것이 아니라 지식 표현 언어가 상이할지라도 기존 시스템의 지식을 이용하여 손쉽게 지식베이스(KB)를 구축하거나 여러 시스템간의 지식 공유로 보다 큰 일을 수행하기 위한 목적에서 연구가 시작되었다 [Hyacinth, 1996]. 이러한 지식의 구축 및 공유를 위해서는 공통된 언어를 지식 표현에 사용하거나 적어도 서로 번역 가능한 언어를 사용하여야 하며 서로간의 통신이 가능하여야 한다. 따라서 KSE에서는 지식 표현을 위한 공통 언어 및 통신을 위한 언어를 연구 성과로 내놓았다. KSE의 Inter-lingua Group에서 KB의 내용을 표현하기 위한 공통의 언어로 KIF를 개발 중에 있고, External Interface Group에서 지식 기반 시스템(Knowledge Base System)간 혹은 지식 기반 시스템과 다른 프로그램간의 실시간 지식 교환을 위한 KQML을 개발하고 있다. 일반적으로 ACL은 크게 외부 언어로 KQML과 내부 언어로 KIF이 많이 사용되는 추세이다.

#### 1) 외부 통신 언어 (KQML)

KQML은 에이전트간의 지식 교환을 위한 메시지의 형식이자 상호간의 통신 규약이다[Patil & Files et al., 1992]. KQML이 지식 교환을 위한 메시지의 형식이라는 것은 KQML은 지식을 표현하는 것이 아니라 표현된 지식을 통신으로 전달하기 위한 형식이라는 것을 말한다. KQML은 실제 실어 나르는 지식의 내용과 무관하다. 따라서 내용의 표현 언어에 독립적이며 여러 가

지 언어가 쓰일 수 있다. KQML은 다만 이들 내용에 대한 태도(질문, 요구, 명령, 단언)를 나타낸다. 이 개념은 Speech Act 이론에서 나온 것으로 태도를 나타내는 KQML의 기본 요소를 수행자(Performative)라고 부른다. 수행자는 다른 여러 인자와 함께 완전한 KQML 문장을 이루며 또한 상호간의 통신 규약을 나타내는 역할을 한다.

선박 초기설계 에이전트 시스템에서 사용된 다음과 같은 간단한 예를 통해 KQML의 구성요소를 살펴보자[이상욱, 1998].

```
(ask-one          :sender facilitator
                  :receiver agent1
                  :reply-with id1
                  :language KIF
                  :ontology Particular_Dim
                  :content (length h30 ?!))
```

여기서 ask-one이 수행자로 :content가 질문임을 나타내고 암시적인 통신 규약에 따라 다음과 같은 형태의 답변이 오기를 기다린다.

```
(tell            :sender agent1
                 :receiver facilitator
                 :in-reply-to id1
                 :language KIF
                 :ontology Particular_Dim
                 :content (length h30 264))
```

이 대화의 내용은 h30번 배의 길이가 얼마냐고 물은 것에 대해 264m 라고 대답하는 에이전트간의 통신 예이다.

남아있는 다른 파라미터를 살펴보면 :sender는 메시지의 전송자를 나타내며, :receiver는 메시지의 수신자를 나타낸다. ask-one의 :reply-with와

<b>통신 계층</b>	<b>Performative</b> (ask-one, tell, ...) :language :ontology ...
<b>메시지 계층</b>	:sender :receiver :reply-with ...
<b>내용 계층</b>	:content ...

(그림 2) KQML 메시지의 구성

tell의 :in-reply-to는 서로 짝을 이루어 질문에 대응하는 정확한 tell 메시지를 찾을 수 있게 한다. 즉, ask-one을 보내면서 id1이라는 값으로 답변하기를 요청하고 tell은 :in-reply-to 값을 id1으로 설정함으로써 이것이 :reply-with id1으로 설정된 ask-one에 대한 답변임을 밝히는 것이다. :language는 :content를 표현하는데 쓰인 언어가 KIF임을 밝히며 Prolog와 같은 다른 언어를 사용할 수도 있다. 마지막으로 :ontology는 content에 쓰인 어휘의 정의를 담고 있는 것으로 여러 에이전트간에 쓰이는 어휘를 공유하기 위한 것이다. 여기서는 length라는 KIF의 Relation Constant가 Particular\_Dim이라는 ontology에 저장되어 있음을 뜻하는데 이 곳에 length를 포함한 선택의 주요치수와 관련된 용어를 정의해 놓고 여러 에이전트들이 그 뜻을 공유한다.

이상에서 보듯이 KQML은 [그림 2]와 같이 개념적으로 세 개의 계층(Layer)으로 나누어질 수 있는데, 내용 계층(Content Layer), 메시지 계층(Message Layer), 통신 계층(Communication Layer)이 그것이

다. 이 중 내용 계층은 교환하고자 하는 내용을 담은 층으로 KIF 뿐 아니라 여러 가지 다른 표현 언어(예로서 Prolog, Lisp 등)를 쓸 수 있다. 다만 본 논문에서는 KIF를 제한적으로 사용한다. 나머지, 통신 계층은 메시지의 발신자(Sender)와 수신자(Receiver)와 같은 통신 파라미터를 나타내고, 메시지 계층은 내용에 대한 태도(질문, 명령, 단언 등)를 나타내며 또한 이에 따라 상호간의 통신 규약을 결정하게 된다.

### 2) 내부 통신 언어 (KIF)

KIF는 LISP, Prolog언어와 유사하게 지식(knowledge)이나 규칙(rule) 등을 표현할 수 있는 First Order Predicate Calculus의 Prefix 타입의 언어이다[Genesereth & Fikes et al., 1992]. KIF는 상이한 지식 표현 언어를 사용하는 시스템간의 지식 교환을 위한 목적으로 개발되었으므로 풍부한 표현력을 가지고 있다. 결국 ACL의 해석을 위해서 가장 핵심이 되는 것은 내부 언어의 해석이며 이를 위하여 KIF 해석기를 필요로 한다. 본 논문에서는 KIF를 제한적으로 사용하고 있으며, 에이전트 고유의 지식 공유를 위해서 설계 지식을 서로가 이해할 수 있도록 하기 위한 KIF 해석기를 선택설계 전용으로 개발 중에 있다[최제민, 1997].

### 3) 온톨로지 (Ontology)

사람들간의 의미 있는 대화를 나누기 위해서는 적어도 같은 언어 혹은 번역 가능한 언어의 사용과 함께 의미를 공유하고 있는 어휘의 사용이 필요하다. 이는 ACL에서도 그대로 적용되는데 온톨로지는 ACL에서 사용하는 어휘의 집합을 나타낸다[Gruder, 1993a; 1993b]. 앞의 메시지에서 나타난 KQML 메시지의 내용 중에서 내용을

전달하는 :content에서 length의 의미를 명확하게 하기 위해 Particular\_Dim이라는 온톨로지를 정의함으로써 정보를 공유하는 에이전트에서 length의 의미를 배의 길이(LBP)라고 인식하게 된다.

### 2.3 선박 초기설계 에이전트 시스템 구현

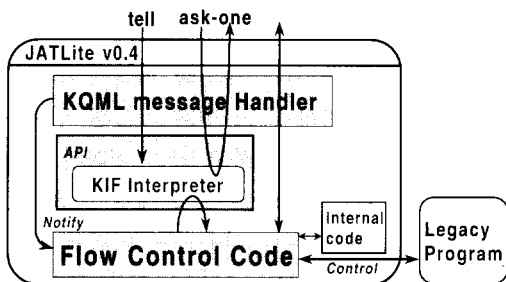
따라서 본 연구에서는 초기설계 에이전트 시스템의 구성에 기초하여 다음과 같은 기본 방향을 정하였다. 첫째, 에이전트의 기본 통신 기능에는 JATLite 0.4의 기능을 활용한다. 둘째로는 표준적인 API를 제공하는 범용적인 KIF 인터프리터를 개발하여 독립된 모듈로서 이용한다[최제민 외, 1997]. 여기서 범용적인 KIF 인터프리터의 개발을 위해서는 추론 기능, 함수 및 관계(Relation)의 정의 기능을 구현해야 한다. 셋째로 기본적인 KQML 수행자를 처리하기 위한 표준적인 알고리즘을 설계한다. 본 논문에서는 우선 interested, handles, tell, ask-one, reply 수행자 등 기본적인 KQML만을 사용하였으며 JATLite 0.4에는 KQML 수행자를 토른 단위로 분석하는 기능이 있을 뿐 그 의미를 구현해 주지는 않는다. 따라서 KIF 인터프리터와 연계해서 KQML의 의미대로 실행되는 KQML Message Handler가 필요하다. 특히 표준적인 KQML Message Handler의

개발은 독립적인 KIF 인터프리터와 마찬가지로 에이전트 개발의 생산성을 크게 높여준다. 다음에서 퍼실리테이터가 ask-one 수행자를 받았을 때 KIF 인터프리터를 통해 적당한 reply를 보내거나 CBR(Content-Based Routing) 기능을 이용해서 다른 에이전트에게 라우팅 하는데 여기서는 기본계획 에이전트가 퍼실리테이터의 역할도 겸하게 하였다.

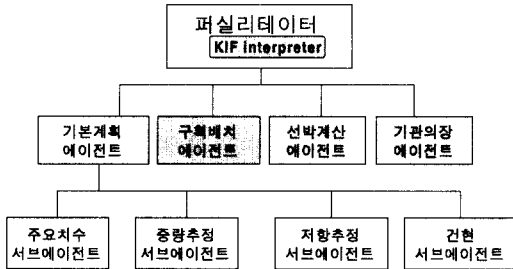
이상의 기본 방향에 따라 [이상욱, 1998]을 바탕으로 [그림 3]과 같이 에이전트의 기본 아키텍처를 개념적으로 설계하였다. 에이전트는 기본적으로 JATLite 0.4의 틀에서 동작하며, ACL은 JATLite 0.4의 통신 인터페이스를 통해 앞에서 설명한 KQML Message Handler에 전달되고 다시 KIF 인터프리터에게 Content가 전달된다. 이때 KIF 인터프리터에서는 tell이나 reply 수행자의 Content는 자신의 지식베이스(Knowledge Base)에 저장하고 handles, interested 수행자는 특별히 관리한다. 또한 ask-one 수행자에 대해서는 reply를 보내고 만약 에이전트가 퍼실리테이터의 역할도 겸한다면 다른 에이전트에게 라우팅 할 수도 있다.

#### 2.3.1 퍼실리테이터(Facilitator)를 이용한 정보 제어

선박 초기설계 에이전트 시스템을 구현하기 위해서 본 연구에서는 [그림 4]와 같은 다중 에이전트 시스템(Multi-Agent System) 개념을 도입하였다. 에이전트 시스템에서 다수의 에이전트들 서로 통신하기 위해서는 각각의 에이전트가 다른 에이전트들의 주소를 관리해야 하고, 또한 각 에이전트에 대한 인터페이스를 알아야 하므로 에이전트 개발을 매우 어렵고 복잡하게 만들 수 있다. 따라서 그 대안으로 각 에이전트간의 통신 문제에



[그림 3] 에이전트화를 위한 기본 구조



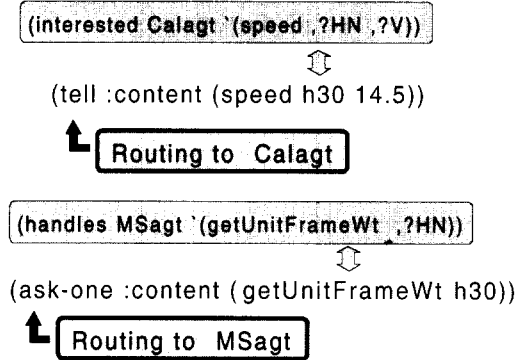
(그림 4) 에이전트기반 선박설계 시스템 구성도

대한 부담을 줄이고 인터페이스에 대한 중앙 집중식 관리를 위해서 연방 체계(Federated Architecture) 개념을 채용한다. 이러한 연방 체계에서는 에이전트들의 연결을 퍼실리테이터(Facilitator)가 담당하며 각 에이전트들은 초기화 단계에서 자신의 주소와 인터페이스를 퍼실리테이터에 등록함으로써 퍼실리테이터가 에이전트간의 통신을 중계하고 업무를 "Monitoring" 하고 "Coordination" 하는 역할을 한다. 즉, 에이전트가 ACL을 퍼실리테이터에게 보내기만 하면 퍼실리테이터는 자신에게 전달된 ACL의 내용과 등록된 인터페이스를 보고 적당한 에이전트에게 그 메시지를 보내주고 응답되는 결과를 다시 발신자에게 중계해 준다.

이와 같은 퍼실리테이터의 역할 중에서 가장 중요한 개념이 Content-Based Routing 이라 할 수 있다.

### 2.3.2 Content-Based Routing (CBR)

퍼실리테이터가 에이전트들 간의 메시지들을 어떻게 정확히 전달할 것인가 하는 것으로서 에이전트가 보낸 ACL의 내용을 보고 그 메시지를 다룰 수 있는 인터페이스를 등록한 에이전트를 찾아서 그 에이전트에게 ACL을 라우팅 해준다

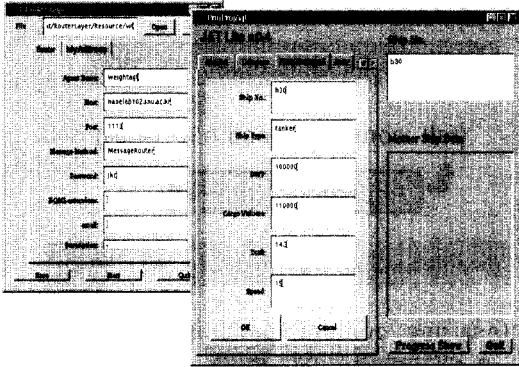


(그림 5) 퍼실리테이터에서의 정보 라우팅 (Content-Based Routing)

[Genesereth & Singh, 1994]. [그림 5]는 각 에이전트로부터 퍼실리테이터에 등록된 정보가 어떻게 라우팅 되는지를 보여준다.

여기서 (interested Calagt '(speed ,?HN ,?V))는 선박계산 에이전트(Calagt)가 (speed xx xx) 형태의 문장에 관심이 있다는 것을 나타내며, 이 KQML 문장의 (speed ,?HN ,?V) 는 실제 지식이나 정보를 담고 있는 내용계층으로 KIF으로 표현되는데 KIF에서 ?로 시작하는 항은 Individual 변수를 가리키며 이 예에서는 변수이름은 중요치 않다. 컴마 다음의 항은 문자열 자체가 아닌 변수의 값에 관심이 있다는 것을 나타낸다. [그림 6]은 에이전트 기반 선박 설계 시스템에서 구현된 입력창으로서 여기서 주어진 데이터들은 이미 등록된 인터페이스에 따라 퍼실리테이터에 의해 Calagt 에이전트에게 전달된다.

또한 (handles MSagt '(getUnitFrameWt ,?HN))는 구조설계 중에서 선박 중앙단면 설계를 담당하는 MS에이전트(MSagt)가 (getUnitFrameWt xx) 형태의 질문에 대답할 수 있다는 것을 나타낸다. 이 때 기본계획 에이전트가 퍼실리테이터에게 다음과 같은 ACL을 보내면 퍼실리테이터는



[그림 6] 선주 요구조건 입력창

등록된 인터페이스와 ACL의 Content를 비교하여 tell 수행자는 선박계산 에이전트(Calagt)에게 ask-one 수행자는 MS에이전트(MSagt)에게 라우팅 해준다.

```
'(tell :content (speed h30 15.0))
(ask-one :content (getUnitFrameWt h30))
```

결국 퍼실리테이터의 CBR(Content-Based Routing) 기능은 에이전트가 ACL을 어느 에이전트에게 보내야 하는가를 알지 못하더라도 퍼실리테이터를 통해 적당한 에이전트에게 전달되도록 해주는 기본적인 기능이다.

### 3. 선박 초기설계에서의 의사 충돌

여기서는 앞서 2절에서 구현된 선박 초기설계 에이전트 시스템에서 나타날 수 있는 설계 시스템간의 의사 충돌 및 해결 사례를 조선소의 설계 업무에서 나타나는 사례를 바탕으로 추출하였다. 실제로 선박 초기설계 과정에서 많은 의사 충돌 사례가 발생하지만 그 중에서 가장 전형적인 설계 시스템간의 충돌문제에 대해 언급하겠다.

이러한 의사 충돌은 근본적으로 각각의 설계 시스템들이 서로 다른 관점에서 설계를 수행하기 때문에 설계 과정에서의 충돌 문제는 피할 수 없는 해결해야 할 문제이다. 선박 초기설계 과정에서는 [그림 7]과 같은 설계 관점의 차이에서 문제가 발생하게 된다. [그림 7]에서 기술한 설계과정의 의사충돌과 이들의 협의 과정은 사실상 설계부서의 영향력에 상당히 영향을 받는다. 기본계획 분야 부서의 영향력이 큰 조선소의 설계에서는 선박의 길이를 최대한 줄이기 위해서 일반적으로 기관실 구획이 짧고 선미형상이 좀 넓다든지 아니면 Web Frame 간격이 넓다. 반면에 이와 반대 현상을 보이는 조선소의 경우는 기관실 구획이 1 Frame 정도 길며 선미형상 역시 다소 날씬한 경우가 많다.

이러한 점들을 고려하여 본 논문에서는 조선 공학적 측면에서 의사 충돌을 해결해 나가는 과정을 대상으로 설계 시나리오를 작성하여 충돌 해결 및 협의 전략을 모색하였다.

### 4. 사례기반 추론에 의한 충돌해결

이미 언급한 바와 같이 에이전트간의 의사 교환을 통하여 동시공학적 협동설계 시스템을 구축함에 있어 무엇보다 중요한 것이 어떻게 각각의 설계 관점의 차이에서 발생하는 의사 충돌 문제를 효과적으로 시스템화하여 처리할 수 있을 것인가의 문제이다. 이를 해결하기 위하여 과거의 문제 해결 사례를 바탕으로 현재의 문제를 해결해 보고자 하는 시도가 쉬우면서도 설득력이 있다고 생각된다. 따라서 여기서는 3절에서 언급한 에이전트간의 논리적인 의사충돌 문제를 사례기반 추론 기법에 의한 해결 방법에



의해 접근하고자 한다.

여기서는 기장설계 에이전트 시스템을 중심으로 기본계획 에이전트, 선형설계 에이전트, 그리고 구조설계 에이전트 간의 설계 관점의 차이에 의한 의사 충돌 문제를 해결하기 위하여 먼저

적으로 처리하기 위해서 의사 충돌 해결 처리기 (Conflict Resolution Handler)를 구현하고, 이를 퍼실리테이터내에 두어 에이전트간의 정보교환 과정에서 발생하는 의사충돌 문제를 해결해 주도록 하였다[Klein, 1991; 1993].

**설계 관점**

- 기본계획 : 경제성 관점 (선가났고 성능이 우수한 배)  
 >> 가급적 배의 길이 짧게
- 선형설계 : 선미부 형상 Slender
- 구조설계 : 구조강도 만족을 위한 보강  
 >> Web Frame 크게
- 기장설계 : 기관실 배치를 위해 E/R 공간 확보

**의사 충돌 발생**

: 기관실 배치과정에서 Hull과의 Clearance (Passage Way(60Cm)) 확보 어려움

**충돌 해결 과정**

- 기본계획에 요청 : 기관실 구획을 1 frame 늘려달라  
 기본계획의 응답 :  
 (Case 1) 1 frame을 늘리겠다.  
 설계변경 전략 적용(기본계획, 기장설계)  
 (Case 2) 경제성 측면에서 불가함
- 선형설계에 요청 : 선미형상을 더 넓게 해달라  
 선형설계의 응답 :  
 (Case 1) 선미형상 수정  
 선미형상 변경에 따른 성능해석 평가  
 (Case 2) 저항추진성능 때문에 불가함
- 구조설계에 요청 : Web Frame을 줄여달라
- 기장설계 자체 해결 : 재배치를 통한 해결

(그림 7) 선박설계 과정에서의 의사충돌 예

효과적인 사례 구현 전략을 수립하고, 이를 바탕으로 사례베이스를 구축하였다. 이렇게 구축된 사례베이스를 이용하여 과거의 의사 충돌 해결 사례 중에서 유사도 평가에 의해 유사사례를 추출하고 이를 바탕으로 해결 방법을 제시하였다. 본 논문에서는 이러한 일련의 과정을 효과

**4.1 사례기반 접근에 의한 에이전트간의 의사충돌 해결**

**4.1.1 사례베이스 구축**

협동설계 과정에서의 의사충돌 해결을 위한 사례베이스 구축 과정을 설명하기 위해서 여기서는 앞서 3절에서 언급한 기관실 기기 배치 문제에서 발생하는 기장설계, 기본계획, 선형설계, 그리고 구조설계 시스템간의 의사충돌 문제를 대상으로 하였다. 이러한 에이전트 시스템간의 의사충돌 문제를 해결하기 위해서 다음과 같은 네 가지 항목으로 세분화하여 사례를 구현하였다.

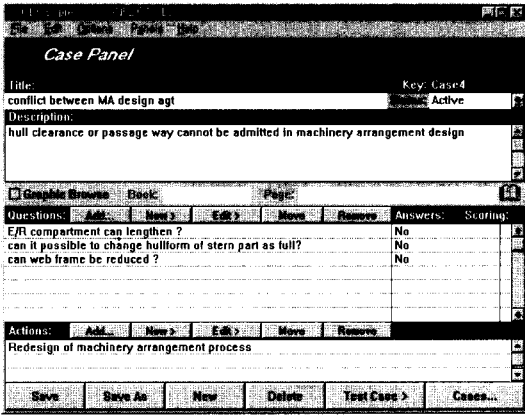
1) Title

CBR(Case-Based Reasoning)을 이용하여 설계 에이전트간의 충돌 해결 시스템을 구성하기 위해서 Title은 실제로 의사충돌이 일어나는 에이전트의 이름을 아래와 같이 기술하였다.

- Conflict between MA Design Agent and Preliminary Design Agent
- Conflict between MA Design Agent and Hullform Design Agent
- Conflict between MA Design Agent and Midship Design Agent
- Conflict between MA Design Agent

2) Description

앞서서도 언급하였듯이 Description은 사례의



(그림 8) 충돌해결을 위해 구현된 사례베이스의 예 특징을 결정짓는 요인으로서 유사사례를 분류하기 위한 개략적인 사항을 기술한다. 즉 사례들 가운데 주어진 문제를 해결하는데 관련이 있는 사례들을 찾아내는 역할을 수행한다. 이 부분이 사례기반 추론 방법의 인덱싱에 해당하는 부분으로서 사례의 분류 및 검색에 매우 중요한 인자이다. 따라서 여기에는 다음과 같은 정보가 키워드(Keywords) 방식이나 자연언어(Natural Language) 방식으로 기술된다.

- 선종(Ship type) : bulk carrier, tanker, container
- 문제가 되는 영역 : hull clearance, passage way, 등
- 문제가 되는 에이전트 시스템 명 : Preliminary Design, MA Design, 등

### 3) Question / Answer

Question은 Description에 기술된 사항들에 의해 관련된 사례를 추출한 후 추출된 사례의 유사도(Similarity Score)를 파악하여 주어진 문제에 가장 가까운 사례를 제시하기 위해 특성치들에 대한 질의응답 과정을 거치게 된다. 즉 세부사항을 통하여 구체적인 유사사례를 찾아내기 위

한 것이다. 여기서는 각 특성치 항목에 대한 가중치(Weight)를 부여하게 되며, 가중치가 고려된 사례의 유사도를 평가하게 된다. 다음은 기술된 Question/Answer의 예이다.

- E/R compartment can lengthen ? (Yes / No)
- Can it possible to change hullform of stern part as full ? (Yes / No)
- Can Web frame be reduced ? (Yes / No)

### 4) Action

Action은 주어진 문제상황에 따라 취해질 수 있는 조치사항을 기술하였다. 만일 기관실 구획을 선수쪽으로 조금 늘릴 경우 이에 수반되는 기관실의 기기 배치를 재배치하고, 기관실 구획이 늘어나면서 발생하게 되는 기본계획 에이전트 시스템의 영향을 고려하여 메시지를 기본계획 에이전트에게 전달하게 된다. 다음은 구현된 사례에서의 Action의 예이다.

- Make E/R longer (Move FWD BHD)
- ☞ Execute redesign process of preliminary design
- ☞ Redesign machinery arrangement process
- Change the hullform of stern part
- ☞ Execute evaluation process of speed/power
- ☞ Redesign machinery arrangement process (Check hull clearance an passage way)
- Reduce web frame size
- ☞ Execute evaluation process of structural strength

이러한 내용을 바탕으로 설계 에이전트간의 의사충돌 해결을 위한 사례베이스를 구축하였고, [그림 8]은 구현된 사례의 한 예를 보여주고 있다.

### 4.1.2 유사도 평가 (Matching & Ranking)

사례베이스로부터 유사사례를 추출하기 위해서는 앞서 언급한 방법에 의해 사례를 조직화한 다음 이를 바탕으로 매칭함수를 도입하여 주어진 문제와 기존의 사례들이 어느 정도 유사한지를 평가하게 된다. 이러한 매칭을 바탕으로 기존 사례들의 유사척도에 따라 랭킹이 정해져서 유사사례가 추출되게 된다.

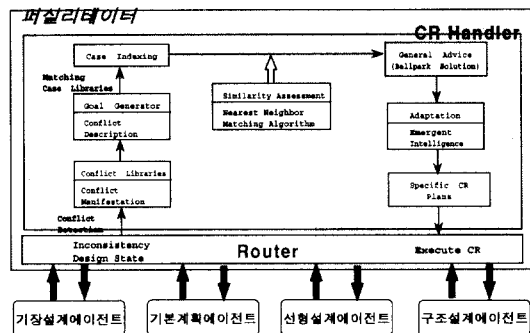
일반적으로 매칭과 랭킹에 관련하여 크게 세 가지의 절차로 구분할 수 있다. 첫째는 수치적인 절차 (Numeric Procedure)와 경험적 절차 (Heuristic Procedure), 그리고 이 두 가지를 합성한 복합 경험적-수치적 절차 (Combined Heuristic and Numeric Procedure)가 그것이다.

그러나 일반적으로 대부분의 사례기반 시스템에서 수치적 절차에 바탕을 둔 Nearest-Neighbor 알고리즘을 이용하고 있다. 이 방법은 수치 평가 함수를 사용하며, 본 논문에서도 이 Nearest-Neighbor 알고리즘을 이용하였다.

## 4.2 사례기반 충돌해결 처리기

4.1절에서 구현된 사례베이스를 바탕으로 본 논문에서는 동시공학적 설계 환경내에서 에이전트간의 정보교환 과정에서 발생하는 의사충돌 문제를 체계적으로 해결하기 위해서 사례기반 추론 기법에 의한 충돌해결 처리기를 [그림 9]와 같이 구현하였다. 이 처리기는 에이전트간의 정보교환 및 공유를 제어하는 퍼실리테이터 내에 두어 설계 에이전트들로부터 들어오는 정보를 바탕으로 의사충돌이 일어났는지를 감지하고, 충돌문제가 발생하였을 때 이를 신속하고 체계적으로 처리하여 관련된 에이전트에게 전달해 주는 역할을 수행한다.

여기서 구현된 사례기반 충돌해결 처리기는 다음과 같은 과정을 거치면서 유사한 문제해결 사례를 바탕으로 설계 과정에서 발생한 의사충돌 문제를 해결해 나간다.



(그림 9) 사례기반 충돌해결 처리기

### 1) 의사충돌 표현 (Conflict Manifestation)

여기서는 현재 발생한 충돌문제에 대한 키워드를 생성하게 된다. 즉 유사사례를 인덱싱하기 위한 문제의 기술(예를 들어, Passage Way, Hull Clearance, ...) 및 충돌이 발생된 에이전트 명(예를 들어, Preliminary Design, Hullform Design, 등)이 생성되어 기술된다.

### 2) Goal Generator

의사충돌 표현 과정의 충돌 라이브러리(Collict Library)에서 생성된 키워드 및 에이전트 명을 바탕으로 인덱싱을 수행하여 관련 사례들을 찾아오기 위해서 API(Application Programming Interface) 기법을 이용하여 사례기반 추론 시스템을 작동시킨다. 생성된 키워드는 CallSetDescription이라는 함수를 통하여 사례기반 추론기에 전달된다.

### 3) 0.25

사례 인덱싱(Indexing)

기술된 description을 바탕으로 키워드의 매칭

을 통한 관련 사례를 추출하게 된다. 인텍싱 과정을 통하여 키워드, 에이전트 이름과 관련이 있는 모든 사례가 추출되어 이를 바탕으로 다음의 유사도 평가가 이루어진다.

4) 유사도 평가 (Similarity Assessment)

여기서는 Nearest Neighbor 알고리즘을 이용하여 특성치 타입별로 유사도를 평가하여 이를 종합한다.

5) General Advice

유사도 평가를 거친 사례들 중에서 유사도가 가장 높은 사례를 개략적인 해(Ballpark Solution)로 제시하게 된다. 일반적으로 이 해는 주어진 문제의 상황과 정확하게 일치되는 경우가 적기 때문에 변용(Adaptation) 과정을 거치게 된다.

6) 변용(Adaptation)

여기서는 앞서 제시된 개략적인 해를 바탕으로 구체적인 해결책을 제시한다. 예를 들어, 개략적인 해가 make longer E/R(move FWD BHD)라고 제시되었을 때 그러면 기관실(E/R) 길이를 몇 프레임 길게 할 것인가 하는 구체적인 해를 찾게 된다.

7) Conflict Resolution 실행

언어진 해를 Router로 구체적인 메시지 형태로 전송하여 관련 에이전트에 전달되고, 전달된 메시지에 따라 각 에이전트에서 문제 해결을 위한 재설계를 수행한다.

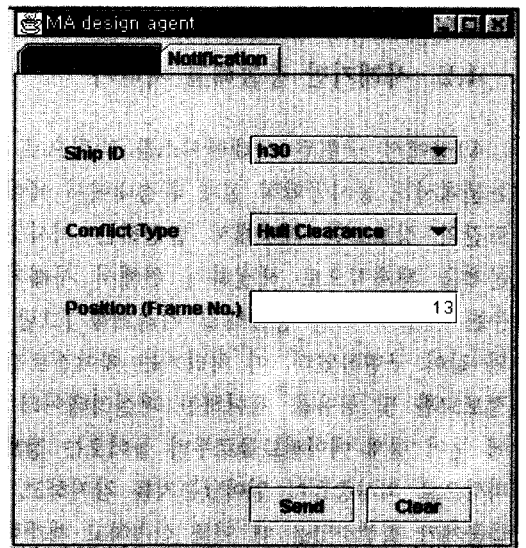
4.3 충돌해결 처리기에서의 문제 해결 시나리오

- 1) 기장설계
  - 문제 발생

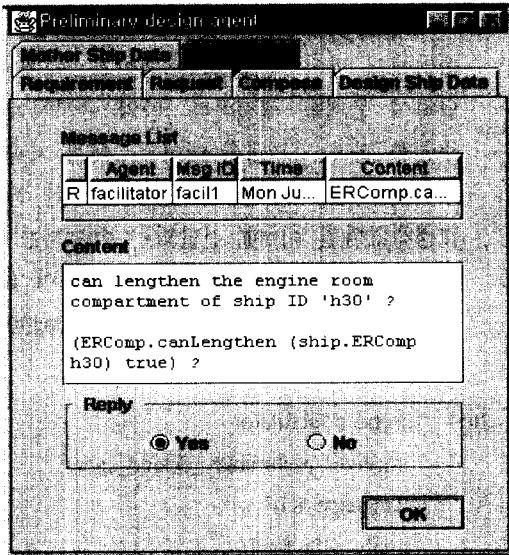
기관실의 기기 배치 과정에서 A기기와 선각(Hull)과의 여유공간(Clearance) 및 통로(Passage Way)를 위한 공간을 확보할 수 없음

```
(tell :sender MA_design_agent
:receiver facilitator
:language KIF
:reply-with MA0
:content (and (conflict CF1)
(conflict.type CF1 hullClearance)
(conflict.shipId CF1 h30)
(conflict.position CF1
(hullform.frame h30 13))
(conflict.source CF1
MA_design_agent)) )
```

[그림 10]은 기장설계에서 문제발생을 정의하는 창으로서 여기서 보내어진 정보는 위와 같이 에이전트 통신 언어로 변환되어 퍼실리테이터에게 전달된다.



(그림 10) 기장설계 에이전트에서 충돌 문제 정의를 위한 창



(그림 11) 기본계획 에이전트에서 응답하기 위한 메시지 창

## 2) 퍼실리테이터

- 메시지 전송 (충돌해결 처리기에게 라우팅)  
Conflict에 대한 메시지를 받아 이를 충돌해결 처리기(CR Handler)에 전달

## 3) 충돌해결 처리기(CR Handler)

- 퍼실리테이터로부터 전달받은 'Conflict 내용', 'Conflict 일어난 에이전트명'을 바탕으로 유사사례를 찾기 위한 문제 기술(Problem Description)

키워드 (Conflict 내용의 키워드 및 Conflict가 일어난 에이전트명) 위주로 기술 후 이를 Conflict Library에 저장

- 이러한 Conflict와 관련된 내용이 API 함수를 이용한 프로그램에 의해 충돌해결을 위한 사례기반 시스템으로 전달된다. 이 때 API 함수를 이용하여 정보가 전달되는 형태는 다음과 같다.

CallSetDescription("hull clearance in MA design agent");

- 관련 에이전트명, Conflict 내용을 담고 있는 사례들을 사례베이스로부터 인덱싱 과정을 통해 추출한다.

이렇게 퍼실리테이터로부터 전달받은 정보를 바탕으로 충돌해결 처리기에서 의사 충돌 문제를 해결하기 위해 사례기반 추론 시스템을 이용하여 유사 문제 해결 사례를 찾아내는 작업을 수행하게 된다.

## 4) 사례기반 추론 시스템

- 유사도 평가 과정 : 충돌 해결 처리기로부터 사례기반 추론 시스템으로 전달된 문제의 기술(의사충돌 문제, 문제가 일어난 에이전트 명)을 바탕으로 충돌 문제와 관련된 유사한 사례들이 인덱싱 과정을 통해 얻어지면 Question/Answer 과정을 통해 추출된 유사사례들을 평가하게 되는데, 이를 위하여 퍼실리테이터는 아래와 같이 관련 에이전트 시스템과의 정보교환을 통하여 유사도 평가에 필요한 Question/Answer 정보를 수집하여 이를 충돌해결 처리기로 라우팅하게 되고 이것이 다시 사례기반 추론 시스템에 전달된다.

- 퍼실리테이터 <--> 기본계획 에이전트  
- 퍼실리테이터는 기본계획 에이전트에게 '기관실 구획을 1프레임 늘릴 수 있는가?' 라는 메시지를 다음과 같이 보낸다.

```
(ask-if :sender facilitator
        :receiver preliminary_design_agent
        :language KIF
```

```

:reply-with facil1
:content (ERComp.canLengthen
          (ship.ERComp h30) true) )
    
```

- 잠시 후 기본계획 에이전트는 나름대로의 평가를 통하여 [그림 11]의 화면을 통하여 기관실 구획의 변경이 가능한지의 여부를 퍼실리테이터에게 통보하여 준다(Yes/No). 이렇게 보내진 정보는 아래와 같은 에이전트 통신 언어로 변환되어 퍼실리테이터에게 전달된다.

```

(tell :sender preliminary_design_agent
      :receiver facilitator
      :language KIF
      :reply-with pre0
      :in-reply-to facil1
      :content true )
    
```

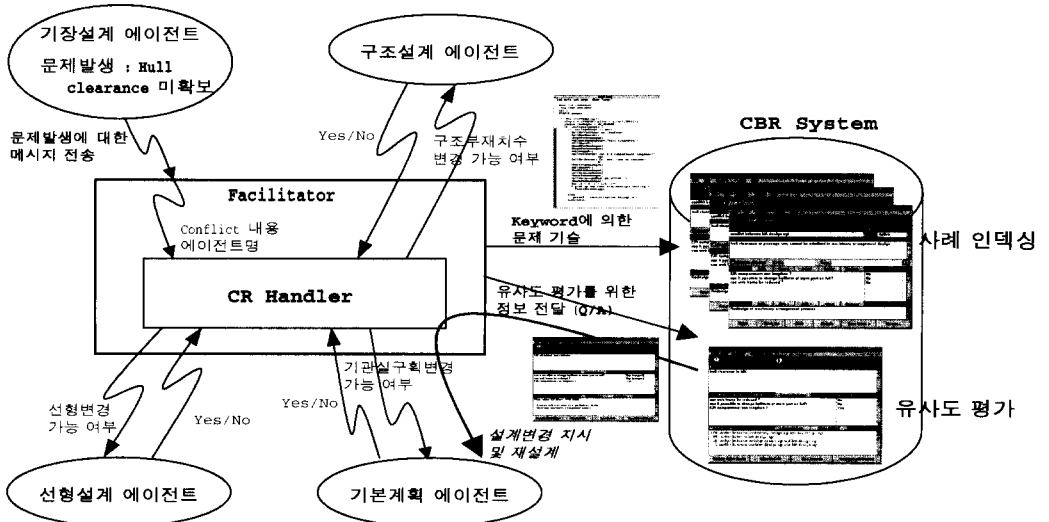
이와 마찬가지로 퍼실리테이터에서는 선형설계 에이전트, 구조설계 에이전트, 등에게도 같은 방법으로 정보를 수집하게 된다.

- 사례기반 시스템에서 제시된 해를 퍼실리테이터에서 받아 이를 관련 에이전트에게 전달하여 문제를 해결한다. 만일 유사사례를 추론한 결과 기본계획 에이전트에서 기관실 구획을 늘리는 방향으로 문제에 대한 해를 결정하였다면 이 때 전달되는 메시지는 다음과 같다. \* 메시지에서 알 수 있듯이 h30 호선의 기관실 구획의 길이(frameLength)를 1프레임 늘리게 된다.

```

(tell :sender facilitator
      :receiver preliminary_design_agent
      :language KIF
      :reply-with facil4
      :content (ERComp.Lengthen
                (ship.ERComp h30)
                (* 1 frameLength)) )
    
```

여기서는 기관실 배치설계 과정에서 발생하는 의사 충돌 문제중에서 한 가지에 대한 시나리오를 통하여 에이전트간의 전달되는 KQML 메시지를



(그림 12) 에이전트기반 선박 설계 시스템에서의 의사충돌 해결 시나리오

와 사례기반 추론 기법에 의해 충돌 문제를 해결하기 위한 충돌해결 처리기와의 의사 교환 및 이를 통한 문제 해결 과정을 기술하였다. [그림 12]는 위에서 언급한 의사 충돌 문제를 해결해 가는 과정에서의 정보 전달과 각각의 에이전트 시스템, 퍼실리테이터, 충돌해결 처리기, 그리고 사례기반 추론기간의 의사 결정에 대한 개념을 나타내고 있다.

## 5. 결론

본 논문에서는 크게 두 가지부분에 대해 언급하였다. 첫째는 선박설계 분야에서 동시공학적 협동설계 환경을 구축하기 위한 에이전트 시스템 구현이고, 또 하나는 이들간에 발생하는 의사충돌 문제로서 에이전트간의 논리적인 의사충돌 문제를 사례기반 추론 기법에 의해 해결 방법을 제시하였다.

에이전트 기반 선박설계 시스템 구현을 통하여 다양하면서 독립적인 설계 프로그램들이 원활한 정보교환 및 지식의 공유를 통한 지적 시스템을 구현할 수 있었다. 또한 의사충돌 문제를 해결하기 위하여 사례기반 추론에 의한 접근을 시도하였다. 특히 사례기반 추론의 일련의 과정을 효과적으로 처리하기 위해서 의사 충돌 해결 처리기(Conflict Resolution Handler)를 구현하고, 이를 퍼실리테이터내에 두어 에이전트간의 정보교환 과정에서 발생하는 의사충돌 문제를 감지하여 이를 해결해 주도록 하였다. 여기서 제시된 사례기반 추론 방법은 문제를 해결함에 있어 그 방법이 매우 단순하면서도 인간의 문제해결 방법과 또한 설계 개념이 매우 유사하여 에이전트간의 의사충돌 문제를 해결하기 위한 방법으로 매우 유용한 접근 방법이라 할 수 있다.

## 참고 문헌

- [1] Finin, T., Fritzson, R., et al., "KQML : an information and knowledge exchange protocol", *International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, 1995.
- [2] Finin, T., Labrou, Y. and Mayfield, J., "KQML as an agent communication language", In *Software Agents* edited by Bradshaw, J.M., MIT Press, 1995.  
Also available as  
<http://www.cs.umbc.edu/kqml/papers/KQML-acl.ps>
- [3] Finin, T., Weber, J., et. al., *Specification of the KQML Agent Communication Language*, The DARPA Knowledge Sharing Initiative External Interfaces Working Group, 1994.2.
- [4] Genesereth, M.R., Fikes, R., et al., *Knowledge Interchange Format Ver. 3.0 Reference Manual*, Technical Report of Computer Science Department, Stanford University, 1992.
- [5] Genesereth, M.R. and Ketchpel, S., "Software Agents", *Communication of the ACM*, Vol.37, No.7, 1994.
- [6] Genesereth, M.R., Singh, N.P. and Syed, M.A., "A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation", *Proc. of the International Symposium on Fifth Generation Computing Systems*, pp.125-139, 1994.
- [7] Gruder, T.R., *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, Tech. Report of Knowledge Systems Laboratory, Stanford University, 1993a.
- [8] Gruder, T.R., *A Translation Approach to Portable Ontology Specifications*, Tech. Report of Knowledge Systems Laboratory, Stanford University, 1993b.

- [9] Hyacinth, S. N., "Software Agents : An Overview", *Knowledge Engineering Review*, II(3), 205-244, 1996.10/11.
- [10] Klein, M., "Supporting Conflict Resolution in Cooperative Design Systems", *IEEE Trans. on Stems, Man, and Cybernetics*, Vol.21, No.6, Nov., 1991.
- [11] Klein, M., "Supporting Conflict Management in Cooperative Design Teams", *Group Decision and Negotiation*, Kluwer Academic Publishers, 2:259-278, 1993
- [12] Labrou, Y. and Finin, T., *A Proporsal for a new KQML Specification*, TR CS-97-03, Computer Science and Electrical Eng. Dept., Univ. of Maryland Baltimore County (UMBC), Feb., 1997.
- [13] Patil, R., Files, R., et al., "The DARPA Knowledge Sharing Effort : Progress Report", *Proc. of the 3rd International Conference on Principals of Knowledge Representation and Reasoning*, Nov., 1992.
- [14] 이상욱, *시스템 통합을 위한 에이전트 기본 아키텍처에 관한 연구*, 서울대학교 조선해양공학과 석사논문, 1998.
- [15] 최제민, 외, "에이전트 기반의 선박설계 시스템 구축을 위한 시험형 KIF 번역기 개발", *대한조선학회 추계 학술회의 발표집*, 1997. 11.