JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

기반 기술

J. Adv. Navig. Technol. 27(5): 635-642, Oct. 2023

지능형 소프트웨어 개발을 위한 통합개발환경 및 연동 에이전트 설계

Designing Integrated Development Environments and Integration Agents for Intelligent Software Development

서 민 기[1*] · 정 다 나[1] · 조 연 제[2] · 신 주 철[2] · 김 성 우[2]
[1]LIG넥스원 항공연구소,
[2]LIG넥스원 드론개발단

Min-gi Seo[1*] · Da-na Jung[1] · Yeon-je Cho[2] · Ju-chul Shin[2] · Seong-woo Kim[2]

[1*]Avionics System & Testbed, LIG Nex1, Daejeon 34115, Korea,

[2]Fixed Wing Drone System, LIG Nex1, Daejeon 34115, Korea

[요 약]

드론은 인공지능 기술의 발달로 단순한 원격 조종 도구를 넘어서 자율적으로 임무를 수행하는 지능형 드론으로 진화하고 있다. 해외 군사 분쟁에서의 드론 활용 사례와, 국내에서 전망한 미래 작전환경 분석에 따라 드론의 중요성이 점차 주목받고 있다. AMAD는 지능형 드론의 신속한 개발을 위해 제안되었다. AMAD를 기반으로 지능형 소프트웨어를 개발하기 위해서는 디버깅, 성능 평가, 모니터링 등의 기능을 사용자에게 지원하는 통합개발환경(IDE)이 필수적이다. 본 논문에서는 지능형 소프트웨어 개발에 필요한 개발환경의 개념들을 정립하여, 이를 IDE 및 IDE와 연동하는 AMAD의 에이전트인 SVI, MPD의 설계에 반영한 결과를 설명한다.

[Abstract]

With the development of artificial intelligence technology, drones are evolving beyond simple remote control tools into intelligent drones that perform missions autonomously. The importance of drones is gradually gaining attention due to the use of drones in overseas military conflicts and the analysis of the future operational environment in Korea. AMAD is proposed for the rapid development of intelligent drones. In order to develop intelligent software based on AMAD, an integrated development environment (IDE) that supports users with functions such as debugging, performance evaluation, and monitoring is essential. In this paper, we define the concepts of the development environment required for intelligent software development and describe the results of reflecting them in the design of the IDE and AMAD's agents, SVI and MPD, which are interfaced with the IDE.

Key word : Autonomous mission agents for drones, Framework, Integrated development environment, Intelligent drone, Multi-agent system.

http://dx.doi.org/10.12673/jant.2023.27.5.635

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-CommercialLicense(http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 27 September 2023; **Revised** 12 October 2023
Accepted (Publication) 24 October 2023 (30 October 2023)

***Corresponding Author : Min-gi Seo**

Tel: +82-42-723-2490
E-mail: mingi.seo@lignex1.com

Ⅰ. 서 론

드론은 무인 항공기의 한 분류로, 원격 조종 또는 자율 비행이 가능하다. 초기에 군사적인 용도로 개발되었지만, 현재는 농업, 산업, 과학, 교육, 레저 등 다양한 분야에 활용되고 있다. 드론의 활용 영역 확장에 따라 성능과 기능 역시 계속해서 발전하고 있다. 특히 인공지능 기술의 발달로, 드론은 단순한 원격 조종 도구를 넘어서 자율적으로 임무를 수행하는 지능형 드론으로 진화하고 있다. 19년 사우디아라비아의 석유 시설 공격, 리비아 내전의 Kargu-2 활용, 20년 아제르바이잔-아르메니아 전쟁, 22년 우크라이나-러시아 전쟁에 이르기까지 최근의 군사 충돌에서 드론의 역할은 더욱 중요해져, 전쟁의 새로운 양상과 전략에 큰 영향을 미치고 있다. 국내에서 전망하는 미래 작전환경은 병력자원 감축, 인명 중시 사상의 확산, 정보 및 화력 집중 필요성 증대, 드론의 위협 증대가 주요 이슈로 보고, 육군은 드론을 5대 게임체인저 및 평화 구축 역량 확충 용도로 제시하고, 2018년 9월 28일 드론봇 전투체계단을 창설하는 등 드론의 중요성을 인식하고 이를 위한 전략과 체계를 지속적으로 강화하고 있다[1].

드론은 인력과 자원의 손실 최소화, 적이 은닉한 위치 탐색, 정밀 공격, 다른 무기체계와의 협력 등 다양한 임무에 효율적으로 대응할 수 있다. 이러한 자율적 임무 수행 능력[2][3]과 비용적 효과를 기대하며 공용화 특성과 다양한 인공지능 기술[4][5]이 도입된 소프트웨어 프레임워크인 AMAD(autonomous mission agents for drones)가 제안되었다[6]. AMAD는 멀티 에이전트 시스템, 인지 아키텍처, 지식 기반의 상황추론 등의 기술들을 통합하여 드론의 독립적인 임무 수행 및 다른 드론과의 협력, 상위 시스템과의 통신을 위한 호환성을 지원한다. AMAD는 공용화된 운용 플랫폼을 통해 다양한 유형과 규모의 드론에 적용할 수 있으므로 드론의 신속한 개발과 효율적인 관리를 가능하게 한다.

AMAD 기반의 지능형 소프트웨어의 개발을 위해서는 디버깅, 성능 평가, 모니터링 등의 기능을 지원하는 개발환경이 필수적이다. 본 논문에서는 지능형 소프트웨어 개발을 위한 통합개발환경(IDE; integrated development environment)과 관련 에이전트에 대해 설명한다. IDE는 지능형 드론의 임무 수행을 위한 임무 패키지 배포와 모니터링 및 평가를 수행한다. IDE와 관련되는 에이전트는 2가지로 식별된다. 하나는 디버깅 및 모니터링을 지원하는 에이전트로, AMAD를 구성하는 에이전트들이 생성한 정보를 IDE에서 확인할 수 있도록 직관적인 형태로 변환하여 제공하거나, IDE-디버깅 대상 에이전트 간의 디버깅 명령과 디버깅 상태정보를 중개한다. 또 다른 하나는 AMAD 에이전트들의 실행에 필요한 초기 정보들을 제공하는 에이전트이다.

본 논문은 먼저 AMAD를 상세히 살펴본 후, IDE 및 IDE와 관련된 에이전트들의 설계를 설명하고, 구현 결과와 결론을 제시하는 순서로 기술하였다.

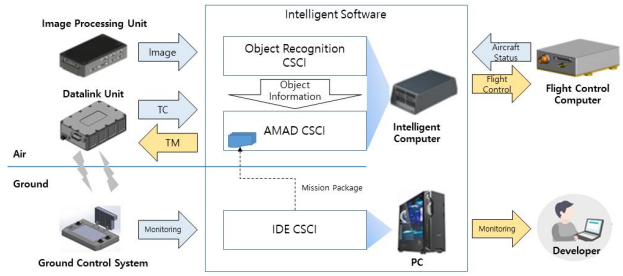

그림 1. 지능형 소프트웨어의 컨텍스트 다이어그램
Fig. 1. A context diagram for the intelligent software

Ⅱ. AMAD

지능형 드론은 외부 또는 내부의 사건에 대응하여 자율적으로 결정을 내리는 무인 항공기로, 인간의 개입 없이 임무를 수행하며, 이는 인공지능 기술을 이용하여 구현한다. 지능형 드론에 장착된 지능형 소프트웨어는 전장의 상황과 드론의 상태를 고려하여, 임무 달성을 위한 작업을 자율적으로 선택하고 실행하는 소프트웨어들을 일컫는다.

2-1 시스템 컨텍스트

지능형 소프트웨어의 시스템 컨텍스트 다이어그램은 그림 1에 제시하였다. 지능형 소프트웨어는 컴퓨터 소프트웨어 형상 항목(CSCI; computer software configuration item)과 임무 패키지로 구성된다. CSCI는 3종으로, 객체 탐지 알고리즘에 딥러닝을 활용한 객체 인식 CSCI, AMAD의 CSCI 명칭인 자율임무 CSCI, 그리고 이 논문에서 주로 다루게 될 통합개발환경 CSCI로 구성되어 있다. 임무 패키지는 지능형 드론이 임무를 자율적으로 수행하는데 필요한 지식, 상황 추론 규칙, 작업 계획 등 설정 데이터가 담긴 파일이다.

지능형 컴퓨터는 객체 인식 CSCI와 자율 임무 CSCI를 탑재하고 있으며, 이를 통해 드론의 영상처리장치에서 입력되는 영상, 데이터링크를 통한 제어 명령, 그리고 비행조종컴퓨터에서 제공되는 드론 상태정보를 수신한다. 수신된 영상에서 객체 정보를 인식하고, 드론의 상태정보와 제어 명령을 통합하여 객체 인식, 상황 판단, 의사결정 및 자율행동을 수행한다. 이러한 자율행동의 수행 결과는 드론과 센서에 대한 제어 명령으로 변환되어 비행조종컴퓨터로 전송된다. 드론의 상태정보와 자율 임무 수행에 대한 상태정보는 데이터링크를 통해 지상으로 전송된다. 개발자는 통합개발환경 CSCI를 활용하여 지능형 컴퓨터에 임무 패키지를 탑재하고, AMAD가 수행 중인 상황추론 및 의사결정 과정을 모니터링하고 디버깅할 수 있다.

2-2 핵심 설계 개념

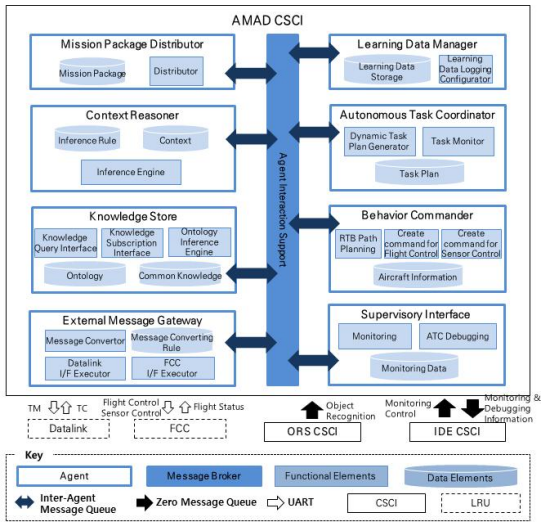

그림 2. AMAD의 멀티 에이전트 시스템 구성과 동작 개념
Fig. 2. Concepts of multi-agent system composition and behavior in AMAD

AMAD의 5가지 핵심적인 설계 개념은 1)인지 아키텍처 기반의 멀티 에이전트 시스템 적용, 2)지식 기반 에이전트 적용, 3)의사결정 알고리즘, 4)규칙 기반 추론, 5)프레임워크와 응용 모델의 분리가 있다[6].

인지 컴퓨팅은 인간의 인지 능력을 모사하는 기술로, 인간의 마음(mind)을 연구하는 고전 인지 이론으로부터 시작하여, 기호주의 인공지능 범주에 속하는 인지 로보틱스로 발달하였다. 이 과정에서 인지 아키텍처(cognitive architecture)라는 개념이 제안되었다[7][8]. 인지 아키텍처는 문제해결을 위한 추론을 지원하는 통합된 지식과 다양한 도메인에 적용될 수 있는 범용성이 요구된다. AMAD는 드론이 자율적으로 임무를 수행하기 위한 프레임워크로서, 이 인지 아키텍처에 매우 적합하다. AMAD는 인지 아키텍처의 특성을 구현하기 위해 멀티 에이전트 시스템을 적용하였다. 시스템 구성과 동작 개념

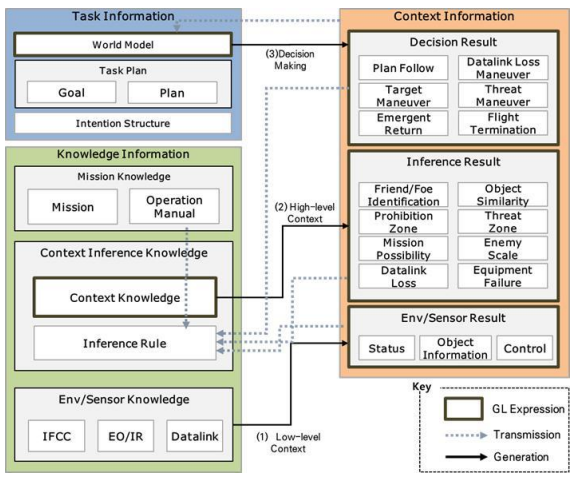

그림 3. AMAD의 지식의 분류와 흐름
Fig. 3. Classification and flow of knowledge for AMAD

은 그림 2와 같다.

지식 기반 에이전트는 자율 임무 수행에 필요한 추론과 의사결정이 지식 기반(knowledge base)으로 실행된다. 이 지식은 문장 형태로 구성되며, 추론은 문장으로 표현된 규칙에 따라 수행한다. 지식과 추론 규칙은 도메인으로부터 독립적이다.

의사결정 알고리즘은 BDI(belief, desire, intention) 아키텍처를 활용하였다. BDI 아키텍처는 인간의 행동 결정 과정을 에이전트에 적용한 구조로, 목표(goal)와 그 목표를 달성하기 위한 일련의 행동을 서술하는 작업계획(plan)을 토대로 의사결정이 이루어진다.

규칙 기반 추론은 규칙으로 정의된 조건(premise)과 결론(conclusion)을 통해 상황을 추론하는 설계 개념이다. 다른 에이전트와의 통신을 고려하여 조건과 결론은 지식 기반에서 파생된다.

프레임워크와 응용 모델의 분리는 지식의 구성을 중심으로 결정되었다. 그림 3은 AMAD의 지식 분류와 흐름을 보여준다. 그림의 왼쪽 항목에 해당하는 사전에 정의된 지식은 실행 간 결과적인 상황정보로 파생되는데, 그림의 오른쪽 항목에 해당한다. AMAD는 사전에 정의된 지식을 응용 모델로 정의하였으며, 변경에 유연하도록 응용 모델의 배포 개념을 반영하였다. '응용 모델'이라는 표현은 상대적으로 추상적이므로, 이를 '임무 패키지'라는 용어로 치환하였다.

III. IDE 및 연동 에이전트

IDE는 지능형 소프트웨어 개발 지원을 위한 목적으로 설계되었으며, 주요 기능은 프로젝트 관리, 제어, 배포, 디버깅, 평가, 오프라인 분석 및 개발환경 지원으로 구성하였다.

3-1 프로젝트 관리 기능

IDE는 작업 구분을 위한 프로젝트 개념을 도입하였다. 이에 따라, 프로젝트 관리 기능은 프로젝트 생성, 저장, 불러오기 등의 하위 기능을 포함하였다. 사용자는 임무 패키지의 이름을 지정하고, 임무 패키지 내부의 세부 구성을 결정할 수 있다. 임무 패키지의 구성은 표 1과 같으며, Plan, Knowledge, Rule은 그림 3의 작업 및 지식 정보와 연결된다.

표 1. 임무 패키지 세부구성
Table. 1. Details for the mission package

Category	Description	Extension
Plan	Task plans for decision-making	.plan
Knowledge	Knowledge required to do the mission	.owl
Rule	High/Low-level context reasoning rule The lif includes a interface protocol	.lif .lrule, .hrule
Alarm	Information to visualize context reasoning and decision making	.table
Package description	Information and meta-data describing the mission package	.mdp .mta
Etc	Platform-dependent information(mission plan, terrain elevation, etc.)	*

3-2 제어 기능

제어 기능의 핵심은 AMAD 에이전트와의 연동 및 데이터 송수신 처리에 있다. IDE는 AMAD 에이전트 중 하나인 SVI(supervisory interface)와 연동하는데, 기본적인 연동 개념은 IDE가 SVI로 연동을 요청하면, SVI가 모니터링 대상 데이터들을 IDE에 지속적으로 전송하는 것이다. 모든 메시지는 JSON(javascript object notation) 포맷으로 구성되며, 메시지는 연동 제어 메시지, 에이전트 메시지, 그리고 기타 메시지로 세분화된다. 연동 제어 메시지는 IDE와 SVI 간의 연동을 위한 메시지로, 연동의 시작, 변경, 종료와 관련된 내용이 포함된다. 에이전트 메시지는 FIPA(foundation for intelligent physical agents)에서 제시한 ACL(agent communication language) 통신 표준을 구현[9]한 것으로, 에이전트의 행위를 표현하는 에이전트 액션 메시지와 에이전트 간 연동 메시지가 포함된다. 기타 메시지에는 디버깅과 평가 기능을 위한 SVI의 액션 메시지가 해당된다. 그림 4는 IDE와 AMAD(SVI) 간의 연동 프로토콜을 보여준다. IDE는 메시징 라이브러리인 ZMQ(zero message queue)를 초기화하여 SVI로 접속 후, 연동 시작을 의미하는 'create monitor' 메시지를 송신한다. SVI는 수신한 'create monitor' 메시지 내 포함된 필터 정보를 활용하여 프록시(proxy) 서비스 객체를 생성한다. 에이전트 통신, 지식 표현, 추론 규칙 표현, 에이전트 액션 로깅 등인 멀티 에이전트 시스템의 기본 기능을 제공하는 기반 에이전트인 AIS(agent interaction support)는 AMAD의 모든 에이전트 액션 메시지와 에이전트 간의 연동 메시지를 SVI로 전송하는데, SVI는 프록시 서비스를 통해 모니터링 대상의 에이전트 메시지를 선별해 IDE로 전송한다. 'change filter'는 IDE가 모니터링 대상 메시지를 변경하기 위해 전송하는 메시지로, SVI는 이 메시지를 수신하면 프록시 서비스 객체 내 필터 정보를 수신한 필터 정보로 대체한다. 'delete monitor'는 IDE가 연동을 종료하기 위해 전송하는 메시지로, SVI는 이 메시지를 수신하면 해당하는 프록시 서비스를 삭제한다.

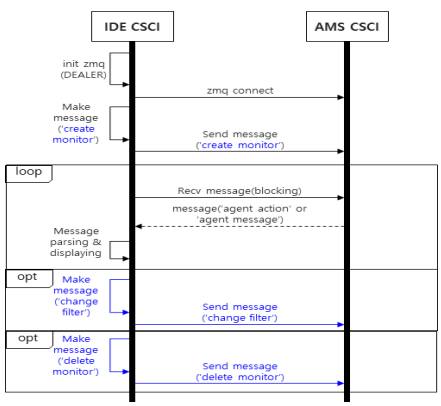

그림 4. IDE-AMAD 간 연동 프로토콜
Fig. 4. IDE to AMAD communication protocol

```
{
"LogType":"SystemLog",                  // type
"Actor":"agent://ContextReasoner",      // agent name
"Action":"MonitorReasoning",            // action name(action type)
"Content": {
    "ID":"OwnshipPosition",             // action's sub field
    "KnowledgeUsed":"(rule (...) --> (assert a))", // action's sub field
    "Result":"None",                    // action's sub field
    "Time":"20"                         // action's sub field(consume time, us)
    },
"Time":"1662460791883.000000"           // time stamp
}

{
"LogType":"MessageLog",                         // type
"Sender":"agent://ExternalMessageGateway",      // sender agent
"Receiver":"agent://KnowledgeStore",            // receiver agent
"Action":"Request",                             // message action
"Content":"(update (pos $a $b) (pos 1 2))",     // contents
"Time":"1662460791122.000000"                   // time stamp
}
```

그림 5. 에이전트 액션 메시지와 에이전트 간의 연동 메시지 형식
Fig. 5. A format for the agent action/message

그림 5는 에이전트 액션 메시지와 에이전트 간의 연동 메시지 형식을 나타낸다. 메시지에서 전달하려는 주요 내용은 'content' 속성에 포함되어 있으며, 이 'content' 속성의 값은 JSON 포맷에서 허용되는 모든 형식을 지원한다. 'content'가 지식 기반과 연결되어야 할 때는 GL(generalized list) 형식이 적용된다. GL은 일차 논리(first-order logic) 서술 외에도 문장 목록 내의 하위 문장 목록 구성이 가능하므로, 복수의 정보 간의 관계를 표현하는데 유용하다. 그림 5의 'MessageLog' 타입에 대한 'content'가 GL로 표현된 문장이며, GL 내 문장 구분은 괄호 기호를 이용한다.

3-3 배포 기능

IDE의 배포 기능은 작성한 임무 패키지를 지능형 컴퓨터로 전송하는 기능이다. Windows OS(operating system)에서 실행되는 IDE는 Linux OS의 지능형 컴퓨터와 SMB(server message block) 프로토콜을 통해 공유 영역을 정하며, 지능형 컴퓨터에서 임무 패키지의 배포를 담당하는 MPD(mission package distributor) 에이전트가 임무 패키지를 읽을 수 있게, 압축된 임무 패키지 파일을 공유 영역에 복사한다.

MPD의 실행 프로세스는 압축된 임무 패키지의 압축 해제부터 시작된다. 이후, 임무 패키지의 각 구성 요소에 대한 처리 핸들러들을 생성한다. 핸들러는 파일의 확장자에 따라 지정된다. 대상 파일의 확장자 및 파일 경로 확인 같은 핸들러 공통 기능은 핸들러 상위 클래스에 구현되며, 각 파일 별 파싱 방법과 AMAD의 다른 에이전트들에게 임무 패키지를 전달하는데 필요한 GL 작성 방법과 같이 확장자마다 다르게 구현되어야 하는 기능은 구현 클래스로 상속받은 순수 가상 함수를 오버라이드하여 구현한다. 이러한 핸들러 생성은 factory 패턴을 활용하여 생성에 대한 책임을 위임하였고, 핸들러의 전체적인 설계는 facade 패턴을 적용하였다. 팩토리 함수에서 핸들러가 생성되면, 핸들러별로 해당하는 임무 패키지 구성 파일을 읽고 파싱하고, GL 구문을 도출한다. 도출된 GL 구문을 구성 요소 별 해당하는 에이전트에게 요청(request)한 후, IDE에 임무

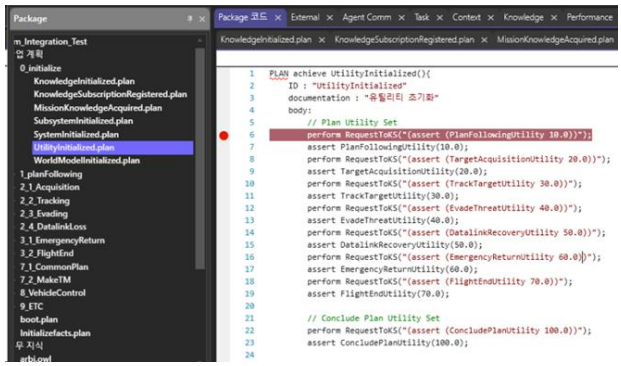

그림 6. IDE에서의 작업계획 중단점 설정 예시
Fig. 6. Example of setting a plan break point in the IDE

패키지 배포 상태를 전달하기 위한 액션을 수행한다. 앞서 언급한 IDE의 제어 기능과 SVI의 역할을 고려할 때, 패키지 배포 액션은 에이전트 액션 메시지 형태로 IDE에 전달되며, 이를 통해 IDE는 임무 패키지가 성공적으로 전송되었음을 확인한다.

3-4 디버깅 기능

AMAD의 에이전트 중 하나인 ATC(Autonomous Task Coordinator)는 BDI 아키텍처 기반 의사결정 알고리즘을 이용하여 현재 상황에 적합한 작업(task)을 선택하여 수행한다. BDI 아키텍처는 에이전트의 주변 환경 인식 결과인 월드 모델(belief)을 기초로, 에이전트의 목표(desire) 달성을 위한 현재 상황에 적합한 작업 계획을 탐색하고 선택하는 방식(intention)으로 동작한다. 개발자는 임무 모델링 시 goal(desire)과 작업 계획(plan)을 구분하며, 작업 계획은 'goal' 달성을 위한 절차적 사양(procedural specification)으로 구성되어 있다. 이 사양은 'goal' 달성 방법, 진입/진출 조건, 실행 중 조건, 액션 절차, 우선순위 결정 기준(유용도, utility) 등을 포함한다. 작업 계획이 의도대로 동작하는지 확인하기 위해서는 실행 및 결과 확인이 필요하다. 그러나 실시간 변화하는 월드 모델에서 액션의 결과를 정확하게 확인하려면, 실행 중단점(break point)과 현재 월드 모델 상태 확인 기능이 필요하다.

plan의 디버깅 기능은 IDE, SVI, 그리고 ATC가 상호작용하며 동작한다. 프로그램 시작 후 IDE에서는 디버깅 대상 작업 계획을 실행하고, 원하는 코드 라인을 클릭하여 중단점을 설정한다. 사용자에 의해 중단점이 설정되면 IDE는 중단점 액션 메시지를 SVI로 전송한다. SVI는 중단점 액션을 확인 후, 'content'의 값을 GL로 변환하고 ATC에 질의(query)한다. ATC는 중단점 액션인지 판별하고, 서브 타입이 중단점 추가인 것이 확인되면 ATC 중단점 관리자에서 제공하는 중단점 추가 함수를 호출한다. 중단점 삭제나 디버깅 제어 명령도 이와 유사한 방식으로 처리된다. 이러한 디버깅 제어 명령은 'stepinto', 'stepover', 'stepout', 'resume' 등이 포함된다. 디버깅 상태 정보는 ATC에서 SVI, 그리고 IDE로 전달되며, ATC는 중단점에 도달(hit)하거나 'step' 명령을 수행할 때마다 디버깅 상태를 보고한다. 결과적으로, IDE는 GUI를 통해 현재 ATC의 작업 계획 진행 상태를 표현한다. 그림 6은 IDE에서 작업 계획에 설정된 중단점을 보여준다.

3-5 평가 기능

AMAD의 성능 평가 지표는 표 2에 제시된 대로 상황추론 속도, 의사결정 소요시간, 그리고 상황추론 및 의사결정의 정확도를 포함한다. 상황추론은 저수준 및 고수준으로 나뉘며, 저수준 상황추론은 객체 인식 CSCI와 같은 AMAD의 외부 CSCI 및 데이터링크와 비행조종컴퓨터로부터 지능형 컴퓨터 장치가 수신한 센싱(perception) 및 환경(proprioception) 정보를 기반으로 내부에서 유통되는 상황정보로 추론하는 작업이다. 반면, 고수준 상황추론은 현 상황정보로부터 추가로 유추될 수 있는 새로운 상황정보를 추론하는 작업이다.

상황추론 속도는 IDE에서 측정한다. 운용자가 측정 시작을 요청하면, SVI는 IDE에 측정 시작 액션 메시지를 전송하며 측정이 시작된다. IDE는 이 명령을 받은 후, 초 단위로 저수준 및 고수준 상황추론 정보를 분석하고 집계하며, 시뮬레이션 동안의 평균을 산출한다. 운용자가 측정 종료를 요청하면, 측정 시작과 마찬가지로 SVI가 IDE에 측정 종료 액션 메시지를 전송하는 것으로 측정을 중지한다.

의사결정 속도는 ATC가 작업 계획을 선택한 순간부터 그 계획이 완료될 때까지의 시간을 나타낸다. 측정 시작/종료는 상황추론 속도 측정과 동일하지만, 측정 주체가 IDE가 아닌 ATC인 점이 다르다. 상황추론 속도는 상황추론 지식 카운팅을 매초 수행하지만, 의사결정 속도는 의사결정이 필요한 순간부터 시간을 시작하여 의사결정이 완료되는 순간까지의 시간을 기록한다. 이렇게 수집된 의사결정 소요 시간은 의사결정 횟수를 분모로 하여 평균을 산출한다.

표 2. 성능평가 방안
Table. 2. Performance evaluation methods

Metric			Measurement		Target performance
			Subject	Method	
Speed					
	context reasoning	Low-level	IDE	Evaluate by counting and averaging the number of logs received by the IDE every second	O O O O predicates/sec or more
		High-level	IDE	〃	O O O O predicates/sec or more
	Decision making		AMAD (ATC)	Calculate the average time spent in a log	O sec or less
Accuracy					
	Context reasoning		IDE	Evaluate by achievement rate for target test scenario (Achievement rate: the percentage of items that are correct against a predefined set of targeted outcomes)	OO% or more (only high-level context reasoning)
	Decision making		IDE	〃	OO% or more

그림 7. 상황추론 정확도를 판단하기 위한 정답지 예시
Fig. 7. Example answer sheet for judging context reasoning accuracy

상황추론과 의사결정의 정확도는 미리 정의된 상황추론 및 의사결정 결과를 기반으로 시나리오를 구성하고 실행하여 평가된다. 이에 따라 IDE는 상황추론 및 의사결정이 정확하게 이루어졌는지 확인할 정답지가 필요하다. 정답지의 형식은 그림 7과 같으며, 그림 7은 2개의 정답 조건(condition)을 포함한 예제이다. 첫 번째 조건은 3개의 predicate를 포함하며, 그 의미는 0번 항로점 도착 후 1번 항로점 도착까지 객체가 탐지된 상황임을 의미한다. 두 번째 조건은 4개의 predicate를 포함하며, 그 의미는 1번 항로점 도착 후 2번 항로점 도착까지 ID가 1인 표적을 확인한 후의 배터리 사용량이 50% 미만임을 나타낸다. 만약 1번 항로점 도착 이후 획득한 표적 ID가 1이 아니라면, 획득한 표적 ID 결과는 폐기하고, 다음 획득하는 표적 ID가 1일 때까지 대기한다. 시뮬레이션이 종료될 때까지 표적 ID 1을 수신하지 못한다면 해당 조건은 달성하지 못한 것으로 판정된다. predicate가 3개였던 첫 번째 조건이 1번 항로점 도착으로 정의된 마지막 predicate까지 달성했다면, 두 개의 condition 중 하나의 condition만 달성되었기 때문에 종합 결과 50%의 정확도로 계산된다. 따라서 정답지를 작성하는 개발자는, 상황추론 및 의사결정에 대한 지식의 발생 순서와 인자의 범위를 작성하고, 상황에 따라 선택 가능한 작업들 중 최종 선택된 작업의 정보, 센싱 정보의 잡음, AMAD가 탑재되는 하드웨어의 성능과 통신 인터페이스의 타이밍 등의 변수들을 고려하여야 한다.

개발환경에서의 AMAD는 네트워크 대역폭을 제한 없이 사용할 수 있으나, 드론에 탑재 후에는 지상체와의 데이터링크 대역폭에 제약을 받게 된다. 이러한 환경 변화를 고려하여, AMAD는 SVI의 모든 모니터링 정보 중에서 개발 관련 정보를 제외하고, 자율임무모드와 관련된 메시지만을 생성하도록 설계되었다. 여기서 자율임무모드란, 지능형 드론 운용자의 직접적인 제어 없이 지능형 소프트웨어가 독립적으로 임무를 수행하는 모드를 의미한다. 운용자는 이 모드에서 드론의 상태를 모니터링하며, 필요시 개입하여 통제권을 갖게 된다. 따라서 이와 관련된 메시지는 운용자가 통제권을 행사해야 할지의 판단을 위한 정보로 구성되며, 이를 AMAD에서는 '알람 정보'라고 한다.

그림 8은 알람 정보의 처리 과정을 상세히 나타낸다. 임무

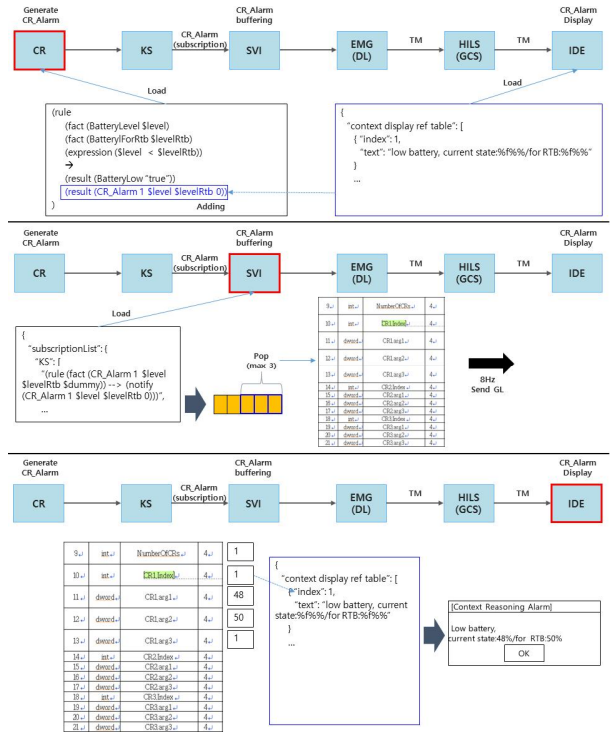

그림 8. 알람 정보를 처리하는 과정
Fig. 8. The process of handling alarm information

모델링 과정에서, 개발자는 각 상황추론 및 의사결정 항목에 대한 알람의 필요성을 검토한다. 필요한 경우, 해당 상황추론 규칙 또는 작업계획에 알람 predicate 생성 코드가 포함될 수 있다. IDE 환경 내에서는 이를 기반으로 인덱스 중심의 상황추론 및 의사결정 테이블을 구성한다. 예시로, 1번 인덱스의 상황추론 결과는 배터리 레벨과 기지 귀환(RTB; return to base)에 요구되는 배터리 레벨을 포함한다. 상황추론 규칙에서 배터리 잔량을 결정할 때, 해당 결론에는 알람을 위한 GL 문장이 포함되며, 이는 IDE의 상황추론 결과 디스플레이 테이블에 변수 바인딩과 함께 표시된다. SVI에서는 알람 관리자가 데이터링크 대역폭을 고려하여 알람 정보를 버퍼에 저장하며, 정해진 전송 주기마다 해당 정보를 EMG(external message gateway)에 요청하여 전송한다. EMG, GCS(ground control system), 최종적으로 IDE에 수신되는 메시지는 테이블을 참조하여 운용자에게 제시된다.

3-6 IDE 오프라인 분석 기능, 개발환경 지원 기능

드론의 데이터링크 대역폭 제한으로 인하여 주로 알람정보만이 연동되나, SVI 및 EMG는 운용 중의 모니터링 데이터를 로그 형태로 보존한다. 이를 통해, 운용자는 비행 종료 이후에 저장된 로그 데이터를 지능형 컴퓨터에서 추출하고, IDE에서 이를 오프라인으로 분석할 수 있다. 이 오프라인 분석은 기록된 로그를 기반으로 데이터 재생의 개념을 활용한다.

개발환경 지원 기능은 IDE의 IP(internet protocol)와 port

표 3. AMAD 핵심 설계 개념을 SVI, MPD에 적용한 결과
Table. 3. Applying AMAD key design concepts to SVI, MPD

Key design concepts	SVI & MPD
Apply cognitive architecture to multi-agent systems	Both SVI and MPD are intended to support the development of AMAD major agent tiers based on cognitive architectures, which are multi-agent systems but not cognitive architectures.
Knowledge-based agents	Both agents deal with knowledge and rules organized in sentences.
Decision making algorithms	Not applicable. Both agents are responsible for deploying task plans for decision making and monitoring the results of decision making.
Rule-based inference	Not applicable. Both agents are responsible for deploying rule for context reasoning and monitoring the results of context reasoning.
Decoupling frameworks and application models	The SVI is separately configured as an environment model where you can set up subscription tables for alarm of context reasoning and decision making, whether to log, etc. MPD is responsible for embedding a set of application models into AMAD.

설정, 그리고 모니터링 대상의 액션과 에이전트 메시지 선택을 위한 메시지 필터 설정 등의 환경을 구성하는 기능이다.

3-7 주요 설계 개념의 반영 결과 및 구현 결과

IDE를 제외하고, SVI와 MPD에서는 AMAD의 주요 설계 개념이 어떻게 적용되었는지의 상세는 표 3에 요약되어 있다.

그림 9는 IDE의 AMAD 성능 평가 기능을 구현한 화면이다. 5개의 차트들은 각각 AMAD 성능 평가 지표를 표현하였다. 성능 평가 데이터는 CR, ATC, EMG 에이전트에서 발생시킨 액션 메시지를 SVI가 변환하여 IDE로 전달하는 흐름을 갖는다. IDE는 수신한 데이터를 차트 또는 표 형태로 표현하고, 평균값을 도시하여 현재 시나리오 기준의 목표 성능 달성 여부를 판별할 수 있도록 구현하였다. AMAD 연구의 목표 성능 달성 확인은 성능평가 기능을 이용해 진행하였다.

Ⅳ. 결 론

지금까지, 지능형 드론에 탑재되는 지능형 소프트웨어 개발을 위한 IDE와 관련 에이전트인 SVI, MPD에 대해 살펴보았다. 먼저 지능형 소프트웨어의 시스템 컨텍스트와 AMAD 핵심 설계 개념을 살펴보았고, IDE의 주요 기능에 따른 설계 개념을 탐구하였으며, 임무 패키지 구성, IDE-SVI 간 메시지 연동, 임무 패키지 배포 설계 개념, ATC 디버깅 기능, 성능 측정을 위한 IDE 및 SVI 설계 개념, 그리고 데이터링크 대역폭 제한을 고려한 알람 정보 정의 및 구현에 관한 내용을 다루었다. 구현 중 겪은 난관은 방대한 양의 모니터링 메시지를 처리하는 IDE의 성능이었으며, 이는 버퍼링과 선별적 시현을 통해 극복하였다.

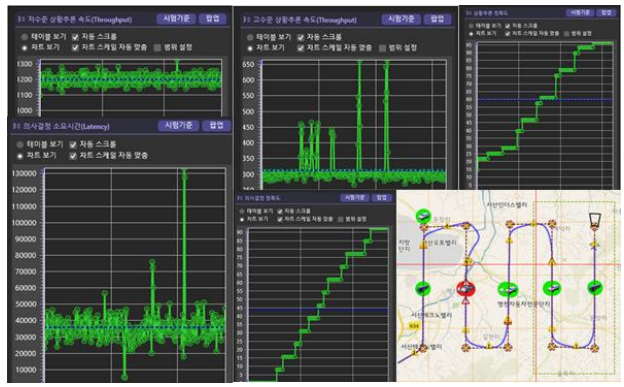

그림 9. AMAD 성능 평가 기능 구현 결과
Fig. 9. AMAD performance evaluation feature implementation results

향후 연구는 에이전트의 추가와 IDE 모델링 도입을 검토할 예정이다. 추가될 대상은 드론의 해킹과 같은 보안 문제에 대응하거나, 강화학습 기반의 의사결정 에이전트 등을 생각해 볼 수 있다. IDE 모델링은 작업 계획과 고수준 상황추론 규칙을 BT(behavior tree), HFSM(hierarchical FSM)와 같은 표준화된 개발 방법론의 적용을 의미한다. 이를 통해 공용화 개념을 공고히 하며, 효율적인 표준화 개발환경을 제공할 수 있을 것으로 기대한다.

Acknowledgments

본 연구는 2020년 국방과학연구소 미래도전국방기술 연구개발사업(912904601)의 지원을 받았음

References

[1] Korea Research Institute for defense Technology planning and advancement. Korea Research Institute for defense Technology planning and advancement [Internet]. Available: https://www.krit.re.kr/common/download.do?atchFileId=F2022032909350805O&fileSn=0

[2] M. Sarosa, M. N. Zakaria, S. Wirayoga and N. Muna, "Autonomous pilot drones for detecting objects in a space," in *Proceeding of the 1st The 1st Annual Technology, Applied Science, and Engineering Conference*, East Java, Indonesia, 2019.

[3] D.D. Nguyen, J. Rohacs, D. Rohacs, "Autonomous Flight Trajectory Control System for Drones in Smart City Traffic Management," *The Journal of International Society for Photogrammetry and Remote Sensing*, Vol. 10, pp. 338, 2021.

[4] J. S. Rao, K. Choragudi, S. Bansod, S. C. Paidipalli V V, S. K. Singh and P. Pal, "AI, AR Enabling on Embedded systems for Agricultural Drones," in *2022 International Conference on Futuristic Technologies (INCOFT)*, Belgaum, India, pp. 1-4, 2022.

[5] S. Prathibha, K. R. Saradha, P. Sharmila and S. Kaveya, "AI and Web Application in Medicine Delivery Drones," in *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India, pp. 48-53, 2022.

[6] J. C. Shin, S. W. Kim, G. H. Baek, and M. G. Seo, "A Proposal for Software Framework of Intelligent Drones Performing Autonomous Missions," *The Journal of Korea Navigation Institute*, Vol. 26, No. 4, pp. 205-210, Aug. 2022.

[7] B. T. Zhang and M. S. Yeu, "Cognitive computing I: Multisensory perceptual intelligence in real-world," *Communications of the Korean Institute of Information Scientists and Engineers*, Vol. 30, No. 1, pp. 75-87, Jan. 2012.

[8] A. Newell, "Unified Theories of Cognition," *Harvard University Press*, pp. 15-20, 1990.

[9] E. German and L. Sheremetov, "Specifying Interaction Space Components in a FIPA-ACL Interaction Framework," in *Languages, Methodologies and Development Tools for Multi-Agent Systems, First International Workshop*, Durham, UK, Sep, 2007.

서 민 기 (Min-gi Seo)
2012년 02월 : 한국항공대학교 컴퓨터공학과 (공학사)
2011년 10월 ~ 현재 : LIG넥스원 항공연구소 선임연구원
※관심분야 : 항공전자, 내장형 소프트웨어, 검증환경

정 다 나 (Da-na Jung)
2021년 08월 : 아주대학교 정보통신공학과 (공학석사)
2007년 2월 ~ 20021년 12월 : 삼성전자 네트워크사업부
2022년 05월 ~ 현재 : LIG넥스원 항공연구소 수석연구원
※관심분야 : 항공전자, 내장형 소프트웨어, 검증환경

조 연 제 (Yeon-je Cho)
2016년 02월 : 경희대학교 전자전파공학과 (공학사)
2018년 07월 ~ 현재 : LIG넥스원 드론개발단 선임연구원
※관심분야 : 소프트웨어 아키텍처, 항공전자, 인공지능

신 주 철 (Ju-chul Shin)
2009년 02월 : 포항공과대학교 컴퓨터공학과 (공학사)
2022년 03월 ~ 현재 : 한국과학기술원 소프트웨어대학원 석사과정
2009년 01월 ~ 현재 : LIG넥스원 드론개발단 수석연구원
※관심분야 : 소프트웨어 아키텍처, 인공지능, 항공전자

김 성 우 (Seong-woo Kim)
2002년 08월 : 부산대학교 정보통신공학과 (공학석사)
2002년 10월 ~ 현재 : LIG넥스원 드론개발단 수석연구원
※관심분야 : 자율화 시스템, 인공지능, 항공전자