5

ASADAL : 내장형 시스템의 객체 지향적인 개발 및 검증을 지원하는 CASE 도구

포항공과대학교 ▌안미영 · 김혜정 · 강교철*

개 요

최근에 소프트웨어 산업이 급속도로 발전함에 따라, 소프트웨어 시스템이 산업 전반에 걸쳐 두루 영향을 미치게 되었고, 소프트웨어의 개발 및 유지 보수비용이 제품 비용의 중요한 결정요소로 자리잡게 되었다. 특히, 기업이 경쟁력을 확보하기 위해서는 기업의 노하우가 축적될 수 있고, 빠르게 변하는 기술과 다양한 사용자의 요구사항에 쉽게 대처하여 고품질의 소프트웨어를 적기에 출시할 수 있어야 한다. 이러한 상황을 효과적으로 대처하기 위해서는 소프트웨어의 개발에서부터 테스트 및 유지보수까지의 전 과정을 체계적으로 관리하는 소프트웨어 공학적 방법이 필수적으로 요구된다.

본고에서는 여러 기업에서 소프트웨어를 효율적으로 개발하고 유지 보수하기 위해 사용하고 있는 CASE (Computer Aided Software Engineering) 도구인 ASADAL/OBJ(A System Analysis and Design Aid Tool/Object-oriented)에 대해서 소개한다. 먼저 ASADAL/OBJ의 기술적 특징 및 내부 구성 모듈에 대해서 설명하고, 공기압 시스템 예제를 통해 모델링과 시뮬레이션 과정에 대해 설명한다.

1. 서 론

ASADAL은 포항공대 소프트웨어 공학 연구실에서 지난 1995년부터 연구 개발해 온 방법론 및 CASE 도구로서, 요구 분석 단계에서부터 코드 생성 및 테스트, 유지 보수에 이르는 전 소프트웨어 개발 과정을 포괄하는 통합 개발 환경을 지원한다. ASADAL은 정형적인 요구 명세 및 분석 도구, 3차원 시뮬레이션 도구, 정형적인 검증 도구, 영역 분석 도구, 아키텍처 및 재사용 컴포넌트 설계 도구, 그리고 자동 코드 생성 도구 등을 포함하고 있으며, 실제 여러 산업용 소프트웨어 개발에 성공적으로 적용되어 그 실효성을 입증 받았다.

이 중에서 ASADAL/OBJ 도구는 객체 지향적인 소프트웨어의 명세 및 명세의 시뮬레이션을 통한 검증 기능을 제공하며, 여러 산업 분야의 소프트웨어 개발에서 사용되고 있다. 특히, 실시간 제어 시스템의 경우에는 소프트웨어의 안전성 및 신뢰성이 매우 중요한 요소이므로, 소프트웨어의 정형적인 분석이 필수적이다. ASADAL/OBJ에서는 소프트웨어의 개발 초기 단계에서부터 정형적인 명세를 바탕으로 시뮬레이션을 수행하고, 그 결과를 분석함으로써 쉽게 명세의 오류를 찾아낼 수 있게 한다. 또한, 제어 소프트웨어에 대한 시뮬레이션 결과는 가상 테스트베드를 통해 가시화됨으로써, 소프트웨어의 명세에 대해 잘 알지 못하는 사용자도 개발될 시스템을 직관적으로 이해할 수 있어 사용자의 요구 사항을 정확하게 분석할 수 있게 된다.

이를 위해, ASADAL/OBJ에서는 제어 소프트웨어와 상호 작용하는 환경 객체에 대한 모델링을 통한 가상 테스트베드의 구축과 시뮬레이션 결과의 3차원 가시화 기능을 제공한다. 3차원 시뮬레이션에 의한 소프트웨어 검사는 실제 현실 세계와 유사한 환경을 3차원 모델링을 통해서 구축하고 이를 바탕으로 소프트웨어를 검사 및 평가함으로써, 다양한 검사를 적은 비용으로 할 수 있다는 장점이 있다. 특히, 실시간 제어 시스템의 경우, 내장형(Embedded) 시스템인 경우가 대부분이므로, 외부의 하드웨어 환경과 내부 소프트웨어

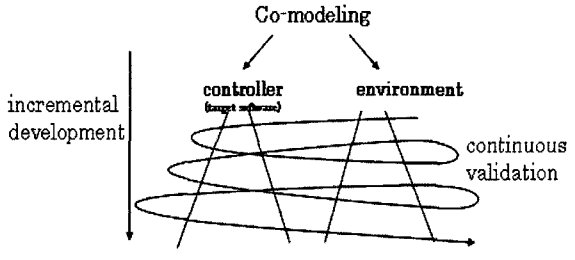

그림 1 ASADAL/OBJ의 점진적인 개발 프로세스

* 종신회원

시스템이 정상적으로 상호 작용을 하는지를 소프트웨어적으로 검사하고 평가함으로써, 소프트웨어의 신뢰도를 높일 수 있을 뿐만 아니라, 점진적인 개발과 계속적인 검사를 통해 개발 초기 단계에서 중대한 오류를 찾아낼 수 있으므로 개발 비용을 크게 줄일 수 있다.

본고의 구성은 다음과 같다. 2장에서는 ASADAL/OBJ의 기술적 특성 및 구성 요소에 대해서 설명하고 3장에서는 공기압 시스템 예제를 통해 모델링과 검증 과정에 대해 설명하며, 4장에서는 본고의 내용을 요약하고 결론을 내린다.

2. ASADAL/OBJ 소개

2.1 구성 요소

ASADAL/OBJ는 객체 지향적이고 점진적인(Incremental) 방법으로 내장형 시스템의 제어 소프트웨어(Target Software)와 그 제어 소프트웨어가 상호 작용하는 환경 객체(Environment)들을 동시에 모델링 할 수 있는 프레임워크를 제공하며, 명세의 시뮬레이션 및 3차원 가시화를 통해 사용자의 요구사항을 만족하는지 검사할 수 있는 기능을 제공한다.

ASADAL/OBJ 도구는 객체의 모델링을 위해 행위 명세 편집기, 기능 명세 편집기, 형태 명세 편집기를 지원하며, 시뮬레이션을 위해서 자동 코드 생성기와 시뮬레이션 실행 환경을 제공하는데, 이러한 도구를 이용한 제어 소프트웨어 명세와 가상 테스트베드 명세를 통한 시스템 모델링 과정과 시뮬레이션 환경에 대해 설명한다.

2.2 행위 및 기능 명세 언어

객체의 행위 명세는 Statechart를 이용하여 외부에서 들어오는 자극에 대해 어떻게 반응하여야 하는지를 기술하며, 기능 명세는 DFD(Data Flow Diagram)를 이용하여 기능과 데이터의 흐름 및 변환에 대해 기술한다. Statechart와 DFD를 이용하여 요구사항을 명세하는 자세한 방법에 대해서는 [6, 7]을 참조할 수 있다.

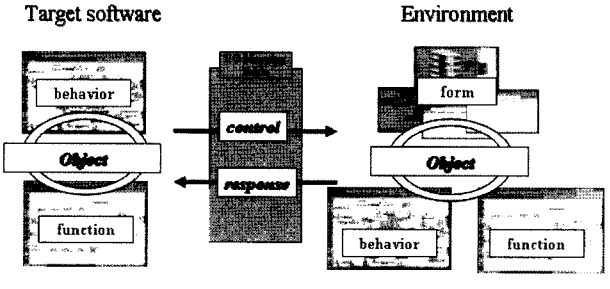

그림 2 ASADAL/OBJ의 모델링 프레임워크

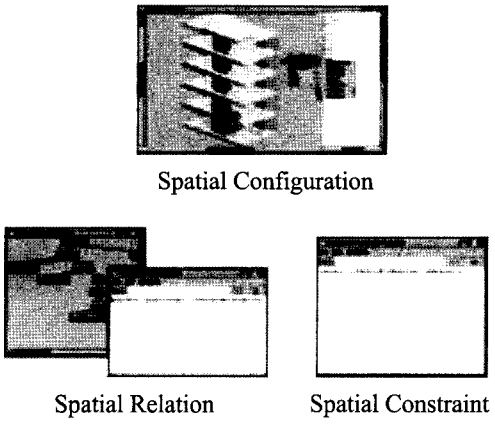

그림 3 형태 명세 편집기

2.3 형태 명세 언어

외부 환경 객체는 행위 명세와 기능 명세 외에도 공간적인 형태를 가지고 있기 때문에, 이를 기술하기 위해 FSL(Form Specification Language)이라는 형태 명세를 사용한다. 형태 명세에는 3차원 형태를 기술하기 위한 공간적인 구성 관계(Spatial Configuration)에 대한 명세, 환경 객체 사이에 존재하는 부딪힘(Interfere), 움직임 종속성(Motion Dependency) 등을 기술하기 위한 공간적인 관계(Spatial Relation)에 대한 명세, 환경 객체들 간에 지켜져야 할 공간적인 제약 사항(Spatial Constraint)에 대한 명세가 포함된다.

2.4 객체 합성에 의한 시스템 모델링

각 객체는 행위, 기능, 형태 명세에 의해 정의되고, 객체의 인스턴스가 생성될 수 있으며, 하위 객체들을 합성(Composition)하고 그들의 상호 작용을 통해 더 큰 역할을 수행하는 상위 객체를 정의하는 객체 지향적 프레임워크 상에서 시스템의 모델링을 수행한다.

제어 소프트웨어의 개발 시에는 행위와 기능 명세를 이용하여 지역적인 제어를 수행하는 객체를 생성하고, 이들 객체의 인스턴스들을 생성하여 구성하고 이들의 상호 작용을 통해 조정(Coordination) 역할을 수행하는 상위 제어 객체를 생성하며, 이런 식으로 점진적인 합성 과정을 통해 최종적으로 글로벌 제어를 수행하는 객체를 생성하게 된다.

가상 테스트베드 시스템 개발 시에는 말단의 환경 객체를 기능, 행위, 형태 명세를 이용하여 생성하고, 이들을 합성(Composition) 하고 상호 작용을 명세함으로써 좀 더 큰 환경 객체를 생성하며, 이는 사용자가 원하는 가상 테스트베드가 생성될 때까지 계속 수행된다. 이렇게 각각 개발된 제어 소프트웨어와 가상 환경 객체의 연결은 I/O 대응(Mapping) 과정을 통해, 대응하는 이벤트와 데이터를 연결함으로써 쉽게 가능해진다.

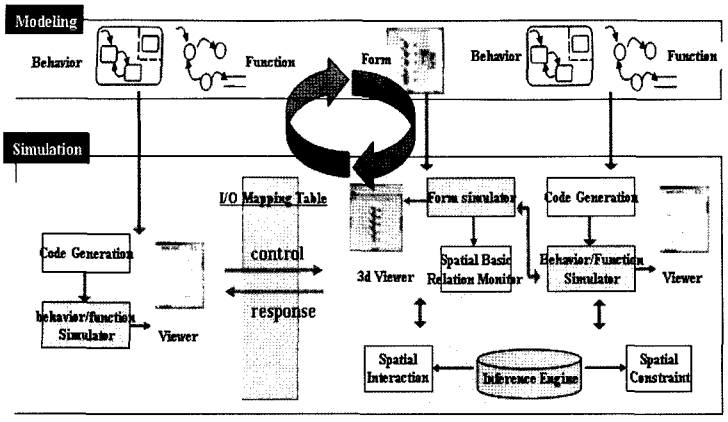

그림 4 ASADAL/OBJ의 시뮬레이션 환경

2.5 자동 코드 생성

제어 소프트웨어와 가상 테스트베드의 명세가 완성되면, I/O 대응 관계를 설정하고, 이를 바탕으로 타겟 언어(예, 자바 언어)로 된 소스 코드를 자동 생성하게 된다. 이렇게 자동 생성된 코드들은 시뮬레이션 환경 하에서 실행되어 가상 테스트베드를 통해 검증한 후에, 제어 소프트웨어 명세에 대해서만 코드 생성하여 실제 장치 드라이버 프로그램과 연결하는 코드를 추가 작성하면 실제 환경에서도 사용될 수 있다.

2.6 3차원 시뮬레이션

시뮬레이션은 시스템의 명세가 요구 사항을 만족하는지를 테스트 해 볼 수 있는 방법으로, 외부 이벤트 및 데이터의 변경 등과 같은 외부 환경을 직접 변경시켜 가면서 제어 소프트웨어 명세를 테스트하게 된다.

가상 테스트베드 상에서 시뮬레이션 하는 경우에는, 자동 생성된 제어 소프트웨어 코드와 가상 테스트베드 코드가 실행되면서, 서로 이벤트나 데이터를 주고받으며 시스템의 상태 및 행위를 변경시키는데, 현재의 시스템 상태는 Statechart 상에 표시가 되고, 제어 소프트웨어의 이벤트나 데이터 값의 변화에 의해 환경 객체의 위치, 형태 등의 변화가 가시화됨으로써, 시스템의 행위에 대한 직관적인 이해가 가능해진다.

제어 소프트웨어와 가상 테스트베드는 최종적인 단계에서만 시뮬레이션 하여 검증하는 것이 아니라, 개발 중간의 어느 단계에서도 시뮬레이션이 가능하므로, 중요한 시스템 요소부터 개발하여 검증해 보고, 나중에 다른 요소들을 더해 가면서 개발하고 검증하는 식으로 점진적으로 개발함으로써, 효율적으로 시스템을 개발할 수 있게 된다.

3. 공기압 시스템 모델링과 검증

본 장에서는 ASADAL/OBJ 도구를 이용하여 공기압 시스템(Pneumatic System)을 모델링 하는 방법을 기술하도록 한다. 공기압 시스템은 실제 시스템을 축소하여 만든 테스트베드용 시스템으로서, 공기압을 통해 실린더를 움직여 물건을 테이블로 밀어낸 후, 광 센서를 통해 물건의 이동을 감지한다. 테이블 위로 물건이 옮겨지면, 모터를 통해 테이블을 90도 회전하고 위치 검사기를 통해 물건이 제 위치에 왔는지 검사를 한 후 물건을 컨베이어(Conveyer)로 이동시키는 간단한 시스템이다.

그림 5는 ASADAL/OBJ에서 모델링한 공기압 시스템을 보이는데, 앞으로 그 과정에 대해서 자세히 설명한다.

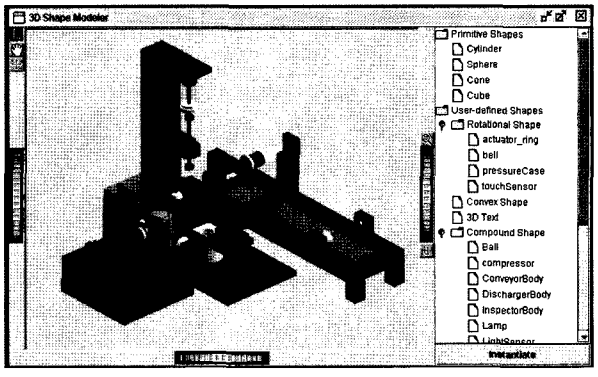

그림 5 ASADAL/OBJ 의 공기압 시스템 모델링

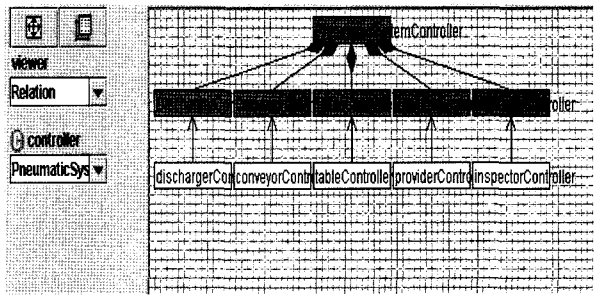

그림 6 제어 소프트웨어 합성 관계

3.1 제어 소프트웨어 명세

그림 6은 제어 소프트웨어의 합성 관계를 보이는데, 각 설비를 제어하기 위한 입력 제어기, 위치 검사기, 턴테이블 제어기, 출력 제어기, 컨베이어 제어기와 이들 설비의 초기화 및 연결 관계, 종료 처리를 위한 글로벌 제어기로 이루어져 있다.

각 제어기는 행위 및 기능 명세로 이루어지며, 그림 7은 턴테이블 제어기의 행위 명세인 Statechart를 보이고 있는데, 턴테이블이 멈춘 상태와 움직일 각도를 계산하는 계산 상태, 그리고 움직이는 상태로 이루어지며, 이들 사이의 상태 전이가 정의되어 있다. Statechart의 상태들은 상태 명세(State Spec.)를 가지고 있어서 이 안에서 DFD의 특정 프로세스를 실행하도록 명세 한다.

그림 8은 공기압 시스템의 턴테이블 제어기의 기능 명세인 DFD를 보이고 있다. 이 DFD에 기술된 프로세스들의 활성화를 제어하는 제어기의 이름은 둥근 네모 안에 표시하므로, 이 DFD는 TableActuatorController(그림 7의 Statechart)에 의해 제어됨을 알 수 있는데, 예를 들어 테이블의 움직임의 각도를 계산하는 기능을 수행하는 'rotate90' 프로세스는 TableActuatorController의 'calcPosition' 상태에서 활성화된다.

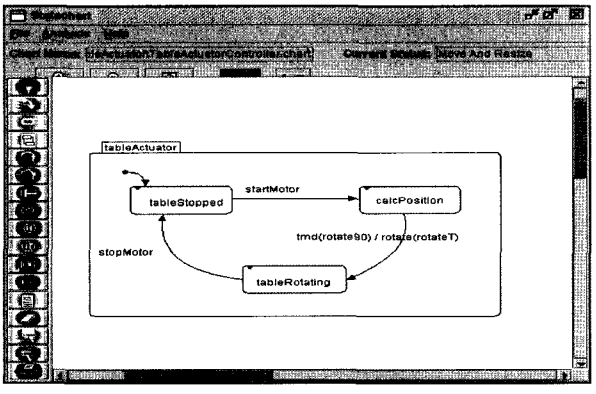

그림 7 턴테이블 제어기의 행위 명세

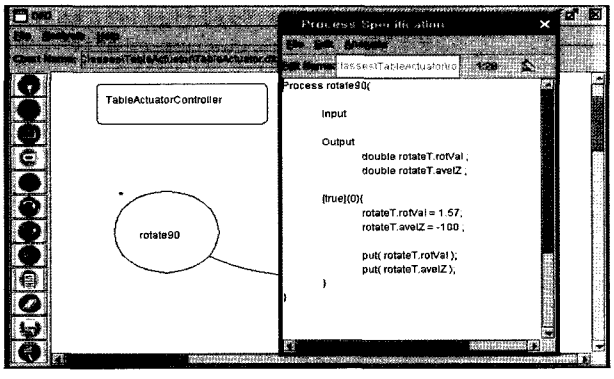

그림 8 턴테이블 제어기의 기능 명세

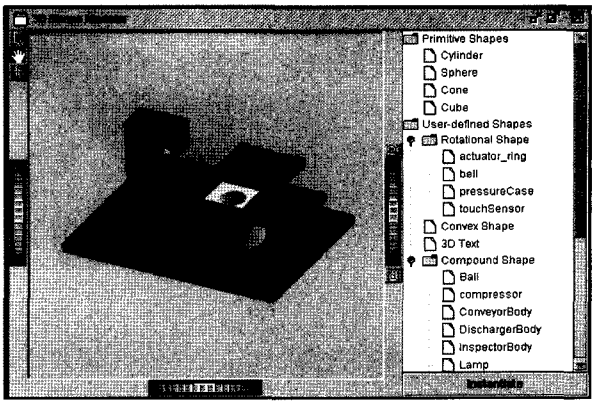

그림 9 턴테이블 객체의 3차원 형태 명세

3.2 가상 테스트베드 명세

가상 테스트베드는 각 환경 객체의 행위 및 기능 명세 외에도, 형태 명세를 필요로 한다.

그림 9는 턴테이블 객체의 3차원 형태 명세를 보인다. 3차원 형태는 형태 모델링 도구를 이용하여 쉽게 구성되며, 이들 사이의 움직임 의존성, 공간적 관계에 대해 명세하여 형태 명세를 정의하고, 행위 명세에서는 상태에 따른 객체의 동적 움직임이나 형태의 변형을 기술하며, 기능 명세에서는 동적 움직임을 위해 필요한 계산을 수행하게 된다.

이런 환경 객체들을 구성하여 최종적으로 가상 테스트베드로서 사용될 공기압 시스템을 완성한다.

3.3 자동 코드 생성

제어 소프트웨어와 가상 테스트베드의 명세가 완성되면, 이들 사이의 이벤트나 데이터를 대응시켜 서로 상호작용할 수 있게 I/O 대응을 시키고, 3가지 모델로부터 타겟 언어로 소스 코드를 생성시킨다.

3.4 시뮬레이션

3.3에서 코드가 생성되면, 시뮬레이션 환경에서 생

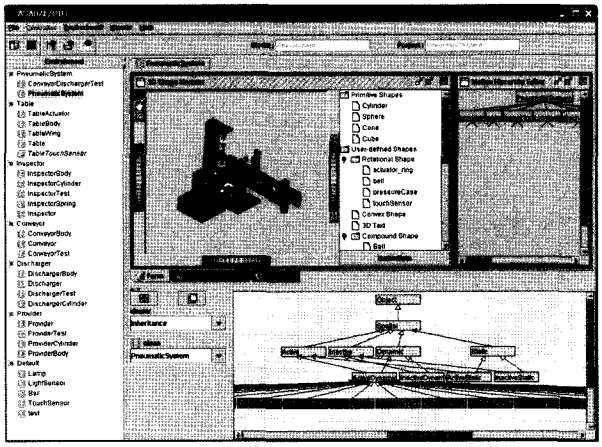

그림 10 공기압 시스템의 가상 테스트베드 구성

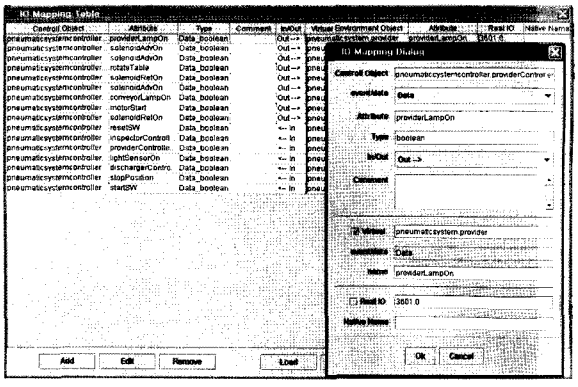

그림11 I/O 대응 화면

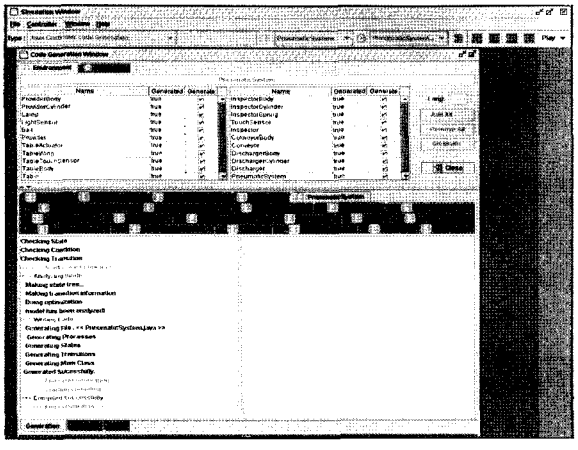

그림 12 ASADAL/OBJ의 코드 생성기

성된 코드를 실행시키면서, 가상 테스트베드를 통해 시뮬레이션 결과를 가시화해 줌으로써, 제어 소프트웨어 명세에 대한 검증을 실시한다.

공기압 시스템의 경우 최종 시스템을 완성하기 전까지 각 지역 제어기와 환경 객체를 명세하여 단계적으로 시뮬레이션을 통한 검증을 실시하였고, 최종 단계에서는 지역 제어기 간의 동기화가 잘 되는지에 대

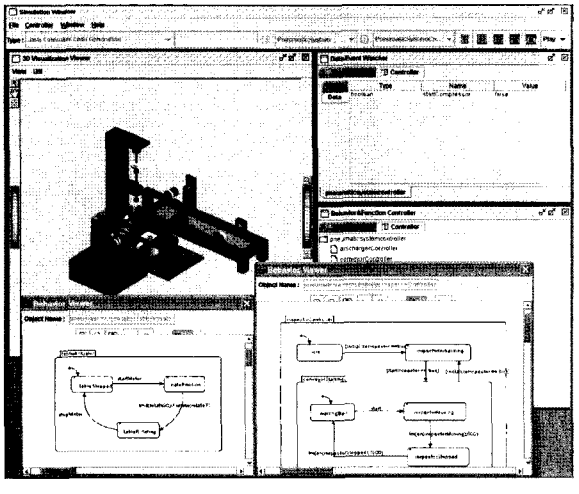

그림 13 공기압 시스템의 시뮬레이션 결과

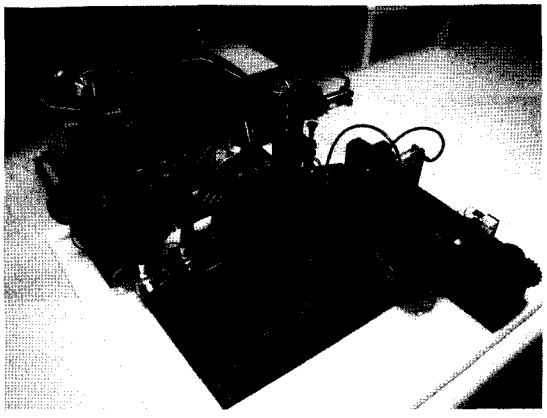

그림 14 실제 시스템 적용 결과

해서만 검사하여 제어 소프트웨어의 검증을 완료하였고, 이를 실제 시스템에 적용하여 잘 작동하는 것을 확인할 수 있었다.

4. 결 론

ASADAL/OBJ는 소프트웨어 개발 시 빠른 프로토타이핑과 정형 검증 및 3D 시뮬레이션을 통한 직관적인 테스트 등을 장점으로 하며, ASADAL이라는 방법론 및 개발 환경에 통합되어 있다. Statechart 와 DFD 등과 같은 시스템의 행위 명세에 많이 사용되는 언어를 이용하여 시스템을 명세해 나가는 과정에서, ASADAL/OBJ와 같은 CASE 도구를 이용하여 계속적으로 시스템을 테스트해 나가고 사용자의 피드백을 얻는다면, 정확하고 요구사항에 부합하는 소프트웨어를 빠른 시간에 개발하는 것이 가능해 질 것이다.

ASADAL/OBJ는 공기압 시스템 이외에도 전자 오븐, 프린터, 자동판매기, 자동차 속도 조절 시스템, 인버터 시스템, 분류(Sorting) 시스템 등 여러 내장형 시스템에 성공적으로 적용되었으며 일부는 실제 시스템 개발에 이용되었다. 현재는 FORM(Feature Oriented Reuse Method)[10]이라고 불리는 재사용 방법론과 연결하여 사용하는 방법에 대해 연구되고 있으며. 이로써 ASADAL은 영역 분석, 요구 사항 분석, 아키텍처 디자인, 재사용 가능한 컴포넌트 설계, 코드 생성 및 유지 보수에 이르는 전 소프트웨어 개발 과정을 포괄하는 개발 방법론 및 도구로서 발전해 나갈 것이다.

참고문헌

[1] Ian Sommerville. Software Engineering, Addison-Wesley, 2001

[2] Stephan Romahn, Annette Kaster. "Creating intelligent user interfaces using prototyping and knowledge based support technologies: the rapid prototyping

tool MacEMSIG," Proceedings of the international workshop on Intelligent user interfaces, 1993, Pages 259-262

[3] Jonathan Knudsen, Java 2D Graphics, O'Reilly, 1999

[4] Ken Arnold, James Gosling. The Java Programming Language, Javasoft, 1996

[5] Kyo C. Kang, et. al., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Engineering, 5, 143-168(1998).

[6] Kyo C. Kang, Kwan W. Lee, Ji Y. Lee, Gerard J. Kim, and Hye J. Kim, "Refinement and Validation of Software Requirements using Incremental Simulation," IEICE(The Institute of Electronics, Information and Communication Engineering) Trans. on Information and Systems, Vol. E81-D, No. 2, pp. 171-182, February 1998.

[7] K. C. Kang, K. W. Lee, J. Y. Lee, and G. J. Kim, "ASADAL/SIM: An Incremental Multi-level Simulation and Analysis Tool for Real-Time Software Specifications," Software: Practice and Experience, Vol. 28, Issue 4, pp. 445-462, April 1998

[8] Kwang I. Ko, Kyo C. Kang, "ASADAL/PROVER: A Toolset for Verifying Temporal Properties of Real-Time System Specifications in Statechart, "The Institute of Electronics, Information and Communication Engineering(IEICE) Transactions on Information and Systems, Vol. E82-D, No. 2, 1999, pp. 398-411.

[9] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson, "Feature-Oriented Domain Analysis(FODA) Feasibility Study", Technical Report CMU/SEI-90-TR-21, Pittsburgh, Pa., Software Engineering Institute, Cargegie Mellon University, 1990.

[10] K. Kang, S. Kim, J. Lee, K. Kim, E. Shin and M. Huh, FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, Vol.5, 143-168, 1998.

[11] Statemate 4.5 User Reference Manual, I-Logix Inc., Burlington, MA, 1992.

[12] Statemate MAGNUM(ver. 1.2) User Reference Manual, I-Logix Inc., 1997.

[13] http://selab.postech.ac.kr/realtime/public_html/index.html

안 미 영

2003 숭실대학교 기초과학 졸업(학사)
2004~2006 포항공과대학교 정보통신연구소 연구원
2007~현재 포항공과대학교 정보통신대학원 석사과정
관심분야 : 소프트웨어 재사용, 제품 라인 공학
E-mail : myahn@postech.ac.kr

김 혜 정

1997 포항공과대학교 컴퓨터공학과 졸업(학사)
1999 포항공과대학교 컴퓨터공학과 졸업(석사)
2000 포항공과대학교 정보통신연구소 연구원
2001~2003 일본 (주)SRA-KTL 연구원
2004~현재 포항공과대학교 정보통신연구소 연구원
관심분야 : 소프트웨어 공학, 실시간 내장형 시스템, 소프트웨어 재사용, 제품 라인 공학
E-mail : unicorn@postech.ac.kr

강 교 철

1973 고려대학교 통계학과 졸업(학사)
1974 콜로라도 대학교 산업공학 졸업(석사)
1982 미시간대학교 소프트웨어공학 졸업(박사)
1982~1984 미시간대학교 Visiting Professor
1984~1985 (미국)벨 통신 연구소 Technical Staff
1985~1987 At&T 벨 연구소 Technical Staff
1987~1992 카네기멜론 대학교 Project Leader
1992~현재 포항공과대학교 교수
2000~2001 카네기멜론대학교 Visiting Scientist
관심분야 : 소프트웨어 공학, 실시간 내장형 시스템, 소프트웨어 재사용, 제품 라인 공학
E-mail : kck@postech.ac.kr