2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

CASE 도구를 이용한 내장형 시스템 개발 프로세스 구축

오광근*, 한광수*, 오기훈*, 권용관*, 문전일*, 임계영*, 강교철**
*LG 산전연구소
**포항공과대학교
e-mail : kkoh@lgis.com

A Development for Embedded Real Time System with CASE Tool

KwangKeun Oh*, KwangSoo Han*, KiHoon Oh*, YoungKwan Kwon*,
JeonIL Moon*, KyeYoung Lim*, Kyo Chul Kang**
LG Industrial Systems R&D Center* POSTECH SE Lab**

요 약

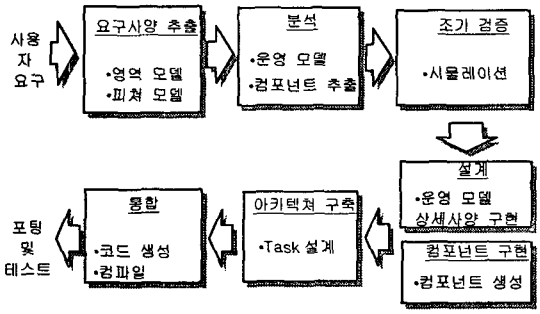
내장형 시스템 개발에 있어서도 복잡해져가는 소프트웨어 구조에 대한 관리 필요성과 개발 프로세스 및 개발된 제품에 대한 신뢰성 요구가 날로 증가되고 있다. 이런 당면 문제를 극복하고자 LG 산전에서는 내장형 시스템 개발에 적합한 소프트웨어 개발 프로세스를 구축 하였다. 본 연구를 통해 인버터 시스템 개발에 구축한 프로세스가 성공적으로 적용되었음을 확인할 수 있었다. 또한 CASE 도구를 통한 시스템 개발에 있어서도 내장형 시스템 개발에 가장 중요한 성능 사양인 시간제약조건의 만족 여부도 확인할 수 있었다.

1. 서론

감시제어 시스템등 크고 복잡한 산업용 시스템의 개발에 있어서는 체계적인 소프트웨어 개발 프로세스 및 CASE 도구에의한 성공적인 개발 사례가 많이 소개되고 있다. 그러나 상대적으로 규모가 작은 내장형 시스템 제품은 하드웨어 중심의 시스템 개발 특성과 시스템 제품군에 비해 상대적으로 작은 개발 규모때문에 소프트웨어 개발에 대한 투자 및 연구가 소홀히 진행되어왔다. 하지만 근래의 내장형 시스템 제품군에서도 제품 경쟁에따른 life cycle 은 점차 줄어들고 있으며, 고기능 사양 요구로 제품 기능 범위가 점차 확대 다양해지고 있는 현실이다. 이런 제품 개발 현실에따라 기존의 소프트웨어 개발방법론으로는 시장의 요구를 만족시킬수가 없게 되었다. 다양한 기능사양의 추가와 개발 life cycle 단축등의 요구를 만족하기 위해서는 많은 소프트웨어 산출물들이 재사용이 용이한 구조로 관리 되어져야하며, 점차 복잡해지는 시스템 사양을 만족시키기 위해서도 정형화된 표준 프로세스에의한 개발 및 CASE 도구지원이 필요하게 되었다. 또한 기존의 내장형 시스템 개발이 구현중심으로 진행 되므로써, 체계적인 분석과 설계 모델이 없기에 새로운 시스템을 개발하고자 할때에는 기존 개발자들의 경험과 구현된 코드에 의존적일 수 밖에 없었다. 이런 기존의 개발 환경에서는 신규 소프트웨어 개발에 대한 지속적인 품질보증이 힘들어졌으며, 신규개발 인력에 대한 교육 및 내용 전달이 코드 중심의 제한적일 수 밖에 없었다. 이런 필요성에의해 LG 산전에서는 포항공대와 공동으로 내장형 시스템 개발에 적합한 개발 프로세스 및 CASE 도구를 연구 개발 하였다. 본 논문에서는 이 연구의 일환으로 LG 산전 제품인 인버터 소프트웨어 개발에 내장형 시스템 개발 프로세스 및 CASE 도구를 적용한 사례를 소개한다.

2. 내장형 시스템 개발 프로세스 모델링

표준 프로세스 모델은 포항공대의 실시간 시스템 개발 방법론인 Asadal 방법론을 기반으로 구축하였다.[1] 그림1은 내장형 시스템 개발을 위한 상세 개발 프로세스를 나타낸 것이다. 개발 프로세스는 영역 모델(Context Diagram)과 피쳐 모델 을 통해 개발 사용자로부터 개발 요구 사양을 추출 하는것으로 시작된다. 요구사양 추출후 분석 단계는 먼저 시스템의 기능 모델 및 행위모델을 Data Flow Diagram과 State Chart를 통해 구현 하는 운영모델 단계와 피쳐 모델로부터 구현 단위의 재사용이 가능한 컴포넌트를 추출하는 컴포넌트 모델 단계로 나누어 진행된다. 분석단계 후 시뮬레이션 단계를

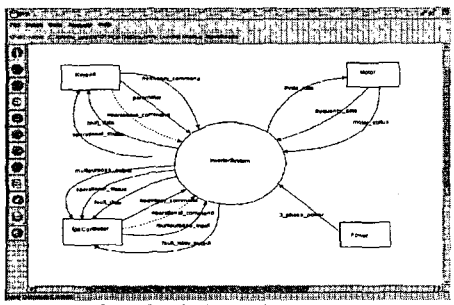


[그림 1] 내장형 시스템 개발 프로세스

통해 운영 모델 논리를 조기에 검증하고, 분석 단계에서 발견할 수 있는 오류를 추출 제거한다. 시뮬레이션이 성공적으로 수행되면 분석단계의 운영 모델을 구현 환경을 고려한 설계단계의 모델로 변환한다.또한 분석단계에서 추출한 컴포넌트를 재사용 가능한 모듈로 구현한다.설계단계후 아키텍처 구축 단계에서는 시스템에 장착될 Task설계를 하고, 각각의 Task에 운영모델의 관련 프로세스와 State Chart를 할당하고, 초기화 컴포넌트도 할당한다. 통합단계에서는 먼저 CASE 도구를 통해 각 Task에 해당하는 코드를 자동 생성한다.이렇게 생성된 코드는 콤포넌트와 통합 컴파일 후 시스템에 장착된다. 장착된 후 테스트를 통해 시스템의 성능 및 기능 사양을 만족하느지 여부가 확인되면, 한 Release에 대한 개발 life cycle이 완료된다. 이 모든 개발 단계는 Asadal이라는 CASE도구를 통해 관리되어진다. Asadal도구는 내장형 표준 프로세스 개발을 지원하기 위해 개발된 CASE도구이다.[2]

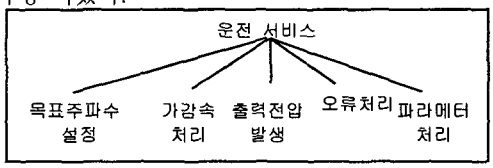
3. 영역분석을 통한 운영 및 컴포넌트 모델 구축

이렇게 구축된 내장형 시스템 개발 프로세스는 LG 산전의 내장형 시스템 제품인 인버터 시스템을 목표 시스템으로 그 적용 가능성을 확인하였다. 먼저 시스템 개발 영역을 결정하기위해 영역 분석을 통해 시스템의 이해를 도모하였다.



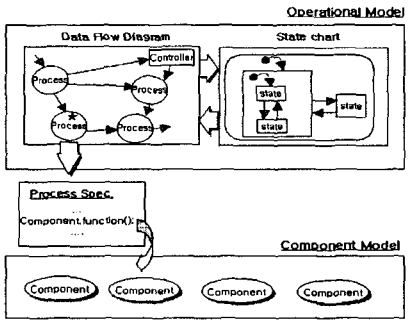
[그림 2] 인버터 영역 모델(Context Diagram)

그림 2 는 영역모델을 통해 인버터 개발 영역을 나타낸것이다. 영역모델은 인버터 시스템과 외부기기들과의 상호작용을 도출함으로써 구현할 인버터 시스템의 입출력 정보를 조기에 도출하는데 그 목적이 있다. 영역 모델과 더불어 피쳐모델은 영역 분석의 중요한 역할을 수행하다. 영역 모델이 개발 영역등 시스템 외부적인 요소를 구분짓는데 사용된다면, 피쳐모델은 개발하고자 하는 시스템의 내부 구성요소를 개념적인 컴포넌트로 구분지어 관리하는데 그 목적이 있다. 피쳐란 사용자나 개발자에게 보이는 시스템의 구분되는 특징을 말한다. 시스템 특징들은 서비스,운영기능,일반기술,특성화 기술들로 구분되어 추출된다.[1] 피쳐 모델은 4 가지 계층으로 구분되는데 사용자관점에서 영역내의 공통점과 차이점을 도출한 특성계층인 CA 와 영역내의 운영환경을 나타내는 OE, 영역내에서만 사용되는 이론이나 규칙등을 나타내는 DF 그리고 일반적인 기술인 IT 로 나누어진다. 서비스 피쳐인 운전 서비스를 중심으로 총 77 개의 피쳐로 인버터의 영역을 특성 지었다.



[그림 3] 인버터 특성 계층(CA Layer)

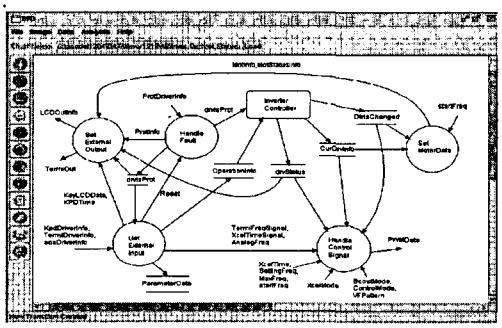
그림 3 은 인버터 영역분석인 피쳐모델을 통해 추출한 특성 계층을 나타내고 있다. 이렇게 추출한 피쳐는 그 특성에 따른 운영모델 및 컴포넌트 모델로 구현된다. 시스템이 가지고 있는 특징을 트리 구조로 구조화 한 것이 피쳐모델이며, 그로부터 재사용이 가능한 요소들을 추출하여 구현한 것이 컴포넌트 모델이다. 피쳐모델의 특징들이 어떻게 운영되는가에 초점을 맞추어 기술된 것을 운영모델이라고 한다. 인버터 운영모델링은 영역모델링에 나타난 경계를 추상화한 상태에서 하향식 접근 방법에 의해 세분화 한 것을 Data Flow Diagram 과 StateChart 로 나타내었다. 피쳐가 개념적인 재사용 단위라면 컴포넌트는 구현 수준의 재사용 단위이다. 즉,피쳐를 통해 재사용이 가능하도록 구현된 것이 컴포넌트이다.



[그림 4] 운영모델 과 컴포넌트 모델과의 관계

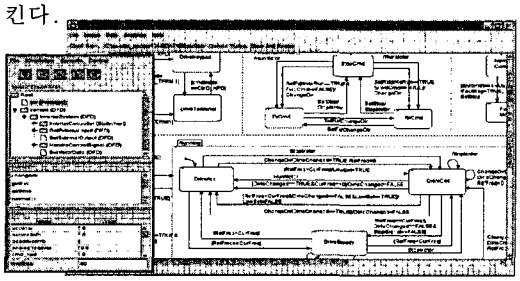
그림 4 는 도메인 분석후 도출된 운영모델과 컴포넌트 모델사이의 관계를 나타낸 것이다. 그림에서 보면 DataFlow Diagram 의 프로세스 사양서의 함수 호출을

통해 컴포넌트 모델로부터 구현된 컴포넌트 라이브러리 함수를 호출하는 것을 보여준다[3]. 호출된 컴포넌트의 함수는 그 결과값을 프로세스 사양서에 돌려주며. 프로세스 사양서는 그 결과값을 운영모델 변수로 할당하여 계산 및 출력에 사용한다. 이렇게 운영모델과 컴포넌트모델과의 관계가 프로세스사양서를 통해 연결되어져 있다. 운영모델은 기능 모델과 행위 모델로 나누어 관리되어진다. 인버터 시스템 기능 모델은 영역모델(Context Diagram)에 묘사된 Inverter System 을 데이터 입력, 데이터 출력, 모터처리, 오류처리, 속도제어처리의 5 개의 프로세스로 분할하고, 이 5 개의 프로세스의 상세 역할을 최종 23 개의 말단 프로세스로 분할 구현하였다. 그림 5 는 최상위 프로세스인 Inverter System 과 상위 5 개 프로세스를 나타내고 있다.



[그림 5] Inverter System Data Flow Diagram

인버터의 행위특성들은 관심분야에 따라 운전 상태, 주파수 모드상태, 가감속 시간모드 상태를 나타내는 StateChart들로 모델링 하였다[4]. 이렇게 행위모델이 완성되면, Data Flow Diagram의 관련 프로세스에 따라 statechart를 할당한다. 데이터흐름도에 할당된 statechart는 state사양서를 통해, 해당 Data Flow Diagram의 프로세스를 활성화 시킨다.

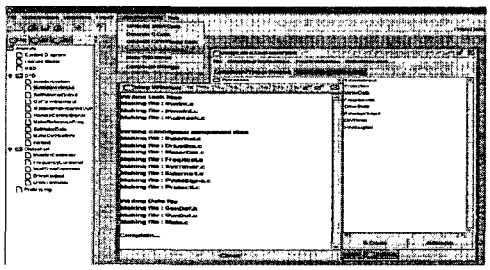


[그림 6] 시뮬레이션 과정

운영모델이 완성되면 운영모델의 논리 구조를 검증하기 위해 시뮬레이션을 수행한다. 그림 6 은 시뮬레이션 도구를 통해 상위 StateChart 인 InverterController 를 시뮬레이션하는 과정을 보여준다. 시뮬레이션 수행은 입출력 데이터 값 변화에 따라 이루어지며, Data Flow Diagram 에서의 프로세스 호출과 StateChart 의 상태변화로 확인 되어진다. 시뮬레이션 논리검증 만족 여부를 확인하기위해 사용자 요구사항으로부터 시스템 수준의 테스트 케이스를 추출하였다. 테스트 케이스는 사용자가 시스템을 이용하는 시나리오를 Use Case 라는 모델링 방법을 통해 추출하였다[5].

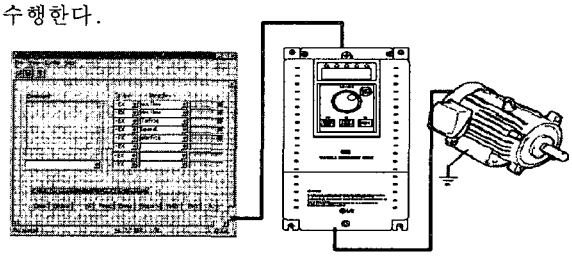
4. 아키텍쳐 모델 및 코드 발생

운영모델에서 구현한 논리적 구조를 실질적으로 어떻게 배치 할 것인지에 초점을 맞춘 것이 아키텍쳐 모델이다. 먼저 아키텍쳐 모델링은 시스템 구동시 독립적으로 수행될 Task 를 실시간 제약 조건등의 성능을 만족시키는 업무중심으로 Task 를 설정하였다. 설정후 각 Task 에 관련 운영모델 및 컴포넌트 모델의 요소들을 수행순서를 고려하여 배치하므로써 아키텍쳐 모델은 완성된다. 코드 발생을 하기위해서는 먼저 운영모델에서 생성되고 사용되는 모든 데이터를 정의하고, 초기화하여야 한다. Task 설계와 데이터 파일 및 초기화 파일 등록이 끝나면 Asadal 도구의 코드발생기를 통해 코드를 발생한다. Asadal 도구는 인버터전용의 코드발생기를 가지고 있으며, 그림 7 과 같이 표준 C 를 지원하는 코드를 발생시킨다.



[그림7] 코드 발생기를 통한 운영모델 및 Task 관련 코드생성

코드발생기를 통해 발생된 코드는 Task관련 파일들과 운영 모델 관련 파일들이다. 발생된 코드와 운영모델에서 호출하는 컴포넌트 파일들을 결합하여 컴파일작업을 수행한다. 컴파일 작업 후, 그림8과 같이 Downloader 프로그램으로 인버터에 포팅작업을 수행한다.



[그림 8] Downloader 프로그램을 통해 인버터 SW 포팅

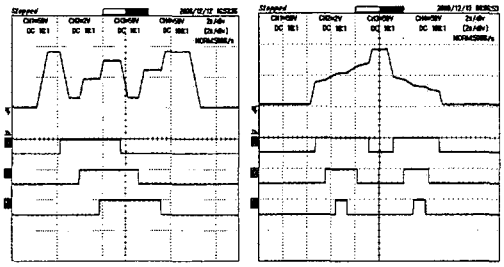
5. 테스트

포팅이 완료된 후, 개발사양 단계에서 작성한 테스트 기준서를 가지고 테스트를 수행한다. 테스트 기준서는 시스템의 기능 및 성능 사양의 만족여부를 검증 할

수 있게 표 1 과 같은 24 개의 체크리스트로 되어 있다.

	항목	내용
1	다기능 입력 변경	다기능 입력선택을 다단 가감속으로 변경하고 입력시 키패드 표시 없음
2	다단속 가감속전	운전 중 P1, P2, P3 입력하면서 설정된 가감속 시간대로 운전하는지 확인

[표 1] 다단 가감속 테스트



[그림 9] 다단속 운전 및 가감속 운전 테스트 결과

그림 9 는 표 1 ' 다단 가감속 테스트' 의 결과로써 실제 입력 및 출력 주파수를 오실로스코프로 출력하여 나타낸 것이다. 이렇게 24 개의 항목에 대한 점검 결과,모든 항목에대해 테스트가 성공적으로 수행됨을 알 수 있었다.

6. 성능 분석

내장형 시스템에 있어 성능 평가는 매우 중요하다. 대부분의 내장형 시스템들이 실시간성을 요구 하는바, 개발된 제품의 성능사양 만족여부는 개발된 제품의 성공여부를 판단하는 중요 자료로 활용된다. 인버터 시스템인 경우, 기능 외적인 요구사양으로 시스템 메모리 제약에 따른 파일크기 제한과 Background 프로세스 수행시간이 기존 개발 사양을 만족해야하는 조건이 있었다. 크게 이 두가지 관점에서 CASE도구에 따른 소프트웨어 개발프로세스 적용과 기존 소프트웨어 개발과의 성능 분석을 표2을 통해 나타내었다.

항 목	기존 구조	CASE 도구 적용
구성 파일 수	.H(37 개)	.H (34 개)
	.C (45 개)	.C(37 개)
파일 평균 크기	.H(1.53kB)	.H(1.66kB)
	.C(3.71kB)	.C(4.38kB)
Binary 크기	20kWord	21kWord
변수	0.62kWord	0.59kWord
Text	16.32kWord	17.02kWord
Const	3.5kWord	3.43kWord
Background Task 수행시간	0.6ms	0.6ms

[표 2] 개발 프로세스 성능 분석표

표2에 나타난 파일수를 살펴보면 새로 개발된 소프트웨어의 파일 수가 기존에 개발된 파일수보다 .H파일3개와 . C파일 7개등 총 11개가 적어진 것을 알수 있다. 이것은 새로 개발된 소프트웨어의 컴포넌트 설계가 그만큼 관련성있는 사항들을 잘 묶어 관리 되어지는 것을 보여준다. 기존 코드와 비교하여 볼 때 코드발생기를 통한 코드크기가 약간 증가된 것을 알 수 있었으나, 시스템 포팅에 전혀 영향을 미치지 않았다. 가장중요한 소프트웨어 수행시간을 보면 Background Task 수행시간은 6msec으로 기존 소프트웨어와 동일한 수행시간을 보여줌을 확인 할 수 있었다.

7. 결론 및 향후 연구

인버터 시스템과 같은 실시간 내장형 시스템인 경우, 그 특성상 파일크기와 실행시간에 제약이 따른다. 아무리 정형화된 툴을 사용하여 시스템을 개발하였다 하여도 개발된 소프트웨어의 성능이 시스템 요구사항을 만족 못 시킨다면 결국 사용할 수 없기 때문이다. 그 동한 이러한 이유로 인해 내장형 시스템에 CASE 도구를 이용한 개발이 주저되어 왔던 바이다[6]. 이런 환경적인 요인을 감안하여 시스템 성능사양을 만족할만한 수준의 CASE 도구 및 표준 프로세스를 포항공대와 공동으로 개발하였다. 본 연구를 통해 실시간 내장형 시스템에서도 CASE 도구를 이용한 정형적 개발 프로세스의 도입이 성공적으로 수행됨을 확인 하였다. 또한 시각적 모델링 방법론을 사용함으로써 소프트웨어 복잡도를 감소시켰으며, 시스템 이해도를 향상 시킬 수 있었다. 향후 효율적인 소프트웨어 관리를 위해서는 재사용 컴포넌트에 대한 설계 및 관리 방안에 대한 연구와 도메인에 적합한 아키텍쳐 구축에 대한 연구가 필요하다.

참고문헌

[1] K.C Kang. S. Kim, J.Lee, Kim, G.J.Kim and E. Shin, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures", Annals of Software Engineering, Vol 5, pp.143-148, 1998

[2] 강교철, "산전 소프트웨어 개발 자동화 환경 구축", LG 산전 위탁 연구결과 최종 보고서,2000

[3] Paul T. Ward, Stephen J. Mellor,"Structured Development for Real-Time Systems" Yourdon press Prentice Hall Inc, 1985

[4] David Harel, "StateChart : A VISUAL FORMALISM FOR COMPLEX SYSTEMS, Science of Computer Programming ,Elsevier Science Publishers, pp 231-274, 1987

[5] Booch,G.,J.Rumbaugh, and I. Jacobson, " The Unified Modeling Language User Guide",Addison Wesley, 1999

[6] 전태웅, "실시간 소프트웨어 프로세스 정형화 연구",LG 산전 위탁 연구결과 최종보고서, 1998