천이 사건 순서의 표현과 정형화

김 정 술† · 강 병 욱††

요 약

이 논문에서 우리는 OARTS(Object based Approach for Real-Time Systems)를 위한 시나리오를 표현하는 방법과 명세언어 및 확인기법을 제안한다. 이 방법은 지금까지 방법론 차원의 시나리오를 다루지 않았기 때문에 일반적인 모델링기법(사건추적도)에 명세언어와 확인방법을 포함한다. 본 논문은 객체에 기초한 통신 인터페이스인 외부 모듈 천이의 동기를 중심으로 내부 액션 천이와 외부 사건들의 열의 표현에 중점을 둔다. 실제의 예를 통하여 제안된 방법이 분석 단계의 요구사항들을 잘 반영하였으며 개념적인 확인기를 통하여 그 표현의 정당성을 확인할 수 있었다. 또한 이 방법은 일반적인 실시간 시나리오를 표현하기 위한 분석 도구로도 이용될 수 있을 것이다.

Specification and Formalization of Transition Event Sequence

Jung Sool Kim† · Byung Wook Kang††

ABSTRACT

In this paper, we propose a scenario representing method, a specification language, and a verification technique for OARTS(Object based Approach for Real-Time Systems). As well as the general modeling method(event trace diagram), this study includes a specification language and a verification technique because there was no study about methodological level technique for scenario development as yet.

Centering around the synchronization problem of transition of external modules which are the communication interfaces based on the objects, we lay stress on the representation of sequence of external events and internal action transitions.

From the results of practical experiences, it has been ascertained that the proposed method reflect well the requirements in the analysis step, and its validity of the representation has been identified by a conceptual verifier. We support that it can serve as an analyzing tool for representing a general real-time scenarios also.

1. 서 론

최근, 다양해지는 시스템 분석, 설계방법의 추세를 보면, 크게 실시간 시스템과 비실시간 시스템으로 분류되어진다. 실시간 시스템은 시간과 병행성의 요소들이 주요 성분들로 구성되지만, 비실시간 시스템은 그렇지가 않다. 이 시스템은 일반적인 경영정보시스템의 차원에서 다루어지는 순서적인 프로세스들의 성분으로 이루어진다. 반면에 실 시간적 성분을 구성하는 요소들은 시스템의 개발 초기단계에서부터 정확하고 완전한 시스템의 분석이 요구된다. 이러한 성분을 구성하는 요소들 중의 하나로 시나리오의 개발이 필요하다. 이 시나리오는 시스템의 외부환경에서 발생되는 사건들을 중심으로 개발되어 지는데, 특히 상태 천이도와 함께 시스템의

†정 회 원 : 영남대학교 컴퓨터공학과 박사과정
††종신회원 : 영남대학교 컴퓨터공학과 교수
논문접수 : 1997년 8월 13일, 심사완료 : 1998년 3월 10일

초기 분석을 위한 도구로 이용되어왔다. 시나리오는 시스템의 외부사건으로부터 내부사건을 유발시켜 필요한 시스템의 행위들을 취하게 하는 것으로, 이러한 사건들을 목록으로 작성하여 시스템의 행동들을 결정하는데 중요한 역할을 수행해 왔다. RTSA[11](Real-Time Structured Analysis)에서는 이런 사건들의 열이 비정형적인 목록으로 작성되고, 자료흐름도와 함께 시스템의 분석에 이용되고 있으며, DARTS[2,3,4]방법은 자료흐름도나 제어 흐름도상에 순서번호를 기입하여 사용하고 있다. 객체모델링의 경우는 Rumbaugh[6]방법에서 처음, 사건 추적도라는 다이어그램을 제시하여 전체 사건들의 흐름 순서관계를 표현하고 있다. 현재 대부분 객체 모델링의 경우는 Rumbaugh방법의 사건 추적도를 사용하고 있다. 본 논문도 객체모델링을 위해 이 사건 추적도에 기반을 둔다. 그러나, 본 논문에서는 사건 추적도가 천이 사건 순서도로 더욱 확장되며, 시나리오의 개발을 위해 시나리오 명세언어를 제공하며, 모델링과 명세언어를 기반으로 검증할 수 있는 확인기를 통하여, 작성된 시나리오가 정당한지를 확인할 수 있도록 한다. 특히 본 논문은 OARTS[14]라는 실시간 시스템 모형화 방법(2장에서 설명된다.)을 위해 사용되는데, 이 방법은 기존의 일반객체를 활성(Active)객체로, 객체간의 메시지 인터페이스인 통신 모듈을 수동(Passive)객체로 인식하여 시스템의 행위들을 묘사하는 방법이다. 그래서 이러한 수동객체들의 천이관계를 동기 순서화하며(내부 액션천이는 물론이고), 외부사건 및 내부사건들의 열을, 시간을 중심으로한 축을 따라 표현하며 명세한다. 아울러, 정형화된 표현을 위해서는 시제논리[7]를 이용한다. 본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를, 3장에서는 천이사건 순서도와 명세언어를 제공하고, 4장에서는 실 사례를 통하여 적용하며, 표현 방법이 정당한가를 확인한다. 끝으로 5장에서 결론을 내린다.

2. 관련 연구

2.1 시나리오 표현방법

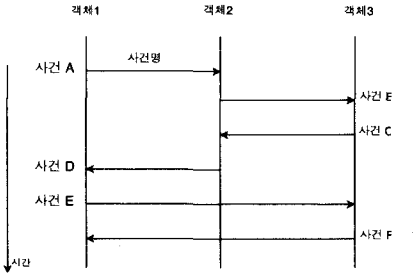
일반적으로 행위분석(Behavior Analysis)의 도구로 사용되는 시나리오는 사용자와 분석자간의 원활한 대화를 위한 통신의 수단으로 이용되어 왔다. 즉, 시나리오는 사용자와 시스템간의 상호 작용(Interaction)을 사용자가 알기 쉽도록 순서대로 기술한 것이다. 시나리오의 일반적인 정의는 주어진 조건에 대한 대상 시스템의 외부 동작을, 사용 경우(Case)별로 기술한 것으로 완성된 소프트웨어가 외부적으로 어떤 기능을 가져야 하는가를 입력 조건에 따른 반응으로 나타낸 것이다[13]. 이것이 사건 순서도(Event Sequence Diagram)[2]이다. 일반적인 시나리오 표현의 방법으로는 첫째, Agent와 Action의 연속으로 표현하는 방법[1]과 둘째, 다이어그램으로 표현하는 방법[4,5]과 그리고, 셋째, Tree로 표현하는 방법[10]이 있는데 그 장단점은 다음과 같다.

① Agent와 Action의 연속으로 표현하는 방법

이 방법은 〈표1〉과 같이 특정의 시나리오 집합들을 사용자의 관점에 따라 분류하여 사용자가 그 시스템을 사용하는 동작을 순서대로 표현하는 방법으로 RTSA에서 주로 사용되었다. 이 방법은 비정상적인 경우를 포함하여 시스템에서 발생 가능한 모든 경우를 나타낼 수 있지만 비정형화되게 기술되므로 혼란스럽고 전체구조가 보이지 않는 단점이 있다.

〈표 1〉 Agent와 Action의 표현
〈Table1〉 Representation of Agent and Action

시나리오 #1	정상수행시
Agent	Action
1. Caller	수화기를 든다
2. 전화국	DialTone을 Caller에게 보낸다
3. Caller	7자리의 전화번호를 누른다
4. 전화국	Callee에게 Ring을 발생시킨다. 동시에 Caller에게 RinbackTone을 발생시킨다.
5. Callee	수화기를 들면 연결이 설정되고 Caller와 대화를 한다.



(그림 1) 사건 추적도
(Fig. 1) Event Trace Diagram

② 다이어그램으로 표현하는 방법

이 방법은 (그림1)과 같이 각 객체에 대해 수직선은 시간 축으로 수평선은 메시지를 의미한다. 대부분의 객체 모델링에 사용되며 통신분야에 많이 이용되는데 장점은 시간 축을 따라 메시지의 전달순서와 방향 등이 나타나고 전체 구조가 한꺼번에 파악되므로 사건 분석이 쉬울 뿐만 아니라 프로그램의 실행시간의 동작을 이해하는데 도움을 준다. 이를 사건 추적도[6]라고 한다.

③ Tree로 표현하는 방법

이 방법은 (그림2)와 같이 사용자의 관점을 트리(Tree)구조로 표현[10]한 것이다. 루트(Root)노드는 시스템의 초기 상태이고 초기 상태에서 발생 가능한 사건에 대해서 루트 노드로부터 가지를 만들어 나간다. 각 노드는 시스템의 상태를 나타낸다. 이렇게 전체 사건들을 시스템의 상태와 관련하여 작성한 후 시스템은 초기 상태로 간다. 이 방법의 장점은 형식적인 표현 형태이고 결국은 상태천이도의 모습이 되어버린다.

④ CFD(Control Flow Diagram)상에 표현하는 방법

이 방법은 (그림3)과 같이 DARTS에서 사용되는 것으로 초기 DFD나 CFD가 작성되었을 때 그 위에다 사건이 발생된 순서대로 번호를 기입하는 방식[2,4]이다. 이 방법도 비정형화되게 표기되고 사건 간의 변화 모습들이 제대로 보이지 않는다.

위에서 제시된 표기법들은 너무 비정형화되었고, 또한 전체 사건의 열이 한 눈에 보이지 않아 정확하게 사건 관계들을 분석하기가 어렵다. 더구나 동시에 발생하는 사건을 위한 표현과 천이 조건 등을 기술하지 못하는 한계를 가지고 있다. 그래서 본 논문에서는 천이 사건 순서의 표현을 위해서 다이어그램 표기법에 기초하여 정형화하며, 병행사건이나, 천이 조건 등도 기술 가능하도록 확장한다. 또한 지금까지의 시나리오는 전부 수동으로(Manually) 작성되었다. 그래서 본 논문에서 제시하는 방법은 향후 시나리오 표현의 자동화를 위한 기본 구조로 사용될 수 있을 것이다.

2.2 시나리오의 적용단계

일반적으로, 시나리오는 개발단계, 실행단계, 완료단계의 세 가지 단계[4]로 구성된다.

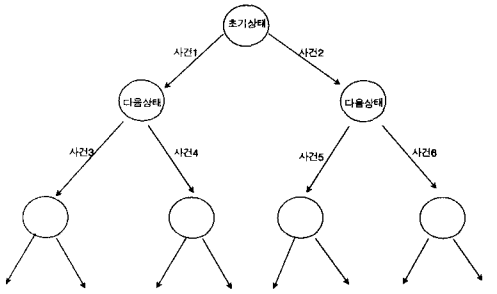
① 시나리오의 개발단계는 각 상태 천이도와 외부 사건들을 사용하여 하나 이상의 시나리오를 개발하는 단계이다.

② 시나리오의 실행 단계는, 각 외부 사건이 순서적으로 실행되는 단계이다.

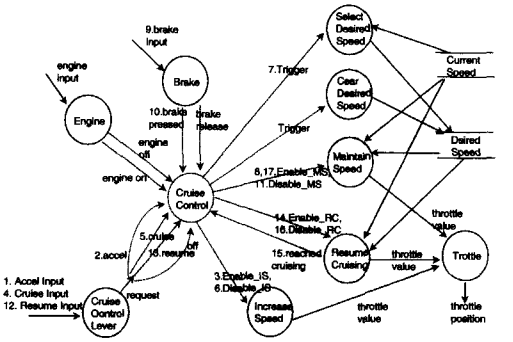
③ 시나리오의 완료 단계는 실행된 시나리오의 모든 상태는 상태 천이도를 중심으로 이루어 진다. 그래서 각 액션(Action)이 수행될 수 있음을 보장하는 단계이다.

지금까지는 이러한 시나리오의 적용단계를 위한 표현으로 주로 사건 추적도와 상태 천이도만을 가지고 이용해 왔다. 그런데 이 두 가지 요소만을 가지고는 사건 및 액션들의 정형성을 보장할 수는 없다. 그래서 본 논문에서는 위의 기본적인 원리에 따라 천이 사건 순서제어를 위해 다음과 같이 구성하고자 한다.

① 문제정의 단계 : 시스템의 외부 사건들의 열과



(그림 2) 트리형태의 표현
(Fig. 2) Tree Representation



(그림 3) CFD상에 표현하는 방법
(Fig. 3) Representation of CFD

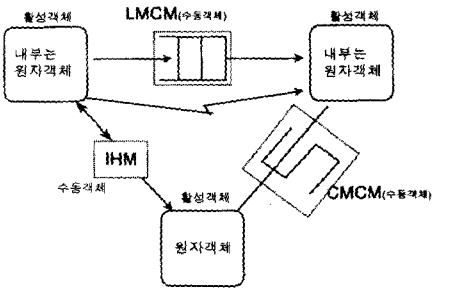
외부천이의 열 그리고, 내부사건과 내부천이의 열을 정의한다(이는 비정형하게 문제를 기술하는 단계이다).

② 모형화 및 명세단계 : 이 단계는 순서적인 시나리오의 실행 단계로서, 각 사건 및 천이들의 동기순서 문제를 정형하게 기술하는 단계이다.

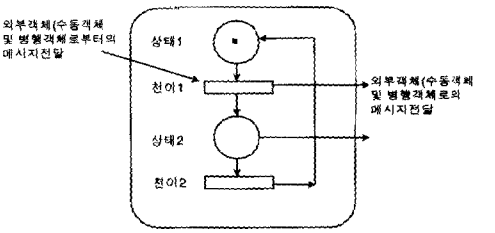
③ 확인 단계 : 이 단계는 실행된 시나리오의 확인(Verify)을 위한 단계[12]로, 각 사건 및 액션들의 수행이 정당함을 보장할 수 있는 단계이다.

2.3 OARTS(Object based Approach for Real-Time Systems)

OARTS는 실시간 시스템을 위한 모형화 방법[14]이다. OARTS는 원래 실시간 시스템 설계 방법인 DARTS를 객체에 기반한 방법으로 변환한 방법에 기초한다. 이 방법에서는 병행 타스크들이 액티브(Active)한 객체로 변환되고, 그 인터페이스(통신모듈) 또한 패시브(Passive)한 수동객체로 변환된다. 여기서의 대상은 활성객체와 수동객체가 된다. 전체시스템은 이러한 객체들로 구성되는데 액티브한 객체는 페트리 넷의 플레이스(Place)로, 수동객체는 트랜지션(Transition)으로 인식된다. 그래서 액티브한 객체는 수동객체가 점화(Fire)됨으로서 메시지 토큰을 수신 객체에 전달한다. 결국 OARTS에서 필요한 분석을 위한 도구로 본 논문이 이용되는데 이러한 천이들의 순서제어와 사건제어를 위한 것이다. 수동객체의 대상이 되는 모듈은 다음과 같다. 강결합 통신모듈(CMCM), 약결합 통신모듈(LMCM), 정보은닉모듈(I.H.M) 및 동기모듈(Synchronization Module)이다[3,4]. 여기서 동기모듈은 그 성질상 활성객체가 된다. 이 모든 모듈들의 내부 행위는 고 수준 페트리 넷(High Level Petri Net)으로 모델링된다. 시나리오는 전체 시스템의 객체를 대상으로 하며, 또한 입력 사건이 중요하기 때문에 이들의 모습을 전부 보여줄 수 있도록 천이모듈 및 천이액션들로 구성된다. 그래서 본 논문에서는 이들의 외부천이를 위한 동기와 내부천이 동기명세가 제공된다. 외부천이 동기는 시나리오를 구성하는 각 객체와의 사건이 내재된 메시지의 흐름관계를 표시하며, 각 구성된 모듈 속의 내부 Queue와 관련된다. 실제로 LMCM과 CMCM의 경우는 두 개의 메시지 큐가 서로 다른 방향으로 하나의 통신모듈을 구성한다. (그림4와 5)를 참고하면 될 것이다. 여기서, 둥근 모서리의 사각형은 활성객체이며, 직사각형(점선으로 표현된)은 수동 객체이다. 그래서, 본 논문에서 표현하고자 하는 부분은 시스템의 외부 뷰(View)를 제공하는 "동기 제한자(Synchronization Constraint)"[8]의 역할을 하는 외부 모듈천이(내부엔 결국 Action 천이이지만)와 시스템의 내부 뷰(View)를 제공하는 역할을 하는 "계약적인 의무(Contractual Obligation)제한자"[8]인 Action천이의 순서(Sequence)를 정의하고자 한다. 본 논문에서는 이를 천이 사건 순서도(Transition Event Sequential Diagram : TESD)라고 부르기로 한다. 결국, 이 천이 사건을 수행하는 객체는 병행객체이며 병행객체 내부의 객체 동기모듈(OSM)[13]인 것이다. 이 OSM은 병행객체의 내부에 존재하는 제어 모듈(C언어의 main 모듈과 같다)이다. 그리고 이 OSM은 각 병행객체에 하나씩 존재하며 신호(사건)를 송신하거나 수신한다. 아울러, 이 천이 사건 순서도를 정형화하게 표현하기 위해서 우리는 실시간 시제 논리[7]를 이용한다. 다시 말하면, 본 논문에서 채택하는 TESD는 일반적인 시각에서의 시나리오가 내, 외부 사건을 중심으로 기술하는 방식을 추구하면서도(부수적으로), 주 시각을



(그림 4) OARTS의 구조
(Fig. 4) ARCH. of OARTS



(그림 5) 원자객체 구조
(Fig. 5) Arch. of Atomic Object

천이모듈 객체의 동기와 일반객체(병행객체 또는 원자객체)의 Action천이의 동기화 문제에 주안점을 둔다는 것이다.

3. 천이 사건 순서도(Transition Event Sequence Diagram)

먼저 이 천이 사건 순서도의 천이 요소에는 동기 제한자의 역할을 하는 외부 모듈천이(즉, 외부천이)와 계약적인 의무의 역할을 하는 Action천이인 내부천이의 두 요소로 나뉘어진다. 메시지의 흐름을 나타내는 요소는 MF(Message Flow) = (발생 객체명, 사건명, 천이명, 도착 객체명)으로 이루어진다. 이 천이 사건 순서도는 천체 시스템의 사건 및 천이와 그 관계를 한눈에 볼 수 있어 편리하게 이용할 수 있다. 현재의 천이 사건 순서도에 표현된 각 외부천이, 내부천이, 사건들은 시제 논리로 표현 가능하다. 먼저 수동 객체의 천이 순서로 POT(Passive Object Trans.)1의 점화 실행 후에 POT2나 POT3이 발생한다면 다음과 같이 기본적인 시제 연산자 만으로도 충분히 기술 가능하다.

$\Box$(POT1 → ○((POT2 → ○(POT3) ∨ (POT3 → ○(POT2))))

내부 Action천이 $\tau 1$의 점화 실행후 $\tau 2$나 $\tau 3$ 의 발생표현은 다음과 같이 기술 가능하다.

$\Box(\tau 1 \rightarrow \bigcirc((\tau 2 \rightarrow \bigcirc(\tau 3) \vee (\tau 3 \rightarrow \bigcirc(\tau 2))))$

아울러 발생 사건에 대한 시제 표현도 다음과 같다.

$\Box$(E5 → ○((E6 → ○(E7) ∨ (E7 → ○(E6))))

그래서 어떠한 천이의 순서 표현이나 사건의 흐름 표현도 시제 논리로 표현 가능하다. 더욱이 사건의 발생은 순서적으로 동기를 형성해야 하기 때문에 과거, 현재, 미래를 표현할 수 있는 시제논리로서는 그야말로 적당한 표현법이라 할 수 있다. 물론 위의 경우에는 아직 시간 요소가 첨가되지 않은 순서적인(Sequential)경우다. 그래서 TESD에서 실시간 성질중의 하나인 상한시간 Hard Dead Line[7,9]을 기술해보면, 만약 사건 E1을 송신한 결과 최대 t초 이내에서 반드시 내부 천이가 발생하여 그 결과로 어떤 사건 E2나, 응답Action A를 발생시켜야 한다면 다음과 같이 기술할 수 있다.

$(E1 \wedge t = T) \rightarrow \Diamond((E2 \wedge t \leq T + t) \vee (A \wedge t \leq T + t))$

본 논문에서는 상한시간의 공식 within(e, t) 및 하한시간의 공식 delay(e, t)등을 기본적으로 제공한다. 아울러 TESD의 계층성을 위해서 객체 내부에 또다른 객체가 존재할 시에는 부모 객체 옆에 같이 확장하여 내부천이 및 사건들을 기술하도록 허용한다.

3.1 TESD의 문법 정의

이는 TESD에 관한 문법 정의이다. TESD는 다음의 5개의 tuple로 구성된다.

TESD = (O_i, E_MSG$_i$, MF$_i$, SYNC$_i$, TRANS$_i$)

- O_i : 모든 객체들의 집합,
- E_MSG$_i$: 모든 메시지들의 집합.
- MF$_i$: 메시지흐름관계의 집합.
- SYNC$_i$: 동기 순서의 집합,
- TRANS$_i$: 모든 천이의 집합.

여기서 E_MSG = (초기 메시지, 내부사건(IE), 외부사건(EE)),

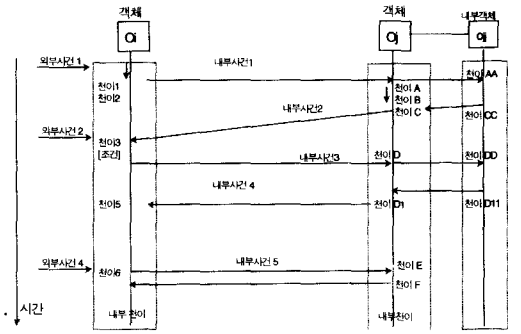
MF = (구분번호, 근원지객체, 사건, 천이액션, 목적지객체),

SYNC = (외부사건동기(EE_SYNC),내부사건동기(IE_SYNC),내부천이(ITR_SYNC),외부천이(ETR_SYNC))

TRANS = (외부천이(E_TRAN), 내부천이(I_TRAN))로 다시 나뉘어진다.

각 구성요소에 대한 설명은 다음과 같다. 먼저, 첫번째 구성요소인 O는 이 TESD를 구성하고 있는 각 대상 객체들이다. 일반적으로 좌측에서 오른쪽으로 하드웨어 객체를 포함한 종단객체를 시작으로 각 대상 객체들을 기술해 나간다. E_MSG는 외부(External)에서 최초로 입력되는 메시지들을 나열한다. 이 요소는 외부사건과 내부사건을 대표한다. EE는 외부에서 천이를 일으키고자 들어오는 사건들이고, IE는 천이후 발생되는 내부사건들이다. MF는 메시지의 흐름 방향으로 각 시간 축을 따라 발생하는 정보들을 표현하고자 사건 발생 순서번호, 시작 객체, 사건명, 천이명, 목적지 객체명 등으로 구성되어 있다. 그 다음 요소들은 전부 순서적인 관계를 표현하기 위한 동기목적으로 기술되는 부분으로, 요소 각각에 대해 EE_SYNC는 외부사건을 위한 발생순서 표현이고, IE_SYNC는 내부사건을 위한 순서표현이며, ITR_SYNC는 내부 액션천이를 위한 표

현이며, ETR_SYNC는 외부천이 순서를 위한 표현이다. 이들 각각은 전부 상태나 시제 공식 연산자로 표현되어 논리적으로 정확함을 보장한다. E_TRAN은 외부천이요소 즉, 메시지 통신 모듈(MCM) 이나 정보은닉 모듈(IHM)을 기술한다. I_TRAN은 내부(Internal)액션 천이 요소들을 기술하는데 일반적인 액션 천이시의 천이 조건도 아울러 제시하여 분석자나 설계자들을 돕고자 한다. 내부의 조건 부분에는 술어공식(하한시간을 위한 delay(e, t)나 상한시간을 위한 within(e, t))이나 조건 변수들을 기입할 수 있다. TESD의 모델링도구로 (그림6)을 참고하기 바란다.



(그림 6) 천이 사건순서도 •
(Fig. 6) Transition Event Sequence Diagram

(그림6)에서 객체의 각 시간 축을 따라가면 굵은 점선(도트 선)이 나오는데 이는 각 객체에 대해 동시(병행) 사건이 발생함(즉, t = 0)을 의미한다. 아울러 큰 점선으로 구성된 네모 박스는 내부천이의 발생을 의미하며 발생 순서적으로 기술되어진다. 또한 본 논문에서는 외부천이 발생순서를 제시하고자 비슷한 형태의 외부천이 순서도(그림11)를 제공한다. 이는 내부 천이와는 달리 사건과는 직접적인 관계가 없기 때문에 동일한 TESD에 표기가 불가능하여 따로 작성하였다. 기본적인 기호는 TESD와 동일하며 그 명세도 동일한 TESD 명세서에 같이 기술 가능하다. TESD의 명세언어는 다음과 같다.

TESD 〈시나리오_명〉
OBJECT : 〈composed_object_list〉;
E_MSG : 〈external_msg_name_list〉;
E TRAN : 〈mcm or ihm name_list〉;
I_TRAN : 〈internal transition_name〉{condition};
EE : 〈external event_list〉;
IE : 〈internal event_list〉;
MF : 〈seq_#,start_obj_name,event_name,transition_name,target_obj〉;
EE_SYNC : 〈state or temporal operator representation〉;
IE_SYNC : 〈state or temporal operator representation〉;
ITR_SYNC : 〈temporal operator representation〉;
ETR_SYNC : 〈temporal operator representation〉;

3.2 TESD의 의미(Semantics)정의

이 의미는 TESD의 명세에서 기술되는 시스템 행동이 정당한가 하는 것이다[12]. 행위(Behavior)들이 정당한(Valid) TESD는 사건 및 천이들의 열로 표현되는데 이 정당한 TESD를 개발하기 위한 모든 정보는 TESD의 개념적인(Conceptual) 확인기(Verifier)에 모여있다. TESD명세서가 작성된 후 이 명세서를 위한 확인기는 다음의 5개의 tuple로 구성된다.

V_{TESD} = (Ei, Ti, EEi, ITi, VUi)

여기서, Ei : TESD 명세시 사용되는 모든 사건들의 집합.
Ti : TESD 명세시 사용되는 모든 천이들의 집합.
EEi : 시스템의 외부에서 주어지는 외부 사건들의 집합.
ITi : 모든 천이들의 집합 중에서 내부 액션 천이들의 집합.
VU : 이는 확인 단위(Verify Unit)로서 인접하는 내부천이들과 그 사이에 존재하는 외부사건들을 가진 확인 단위의 집합인데, 확인단위는 다음의 tuple을 가진다.
vui = (seq_#,선천이,후천이,외부사건,사건순서#)
여기서, seq_#는 확인 단위의 구별번호이고, 선 천이는 인접하는 천이들 중에서 먼저 천이하는 천이집합이고, 후 천이는 뒤이어 천이하는 천이집합이다. 외부 사건은 인접하는 천이사

이에 들어오는 외부사건이고, 사건순서 번호는 외부사건들의 발생 순서번호이다.

또한, 이 확인기에서의 가정은, 미리 언급하지 않아도 초기천이 initial과 종단천이 terminate가 존재함을 내포한다.

3.3 TESD의 정당성(Validity)

초기에 TESD의 명세와 확인기를 통하여 정당한 TESD는 초기화 과정의 시나리오[12]와 이 초기화된 시나리오를 구성하고 있는 사건들의 열과 천이 열들의 집합이다.

초기화된 시나리오 δ는 다음과 같이 사건 E와 천이 T의 집합 열로 표현된다.

δ = (e1,e2,e3, ... ,{ei, ... ,ej||ek, ... ,el},...., en-1,en) ∪ (τ1,τ2,τ3, ... ,{τi, ... ,||,... , τk},.....τn-1,τn)

이 사건 및 천이의 열에서 내부사건과 일치하는 내부 액션천이들을 골라 나열하면 다음의 내부열(Internal Sequence)을 구할 수 있다.

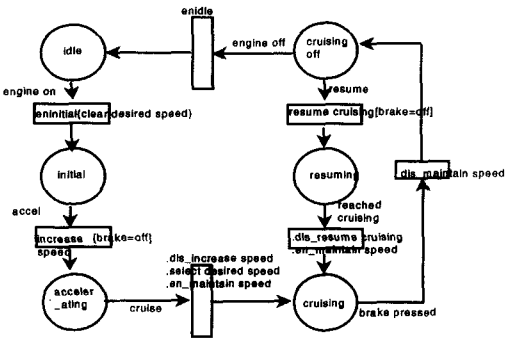
$\delta_{internal}$ = (iτ1, iτ2, iτ3, ... {iτi, ... ,||, ... , iτk},....iτn-1,iτn)

여기에서 동시발생 기호(||)가 사용된 사건 및 천이들은 모두 외부사건, 외부 천이이거나 또는 모두 내부사건 ,내부 천이들이다. 이는 내부와 외부사건이 동시에 발생 시는 외부 사건이 무시되기 때문이고, 천이는 사건신호를 다루는 측면에서는 외부천이(모듈천이)가 내부천이(액션천이)와는 전혀 관계가 없기 때문이다. 그러나 간략함을 위해서 이 예제에서 제시되지 않은(다른 병행객체에서의 사건 송수신시) 천이를 일으키는 액션천이와는 깊은 관계가 있다. 확인기의 정당성을 보증하는 방법은 다음과 같이 구성된다.

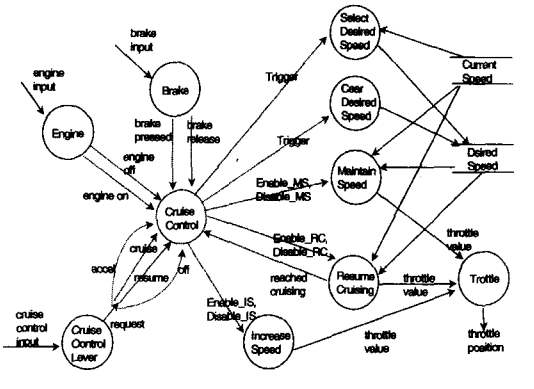
1. 초기화된 시나리오 δ열에서 내부사건과 일치하는 내부 액션천이들을 순서적으로 선택한다.
2. 첫 번째 액션천이와 그 다음의 액션천이 사이에 외부사건이 존재하는지를 체크한다.
3. 만약 외부사건이 존재한다면, 확인단위의 외부사건 요소와 일치하는지를 체크한다.
4. 외부사건이 존재하지 않으면, 확인단위도 Null인가 체크한다.
5. 3,4의 경우가 일치하면 TESD의 표현이 정당함을 보증한다.
6. 3,4의 경우가 비일치하면 TESD의 표현이 정당하지 않다.

4. TESD의 적용 사례

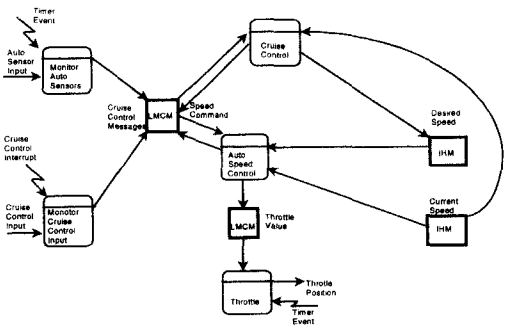
본 예제에서는 실시간 시스템의 응용을 통하여 본 논문의 타당성을 보이기 위해 CCS(Cruise Control System : 순항제어 시스템)[4,11]에 적용하고자 한다. 이 시스템은 어떤 특정의 속도를 유지하며 순항하다가 기준 이하의 속도로 떨어지면 다시 추진(가속)하여 특정의 속도를 계속 유지하는 시스템이다. OARTS의 방법론에 따라 작성된 이 시스템의 상태천이도와 초기 자료/제어 흐름도 및 객체 구성도는 (그림7,8,9)이다.



(그림 7) 순항제어시스템의 상태천이도
(Fig. 7) CCS State Transition Diagram



(그림 8) 순항제어시스템의 초기 자료흐름도
(Fig. 8) CCS initial Data Flow Diagram



(그림 9) 순항제어 객체 구성도
(Fig. 9) CCS Object Config. Diagram

① 문제정의 단계(시나리오의 개발)

시스템의 외부 사건들의 열과 내부 사건의 열 그리고, 외부천이와 내부천이의 열을 정의하면서, 시나리오를 개발한다.

상태천이도를 작성한 후 상태변화를 일으키는 외부환경으로부터 입력을 고려한다. 여기서의 외부입력 사건(External Event : 이하 EE)들은 주로 데이터 근원지 종단(Source Terminator : 이하 ST)과 관련되는 것들인데, ST = { 조종자, 순항제어 레버(Cruise Control Lever), Brake, Engine }이고, 이들 ST와 관련되어 생산하는 사건 EE = {accel_input, accel, cruise, resume, off, brake_input, brake_pressed, brake_released, engine_on, engine_off }이다. 내부사건(Internal Event : 이하 IE)은 소프트웨어 객체에서 생산되는(위의 경우 순항제어부분) 사건으로 IE = { Trigger_select_desired_speed, Trigger_clear_speed, Enable_maintain_speed, Disable_maintain_speed, Enable_resume_cruing, Disable_resume_cruising, Reached_cruising, Enable_increase_speed, Disable_increase_speed }이다. 외부천이(External Transition : 이하 ET)의 열로는 ET = {Cruise_Control_Message, Speed_Command, Throttle_Value, Desired_Speed, Current_Speed}이고, 내부천이(Internal Transition : IT) IT = { en_initial, en_increase_speed, dis_increase_speed, select_desired_speed, en_maintain_speed, dis_maintain_speed, en_idle, en_resume_crusing, dis_resume_crusing }으로 구성되어진다. 이러한 시나리오의 개발은 다음의 예와 같이 비정형하게 기술할 수 있다.

〈가정〉 : 시나리오의 시작시, 상태 천이도는 초기상태에 토큰이 위치한다[4].

1. 조종자는 Accl위치에 순항 제어 레버를 놓는다(engage).

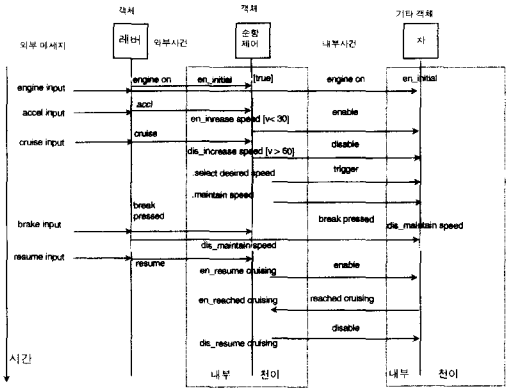
2. 조종자는 세팅된 어떤 속도로 순항하기 위해 순항제어 레버를 해제한다(release).

3. 조종자는 순항제어를 하지 않기 위해 브레이크(brake)를 누른다.

4. 조종자는 순항제어를 재개하기 위해 Resume 위치에 레버를 갖다 놓는다.

② 모형화 및 명세단계(시나리오의 실행)

이는 시나리오를 구성하는 사건이나, 외부천이, 내부천이 등에 관하여 순서적으로 실행하게 해주는 동기화 부분이다. 이들을 보여주기 위한 부분으로 천이 사건 순서도를 작성한다. 먼저, 사건을 기초로 한 TESD는 (그림10)과 같다.



(그림 10) 천이 사건 순서도
(Fig. 10) Transition Event Sequence Diagram

다음은 TESD를 기반으로 작성된 명세언어이다.

TESD 〈CCS#1〉

OBJECT : 〈레버〉,〈순항제어〉,〈차〉;

E_MSG : 〈engine input, accel input, cruise input, brake input, resume input〉;

E_TRANS : 〈ccm.Q1, cs, ccm.Q2(sc), tv.Q1, ct(concurrent transition)〉;

I_TRANS : 〈en_initial[true], en_increase_speed[v 〈 30], dis_increase_speed[v 〉 60], select_desired_speed, maintain_speed, dis_maintain _speed, en_resume_cruising, en_reached_ cruising〉;

EE : 〈engine_on, accel, cruise, brake_pressed, resume〉;

IE : 〈engine_on, enable_increase_speed, disable _increase_speed, trigger_select_desired_speed, maintain_speed, disable_maintain speed, enable_ resume_cruising, reached cruising, disable_resume_ cruising〉;

MF : 〈1, 레버, engine on, clear_desired_speed, 순항제어〉 ||
〈2, 레버, engine on, , 차〉,
〈3, 레버, accel, en_increase_speed, 순항제어〉,
〈4, 순항제어, enable_increase_speed, , 차〉,
〈5, 레버, cruise, dis_increase_speed, 순항제어〉,
〈6, 순항제어, disable_increase_speed, , 차〉,
〈7, 순항제어, trigger_select_desired_ speed, , 차〉,
〈8, 순항제어, maintain_speed, , 차〉,
〈9, 레버, break_pressed, dis_maintain _speed, 순항제어〉 ||
〈10, 레버, break_pressed, , 차〉,
〈11, 레버, resume, en_resume_cruising, 순항제어〉,
〈12, 순항제어, enable_resume_ cruising, , 차〉,
〈13, 차, reached_cruising, en_reached_cruising, 순항제어〉,
〈14, 순항제어, disable_resume_cruising, , 차〉;

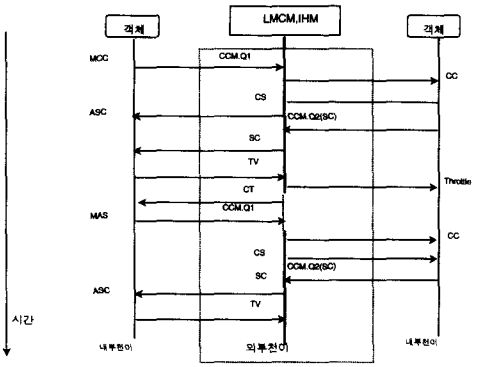
EE_SYNC : 〈 idle(θ : engine input) = TRUE , □(engine on → ○(accl → ○(cruise) ∧ (cruise → ○(brake pressed)) ∧ (brake pressed → ○(resume)))) 〉;

IE_SYNC : 〈 □(engine on → ○(enable_increase_speed → ○(disable increase_speed) ∧ (disable_increase_speed → ○(trigger_select_desired_speed)) ∧ (trigger_select _desired_speed → ○(maintain_speed)) ∧ (maintain_speed → ○(break pressed)) ∧ (brake_pressed → ○(enable_resume_cruising)) ∧ (enable_resume_cruising → ○(reached cruising)) ∧ (reached cruising → ○(disable_resume _cruising)))) 〉;

ITR_SYNC : 〈 (¬((¬en_initial) μ en_increase_speed)) ∧ (¬((¬ en_increase_speed) μ dis _increase_speed)) ∧ (dis_increase_speed P select_desired_speed) ∧ (select_desired_speed P maintain_speed) ∧ (maintain_speed P dis_maintin_speed) ∧ (dis_maintain_speed P en_resume_cruising) ∧ (en_resume_cruising P en_reached_cruising) ∧ (en_reached_cruising P dis_ resume_cruising) 〉;

ETR_SYNC : 〈 □(w1 P w2) → ◇($\models_{\sigma}^{=\text{외부천이열}}$ w), (ccm.Q1 P cs) ∧ (cs P sc) ∧ (sc P tv.Q1) ∧ (tv.Q1 P ct) ∧ (ct → ○(ccm.Q1) ∧ (ccm.Q1 →○(ct))) ∧ (ccm.Q1 P cs) ∧ (cs P sc) ∧ (sc P tv.Q1) 〉

외부 천이는 (그림11)과 같다. 이는 메시지 전달을 위한 외부 모듈들의 동작 순서이다. 외부 천이축 중간의 점선부분은 동시에 천이하는 어떤 동시천이 모듈(CT)이 존재할 경우의 모델링 및 명세를 위해서 제공되었다. 내부 천이에 관한 천이 동작은 위의 TESD의 액션천이 부분을 참조하면 될 것이다.



(그림 11)외부천이 순서도
(Fig. 11) External Transition Sequence Diagram

③ 확인단계(시나리오의 확인)

앞에서 제시된 시나리오 명세서를 바탕으로, 작성된 TESD의 확인기는 다음과 같다.

VERIFIER 〈TESD_ccs〉

EVENT : 〈engine_on,accel,cruise,brake_pressed,resume,enable_increase_speed,disable_increase_speed,trigger_select_desired_speed,maintain_speed,disable_maintainspeed,enable_resume_cruisin

g,reached cruising,disable_resume_cruising〉;

TRANS : 〈en_initial,en_increase_speed, dis_increase_speed, select_desired_speed, maintain_speed, dis_maintain_speed, en_resume_cruising, en_reached_cruising, dis_resume_cruising〉;

EE : 〈engine_on, accel, cruise, brake_pressed, resume〉;

IT : 〈en_initial,en_increase_speed,dis_increase_speed, select_desired_speed, maintain_speed, dis_maintain_speed, en_resume_cruising, en_reached_cruising, dis_resume_cruising〉;

VU : 〈1,en_initial,en_increase_speed,{eng_on, accl},{1.e_on,2.accl}〉,
〈2,en_increase_speed,dis_increase_speed,cruise,{}〉,
〈3,dis_increase_speed,select_desired_speed,{} ,{}〉,
〈4,select_desired_speed,maintain_speed,{},{}〉,
〈5,maintain_speed,dis_maintain_speed,{brk_presse d},brk_presed〉,
〈6,dis_maintain_speed,en_resume_cruising,resume, resume〉,
〈7,en_resume_cruising,en_reached_cruising,{},{}〉,
〈8,en_reached_cruising,dis_resume_cruising,{},{}〉,
〈9,dis_resume_cruising,terminal,{},{}〉;

이제, 확인기에 기술된 내용들이 정당한가를 확인 해 보면,

δ = {engine_on,accel,cruise,brake_pressed, resume,enable_increase_speed,disable_increase_speed,trigger_select_desired_speed,maintain_speed,disable_maintain_speed,enable_resume_cruising,reached_cruising,disable_resume_cruising} ∪ {en_initial,en_increase_speed,dis_increase_speed ,select_desired_speed,maintain_speed,dis_maintain_speed,en_resume_cruising,en_reached_cruising,dis_resume_cruising} 으로 구성된다.

여기서 내부 사건열을 구하고, 그와 일치하는 액션천이들의 열을 구하면, 결국 $\delta_{internal}$ = {en_initial, en_increase_speed,dis_increase_speed,select_desired_speed,maintain_speed,dis_maintain_speed,en_resume_cruising,en_reached_cruising,dis_resume cruising}이 된다.

여기에서 주의해야 할 점은 본 예제에서 제공하는 액션 천이들은 문제를 간단하게 하기 위해 순항제어 객체를 중심으로한 천이들만을 나열하였다. 그러나 전체 시스템의 완전한 TESD를 제공하기 위해서는 대상이 되는 모든 시나리오들을 제시해야만 한다. 그렇게 되면 대상 범주에 포함되는 모든 객체들에 대하여 액션 천이들을 선정해야 하기 때문에 대단히 복잡해진다. 그래서 본 논문에서는 간단한 대상들(사건, 액션천이, 객체)만을 중심으로한 개념을 소개하고자 액션천이의 대상들을 순항제어에 국한시켰다. 위의 예제에서 전체 천이요소와 내부 천이 요소가 같은 점은 바로 이 때문이다. 이 내부 액션 천이를 일으키는 인접하는 외부사건들의 열은 다음과 같다.

EE#1(en_initial ~ en_increase_speed) ⇒(engin_on, accl)
EE#2(en_increase_speed ~dis_increase_speed) ⇒ (cruise)
EE#3(dis_increase_speed ~ select_desired_ speed) ⇒ null
EE#4(select_desired_speed ~ maintain_speed) ⇒ null
EE#5(maintain_speed ~ dis_maintain_speed) ⇒ (break_pressed)
EE#6(dis_maintain_speed ~ en_resume_ cruising) ⇒ (resume)
EE#7(en_resume_cruising ~ en_reached_ cruising) ⇒ null
EE#8(en_reached_cruising ~ dis_resume_ cruising) ⇒ null
EE#9(dis_resume_cruising ~ terminating) ⇒ null

이제, 이 외부 사건들의 열을 만족하는 확인 단위가 집합 VU상에 존재하는지 확인하면 될 것이다. 다음은 인접하는 두 천이들과 그 사이에 존재하는 외부 사건들의 열에 대해서 만족하는 확인 단위들이다. 각 확인단위 vui ∈ VU에 대해서 인접하는 두 천이들 사이의 외부사건이, 확인기의 확인 단위와 일치됨을 확인할 수 있다.

(en_initial ~ en_increase_speed) ⊨ vu1
(en_increase_speed ~ dis_increase_speed) ⊨ vu2

(dis_increase_speed ~ select_desired_speed) ⊨ vu3
(select_desired_speed ~ maintain_speed) ⊨ vu4
(maintain_speed ~ dis_maintain_speed) ⊨ vu5
(dis_maintain_speed ~ en_resume_cruising) ⊨ vu6
(en_resume_cruising ~ en_reached_cruising) ⊨ vu7
(en_reached_cruising ~ dis_resume_cruising) ⊨ vu8
(dis_resume_cruising ~ terminating) ⊨ vu9

이로서 TESD의 표현이 정당함을 보증할 수 있다. 사실, 이 개념적 확인기는 실제 자동화도구를 위한 기본 구조(Framework)로 사용되며, 현재 본 연구실에서 OARTS와 더불어 구현 중에 있다.

5. 결 론

지금까지, 우리는 천이사건 순서제어를 위한 표현방법과 명세언어 및 확인 방법에 관하여 기술하였다. 천이사건 순서제어 방안은 세 가지로 구성되는데 첫째, 비정형화된 문제정의 부분과 둘째, TESD와 명세언어 부분, 셋째, 확인기를 통하여 표현된 TESD와 명세언어가 정당한지 확인하는 부분으로 구성된다. 이렇게 구성된 시나리오는 시나리오 작성에 필요한 모든 요소들이 실제 적용사례를 통하여 잘 적용됨을 확인할 수 있었다. 특히, 본 논문에서는 OARTS를 위해 외부 객체 통신 모듈을 구성하고, 이들 모듈들간의 천이 관계들을 시제논리로 표현하여 동기순서들을 명확하게 표현할 수 있었다. 본 논문은 비단 OARTS를 위한 천이사건 제어에만 사용될 수 있는 것은 아니다. 일반적으로 사용되는 요구 분석을 위한 시나리오(외부사건열 및 내부사건열의 취급)는 사건의 열만을 가지고 다루고 있기 때문에 이 방법을 일반적인 시나리오의 표현에 이용해도 전혀 무리가 없을 것이다. 현재 본 연구실에서는 일반적인 시나리오의 모델링 단계에서 그치는 것이 아니라 미묘한 시간 요구사항까지 더 상세하게 표현할 수 있는 시나리오의 개발과 시나리오와 방법론의 접목시 사용되는 기법들과의 일관성의 검증에 관하여 연구를 하고 있다.

참 고 문 헌

[1] C.potts et al. "Inquiry-Based Requirements Analysis," IEEE Software, pp.21-32, March. 1994.

[2] H.Gomaa. "Software Development of Real-time Systems," CACM, pp.657-668, July. 1986.

[3] H.Gomaa. "A Software Design Method for Real-time Systems," CACM, pp.938-949, Sep., 1984.

[4] H.Gomaa. "Software Design Methods for Co-ncurrent and Real-Time Systems," Addison-Wesley Publishing Co. Inc., 1993.

[5] J.D.McGregor "Integrated Object-Oriented Testing and Development Process," CACM, pp.59-77, Sep., vol37, No9.

[6] J.Rumbaugh et al. "Object-Oriented Modeling and Design," Prentice-Hall, 1991.

[7] J.S.Ostroff and W.M. Wonham. "A Framework for Real-time Discrete Event Control," IEEE Trans. Automatic Control. pp.386-397, Apr., 1990.

[8] Lee,Y.K "Object-Oriented High-level Petri Net for Manufacturing System Modeling," TR Dept of MS, KAIST, 1991.

[9] P.Civera. "Microcomputer Systems in Real-time Applications," in Tzafestas,S.G.(ed), Microprocessors in signal processing, Measurement and Control, 1983.

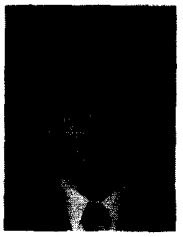
[10] P.Hsia et al. "Formal Approach to Scenario Analysis," IEEE Software, pp.33-41, March, 1994.

[11] P.Ward, and S.Mellor "Structured Development for Real-time Systems," Vol.1, Prentice-Hall, 1985.

[12] 고광일 "다시각적 분석을 제공하는 시제논리 기반의 실시간 소프트웨어 시스템 명세방법," 포항 공과대학교, 석사학위 논문, 1995.

[13] 김은주 "시나리오를 이용한 객체 지향시스템의 기능 테스트," 한국정보과학회, 가을학술발표논문집, Vol.23, No.2, pp.1523-1526, 1996.

[14] 강병욱 "소프트웨어 공학," 영남대학교, Lecture Notes, 1997.



김 정 술

1985년 2월 한양대학교 전자공학과 졸업(학사)
1994년 2월 포항공과대학교 정보통신학과(공학석사)
1998년 2월 영남대학교 컴퓨터공학과(박사수료)
1988년~91년 L.G전자 PC/MNT설계실 근무.
1994년~96년 한진정보통신(주) SI사업부 근무.
관심분야 : 소프트웨어공학(실시간 시스템, Formal Method,재사용)

강 병 욱

1970년 영남대학교 전기공학과(공학사)
1977년 영남대학교 전자공학과(공학석사)
1994년 경북대학교 전자공학과(공학박사)
1979년~현재 영남대학교 전산공학과 교수
관심분야 : 소프트웨어 공학, 프로그래밍 언어, 데이터 압축