실시간 제어시스템의 그래픽 기반 정형명세

윤상호*, 심재환**, 최진영*
*고려대학교 소프트웨어 공학과
**고려대학교 컴퓨터통신공학부
e-mail : morejoy@dreamwiz.com

Graphic-based Formal Specification for Real-time Control System

Sang-Ho Yoon*, Jae-Hwan Shim**, Jin-Young Choi*
*Dept. of Software Engineering, Korea University
** Department of Computer and Communication Engineering, Korea University

요 약

본 논문은 전자제어 시스템들의 소프트웨어의 설계 및 구현 과정에서 나타날 수 있는 의사 전달의 애매모호함을 줄여 요구사항 명세와 구현 시스템 사이의 불일치를 없애기 위한 실시간 제어 소프트웨어의 정형 명세 기법을 제안한다. 실시간 제어 시스템 이론은 미적분학에 바탕을 두고 있는 반면, 실제 동작은 대다수 임베디드 프로세서에 의해 동작함에 따라 제어기의 설계 시 고안된 연속적인 미분 방정식의 이산화 과정을 거쳐 소프트웨어로서 구현이 된다. 이 때, 시스템 설계 엔지니어와 소프트웨어 구현 엔지니어 사이의 이해도의 불일치와 구현 엔지니어의 시스템 이론의 이해 부재로 시스템에 심각한 오류를 야기할 수 있다. 이에 본 논문에서는 이러한 실시간 제어 시스템의 기능 및 동작에 대한 그래픽 기반 정형적 명세 기법을 제안하여 요구사항 명세 과정에서 구현 방식을 구체화하는 방향을 제시한다.

1. 서론

현재 대다수의 실시간 제어 시스템들은 전자 하드웨어 기술의 발전과 함께 소형화되고 저렴해진 다양한 종류의 센서와 마이크로 프로세서들을 활용하여 개발되고 있다. 특히 이러한 패러다임은 과거 기계장치들만으로 구성되었던 자동차, 항공기, 기차, 조선 등에도 빠르게 적용되어 왔으며, 현재에는 한 대의 자동차에도 수십 가지의 임베디드 프로세서를 이용한 제어장치가 장착되어 있으며 그 수는 계속하여 증가하고 있어 이러한 제어 장치의 효율적 개발을 위한 많은 연구가 이루어지고 있다.

이러한 기계장치를 제어하기 위해서는 기계장치의 물리적 특성에 맞는 제어 방식의 설계 및 구현이 필요하다. 기계 장치의 물리적 특성은 대부분 상태 미분 방정식에 의해 표현되며 이에 맞는 제어기 또한 상태 미분 방정식의 형태로 설계된다. 이러한 미분방정식은 시간에 연속적인 특성을 갖기 때문에 정해진 샘플 타임에 맞춰 이산적으로 동작하는 프로세서 내에서 동작시키기 위해서는 이산화 과정을 거쳐 프로그램으로 구현될 수 있다. 그런데 이 과정에서, 시스템 엔지니어의 설계 의도가 구현 엔지니어에게 잘못 전달 될 가능성이 높으며, 이는 시스템의 심각한 오류를 야기할 수 있다. 특히, 자동차, 항공기, 철도 등의 안전필수 시스템(Safety-critical system)은 시스템의 안정성 확보를 위하여 제어기의 설계 과정에서 제어 이론에 입각한 제어 시스템의 안정성을 반드시 고려해야 하며 소프트웨어 구현 과정에서도 제어 소프트웨어의 엄격한 실시간 동작 특성과 신뢰성을 요구한다.

안전필수 시스템의 일반적인 개발 프로세스에는 V-모델이 주로 사용되며 시스템의 설계 시 만들어진 설계사양서를 이용하여 소프트웨어로의 구현과 시험이 이루어 지게 된다. 이때, 설계 사양서가 자연어로 표현될 경우 설계를 담당한 시스템 엔지니어와 임베디드 소프트웨어의 구현 엔지니어 사이에 시스템에 대한 이해의 불일치가 발생한다. 이는 상기 언급한 바와 같이 시스템의 심각한 오류를 초래한다. 또한, 이러한 오류는 시험단계에서 주로 발견되기 때문에 재설계 및 구현에 따르는 프로세스 상의 지연을 야기시키기도 한다. 이러한 문제점을 해결하기 위해서 시스템 설계 시, 자연어 보다는 수학적이고, 엄밀한 명세 기법인 정형 명세 기법을 사용한다. 하지만, 기존의 많은 정형 명세 언어가 연속적인 특성을 가지는 제어 시스템에는 적합하지 않다.

따라서 본 논문에서는 시스템 개발 엔지니어들간의 의사소통의 문제를 줄이고 시스템을 초기에 검증하기 위해, 실시간 제어 소프트웨어에 적합한 그래픽 기반 정형 명세 기법을 제안하고, Fault Tolerant 알고리즘을 포함하는 PID(Proportional Integral Differential) 제어 알고리즘 설계에 본 기법을 적용해 본다.

2. 관련 연구

2-1. 제어 논리의 정형 명세

소프트웨어의 기능적 요구사항의 정형적 표현을 위

해 다양한 연구가 진행되어 왔으며, 크게 집합이론기반의 명세과 행위 명세으로 구분할 수 있다.[3]

집합이론기반의 명세로서는 Z[5]나 B, VDM 등의 명세 언어가 있으며, 이들은 데게 시스템이 어떻게 동작하느냐 보다 시스템이 무엇을 하느냐에 중심을 두고 있다. 이에 반해 CTL(Computational Tree Logic), LTL(Linear Temporal Logic), PLTL(Propositional LTL), TLA(Temporal Logic of Actions)와 같은 행위 명세 기법은 시스템의 동작 특성을 명세하는 것에 중심을 두고 있다. 이러한 명세 언어들은 수학적, 논리적 기반을 둔 언어로서, 소프트웨어의 기능적 측면을 컴퓨터의 실행 형태가 아닌 사람들의 이해를 위한 표현형태를 갖도록 하는 것을 목적으로 한다. 그러나 이러한 명세 언어들은 여전히 사용하거나 이해하기에 어려운 수학이나 논리의 notation 을 사용하기 때문에 여전히 학습의 노력과 실용 경험 및 전문성을 요구한다는 아쉬움을 가지고 있다.

따라서, David Harel 에 의해 행위 명세 기법 중 하나로서 그래픽기반의 명세언어인 상태차트 다이어그램이 제안되었으며, 실제 소프트웨어 엔지니어가 아닌 제어 시스템을 설계하는 시스템 엔지니어들에게도 많이 활용되고 있다. 또한 상태차트 다이어그램은 다양한 도구들을 활용하여 구현 전 simulation 이 가능해짐에 따라, model checking 기법을 활용하여 구현 전 초기 검증이 가능하다는 장점이 있다.[6] 하지만, 상태 다이어그램은 이벤트 기반의 명세언어이기 때문에 연속적인 특성을 가지는 제어시스템을 명세하기에는 여전히 용이 하지 않다.

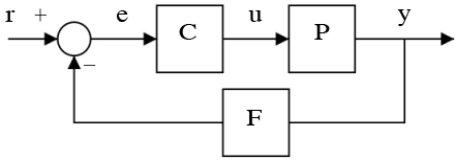
2-2. 실시간 제어 시스템의 특성

실시간 제어 시스템은 다양한 제어 이론에 입각하여 수학적 분석방법을 거쳐 설계가 된다. 최근에는 이러한 제어 이론들을 특히 고전적 제어와 현대적 제어로 분류하기도 하지만, 여전히 PID 제어기와 같은 고전적 제어 방식이 현장에서 가장 많이 사용되고 있다.

제어시스템의 구성은 대게 시스템의 입력에 대해 원하는 출력을 내는 것을 목표로 한다. (그림 1)은 대표적인 제어 시스템의 예이다. 여기서, C 는 제어기를 뜻하며, P 는 제어 대상이 되는 플랜트, F 는 출력을 측정하는 센서이다. 즉 목표로 하는 입력에 대해 원하는 출력이 나오도록 하기 위해 제어 대상의 결과를 센서를 통해 받아오며, 목표 입력과 출력의 차이를 줄이기 위해 제어기가 시스템에 포함된다. 이것을 일반적으로 폐루프 제어 시스템이라고 한다. 식(1)은 이러한 폐루프 제어 시스템을 s-domain 상에 표현한 시스템 방정식이다.

$$Y(s) = \left(\frac{P(s)C(s)}{1 + F(s)P(s)C(s)} \right) R(s) \qquad \text{---------- 식(1)}$$

실시간 제어 시스템의 제어 플랜트의 특성들은 일반적으로 미분방정식의 형태로 표현되며 이를 제어하는



(그림 1) 폐루프 제어시스템의 블록 다이어그램

제어기 또한 미분방정식의 형태로 초기에 설계되고 시스템의 안정성과 응답특성 등이 함께 고려된다. 이렇게 설계된 제어기는 차후 임베디드 프로세서 내에서 동작할 소프트웨어로 구현되기 위해 이산화 및 차분화 과정을 거쳐 차분 방정식이라고 하는 차분 수식의 형태로 표현된다. 따라서 본 논문에서는 이러한 차분 수식을 그래픽 기반으로 표현할 수 있는 형태의 정형 명세 기법을 제안하고자 한다.

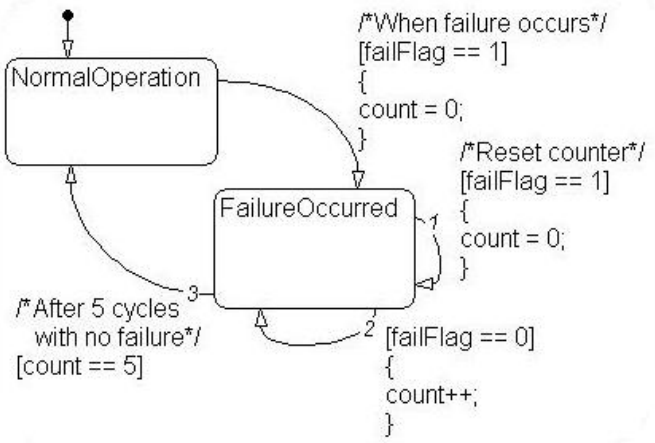
또한, 조건에 따라 제어의 모드를 변환하는 등의 상태차트 다이어그램으로 표현될 수 있는 논리적 패턴들도 제어기 속에 물론 포함되어야 한다. 따라서 본 논문에서는 차분 수식의 명세와 상태차트 다이어그램을 접목시킬 수 있는 형태의 그래픽 기반 명세 기법을 제안함으로써 시스템 엔지니어의 설계사양서를 소프트웨어 구현 엔지니어들이 직관적으로 이해할 수 있도록 보다 구체화하여 의사소통 문제에 따른 오류를 감소시키는 방안을 제시한다.

3. 본론

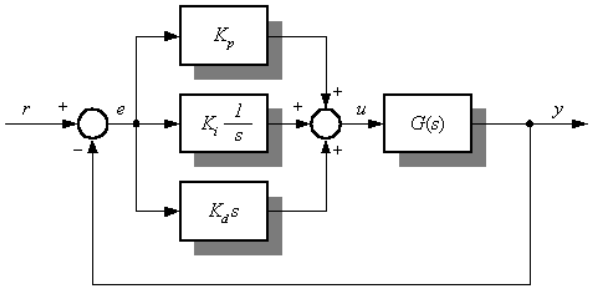
3-1. 제어 모드의 그래픽기반 명세

실시간 제어 시스템 제어 모드 변환의 한 예로서 fault tolerant 알고리즘이 포함된 PID 제어 시스템을 들어보겠다. 현재 예를 들어보려는 시스템은 센서 정보와 같은 입력 데이터의 오류 시 제어 모드를 변경하는 시스템이다. 이러한 시스템은 정상 제어와 비정상 제어의 두 가지 상태로 표현될 수 있으며, 이는 상태차트 다이어그램으로 표현할 수 있는 가장 일반적인 예이다. (그림 2)는 상태차트 다이어그램으로 표현된 가장 간단한 fault tolerant 알고리즘의 한 예이다.

(그림 2)의 fault tolerant 알고리즘은 정상 상태와



(그림 2) Fault Tolerant Algorithm 의 상태차트 다이어그램



(그림 3) PID 제어기로 구성된 페루프 제어시스템

비정상 상태의 두 가지 상태로 구분되며 오류 발생시 정상 상태에서 비정상 상태로 상태의 천이가 일어난다. 비정상 상태에서는 계속하여 오류 발생 여부를 확인하고 오류가 더 이상 발생하지 않는 상태가 5 회 지속되면 다시 정상 상태로 전환된다. 이러한 상태차트 다이어그램은 유한한 개수의 상태들의 명확한 표현을 제시하며, 각 상태의 천이에 대해서도 경로가 명확히 정의되어 있어 소프트웨어 구현에 대한 분명한 방향을 제시하게 된다.

3-2. PID 제어기의 이산화

(그림 2)에 표현된 각 상태에서 취해야 할 액션에 대해서 생각해 보자. 먼저 정상 상태에서는 구현하고자 하는 PID 제어를 수행하고 비정상 상태에서는 warning 을 띄워주며 제어를 중지하는 시스템을 가정하자. 여기서는 warning 을 띄워주는 간단한 mode 의 명세방식에 대해서는 생략하겠다.

먼저 정상 상태의 PID 제어 시스템의 블록 다이어그램은 (그림 3)과 같이 표현된다.
(그림 3)의 G(s)는 앞의 (그림 1)의 플랜트와 센서를 함께 표현한 것이며, Kp, Ki, Kd 항목이 포함된 부분이 PID 제어기 C(s)가 된다. 즉, C(s)에 대한 전달함수는 식(2)과 같다.

$$C(s) = \left(K_p + K_i \frac{1}{s} + K_d s \right) \quad \text{---------- 식(2)}$$

이러한 연속 시스템은 실제 소프트웨어로 구현되기 위하여 이산화되어야 한다. 또한 이산화 시 결정하게 될 샘플 타임(sample time: Ts)에 따라 시스템의 안정도 평가가 재 수행되어야 하며, 이러한 이산화를 위한 다양한 방법이 존재한다.[2] 본 논문에서는 가장 간단한 이산화 방법인 Backward-Euler 기법을 적용하여 시스템을 이산화한다. 식(3)은 s-domain 의 전달함수 식(2)를 Backward-Euler 방식으로 z 변환한 결과 이다. 이 과정에서 이산화에 따르는 Ts 가 새로운 변수로 추가되었으며, 실제 제어 알고리즘이 Ts 의 시간 간격마다 동작함을 의미한다. 즉 이산화 과정에서 정한 샘플 타임이 시스템의 아웃풋에 큰 영향을 주게 된다.

$$C(s) = \left(K_p + \frac{T_s K_i}{1 - z^{-1}} + \frac{K_d (1 - z^{-1})}{T_s} \right) \quad \text{---------- 식(3)}$$

이렇게 이산화된 수식으로 PID 제어기에 대한 차분식을 얻으면 아래의 식(4)와 같다. 식(4)의 e(k-1)은 e(k)의 Ts 시간 이전 값을 의미하며, yi(k-1)은 yi(k)의 Ts 시간 이전 값을 뜻한다.

$$y(k) = y_p(k) + y_i(k) + y_d(k) \quad \text{---------- 식(4)}$$

where

$$y_p(k) = K_p e(k)$$

$$y_i(k) = y_i(k-1) + K_i T_s e(k)$$

$$y_d(k) = \frac{K_d}{T_s} \left[e(k) - e(k-1) \right]$$

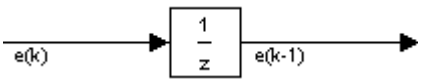
3-3. 차분화된 PID 제어기의 그래픽기반 명세

이산화된 시스템의 차분식 표현 시 식(4)에 나타난 것처럼 각 변수의 Ts 시간 이전 값이 저장되어야 하며, 다음 스텝에서 다시 사용되고 다시 값이 변경되어야 한다. 이렇게 이전 값을 저장하는 메모리 공간을 (그림 4)와 같은 형태로 표현하면 차분식에 대한 그래픽기반의 정형화를 이룰 수 있다.

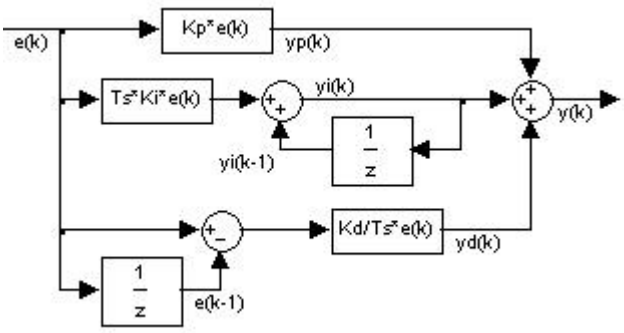
(그림 5)는 식(4)에 대한 (그림 4)의 표현 방식을 이용한 PID 제어기의 표현 결과이다. 단, (그림 5)의 블록(□)내부에는 단순한 제어 파라미터 상수와의 곱셈에 대한 항목만을 포함한다. 즉, 코드 구현 시 이전 값을 저장하는 메모리 공간 할당을 제외한 차분화된 제어기의 나머지 항목은 모두 단순한 사칙 연산이 되는 특징을 활용한 것이다. 결국 제어기의 그래픽 표현은 +/-에 대한 블록과 이전 값을 저장하고 다음 스텝에 활용하게 될 1/z 블록, 상수의 곱셈 연산을 포함하는 블록 만으로 단순화 되며 코드의 구현 방식에 대한 간결한 그래픽 기반 명세를 가능케 해 준다.

3-4. 상태차트 다이어그램과 차분식 명세의 통합

실시간 제어 알고리즘을 구현하는 데 있어 하나의 통합된 정형화 명세 방식을 갖는다는 것은 시스템 엔지니어와 소프트웨어 구현 엔지니어 사이의 의사소통을 원활히 하기 위해 매우 중요한 사항이다.



(그림 4) 차분식의 Ts 이전 값 표현

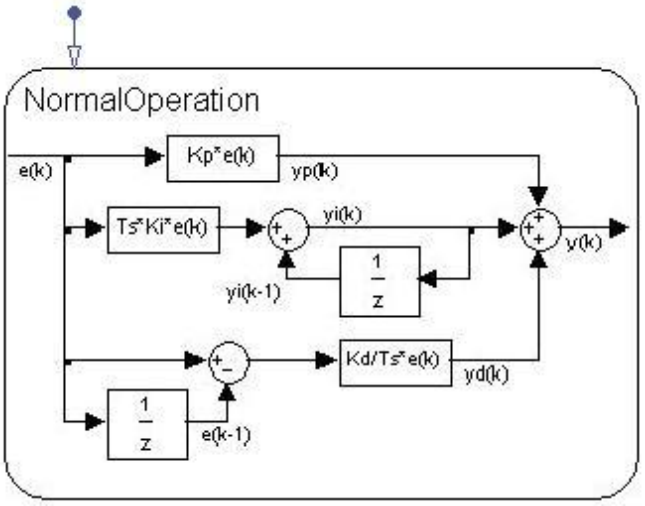


(그림 5) PID 제어 알고리즘의 그래픽 기반 명세

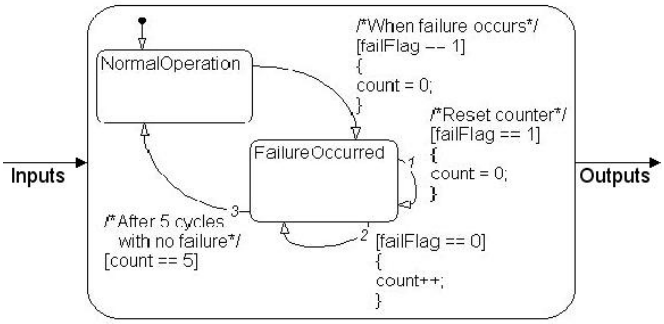
앞에서 다룬 차분 방정식의 그래픽 기반 명세는 실제 각 블록을 연결하는 선들이 데이터의 흐름을 표현한다는 측면에서 데이터 흐름도라고 볼 수 있다. 그러나 상태차트 다이어그램은 소프트웨어의 동작 상태를 구분하여 표현한다는 측면에서 세부 데이터의 흐름을 보여주지는 못하고 있다. 결국 통합을 위해 가장 중요한 부분은 데이터의 처리라 할 수 있겠다. 따라서 상태차트 다이어그램 내에 차분 수식의 블록 선도를 포함시키고 상태 차트로의 입력, 출력 표현을 통해 두 가지를 통합한다.

(그림 6)은 fault tolerant 알고리즘 내 normal 모드에서 PID 제어 알고리즘의 그래픽 기반 명세를 통합한 예이다. 상태 차트내의 한 상태로 전달될 입력 및 출력 데이터는 결국 전체 상태차트로 전달될 입력/출력 데이터 일 수 밖에 없다. 따라서 (그림 7)과 같이 상태차트 다이어그램 외부에 인풋 데이터와 아웃풋 데이터를 데이터 흐름 방식으로 표현하여 그래픽 기반 명세의 통합을 이룬다.

이를 통해 구현 엔지니어가 제어 시스템에 적용된 깊은 이해 없이도 이벤트 기반의 알고리즘과 차분수식을 구현함에 있어 if 문과 간단한 사칙 연산 및 기존 상태의 값을 저장하는 전역 변수의 할당만으로 정확하고 빠른 구현을 가능케 한다.



(그림 6) Normal 모드에서의 PID 제어 구현 명세



(그림 7) 상태차트 다이어그램으로 전달 되는 인풋 데이터와 아웃풋 데이터

4. 결론 및 향후 방향

본 논문에서는 임베디드 실시간 제어시스템의 제어 알고리즘을 정형적으로 명세하는 데에 있어서, 시스템 레벨의 접근을 통해 제어 이론에 입각한 차분 수식의 그래픽 기반 정형적 명세방법을 제시하였다. 또한, 제어 모드의 변화 등을 표현하는 부분에서는 자주 사용되는 그래픽 기반 행위 명세 기법인 상태차트 다이어그램을 활용하였다. 차분식의 그래픽 기반 명세는 데이터 흐름도 내에 시스템 엔지니어들이 쉽게 이해할 수 있는 z-domain 상의 샘플 타임 delay 를 표현하는 블록을 이용하는 방법을 취하였으며, 상태 차트와의 통합을 통해 통일된 방식의 실시간 제어 시스템의 그래픽 기반 정형화를 시도하였다.

본 논문에서 제시한 실시간 제어 시스템을 위한 통합된 명세 기법이 시스템 엔지니어와 임베디드 소프트웨어의 구현 엔지니어 및 시험엔지니어 사이의 견해 차를 줄여 원활한 의사 소통으로 시스템 오류를 감소시키고, 시스템의 개발 기간을 단축 시킬 수 있는 좋은 방향이 될 수 있기를 기대한다.

마지막으로 짚어볼 사항은 현재의 연구가 실시간 제어 소프트웨어의 그래픽 기반 정형 명세 방법에만 여전히 국한되어 있다는 것이다. 그래픽 기반으로 작성된 기능 명세에 대한 검증방법의 추가적인 연구가 필요하며, 상태차트의 model checking 과 함께 각 상태에 포함된 그래픽 기반 차분수식 정형 명세의 증명 기법에 대해 향후 연구를 진행할 예정이다.

참고문헌

[1] Gene F. Franklin, “Feedback Control of Dynamic Systems Fifth Edition”, Prentice Hall, 2006

[2] Katsuhiko Ogata, "Discrete-time Control System", Prentice Hall, 1994

[3] Jean-Francois Monin, “Understanding Formal Methods”, Springer-Verlag London Limited, 2003

[4] Martin Fowler, “UML Distilled Third Edition”, Addison Wesley, 2003

[5] Jim Woodcock and Jim Davies, “Using Z: Specification, Refinement, and Proof”, Prentice Hall, 1996.

[6] 방기석, “Statechart 로 구현된 명세의 정형 검증 기법”, 한국정보과학회 학술발표논문집, 1999

[7] 심재환, “하이브리드 오토마타 기반 실시간 시스템의 모니터링 기법”, 한국컴퓨터종합학술대회 논문집 Vol. 34, No. 1, 2007

[8] Yerang Hur, Jae-Hwan Sim, Jesung Kim, Jin-Young Choi, “Ensuring Sound Numerical Simulation of Hybrid Automata”, Journal of Computing Science and Engineering, Vol.3, No. 2, June 2009, Page 73-87