프로세서 활용도 함수를 이용한 실시간 제어시스템 유연성 분석

채 정 화[†] · 유 철 중[††]

요 약

최근 산업 프로세스의 제어 및 모니터링 분야에서 컴퓨터의 사용이 급증하고 있다. 이와 같은 자동차 기어 및 엔진 제어, 항공기 이착륙/운항 제어, 통신 네트워크 등과 같은 응용분야에서 사용되고 있는 컴퓨터는 시간 임계 제어 및 모니터 기능과 시간에 제한적이지 않은 일괄 프로세싱으로 구분된다. 실시간 제어시스템 또는 임베디드 시스템은 컴퓨터 시스템에서 특정한 역할을 수행하는 다양한 하드웨어와 소프트웨어 요소로 구성되어 있다. 실시간 시스템이 시간적 제약조건을 만족하지 못하면 시스템의 오동작이 발생할 수 있으며, 인명 손상과 같은 큰 재앙이 발생할 수 있다. 그렇기 때문에 시스템의 수행은 예측 가능해야 한다. 실시간 임베디드 시스템은 아키텍처 설계 단계에서의 결정이 시스템의 구현과 성능에 매우 큰 영향을 미친다. 유연성이란 실시간 시스템 환경에서 작업 타이밍에 대한 장애를 유연하게 처리할 수 있는 시스템의 처리 능력을 나타낸다. 이 요소는 시스템의 수행비용과 성능을 분석하는 중요한 요소이다. 본 연구에서는 실시간 임베디드 시스템의 구조를 설계할 때 효율적인 분석을 위하여 유연성 함수를 정의한다. 본 연구를 통하여 실시간 시스템의 하드웨어 및 소프트웨어를 분할할 때 속성과 제약을 분석할 수 있는데, 제약의 측면에서 보면 유연성 문제를 해결 할 수 있고, 속성의 측면에서 보면 시스템 모델 및 유연성과 다른 속성들(비용, 전력소비량 등)과의 상호관계 분석 등을 효율적으로 분석할 수 있다.

On Flexibility Analysis of Real-Time Control System Using Processor Utilization Function

Jung-Wha Chae[†] · Cheol-Jung Yoo[††]

ABSTRACT

The use of computers for control and monitoring of industrial process has expanded greatly in recent years. The computer used in such applications is shared between a certain number of time-critical control and monitor function and non time-critical batch processing job stream. Embedded systems encompass a variety of hardware and software components which perform specific function in host computer. Many embedded system must respond to external events under certain timing constraints. Failure to respond to certain events on time may either seriously degrade system performance or even result in a catastrophe. In the design of real-time embedded system, decisions made at the architectural design phase greatly affect the final implementation and performance of the system. Flexibility indicates how well a particular system architecture can tolerate with respect to satisfying real-time requirements. The degree of flexibility of real-time system architecture indicates the capability of the system to tolerate perturbations in timing related specifications. Given degree of flexibility, one may compare and rank different implementations. A system with a higher degree of flexibility is more desirable. Flexibility is also an important factor in the trade-off studies between cost and performance. In this paper, it is identified the need for flexibility function and shows that the existing real-time analysis result can be effective. This paper motivated the need for a flexibility for the efficient analysis of potential design candidates in the architectural design exploration or real-time embedded system.

키워드 : 임베디드 시스템(Embedded System), 유연성 분석(Flexibility Analysis), 프로세서 활용도 함수(Processor Utilization Function), 실시간 제어 시스템(Real-Time Control System)

1. 서 론

실시간 시스템(Real Time System)이란 기존의 컴퓨터 시스템과는 달리 시스템의 수행 결과가 논리적인 정확성뿐만 아니라 시간적인 정확성까지 만족시켜야 하는 시스템을 의미한다. 실시간 시스템은 산업이 발전하면서 자동차 기어 및 엔진 제어, 항공기 이착륙/운항 제어, 통신 네트워크 등의 많은 응용분야에 널리 적용되고 있다.

이러한 실시간 시스템은 크게 경성 실시간 시스템(Hard

† 준 회 원 : 전북대학교 대학원 전산통계학과
†† 종신회원 : 전북대학교 컴퓨터과학과 부교수
논문접수 : 2004년 9월 10일, 심사완료 : 2005년 1월 20일

Real-Time System)과 연성 실시간 시스템(Soft Real-Time System)으로 구분한다. 경성 실시간 시스템은 정해진 데드라인(deadline) 이내에 정해진 작업의 결과가 절대적으로 출력되어야 하는 시스템으로 시스템이 주어진 종료 시간을 만족시키지 못하면 치명적인 결과가 나온다. 연성 실시간 시스템은 정해진 데드라인 이내에 작업을 처리하지 못해도 치명적인 결과를 가져오지는 않는 시스템으로 경성의 경우와는 달리 치명적인 결과를 초래하지는 않으나 그 결과가 무의미한 것이 될 수 있다.

제어시스템이란 정해진 특정 기능을 수행하기 위해서 하드웨어와 소프트웨어가 결합된 시스템을 의미한다. 초기의 제어 시스템은 주로 간단하고 단순한 순차적인 작업을 처리하였으나 최근에는 마이크로프로세서를 내장하고 운영체제를 탑재하여 네트워크에 연결하거나 멀티미디어까지 지원하는 임베디드(Embedded System) 시스템으로 발전하고 있다. 대부분의 임베디드 시스템은 외부 요청사건에 대해 정해진 특정 시간 이내에 처리 결과를 응답하여야 한다. 즉, 정해진 시간이내에 처리 결과를 응답하지 못하면 시스템 성능에 심각한 저하 요인이 될 수 있을 뿐만 아니라 큰 사고가 발생할 수도 있다.

실시간 임베디드 시스템은 많은 자동화 산업 응용분야에서 응용되고 있다. 특히, 제어 데이터 기록, 모니터 기능 등 시간 제약을 가진 많은 응용분야에서 네트워크로 연결된 시스템에 의해 동작하고 있다[1]. 이러한 실시간 제어 시스템 또는 임베디드 시스템은 컴퓨터 시스템에서 특정한 역할을 수행하는 다양한 하드웨어와 소프트웨어 요소로 구성되어 있다. 실시간 시스템이 시간적 제약조건을 만족하지 못하면 시스템의 오동작이 발생할 수 있으며, 인명 손상과 같은 큰 재앙이 발생할 수 있다. 그렇기 때문에 시스템의 수행은 예측 가능해야 한다.

유연성(flexibility)이란 실시간 시스템 환경에서 작업 타이밍에 대한 장애를 유연하게 처리할 수 있는 시스템의 처리 능력을 나타낸다. 이 요소는 시스템의 수행비용과 성능을 분석하는 중요한 요소이다[2].

실시간 시스템은 시간 제약 응용 프로그램을 결과적으로 명확하게 처리할 수 있도록 설계되어야 한다. 이러한 시스템은 주기적인 작업들로 구성되어 있고 이런 작업들은 일정한 간격을 두고 규칙적으로 발생된다. 또한 실행 완료되는 엄격한 제한 시간을 가지고 있다.

2. 시스템 환경 요소분석

본 절에서는 제어 시스템의 유연성을 계산하기 위하여 시스템 활용도 함수를 도출하고 이를 구하기 위한 환경 요소를 정의한다.

2.1 시스템 환경

시스템 유연성 분석을 위해 제어 시스템의 환경은 다음과 같은 전제조건을 가지고 있다.

- 외부의 태스크(task) 발생과 시스템 제어 처리는 샘플링 비율(sampling rate)에 따라 주기적으로 이루어진다.
- 각 제어 처리는 여러 개의 태스크를 포함할 수 있다. 즉, 적절한 제어 처리 구문은 프로세스 변수를 분석하기 위한 시스템 확인 태스크, 제어 매개변수를 결정하기 적용 태스크, 특정 제어장치에 매개변수를 전송하는 전송 태스크 등으로 구분할 수 있다.
- 각 태스크는 다른 샘플링 비율을 갖거나 다른 통신 채널을 요구하는 모듈로 분해할 수 있다. 각 모듈은 다른 통신 채널과 연계되어 있다.
- 태스크와 모듈간은 상호 선후 관계가 있다. 제어 알고리즘의 처리 모듈은 센서, 구동기의 처리 모듈보다 나중에 실행되고, 이후에는 제어처리 모듈이 실행되어야 한다.

2.2 시스템 동작 및 지연시간(delay time)

실시간 시스템이나 자동 제어 시스템은 3단계의 파이프(pipe) 시스템으로 생각할 수 있다. 즉, 센서로부터 데이터 획득하는 단계, 제어 또는 출력 명령어를 생성하기 위한 데이터 처리하는 단계, 장치 또는 모니터로의 결과 출력하는 단계 등으로 구분할 수 있다[3].

피드백 제어 시스템(feed-back control system)에서는 샘플링 주기 동안 센서링(sensoring), 버퍼링(buffering), 제어처리, 장치구동 등의 순서로 실행된다. 태스크 실행 순서는 장치, 제어, 센서링 순으로 될 수도 있는데, 두 샘플링 주기 이후에 특정 순간에서의 센서 데이터는 구동에 영향을 미칠 수 있고, 폐쇄-루프 시스템(closed-loop system)에서 불필요한 지연을 유발할 수 있기 때문에 시스템 안정성을 떨어뜨리거나 성능을 저하시킬 수 있다. 그러므로 센서로부터 구동까지의 지연은 최소화해야 한다.

시스템에서 발생할 수 있는 지연시간은 각 단계에서의 데이터 전송 지연으로 나타나는데, 각각 다음과 같은 지연이 발생할 수 있다.

- 처리 지연(processing delay) : L_P

제어 알고리즘을 구현한 프로그램을 실행하는데 걸리는 시간을 의미한다. 제어 컴퓨터는 명령어들을 실행함으로써 기초적인 제어 알고리즘을 구현한다. 아날로그 시스템과는 달리 디지털 제어 시스템의 신뢰도는 제어 하드웨어 및 소프트웨어의 MTBF(Mean Time Between Failures) 뿐만 아니라 제어 컴퓨터에서의 제어 알고리즘을 실행하는 지연에 의해 결정된다.

제어 알고리즘의 실행시간은 시작부터 이에 대한 제어

명령어를 생성하는데 걸리는 기간으로 정의된다. 이것은 추가적인 시간 지연으로서 제어 시스템에서는 피드백-루프(feed-back loop)로 알려져 있다.

조건 분기, 자원 공유 지연, 예외처리 때문에 주어진 제어 알고리즘의 실행시간(executing time) 또는 계산시간 지연(computing time delay)은 부분적으로 연속 무작위(piecewise continuous random) 변수로서 일반적으로 이에 해당하는 샘플링 주기보다 작다.

• 버퍼링 지연(Buffering Delay) : L_B

센서를 활성화하고 장치를 구동하기 위해 데이터를 버퍼에 저장하는데 필요한 시간을 의미한다. 시간 t에 수집된 센서 데이터는 그 이후에 제어 처리에서 사용되는데, 시간 $t+L_B$에 프로세스에 영향을 주게 된다. L_B는 폐쇄-루프(closed loop) 시스템에서 순수한 시간지연(time delay)이 된다.

이 길이나 시간 변화는 시스템 성능에 좋지 않은 영향을 준다. 그러므로 L_B나 시간변화는 가능한 작아야 한다.

• 명령 지연(Command Latency) : L_C

오퍼레이터나 호스트 컴퓨터로부터 적절한 처리를 하기 위하여 명령을 받는데 필요한 시간을 의미한다. 간단한 피드백 제어 시스템에서는 이것이 명령이나 입력을 참조하고, 제어 신호를 처리하고, 제어 프로세스를 구동하기 위해 필요한 전체 요청시간을 의미한다. 넓은 의미로는 오퍼레이터로부터 태스크 처리를 위한 명령을 입력받고, 태스크를 처리하고, 제어 프로세스를 구동하는데 걸리는 시간을 의미한다.

• 모니터링 지연(Monitoring Latency) : L_m

수집된 데이터를 처리하고 그 결과를 호스트 컴퓨터에 전송하기 위해 걸리는 시간을 의미한다. 이 지연은 제어 성능에 영향을 주지 않는다. 오히려 모니터링, 운영자의 판단, 명령어 지연 등이 넓은 범위에서 제어 지연을 구성한다.

제어 시스템에 있어서 지연시간의 전체 성능 지표인 L을 피드백 지연, 명령어 지연, 모니터링 지연의 합으로 정의할 수 있다.

$$L=(\sum_{i=1}^{n_f} w_{f_i} L_{f_i})+w_o L_o+w_m L_m \quad (1)$$

w_{f_i} : 피드백 지연 상수,

I : 다중 피드백 루프(loop), $i=1,2,\cdots,n_f$

w_0 : 명령어 지연 상수

w_m : 모니터링 지연 상수

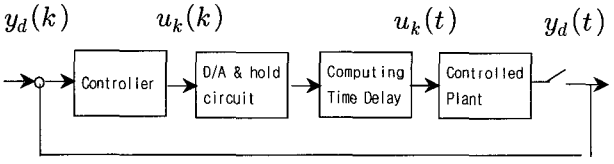
각 상수의 값은 해당 지연상대적인 중요도에 의하여 결정된다. 일반적으로 처리지연이 가장 중요하며 버퍼링 지연, 명령 지연, 모니터링 지연 순으로 중요성을 가진다.

실시간 시스템의 신뢰성은 사용되는 하드웨어와 소프트웨어의 신뢰성 뿐만 아니라 제어를 출력하는 처리에 있어서의 지연시간에 의해서 결정된다. 이는 제어 시스템 성능에 있어서, 계산 시간 지연이 부정적으로 효과가 나타나기 때문이다[5].

고정 샘플링 주기의 시스템 환경에서는 지연 시간이 지연(delay)과 손실 두 가지 문제로 분류된다. 지연의 문제는 지연시간이 0이 아니고, 샘플링 주기보다 작을 때 발생한다. 손실(loss)의 문제는 지연시간이 샘플링 주기보다 크거나 같을 때 발생한다. 예를 들면, 제어 출력을 손실했을 경우가 이에 해당한다.

제어 알고리즘 설계 관점에서는 제어 시스템 안정성과 성능이 계산 시간 지연에 의해 어떻게 영향을 받는지, 특정 제어 알고리즘의 최대로 가능한 계산 지연 시간(computing time delay) 등에 대하여 분석하는 것이 매우 중요하다[6].

제어 시스템에서 허용 가능한 계산 지연 시간(computing time delay)의 크기와 출력 손실의 수(number of output loss)는 디지털 제어 알고리즘의 성능을 평가하는 중요한 지표이다.



(그림 1) 처리 시간 지연이 존재하는 디지털 제어 시스템

3. 알고리즘 구조

RMS(Rate Monotonic Scheduling)는 응답 시간이 짧은 작업의 우선순위를 더 높게 할당하는 스케쥴링이다[4].

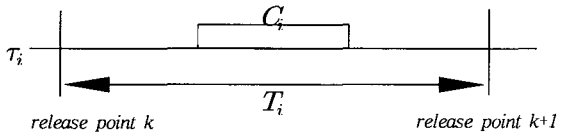
우선순위 할당 방법에는 다음과 같이 두 가지가 있다.

- 평균 응답시간에 의하여 작업 우선순위 부여
- 남아있는 작업 시간이 짧은 작업시간에 의한 우선순위 부여

전자가 후자보다 프로세서 이용도(utilization)가 우

수하다. 단일 프로세서 시스템에서 정적 우선순위, 선점 운영체제 환경에서 RMS는 최적의 효과를 나타낸다[1].

RMS는 작업 요청률에 의해서 우선순위를 부여한다. 즉, 요청률이 높은 작업은 높은 우선순위를 할당받고, 요청률이 낮은 작업은 낮은 우선순위를 부여받는다.



T_i : 작업 발생 주기

C_i : 작업 τ_i의 실행시간

(그림 2) 작업 τ_i의 Release point 및 실행시간

RMS 알고리즘은 매우 안정적인데 이는 최악의 경우 낮은 우선순위의 작업들만 데드라인을 어기기 때문이다. RMS를 적용하여 응용 프로그램을 설계할 때 중요한 작업은 우선순위를 높게 할당하여야 한다. 낮은 우선순위는 주로 백그라운드 작업, 내장된 자가진단, 중요도가 낮은 작업 등에 할당하여야 한다[5].

상호독립적이면서 주기적으로 발생하는 작업군(task set) τ 는 n 개의 작업 $\{\tau_1, \tau_2, \cdots, \tau_n\}$으로 이루어져 있다. 작업 τ_i는 (T_i, C_i)의 튜플로 이루어져 있으며, $1 \leq i \leq n$의 정수 i는 τ_i의 우선순위이다. 즉, 가장 높은 우선순위의 작업은 τ_1이고, 가장 낮은 우선순위 작업은 τ_n이다.

작업 발생주기란 태스크 발생 시점 k(release poin k)로부터 다음 태스크 발생 시점 k+1 (release point k+1)까지의 시간을 의미한다. 작업 τ_i의 데드라인은 작업 발생주기 T_i 끝에 있는 다음 태스크 발생 시점이다. 작업 실행시간 C_i란 다른 작업에게 선점되지 않고 τ_i에게 필요한 실행시간을 의미한다. C_i의 값은 일정하며 시간에 따른 변화가 없다.

RMS는 요청 주기에 의하여 정적으로 우선순위가 할당되기 때문에 짧은 주기를 갖는 작업은 높은 우선순위를 할당받고, 발생주기가 길면 낮은 우선순위를 할당받는다.

작업 τ_i의 프로세서 활용도는 $\frac{C_i}{T_i}$로 정의되며, 전체 프로세서 활용도는 다음과 같다.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{2}$$

모든 작업의 데드라인을 만족시키고 프로세서 활용도를 최대화하는 것은 중요하다. 만일, 모든 작업의 데드라인을 만족시킨다면 그 작업군은 유연성이 있다. 만일, C_i $(1 \leq i \leq n)$ 가 증가하여 작업이 데드라인을 만족하지 못하면 그 작업군은 유연성이 없다.

RMS 알고리즘에 의해 우선순위를 할당받은 작업은 다음의 조건을 만족한다면 유연성이 있다.

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1) \tag{3}$$

실제 제어 시스템의 응용 프로그램은 고정 우선순위, 선점 방식의 운영체제 시스템을 도입한다. 이런 시스템은 일반적으로 동일한 길이의 고정된 간격의 이벤트를 주기적으로 발생하며, 작업 스케줄에 사용된다.

4. 성능분석

만일 모든 태스크를 적절히 스케줄링하고 주어진 프로세서의 실제 처리 성능에 필요한 정확한 처리 성능을 알면 처리시간에 대한 유연성을 도출할 수 있다. 본 논문에서는 프로세서 P의 프로세서 성능에 대한 요구되는 프로세서 성능에 대한 비율을 ρ 라고 정의한다. 그러므로 ρ 는 유연성 메트릭(metric)을 나타낸다.

4.1 Upper Bound Approach

유연성(flexibility)을 계산하는 방법 중 자주 사용되는 방법이 프로세서 활용도(processor utilization)를 기준으로 최고한계치(upper bound)를 구하는 방법이다. 그러므로 본 논문에서도 최고한계치 함수를 이용하여 유연성을 도출한다.

성능분석을 위한 매개변수를 다음과 같이 정의한다.

τ_i : 테스크 i

p_i : 프로세스 주기

d_i : 데드라인

a_i : 구동

c_i : 처리시간

여기서 p_i와 d_i는 다양한 메트릭에 대한 공평하고 일관성 있는 최적의 분산 값을 구하기 위해 사용된다.

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1) \tag{4}$$

$(d_i = p_i, a_i = 0 \ \textit{for all } i)$

$$\sum_{i=1}^{n} \frac{c_i}{d_i - a_i} \leqq n(2^{\frac{1}{n}} - 1)$$

$(d_i \neq p_i, a_i \neq 0 \ \textit{for some } \tau_i)$

즉, ρ의 최고한계치는

$$\rho^u = [n(2^{\frac{1}{n}} - 1)]^{-1} \sum_{i=1}^{n} \frac{c_i}{d_i - a_i} \tag{5}$$

이다.

결국, ρ^u는 값이 작을수록 시스템이 유연성이 더 높다는 가능성을 의미하는 유연성 함수이다. 분명한 것은 $\rho^u \leqq 1$ 이면 시스템은 유연성이 있음을 의미한다. 만일 $\rho^u > 1$이면 더 시스템에 대한 세부적인 분석이 필요함을 의미한다.

4.2 Lower Bound Approach

만일 프로세서 활용도가 U ≥ 1 이면, 작업군(task set)은 제한적으로 유연성이 없다. 따라서 최소한계치(lower bound)를 다음과 같이 정의할 수 있다.

$$\rho^l = \sum_{i=1}^{n} \frac{c_i}{p_i} \tag{6}$$

즉, $\rho^l > 1$이면 시스템은 유연성이 없다. 때문에 시스템은 ρ^l이 [0, 1]사이의 값이면 유연성이 있음을 의미한다.

4.3 Feasibility-Factor Approach

[7]에서는 유연성 평가를 분석하면서 feasibility factor를 정의한 바 있는데, 이는 처리율을 높이기 위하여 요구되는 최소한계치와 최대한계치에 근거하여 도출한 것이다. 본 연구에서는 feasibility factor에 대한 정의를 일반화하여 유연성 함수를 다음과 같이 정의한다.

$$\lambda = \frac{1 - \rho^l}{\rho^u - \rho^l} \quad , \textit{if } \rho^u \neq \rho^l \tag{7}$$

여기서 $\rho^u = \rho^l$ 이면 ρ 를 알 수 있고, $\lambda \geqq 1$이면 프로세서에 할당된 작업은 유연성이 있고, $\lambda < 0$이면 유연성이 없다. 또한 $0 \leqq \lambda < 1$이면 λ만을 이용하여서는 시스템의 유연성을 계산할 수 없다.

5. 결 론

본 논문에서는 실시간 임베디드 시스템의 구조를 설계할 때 효율적인 분석을 위하여 유연성 함수(flexibility metrics)를 정의하였다. 본 연구를 통하여 실시간 시스템의 하드웨어 및 소프트웨어를 분할할 때 속성과 제약을 분석할 수 있다. 제약의 측면에서 보면 유연성 문제를 해결 할 수 있고. 속성의 측면에서 보면 시스템 모델 및 유연성과 다른 속성들(비용, 전력소비량 등)과의 상호관계 분석 등을 효율적으로 분석할 수 있다. 또한 ρ^l 과 λ은 신뢰성 있는 유연성을 나타낸다. 임계 초과 요구율(critical excess requirement ratio)을 $\rho^c = 1 - \rho^l$이라 한다면, ρ^c는 프로세서가 현재의 작업을 충분히 처리하면서 추가적으로 처리할 수 있는 오버로드를 의미한다. λ의 관점에서 보면 $1 - \rho^l$은 $\rho^u - \rho^l$에 비례한다. 임계 초과 요구율은 유연성을 보장하는 시스템을 비교함에 있어 매우 유용하다. 이는 현재의 요구사항 처리가 가능하면서 여전히 가능한 처리용량을 반영하기 때문이다.

참 고 문 헌

[1] C. L. Liu and J. W. Layland, "Scheduling Algorithm for Multiprogramming in Hard Real Time Environment," Journal of ACM, Vol. 20, pp. 46-61, Jan., 1973.

[2] Rajeshkumar S. Sambandam, Xiaobo Hu, "Predicting Timing Behavior in Architectural Design Exploration of Real Time Embedded Systems," Annual ACM IEEE Design Automation Conference Proceedings of the 34th, 1997.

[3] K. G. Shin, Xianzhong Cui, "Computing Time Delay and its Effects on Real-Time Control System," IEEE Trans. on Control Systems Technology, Vol. 3, June. 1995.

[4] R. G. Karl, T. L. Lo, and D. C. Clair, "Effects of Nonsymmetric Release Times on Rate Monotonic Scheduling," 2000.

[5] B. K. Kim, K. G. Shin, "Task Assignment and Scheduling for Open Real-Time Control Systems," Proceedings of the American Control Conference, 1997.

[6] K. G. Shin, C. M. Krishna, and Y. H. Lee, " A unified method for evaluating real-time computer controller and its application," IEEE Trans, Automat. Contr., Vol. AC-30, No. 4, pp. 357-366, Apr., 2001.

[7] X. Hu and J. G. D'Ambrosio, "Configuration-level hardware/software partitioning for real-time embedded

systems", Journal of Design Automation for Embedded Systems, 2002.

채 정 화

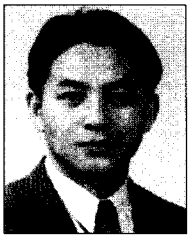
nowcasting@naver.com

1992년 군산대학교 컴퓨터과학과(이학사)

1999년 전북대학교 교육대학원 전자계산교육전공(교육학석사)

2004년 전북대학교 일반대학원 전산통계학과(이학박사)

관심분야 : Web Engineering, Service-Oriented Software Architecture, 객체지향기법 등임



유 철 중

cjyoo@chonbuk.ac.kr

1982년 전북대학교 전산통계학과(이학사)

1985년 전남대학교 대학원 계산통계학과(이학석사)

1994년 전북대학교 대학원 전산통계학과(이학박사)

1982년~1985년 전북대학교 전자계산소 조교

1985년~1996년 전주기전여자대학 전자계산과 부교수

1997년~현재 전북대학교 자연과학대학 컴퓨터과학과 부교수

관심분야 : 소프트웨어공학, 에이전트공학, 컴포넌트기술, 분산객체기술, GNSS(GPS), GIS, 멀티미디어, 인지과학, 교육공학 등임