공정 제어 응용 소프트웨어의 개발을 지원하는 객체 지향 제어 루프 프레임웍의 설계

노 성환*, 전 태웅**
*roh@kucst.korea.ac.kr **jeon@tiger.korea.ac.kr
고려대학교 전산학과

The Design of Object-Oriented Control Loop Framework to Support Process Control Application Software Development

Sung-Hwan Roh, Tae-Woong Jeon
Dept. of Computer Science, Korea University

요 약

제어 루프는 입력값들 만으로써는 정확한 출력값들을 계산하기 어렵거나 불가능한 물리적 공정들을 제어하는 공정 제어 시스템에 핵심적인 구성 요소이다. 본 논문에서는 실시간 공정 제어 응용 시스템의 효율적인 개발을 지원하기 위하여 공정 제어 루프 소프트웨어를 재사용성이 높은 객체 지향 프레임웍으로 설계, 구현한 사례를 기술한다. 본 논문의 제어 루프 프레임웍은 포인트 클래스를 기본 단위로 제어 루프의 공정 변수들과 제어 알고리즘을 캡슐화 하여 다양한 구조와 행위를 갖는 공정 제어 응용 시스템으로 쉽게 개조, 확장이 가능하도록 설계 되었다. 본 논문의 핵심은 공정 변수들에 대한 연속적인 재계산을 수행하는 포인트 객체들의 상호 작용을 통하여 요구된 공정의 감시 제어 기능을 유연하게 구현 할 수 있는 사건/시간 구동적인(event/time-triggered) 포인트 클래스의 설계 패턴이다. 본 제어 루프 프레임웍의 설계에는 Observer, Composite, Strategy, Proxy 등과 같은 객체 지향 패턴들이 사용되었다.

1. 서론

공정 제어 시스템은 끊임없이 변동하는 외부 환경에 적시에 반응하여 공정의 입력 자원으로부터 원하는 형태의 공정 산출물을 만들어내거나 공정의 처리 대상들 사이에 요구된 상호 관계를 유지시키기 위한 공정(들)을 연속적으로 감시, 제어하는 시스템이다. 공정 제어 시스템에서 제어 루프는 입력 값들만으로써는 정확한 출력 값들을 계산하기 어렵거나 불가능하고 출력 값의 계산에 시간 제약이 따르는 물리적 공정들을 감시 제어하는데 핵심적인 구성 요소이다.[4] 본 논문에서는 다양한 형태의 공정 제어 응용 시스템들의 개발에 재사용 가능하도록 제어 루프 소프트웨어를 객체 지향 설계 패턴의 프레임웍[2,3]으로 설계, 구현한 사례를 기술한다.

본 논문의 핵심은 제어 루프의 공정 변수들에 대한 연속적인 재계산을 유연하게 지원하는 포인트 클래스의 설계 패턴이다. 본 논문에서 설계된 제어 루프 프레임웍에서는 여러 제어 루프들의 공통적인 기능을 제공하는 소프트웨어 구조와 행위 부분이 어플리케이션의 개발에 변경 없이 사용될 수 있는 고정 부위(frozen-part)를 구성하고 있다. 제어 루프 어플리케이션은 프레임웍의 가변 부위를 시스템의 요구사항에 맞게 개조, 확장하여 프레임웍의 고정 부위에 합성함으로써 완성할 수 있다.

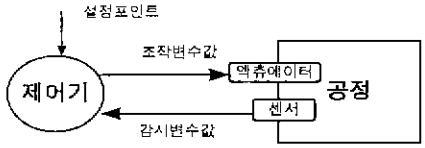
2. 제어루프 프레임웍의 설계

2.1. 제어 루프 모델

공정 제어 시스템은 제어의 대상인 공정과 공정을 제어하는 제어기로 구성된다. 제어기가 감시 제어하는 공정의 현재 상태는 공정 변수들로 추상화하여 표현할 수 있다. 공정 변수들은 제어 변수(controlled variable), 입력 변수(input variable), 조작 변수(manipulated variable) 및 설정변수(reference variable) 등으로 구분된다. 제어 변수는 공정의 제어 대상에 대한 실제 측정값을 나타내는 공정 변수이다. 입력변수는 공정의 직접적인 제어 대상은 아니지만 제어에 필요한 입력 값들을 나타내는 공정변수이다. 제어 변수와 입력 변수는 제어기에 의해 감시 되는 공정 변수, 즉 공정의 감시 변수(monitored variable)로서 사용된다. 조작 변수는 제어기에 의하여 직접적으로 조작, 변경이 가능한 공정 변수이다. 설정변수는 제어변수의 목표 치로 설정된 값(setpoint)을 갖는 변수이다.[4]

공정 제어 시스템의 외부 환경은 시스템과 관계없이 연속적인 물리량으로 스스로, 끊임없이 변동하는 비결정적이고 불안정한 성격을 갖고 있으므로 시스템 입력 값과 시스템 내부 상태 값들 만으로써는 정확한 출력 값을 결정하기 어렵거나 불가능하다. 따라서 설정변수에 의하여 명시된 공정의 산출물들의 요구상태를 만족, 유지시키기 위해 제어기는 설정 변수와 감시

변수 값들을 토대로 조작변수의 값을 계산하여 공정으로 내보내고, 공정은 제어기로부터 입력 받은 조작변수의 값에 따라서 조작변수를 조정한다.[4] 이와 같이 공정 제어 시스템은 제어기와 공정으로 구분된 두개의 컴포넌트들이 상호 비대칭적이고 순환적인 데이타 흐름의 경로를 제공하는 두개의 서로 다른 커넥터들로 연결된 제어 루프 모델로 볼 수 있다.[그림 1]

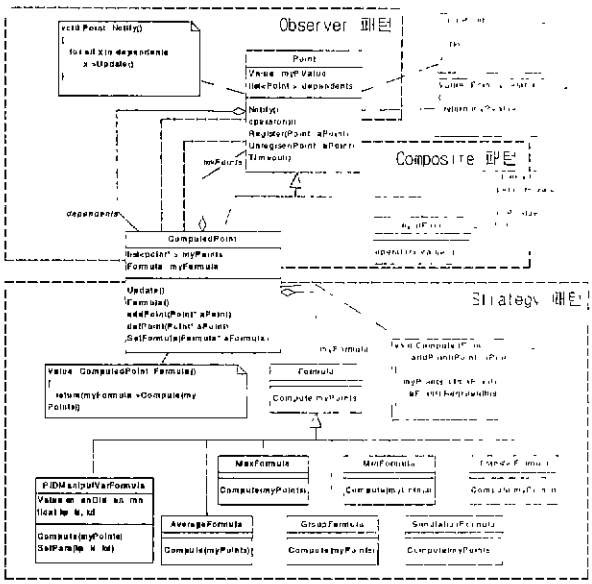


[그림 1] 제어 루프 모델

2.2. 포인트 클래스 설계 패턴

본 논문의 제어루프 프레임웍은 공정 변수를 추상화한 포인트 클래스를 기본 단위로 하여 구성되어 있다. 본 프레임웍의 포인트는 포인트 값이 결정되는 방식에 따라 입력 포인트(input point)와 계산 포인트(computed point)의 두 가지 유형으로 나누어진다. 입력 포인트는 외부의 데이터 소스로부터 입력된 값을 갖는 포인트이다. 계산 포인트는 시스템 내부의 다른 포인트들로부터 계산에 의해 구해진 값을 갖는 포인트이다. 즉, 계산 포인트는 자신의 값을 구하는데 입력 포인트 또는 다른 계산 포인트에 의존하게 되며, 그렇게 의존하는 포인트에 자신을 dependent 포인트로 등록(register)하게 된다. 한 포인트의 값이 변하게 되면 그 포인트는 자신에게 등록된 모든 dependent 포인트들에게 값의 변화를 알린다. 한 포인트의 변화 사실이 dependent 포인트들(dependents)에게 알려지게 되면 dependent 포인트는 자신의 값을 구하는데 필요한 모든 포인트들(myPoints)의 값을 읽어서 자신의 계산방법(formula)으로 포인트 값을 재계산하게 된다. 이러한 방식으로 포인트 값의 변화가 의존 관계로 연결된 모든 포인트들에게 연쇄적으로 전파되면서 재계산이 연속적으로 이루어진다.

그림 2는 위와 같은 행위를 갖는 포인트 클래스와 포인트의 서브 클래스인 입력 포인트와 계산 포인트들의 상호 관계를 보여준다. 그림 2에서 계산 포인트는 자신을 dependent로 갖는 포인트 클래스의 공정 변수 값의 변화에 반응하는 Observer 패턴[2]을 형성한다. 포인트 클래스, 입력 포인트 및 계산 포인트 클래스는 또한 포인트 클래스 객체가 myPoints 참조 변수를 통하여 다른 포인트 객체들을 구성요소로 갖는 계산 포인트 클래스 타입이 될 수 있는 Composite 패턴[2]을 형성한다. 이러한 계산 포인트의 계산 알고리즘은 Strategy 패턴[2]에 따라 동적으로 결합되는 Formula 클래스 객체에 의해 결정된다. 계산 포인트 클래스는 Formula 클래스 객체에 대한 참조인 myFormula을 갖는다. Formula 클래스는 계산 포인트가 가질 수 있는 여러 알고리즘의 공통적인 인터페이스인 Compute 멤버 함수를 제공한다. 계산 포인트는 이 Compute 함수를 이용하여 Formula 클래스의 하위 클래스에 정의된 각 알고리즘을 호출하게 된다.



[그림 2] 포인트 클래스의 설계

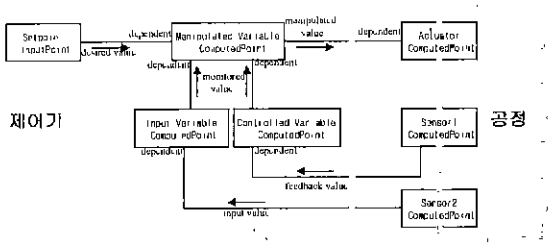
프레임웍을 사용하는 제어 루프 시스템 개발자는 자신의 어플리케이션에 필요한 클래스들을 프레임웍이 제공하는 포인트 클래스 또는 포인트의 서브 클래스들로부터 상속 받아 확장함으로써 원하는 제어 루프 어플리케이션을 구현할 수 있다. 이와 같이 제어 루프 프레임웍을 사용하면 그림 1의 제어 루프 모델을 기반으로한 제어 루프 소프트웨어 컴포넌트의 유연한 설계가 가능하다. 그림 2의 제어 루프 프레임웍으로부터 그림 1의 제어 루프 모델을 구현하는 방식은 다음과 같다. 제어기의 설정 포인트와 공정의 센서는 프레임웍의 입력 포인트의 서브 클래스의 인스턴스로 생성된다. 제어기의 공정 변수들인 입력 변수, 제어 변수, 조작 변수와 공정의 액츄에이터는 프레임웍의 계산 포인트의 서브 클래스의 인스턴스로 생성된다. 제어 알고리즘은 Formula 클래스의 인스턴스로 캡슐화되어 구현되어서 조작 변수 포인트 객체에 동적으로 결합된다.

그림 3은 본 제어 루프 프레임웍으로부터 생성되는 제어 루프 시스템의 전형적인 단일 제어 루프의 포인트 객체 구조를 나타낸다. 그림 3에서 화살표 선은 포인트 값의 변경이 통보되는 방향과 이에 따른 포인트 값의 흐름 방향을 의미한다.

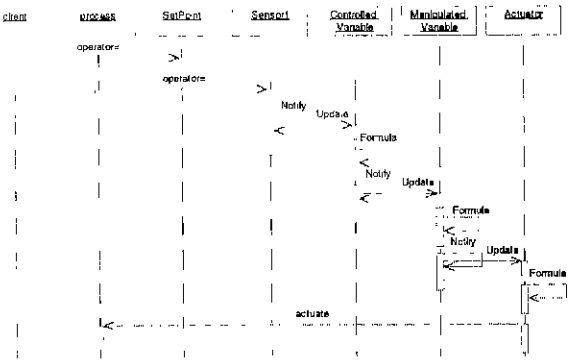
그림 3에서 감시 변수 포인트인 Input Variable과 Controlled Variable은 센서 값의 변화에 반응하여 자신의 값을 재계산한 후 이를 조작 변수 포인트에게 알리고 조작변수 포인트는 다시 이에 반응하여 참조 변수(설정) 포인트 값과 감시 변수 값을 사용하여 자신의 값을 재계산한 후 이를 액츄에이터에게 알리는 연쇄 반응이 일어난다.

그림 4는 setpoint가 설정되어 있을 때 공정의 상태 값을 감

지하여 감시 변수의 값을 트리거 시키는 인터액션 다이어그램의 일부이다.



[그림 3] 전형적인 제어 루프 시스템의 구조



[그림 4] 포인트 객체들의 인터액션 다이어그램

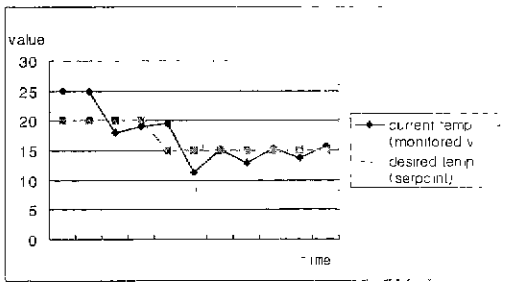
3. 시험 결과

본 논문에서 개발된 제어 루프 프레임웍을 사용하여 방안 온도를 제어하는 시뮬레이션 환경에서의 난방 제어 루프 샘플 어플리케이션을 구현하였다. 그림 5 는 구현된 난방 제어루프 샘플 어플리케이션의 제어 효과에 대한 시험 결과를 나타내는 trend 그래프로써 시간의 흐름에 따른 설정치와 감시 변수 값의 변화의 추이를 보여준다. 그림 5 의 난방 제어 루프 시스템은 액츄에이터, 센서, 제어 변수 및 조작 변수 포인트 객체들이 각각 한개씩인 단일 제어 루프의 예이다. 본 논문의 제어 루프 프레임웍을 사용하면 이외에도 여러 개의 공정 변수들을 갖는 단일 제어 루프 또는 여러 개의 단일 제어 루프들로 구성된 복잡한 구조의 제어 루프 어플리케이션들도 쉽게 구현할 수 있다.

4. 결론

본 논문에서는 객체 지향의 여러 개념들과 디자인 패턴을 응용하여 재사용성이 높도록 공정 제어 시스템에 대한 프레임웍을 설계하였다. 본 제어 루프 프레임웍의 핵심 개념은 공정의 상태 값을 제어 메커니즘과 함께 캡슐화한 포인트 클래스 설계 패턴이다

본 논문의 연구와 관련하여 [5]에서는 공정 제어의 기본 개념을 value 의 연속적인 계산으로 보았다 [6]에서는 여러 종류의 센서와 액츄에이터를 Point 클래스로 추상화 하였다. 본 논문에서는 제어 루프의 각 구성 요소인 제어기, 공정, 센서, 액츄에이터들을 모두 연속적인 계산을 수행하는 공정 변수들로 추상화한 포인트 클래스로 정의하였다. 공정 변수의 값은 구체적인 타입에 관계 없이 연산이 가능하도록 Abstract Class 설계 패턴[1]에 따라 value 클래스로 정의하였다.



[그림 5] 방안 온도 제어 루프 시스템의 제어 Trend 그래프

본 제어 루프 프레임웍의 포인트 클래스들은 Observer 패턴과 Composite 패턴을 사용하여 포인트 객체들 사이의 결합도가 낮게 설계되어 있어서 제어 루프의 여러 포인트 객체들을 쉽게 재구성할 수 있다. 또한 어플리케이션의 실행 중에도 객체의 연결과 분리가 쉽게 이루어진다. 따라서 제어 루프 어플리케이션을 개발하는 프레임웍 사용자는 프레임웍의 포인트 클래스들을 용도에 맞게 개조, 확장하여 프레임웍에 연결함으로써 제어 루프 어플리케이션을 효율적이고 유연하게 완성할 수 있다

대부분의 공정 제어 시스템은 병행 및 실시간 처리를 필요로 한다. 본 논문의 제어 루프 프레임웍은 병행 처리와 실시간 스케쥴링을 지원하는 실시간 커널의 사용을 전제로 하고 있다 따라서 포인트 클래스들이 병행 스레드와 실시간 반응을 직접적으로 지원할 수 있도록 제어 루프 프레임웍을 개선, 확장하는 후속 연구가 필요하다

참고 문헌

[1] Bobby Woolf, "The Abstract Class Pattern"

[2] Erich Gamma and et al, "Design Patterns : Elements of Reusable Object-Oriented Software," Addison-Wesley Publishing Company, 1995

[3] Gregory F. Rogers, "Framework-Based Software Development in C++", Prentice Hall PTR, 1997

[4] Mary Shaw, "Beyond Objects : A Software Design Paradigm Based on Process Control," ACM Software Engineering Notes, Vol 20, No 1, Jan, 1995

[5] Per Dagermo, Jonas Knutsson, "Development of an Object-Oriented Framework for Vessel Control Systems," Technical Report, Dover Consortium 1996

[6] P.Molin and L. Ohlsson, "Points & Deviations – A Pattern Language for Fire Alarm Systems", Pattern Languages of Program Design 3, Addison-Wesley Publishing Company, 1995

[7] Stuart Bennett, "Real-time Computer Control An Introduction, 2nd Edition, Prentice Hall International(UK) Limited, 1994.