Journal of Institute of Control, Robotics and Systems (2012) 18(11):997-1004
http://dx.doi.org/10.5302/J.ICROS.2012.18.11.997 ISSN:1976-5622 eISSN:2233-4335

가변 윈도우 기법을 적용한 통계적 공정 제어와 퍼지추론 기법을 이용한 소프트웨어 성능 변화의 빅 데이터 분석

Big Data Analysis of Software Performance Trend using SPC with Flexible Moving Window and Fuzzy Theory

이 동 헌, 박 종 진*
(Dong-Hun Lee[1] and Jong-Jin Park[2])
[1]SAP Labs Korea TIP
[2]Chungwoon University

Abstract: In enterprise software projects, performance issues have become more critical during recent decades. While developing software products, many performance tests are executed in the earlier development phase against the newly added code pieces to detect possible performance regressions. In our previous research, we introduced the framework to enable automated performance anomaly detection and reduce the analysis overhead for identifying the root causes, and showed Statistical Process Control (SPC) can be successfully applied to anomaly detection. In this paper, we explain the special performance trend in which the existing anomaly detection system can hardly detect the noticeable performance change especially when a performance regression is introduced and recovered again a while later. Within the fixed number of sampling period, the fluctuation gets aggravated and the lower and upper control limit get relaxed so that sometimes the existing system hardly detect the noticeable performance change. To resolve the issue, we apply dynamically tuned sampling window size based on the performance trend, and Fuzzy theory to find an appropriate size of the moving window.

Keywords: performance anomaly, SPC (Statistical Process Control), big data, fuzzy theory, performance monitoring

I. 서론

소프트웨어 제품의 성능 이슈는 지원되는 기능의 구현과 더불어 제품 출시에 영향을 줄 수 있는 중요한 문제이다. 특히 대규모의 엔터프라이즈 소프트웨어 프로젝트에서는 제품을 개발하는 동안, 지속적인 코드 변화에 따른 예기치 못한 성능 저하 문제를 경험하게 된다. 프로젝트의 규모와 복잡도, 개발 인원수와 숙련도 등에 따라 다를 수 있지만, 새롭게 추가되는 코드는 항상 예상치 못한 성능 문제를 야기할 수 있다.

특히 복잡한 기능을 지원하는 엔터프라이즈 소프트웨어의 경우에는 성능 이슈가 발생하는 경우, 원인 파악 및 조치에 많은 비용이 요구되며, 발견 및 조치가 늦어질수록 지속적인 코드 변화로 인해 더 많은 비용이 소요된다. 이러한 문제를 해결하기 위하여 제품을 개발하는 과정에서 지속적으로 다양한 성능 테스트를 수행하게 된다. 특히 DBMS와 같이 다양한 성능 지표를 갖는 제품의 경우에는, 새로운 코드 변화에 대해 수천개 이상의 성능 지표를 확인해야 하며, 하루 수만~수십만개 이상의 성능 지표에 대해 성능 이상 유무를 확인하여야 한다.

이렇게 지속적으로 생성되는 성능 측정 결과는 정형화된 분석 기법이 확립되지 않은 빅 데이터의 성격을 가지며, 관련 개발자들이 일일이 성능 적합성 여부를 눈으로 확인할 수 없기 때문에, 자동으로 성능을 분석하고 의심스러운 성능 변화에 대해 자동으로 리포트 하는 시스템의 구축이 요청된다. 즉, 실시간으로 새롭게 측정된 성능지표의 성능 추이를 분석해서 성능 이상을 감지하고 이상이 있는 경우 개발자에게 즉시 리포트를 발행해서 신속한 조치가 이루어지도록 해야 한다.

소프트웨어 제품의 성능 테스트 수행 및 성능 모니터링을 자동화하는 데에는 많은 이슈들이 존재 한다. 지속적인 테스트가 가능하도록 개발 코드에 대한 주기적인 빌드 및 테스트 시스템을 구성해야 하며, 다양한 테스트 결과를 저장할 수 있는 효율적인 저장 구조가 요구된다. 또한 다양한 유형의 성능 변화에서 성능 이상을 결정할 수 있는 일관된 방안과 성능 이상을 확인하기 위한 효율적인 모니터링 시스템의 구축이 요구된다.

저자는 이전 연구[1,2]를 통해서 성능 이상에 대한 빠른 피드백이 가능한 개선된 소프트웨어 개발 프로세스를 제시하였고, 성능 이상 검출 및 성능 이상의 원인 분석에 소요되는 비용을 최소화 할 수 있는 성능 이상 검출 및 분석 시스템을 구축하였다. 개발 중인 코드에 대한 지속적인 성능 테스트 결과에 대해 통계적 공정 제어(SPC: Statistical Process Control) 차트를 적용하여 자동으로 성능 이상을 검출하였으며, 성능 이상을 결정하기 위한 개별 성능 지표의 기준값을 결정하는 방법을 제안하였고, 얼마만큼의 성능 변화를 성능 이상으로 규정할 수 있는지에 대해 논의하였다.

기 개발된 시스템을 실제 엔터프라이즈 소프트웨어 제품의 개발과정에 적용하면서 처음에 고려하지 못한 다양한 형태의 성능 이상 유형을 발견할 수 있었다. 특히 성능 이상 발생 후 다시 성능이 복구되는 경우에는 성능 이상이 해결되

* 책임저자(Corresponding Author)
논문접수: 2012. 8. 27., 수정: 2012. 9. 14., 채택확정: 2012. 9. 25.
이동헌: SAP Labs Korea TIP(dong.hun.lee@sap.com)
박종진: 청운대학교 인터넷학과(jjpark@chungwoon.ac.kr)

었을 때 기존 방법으로 변화를 제대로 인지하지 못하여 담당자의 지속적인 관찰과 감시가 필요한 경우가 발생한다.

이러한 문제는 기존 연구에 적용된 통계적 공정 제어 기법에서 고정된 크기의 샘플링 구간을 사용함에 따라, 정상 상태의 데이터 집합에 성능 이상에 해당하는 데이터가 유입되면서, 샘플링 구간에서의 데이터 변동성이 확대되고, 이에 따라 SPC 차트 적용을 위한 관리 상 · 하한선의 폭이 확대되어, 성능 이상의 조치에 따른 상태 변화의 검출이 힘들어 지기 때문이다.

본 논문에서는 기존 성능 이상 검출시스템의 성능 이상 검출력 제고 방안으로 기존 시스템에서 제대로 처리하지 못 하는 성능 변화의 유형을 설명하고, 이에 대한 해결 방안으로 퍼지 이론을 활용한 샘플 구간 크기의 자동 조절 방안을 제시한다.

즉, 성능 데이터의 유형에 따라 고려되는 샘플링 구간의 크기를 자동으로 조절해 주는 가변 윈도우 방식을 도입하였으며, 가변 윈도우 구간의 크기를 결정하는 추론 방법으로 퍼지 이론을 이용한 지능형 추론 기법을 사용하였다.

II 장에서는 개발중인 소프트웨어의 성능 측정과 관련한 빅데이터를 다루기 위한 전체 시스템에 대해 설명하고, 기존의 성능 이상 검출 방안을 설명한다. III 장에서는 일반적인 성능 변화의 패턴을 알아보고 SPC 차트의 가변 샘플 구간 적용 및 샘플 크기를 자동으로 추론하기 위한 퍼지 이론의 적용 방안에 대해 기술한다. IV 장에서는 제안된 가변 윈도우 적용을 위한 퍼지 추론 시스템 구성 방안과 경험적 지식 및 성능 데이터에 근거한 퍼지 규칙 설계 방법을 기술하고, 이를 적용한 시뮬레이션 결과를 소개한다. V 장에서 관련 연구를 설명하고 VI 장에서 결론을 맺는다.

II. 연구 배경

1. 성능 변화에 대한 빅 데이터 분석

앞서 언급한 바와 같이 엔터프라이즈 소프트웨어 프로젝트에서는 제품을 개발하는 과정에서 지속적으로 유입되는 코드 변화에 대한 예상치 못한 성능 이상을 확인하기 위하여 수천개 이상의 성능 지표에 대한 하루 수만~수십만개의 데이터를 지속적으로 검사한다.

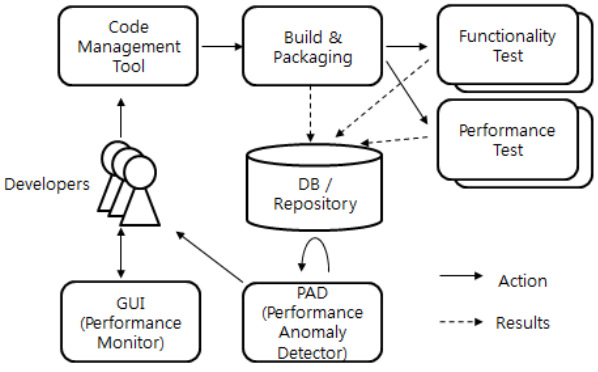

그림 1. 새로운 코드에 대한 지속적인 성능 검사를 위한 시스템 구성.

Fig. 1. System configuration to inspect the possible performance regression against newly added code pieces.

이전 논문 [1]에서 저자는 엔터프라이즈 소프트웨어의 개발 과정에서 성능 이상 검출 및 분석을 자동으로 수행하기 위한 프로세스와 시스템의 구축에 대해 소개하였다.

그림 1에서 보는 바와 같이 개발자에 의해 유입된 코드는 코드 형상 관리 시스템으로 관리되며, 테스트에 사용될 바이너리 구성을 위한 빌드작업이 수행되고, 새로운 코드가 포함된 패키지에 대해 기능 테스트와 성능 테스트들이 수행된다. 수십대의 전용 서버상에서 수백개의 성능 테스트가 분산되어 지속적으로 성능을 측정하며, 테스트 결과들은 관련 데이터베이스 및 저장소에 관리된다. 하루 수만~수십만개의 새로 측정된 성능 변화값들이 PAD (Performance Anomaly Detector)에 의해 자동으로 분석되어 성능 이상이 발견된 경우 관련 개발자들에게 보고된다.

이렇게 성능 변화에 대해 지속적으로 생성되는 빅 데이터는 사람에 의해 일일이 판단될 수 없고, 빠른 피드백을 통한 조치 비용 절감을 위해 더욱 정교한 성능 이상 검출 시스템의 구축이 요청되고 있다.

우리는 앞에서 소개한 시스템을 엔터프라이즈 소프트웨어 개발 과정에 적용하면서, 다양한 형태의 성능 이상들을 파악하였으며, 성능 이상 관리 비용을 획기적으로 줄일 수 있었다. 하지만 더욱 다양한 유형의 성능 변화와 성능이상이 조치된 후 복구된 상태까지 검출하기 위해서는 기존 시스템의 개선이 요구된다. 특히 전체 샘플링 구간내에서 변동성이 큰 데이터 유형을 보이는 경우에 대한 검출 정확성의 제고가 요구된다. 3.2절에서 이에 대해 자세히 설명한다.

2. SPC 차트와 고정 샘플 구간 이동 기법의 적용

저자는 기존 논문[1]을 통해 SPC 차트를 이용해서 소프트웨어의 성능 이상을 효과적으로 검출할 수 있음을 확인하였다. 생산공정에서 SPC 차트를 적용하는 경우에는 산술 평균값을 구하기 위해 사용되는 샘플의 개수가 클수록 더 정확한 확률분포를 갖게 되고, 이상 상황을 더 정확히 감지할 수 있다고 알려져 있다[3]. 하지만 SPC 차트를 소프트웨어 제품의 성능 이상을 검출하는 데에 적용한 결과, 소프트웨어의 경우에는 지속적인 코드의 변화에 따라 일정량의 성능 변화가 계속 유입될 수 있기 때문에, 너무 많은 과거 샘플을 추가하는 경우에는 오히려 표준편차가 커져 최근의 성능변화를 감지하는데에 방해가 되는 경우를 인지하게 되었다.

통계값을 얻기 위해 적절한 크기의 샘플링 구간을 선정할 필요가 있으며, 이전 연구[1]를 통해 대략 40~50개 정도의 샘플을 확보하는 것이 적절함을 보였다.

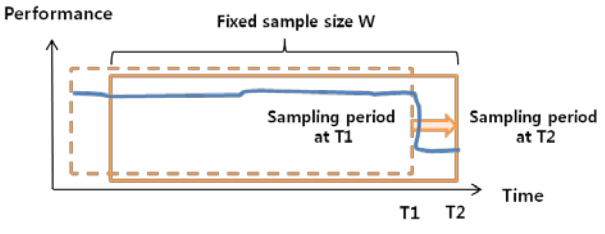

그림 2. SPC 차트적용 시 고정 샘플 구간 이동 기법을 이용한 샘플링 구간 설정 방법.

Fig. 2. Sampling period determination using fixed-size moving window while applying SPC chart.

그림 2는 새롭게 추가되는 성능 데이터와 이를 검출하기 위해 이전 연구[1]에서 도입한 일정 크기의 W를 갖는 고정 샘플 구간 이동 기법을 활용하여 새로운 데이터의 이상 유무를 판단하는 방법을 도식화한 것이다. 즉, 마지막 데이터를 포함하여 일정한 크기 W만큼의 샘플을 이용하여 SPC 차트를 적용하고 성능 이상 유무를 판단한다.

본 연구에서는 일정 크기의 샘플 크기를 이용하는 기존 방식이 성능 이상의 발생 및 조치등을 통해 샘플 구간내에 변동성이 커진 경우 성능 이상의 검출 정확성이 떨어지는 문제점에 대해 논의하고 이에 대한 해결책을 제안한다. III 장에서 이와 관련된 성능 변화 추이 및 문제점에 대해 자세히 설명한다.

III. 성능 변화의 관리 방안

1. 성능 이상 및 조치에 따른 성능 변화 추이 및 이슈 분석

소프트웨어 제품의 성능 이상은 원하지 않는 성능상의 변화를 의미하며, 성능이 나빠진 경우뿐만 아니라 성능이 개선된 경우도 포함되어야 한다. 의도하지 않은 성능 개선은 우리가 소프트웨어 제품의 내부 변화를 제대로 파악하지 못하고 있다는 것을 의미하며, 이러한 변화 역시 중요한 성능 이상으로 정확한 원인을 파악하여 내부 변화를 인지할 필요가 있다.

그림 3은 일반적인 성능 변화의 추이를 단순화한 그림이다. 대체로 특정 성능 지표에 대한 성능 변화는 일정한 범위 내에서 진동을 하듯이 정규분포를 갖으며 변화하다가, 특정 코드가 새로 추가되면서 다양한 크기의 성능 변화가 발생된다. 성능 변화가 발생한 경우, 개발자에게 성능 이상이 보고되고 일정 시간이 지난 후에, 보고되었던 성능 이상을 조치하기 위한 새로운 코드가 추가되어 성능이 다시 복구되는 사각파(square ware) 형태의 모양을 갖는다. 또한 D (성능 감소 크기), G(성능 개선 크기), F(조치기간)의 크기에 따라 다양한 형태의 성능 변화 추이가 가능하다.

그림 3의 (A) 유형의 경우 SPC 차트를 통한 성능 이상 검출이 용이하다. 하지만, (B)와 같이 성능이 복구되는 경우 개

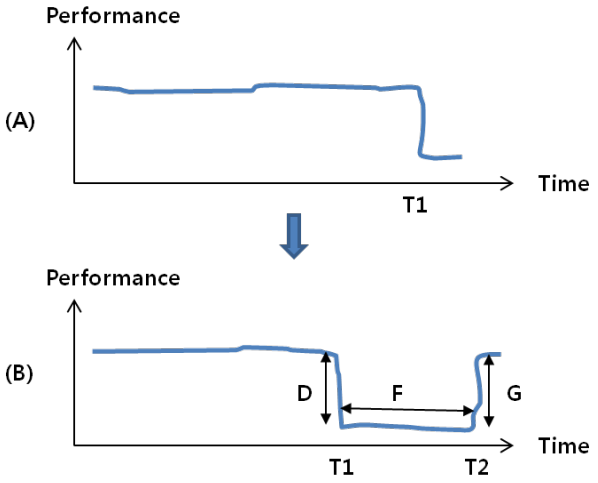

그림 3. 성능 이상 및 조치에 따른 일반적인 성능 변화 추이.
Fig. 3. General performance change by performance regression and recovery.

발자에게 성능 복구에 대한 보고를 하는 것이 바람직하지만, 사각파 형태의 성능 변화 추이에서 조치 기간이 일정 기간 이상 길어지는 경우에는 샘플 구간의 표준 편차가 커져서 SPC 차트 적용시 관리 상 · 하한선(upper/lower control limit)이 커지게 되어 새로운 변화에 대한 검출 변별력이 적어지는 경우가 발생한다. 다음 절에서 이 문제에 대해 좀더 자세히 논의한다.

그림 4는 그림 3에서 설명한 성능 이상 및 조치에 따른 일반적인 성능 변화에 대해, 조치 기간의 변화에 따른 통계값들의 변동으로 SPC 차트의 상 · 하한선이 어떻게 영향받는지 도식화한 것이다. 전체 샘플구간에 해당하는 데이터의 평균값과 표준편차를 이용해서, Shewhart 차트에서 적용되는 상 · 하한선인 표준편차의 2, 3배에 해당하는 관리 기준선이 데이터의 변동성이 커짐에 따라 어떻게 변화하는지 확인할 수 있다.

그림 4(a)는 안정적인 성능을 보이다가 급격한 성능 변화가 발생한 경우로서, 일정 크기의 샘플을 이용한 고정 샘플 구간 이동 기법으로 성능 이상이 검출된 경우이다. 그림 4(b)-(d)는 발견된 성능이상이 비교적 빠른 시간에 조치되어 다시 성능이 이전 상태로 복귀된 경우를 포함하고 있으며, (a)에 비해 조치 기간에 따른 성능 변동성의 확대로 인해 관리기준선이 위 아래로 늘어난 것을 볼수 있다. 성능 이상에 대한 리포트 발행 시 누락되는 성능 이상을 방지하고, 또한 중복된 성능 이상 보고를 줄이기 위하여, 전체 샘플링 구간에서 발견된 모든 성능 이상에 대해 보고하지 않고, 최근 발생된 N개 샘플내에 성능이상이 발견된 경우에만 성능 이상을 보고한다. 이러한 성능 이상 보고 구간은 시스템상에서 테스트별로 설정이 가능하며, 일반적으로 3~5일 혹은 3~5개의 샘플로 국한한다. 이러한 보고 구간 설정 방법은 성능 회복이 발생한 경우 실제 해당 값이 성능 이상으로 검출되지는 않으나, 바로 이전 샘플이 성능 이상으로 검출이 되면 담당자가 쉽게 성능 회복을 함께 확인할 수 있게 되는 간접적인 보고 방법으로 활용된다.

그림 4(e)-(h)의 경우에는 일정 기간이상의 긴 조치 기간이 소요된 경우로 성능 이상에 대한 조치가 성능 데이터에 반영되었지만, 전체 샘플링 구간의 평균값이 낮아지고 표준 편차 값이 커짐에 따라, 관리 상 · 하한선이 커져서 모든 값들이 성능 이상 구간에 포함되지 않는 경우를 도식화한 것이다. 이렇게 일정한 성능 변화가 지속적으로 발생하는 경우 전체 변동성이 커져서 아무런 성능 이상 보고가 발생되지 않는 경우가 발생된다. 이 경우는 그림 4(i)에서 보는 바와 같이 전반적인 성능 데이터가 비교적 큰 변동성을 보여서 성능 이상 검출이 무의미 해지는 경우와 구분될 수 있다. 즉 그림 4의 (e)~(h)와 같은 경우에는 데이터의 변동 특성을 파악해서 샘플링 구간의 크기를 조절 함으로써, 지속적인 성능 변화의 인지가 가능토록 하는 것을 목적으로 한다.

2. 퍼지 이론을 이용한 가변 윈도우 크기 자동 조정

그림 4(e)-(h)의 경우와 같이 조치 기간에 따른 변동성 확대와 이로 인한 통계값의 영향으로 성능 이상이 검출되지 않는 경우에 대해서 아래와 같이 고정된 샘플 구간의 크기를 조절해 주는 방안을 고려하였다. 그림 4(e), (f)의 경우, 즉 그림 5

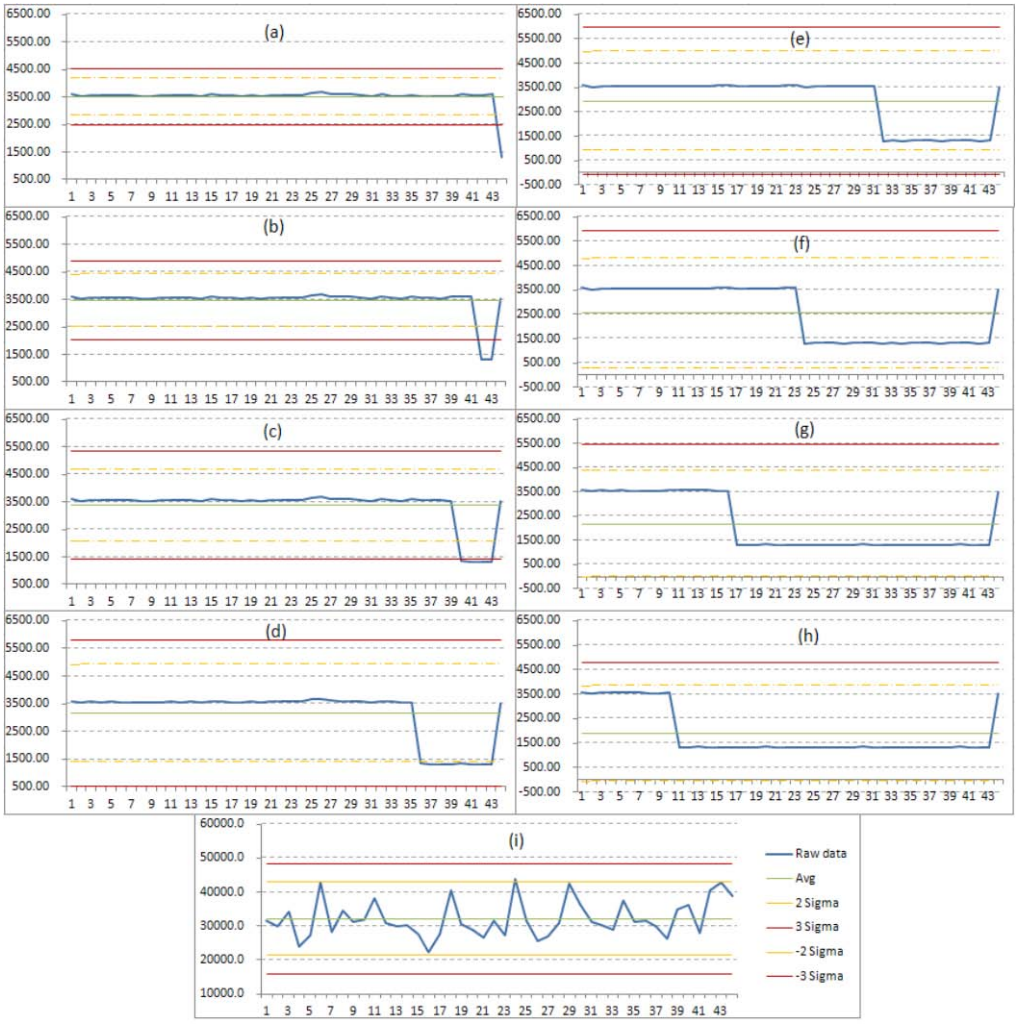

그림 4. 데이터 유형에 따른 통계적 공정관리 기법의 관리 상 · 하한선의 변화.

Fig. 4. Change of the upper/lower limits of SPC chart according to performance data trend.

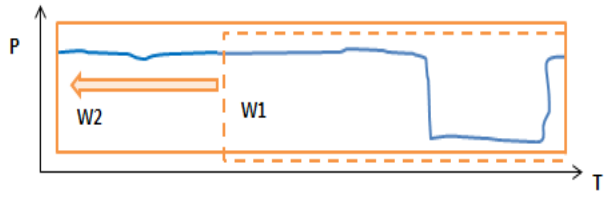

그림 5. 샘플 윈도우 크기 증가를 통한 이상 상황 인지 방안.

Fig. 5. Performance change recognition by increased number of samples.

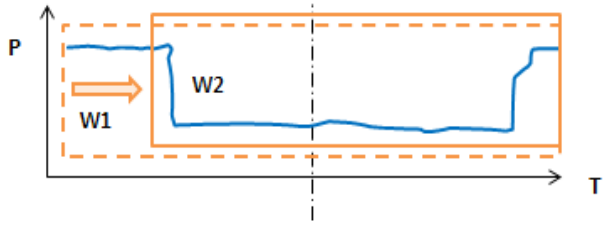

그림 6. 샘플 윈도우 크기 감소를 통한 이상 상황 인지 방안.

Fig. 6. Performance change recognition by decreased number of samples.

에서 보는 바와 같이 조치기간이 너무 길지 않은 경우에는 고정된 샘플 구간, W1을 W2로 늘려주어 좀더 이전의 샘플들을 추가함으로써 전체 변동성을 낮추어 주는 방안이 효과적이다. 즉 늘어난 샘플링 구간에 대해 표준편차의 크기가 줄어들게 되어 SPC 차트의 상 · 하한선이 다시 줄어드는 효과가 있다.

반면 그림 4(g), (h)의 경우와 같이 전체 샘플링 구간에 걸쳐 지속적으로 조치기간이 포함된 경우에는, 그림 6에서 보는 바와 같이 오히려 샘플 구간, W1을 줄여주어 앞쪽의 변동성이 큰 데이터를 줄여서 전체 변동성을 작게 만들어 뒤쪽에서 성능 이상이 조치된 경우를 자연스럽게 성능 이상으로 보고할 수 있도록 유도한다.

위에서 설명한 바와 같이 성능 변화의 특성에 따라 샘플 구간의 크기를 조절함으로써 좀더 정확한 성능 변화의 감지가 가능하나, 샘플 구간의 크기를 얼마만큼 변경해야 하는지가 문제이다. 본 논문에서는 성능 변화의 특성에 따른 정확한 성능 이상의 검출을 위해 사용하는 샘플 구간, 즉 가변 윈도우 크기를 자동적으로 결정하기 위해 퍼지 이론을 적용하였다.

퍼지 이론은 인간이 가진 지식과 경험을 포함하는 제어 알고리즘을 실현하는 방식으로 종래의 수학적인 함수에 의한 제어 방식과 다르게 인간의 제어방식을 모사하는 제어 방식을 구현할 수 있다. 본 논문에서 다루는 소프트웨어 성능 변화의 감시와 성능 이상의 검출과 같이, 발생되는 데이터의 양이 많고 복잡한(complex) 시스템을 다루기 위해서는 전통적인 수학적 방식은 적합하지 않다. 즉, 다루고자 하는 시스템이 복잡하면 시스템을 정확하고 의미 있게 기술하는 것이 불가능해진다. 정확함을 기하려고 하면 시스템 모델의 변수와 파라미터 수가 매우 커지게 되고, 쓸모 없는 모델이 되어 버린다. 따라서 이러한 경우 바람직한 모델은 기존의 정량적인(quantitative) 관계대신 정성적인(qualitative) 관계 모델을 이용하는 것이다. 퍼지 이론에서 사용하는 퍼지 모델은 인간의 경험이나 지식을 정성적인 언어로 표현하고 이를 퍼지 규칙의 형태로 논리화한 것이다[4,5]. 따라서 본 논문에서는 이러한 퍼지 이론을 이용하여 전문가의 경험적 지식과 성능 변화의 샘플 데이터에 근거한 퍼지 추론 시스템을 구성하고 이를 이용하여 각 성능 변화의 특성에 적합한 가변 윈도우의 크기를 자동적으로 추론하도록 하였다.

IV. 가변 윈도우 크기 조정을 위한 퍼지 추론 시스템 구성

1. 퍼지 추론 시스템과 입 · 출력 변수의 선정

서로 다른 변수들이 상호 작용하고 관측되는 신호를 발생시키는 물리적인 시스템에서 이러한 변수들이 서로 어떻게 관계하는가를 이해하는 것은 매우 중요하다. 이러한 관계를 나타내는 시스템의 모델에서 시스템의 복잡성이 증가할수록 이를 나타내는 수학적인 모델은 여러가지 단점을 가지게 되고 이를 통한 시스템의 표현은 의미있고 정확하게 나타내는 것이 점점 어려워진다. 퍼지 추론 시스템은 기존의 수학적인 모델에 의해서는 잘 나타낼 수 없는 복잡하고 잘 정의되지 않는, 그리고 불확실한 시스템을 if-then 형태의 규칙에 의해 잘 나타낼 수 있다고 알려져 있다. 그림 7은 일반적인 퍼지 추론 시스템의 구조를 나타낸다.

퍼지 추론을 이용하여 성능 변화에 따른 적절한 샘플 구간의 크기를 결정하기 위하여 먼저 퍼지 추론 시스템의 입 · 출력 변수를 선정해야 한다. 다양한 변화 추이에 대해서 공통적으로 변화하는 항목을 추출하기 위하여 여러가지 통계값들을 관찰한 결과, 특정 샘플 구간에 대해 전, 후반기의 데이터 추세가 크게 달라지는 점을 주목하고 이에 따른 통계정보들을 이용하였다. 즉 특정 샘플 구간을 반으로 나누어서

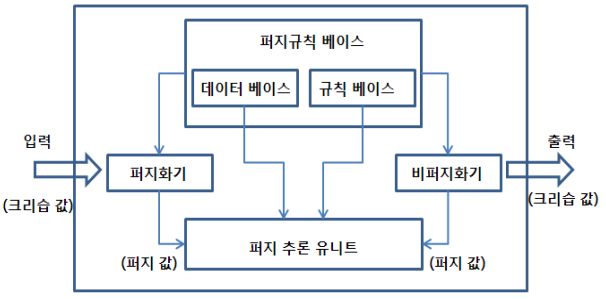

그림 7. 퍼지제어시스템.

Fig. 7. Fuzzy control system.

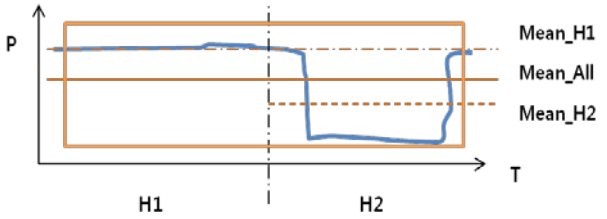

그림 8. 퍼지 변수 결정을 위한 데이터 특성 분석.

Fig. 8. Data trend analysis to determine the Fuzzy member variables.

양쪽의 평균값과 표준편차를 이용해서 변동성을 계산하고 그 값들을 퍼지 추론 시스템의 입력변수로 사용하였다.

그림 8에서 보는 바와 같이 샘플 구간의 전체 평균값 및 표준편차값을 구하고, 또 전체 구간을 반으로 나눈 각 구간, H1과 H2의 평균값(mean)과 표준편차값(stdev)을 구한다. 기존 시스템을 이용해서 검출되지 않는 샘플들의 특성을 분석한 결과, 다음과 같은 세가지 특성값들을 이용해서 해당 성능 변화 유형을 구분할 수 있음을 알게 되었다. 즉 식 (1)로 표현되는 샘플구간의 전반적인 변동성을 통해 성능 이상의 포함여부를 확인할 수 있으며, 식 (2)의 반으로 나눈 각 구간의 평균값의 차이로부터 전, 후반기 데이터 흐름의 기준값의 변화를 알 수 있다. 식 (3)은 구간 H1의 성능 변동성을 나타내는 값으로, 그림 4(e), (f)유형과 (g), (h)유형을 구분하는 데에 도움을 준다. 위 세가지 특성값들은 가변 윈도우의 크기를 자동으로 조정하기 위한 퍼지 추론 시스템의 입력 변수로 선정되었다. 퍼지 추론 시스템의 출력은 가변 윈도우의 크기 (W)가 된다.

$$Fluctuation\_All = 100 * Stdev / Mean\_All \quad (1)$$

$$DM = | Mean\_H1 - Mean\_H2 | / Mean\_All \quad (2)$$

$$Fluctuation\_H1 = 100 * Stdev\_H1 / Mean\_All \quad (3)$$

각 구간내의 평균값(Mean)과 표준편차값(Stdev)은 아래 식 (4)와 식 (5)를 이용해서 구할 수 있다.

$$Mean(\bar{x}) = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (4)$$

$$Stdev(\sigma) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad (5)$$

2. 퍼지 규칙의 설계

가변 윈도우 크기를 자동으로 조정하기 위한 퍼지 추론 시스템에 포함되는 퍼지 규칙은 전문가의 경험적 지식과 성능 변화의 샘플 구간 데이터에 의해 설계되었다. 그림 4에 표현된 데이터 유형에 따른 SPC 차트의 통계값들의 변화와 선정된 퍼지 입 · 출력 변수들의 특성을 고려하여 각 퍼지 변수의 소속함수의 초기값을 결정하였고 시뮬레이션을 통해 그림 9와 같은 결과를 얻었다.

정의된 각 퍼지 입력 변수의 소속함수에 따라 퍼지 규칙을 다음 표 1과 같이 정의하였다. 퍼지 입출력 변수의 조합에 의해 모든 가능한 조건을 검토하였으며, 전체 성능 변동성 (F_All)이 작은 경우에 전, 후반기 데이터의 평균값의 차이 (DM)가 medium이거나 big이 될 가능성이 없으므로 관련된

표 1. 퍼지 규칙.
Table 1. Fuzzy rules.

입출력 변수 규칙	전반부			후반부
	F_All	DM	F_H1	W
Rule 4	Big	Big	Big	Small
Rule 3	Big	Big	Small	Large
Rule 6	Big	Medium	Big	Medium
Rule 5	Big	Medium	Small	Medium
Rule 2	Big	Small	Big	Medium
	Big	Small	Small	Medium
고려 대상 제외	Small	Big	Big	
	Small	Big	Small	
	Small	Medium	Big	
	Small	Medium	Small	
Rule 1	Small	Small	Big	Medium
	Small	Small	Small	Medium

경우들을 규칙에서 제외하였다.

따라서 얻어진 퍼지 규칙은 다음과 같은 형태가 된다.

Rule 1 : If F_All is Small and DM is Small, Then W is Medium
Rule 2 : If F_All is Big and DM is Small, Then W is Medium
Rule 3 : If F_All and DM are Big and F_H1 is Small, Then W is Large
Rule 4 : If F_All, DM, and F_H1 are big, Then W is Small
Rule 5 : If F_All is Big, DM is Medium and F_H1 is Small, Then W is Medium
Rule 6 : If F_All and F_H1 are Big and DM is Medium, Then W is Medium

3. 시뮬레이션 결과 및 분석

설계된 퍼지 규칙과 각 퍼지 변수의 소속함수를 이용하여, 그림 4의 각 성능 변화 데이터 유형에 대해 시뮬레이션을 수행하였다. 시뮬레이션은 Matlab의 fuzzy logic Toolbox를 이용하여 정확성을 높이고 구현 시간을 단축하였다. 표 2는 각 성능 데이터 유형에 대해 퍼지 추론을 적용한 결과로 얻어진 가변 윈도우의 크기 값이다.

그림 4(a)-(c), 그리고 (d)의 경우에는 제안된 퍼지 추론에 의해 기존 시스템에서와 거의 비슷한 윈도우 사이즈가 제안되었으며, 마찬가지로 성능 변화가 성공적으로 검출되는 것을 확인하였다.

그림 10은 각각 그림 4(e)-(g), 그리고 (h)와 같이 기존 시스템상에서 샘플링 구간내의 변동성의 확대로 인해 성능 변화의 감지가 잘 수행되지 않는 경우에 대해, 앞 절에서 설명한 퍼지 추론을 이용하여, 적절한 가변 윈도우의 크기를 구하고, 이를 이용해서 다시 기존 검출 시스템에 적용한 결과이다.

그림 10(e')는 그림 4(e)에 대해 퍼지 추론 결과를 적용한 그림이다. 제안된 퍼지 추론을 이용해서 가변 윈도우 크기로 68개의 샘플이 제안되었으며, 그림에서 보는 바와 같이 이 결과값을 이용한 경우, 3.1절에서 설명한 바와 같이 최근 수개의 성능 이상 보고 구간에 표준편차의 2배에 해당하는 관리 기준선을 초과하는 결과들이 포함되어 담당자에게 전체 차트와 함께 성능 이상 보고가 수행된다. 이를 통해 마지막에 추가된 성능 복구 상황을 인지할 수 있게 된다. 그림 10(f')는 그림 4(f)에 대해 퍼지 추론 결과를 적용한 차트로서, 가변 윈도우 사이즈 100이 적용되었다. 그림 10(g')는 그림 4(g)에 대해 가변 윈도우 사이즈 23이 적용된 결과로서, 퍼지 추론을 통해 정상 상태에서 보다 더 작은 가변 윈도우가 제안되었으며, 이를 통해 복구된 성능이 관리 기준선을 초과하여 성공적으로 성능 변화로 검출됨을 확인할 수 있다. 그림 10(h')는 그림 4(h)에 대한 결과로서 마찬가지로 퍼지 추론을 통해 28개의 샘플로 구성된 가변 윈도우가 제안되었으며, 성공적으로 성능 변화가 검출되는 것을 확인하였다.

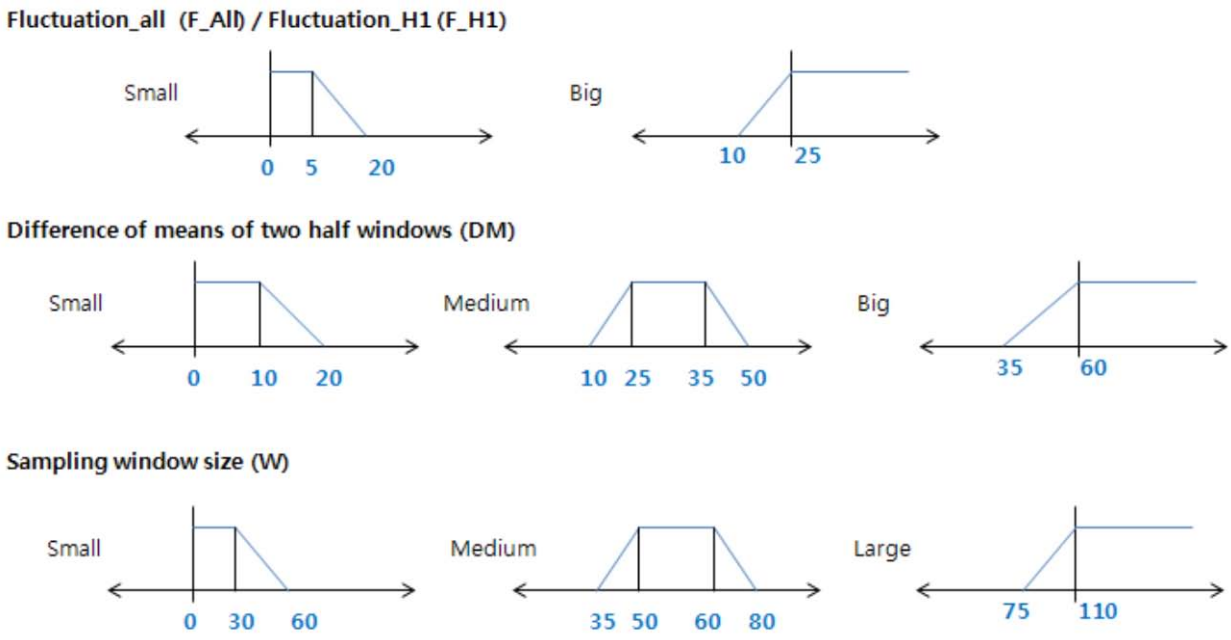

그림 9. 퍼지 소속 함수의 정의.
Fig. 9. Definition of Fuzzy member functions.

표 2. 퍼지 추론에 의한 가변 윈도우 크기 값.
Table 2. Suggested sampling window size by Fuzzy theory.

차트 구분	기대 값	결과 값
그림4-(a)	50 < W	57
그림4-(b)	50 < W	57
그림4-(c)	50 < W	57
그림4-(d)	50 < W	57
그림4-(e)	65 < W	68
그림4-(f)	90 < W	100
그림4-(g)	25 > W	23
그림4-(h)	30 > W	28

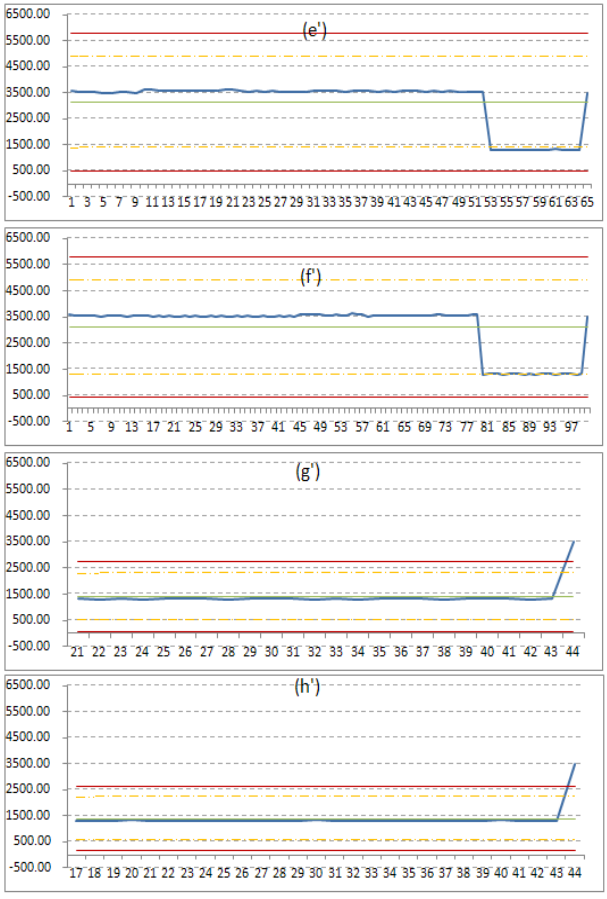

그림 10. 퍼지 추론에 의한 가변 윈도우 크기의 적용 결과.
Fig. 10. Results of flexible sampling window by Fuzzy theory.

V. RELATED WORK

최근 십여년 동안 소프트웨어의 성능 이슈는 소프트웨어 엔지니어링 분야의 핫이슈가 되었으며, 성능 테스트 방법론에 대한 많은 연구가 발표되었다. Scott Barber는 [6,7]에서 소프트웨어의 성능 측정 및 테스트 방법론에 대해 자세히 설명하였다. 소프트웨어의 성능 관리와 관련된 연구 및 노력들은 최근들어 SPE (Software Performance Engineering)이라는 이름으로 하나의 분야를 이루게 되었다. [8]에서 저자는 SPE의 현황과 다양한 이슈들을 잘 정리하였다.

본 논문은 소프트웨어 제품의 개발 과정에서 지속적으로 유입되는 성능 이상을 검출하고 관리하는 방안에 대해 제안하고 있으며, [8]에서 저자가 SPE가 나아가야 할 향후 방향 중 하나로 언급한 '성능 결과를 분석하고 성능 문제를 진단하기 위한 진보된 방법론 및 툴의 개발'과 관련된 연구로 볼 수 있다.

주로 생산 공정의 프로세스 관리를 위해 사용되어 온 SPC는 이미 다양한 control chart들이 개발 및 사용되고 있다. [3]에서 저자는 개개 control chart들에 대해 자세히 설명하고 있으며, 생산 공정 이외의 다른 분야에 SPC를 적용하기 위한 두가지 조건을 설명하였다. [9]과 [10]에서 보는 바와 같이 SPC는 최근에 다양한 분야에 적용되고 있다. 또한 최근들어 소프트웨어 엔지니어링 분야에도 적용되기 시작하였으며, [11]에서 저자는 버그 관리 및 코드 리뷰 프로세스에 SPC를 적용한 경험에 대해 논의하고 있으며, [12]에서는 code coverage와 테스트 에러 개수, 문제의 중요도 등을 활용해서 STP (Software Test Process)를 제어하기 위한 새로운 SPC chart에 대한 연구결과를 보여주고 있다.

VI. 결론

본 논문에서는 기 구현된 소프트웨어 성능 이상 검출시스템의 성능 변화 검출력을 증진하기 위한 방안으로, 기존 시스템에서 쉽게 검출하지 못하는 성능 변화 유형을 설명하고 이에 대한 해결책을 제시하였다. 특히 성능 이상이 발견되고 일정 기간 후에 다시 성능이 복구된 경우, 기존 시스템의 고정 샘플 구간 기법을 적용하게 되면, 샘플 구간내의 성능 변동성이 확대되고 이로 인해 성능 관리 기준선이 이완되어 성능 변화에 대한 검출력이 약화되는 점을 파악하였으며, 이에 대한 해결책으로 퍼지 추론을 이용한 가변 샘플 윈도우 기법을 적용하였다.

퍼지 추론을 적용하기 위하여 기존 성능 모니터링 경험을 바탕으로 샘플 구간의 전체 평균값 및 표준편차값과, 전체 구간을 반으로 나눈 각 구간의 평균값과 표준편차값을 이용하여, 샘플구간의 전반적인 변동성과 반으로 나눈 각 구간의 평균값들의 차이 및 첫째 구간의 성능 변동성을 퍼지 추론 시스템의 입력 변수로 선정하고 가변 윈도우의 크기를 출력으로 선정하였다. 퍼지 입·출력 변수들의 특성을 고려하여 각 퍼지 변수의 소속함수의 초기값을 결정하였고, 세가지 입력 변수에 대한 변동성을 바탕으로 소속함수에 따라 퍼지 규칙을 제안하였다. Matlab을 이용한 다양한 시뮬레이션을 통해 퍼지 변수의 최종 소속 함수를 결정하였으며, 제안된 퍼지추론 결과를 이용한 가변 윈도우 기법을 이용해서 복구된 성능이 관리 기준선을 초과하여 성공적으로 성능 변화로 검출됨을 확인하였다.

제안된 시스템은 끊임없이 코드가 수정되면서 지속적인 성능 변화가 가능한 대규모 소프트웨어 제품의 성능 이상 검출 및 분석뿐만 아니라, 다양한 공정 관리 및 센서 네트워크 관리 등 지속적으로 발생되는 빅 데이터의 변화 상태를 관리하는데에 광범위하게 적용될 수 있다.

본 논문에서는 퍼지 추론의 적용 가능성을 확인하였으며, 향후 좀더 다양한 성능 변화에 대한 데이터를 활용해서 최적의 퍼지 규칙 및 소속함수를 도출해 내고 좀더 종합적이고 정확한 성능 변화 검출이 가능한 시스템 구축 방안에 대해 연구할 계획이다.

참고문헌

[1] D. H. Lee, S. K. Cha, and A. H. Lee, “A performance anomaly detection and analysis framework for DBMS development,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 8, pp. 1345-1360, Aug. 2012.

[2] D. H. Lee, “Performance anomaly detection and management using statistical process control during software development” *Journal of KIISE : Software and Applications (in Korean)*, vol. 39, no. 8, pp. 639-645, Aug. 2012.

[3] D. C. Montgomery, *Introduction to Statistical Quality Control*, 5th Edition. John Wiley & Sons, Inc., 2005.

[4] T. Terano, K. Asai, and M. Sugeno, *Applied Fuzzy Systems*, AP Professional, 1994.

[5] J. J. Park and G. S. Choi, “Fuzzy control system,” Kyowoosa, 2001.

[6] S. Barber, Beyond Performance Testing, http://www-128.ibm.com/developerworks/rational/library/4169.html

[7] S. Barber, http://www.logigear.com/newsletter/explanation_of_performance_testing_on_an_agile_team-part-1.asp

[8] M. Woodside, G. Franks, and D. C. Petriu, “The future of software performance engineering,” *Proc. of International Conference on Software Engineering, 2007 Future of Software Engineering*, pp. 171-187, 2007.

[9] A. de Vries and B. J. Conlin, “Article: Design and performance of statistical process control charts applied to estrous detection efficiency,” *Journal of Dairy Science*, vol. 86, pp. 1970-1984, 2003.

[10] V. S. Puranik, “CUSUM quality control chart for monitoring energy use performance,” *Proc. IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1231-1235, Dec. 2007.

[11] M. Komuro, “Experiences of applying SPC techniques to software development processes,” *ICSE '06: Proc. of the 28th international conference on Software engineering*, pp. 577-584, 2006.

[12] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur, “Monitoring the software test process using statistical process control: a logarithmic approach,” *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 5, pp. 158-167, 2003.

이 동 헌

1993년 연세대학교 전기공학 학사. 1995년 연세대학교 본대학원 석사. 2011년 서울대학교 전기·컴퓨터공학 박사. 1995년~2001년 삼성전자 메카트로닉스 센터. 2003년~현재 SAP Labs Korea. 관심분야는 in-memory DBMS, software performance engineering, quality control, big data analysis 등.

박 종 진

1989년 연세대학교 전기공학 학사. 1991년 연세대학교 본대학원 석사. 1997년 연세대학교 본대학원 박사. 1997년~현재 청운대학교 인터넷학과 교수. 관심분야는 지능시스템 응용, 퍼지 모델링, 임베디드 시스템 등.