제2의 이두문자 시대에 살고 있는 소프트웨어 개발자

양 단 희*

◈ 목 차 ◈

1. 서 론

2000년경을 전후로 Java, Javascript, VC++, Visual Basic, Delphi 등 대부분의 주요 언어들이 한글 명칭을 지원하고 있다.

그런데도 프로그래밍 언어를 배우는 학생들은 변수명으로 a, b, aa, b1, b2 등과 같이 무의미한 알파벳 나열을 선호하고, PhoneUnivDonggi[1], hagBeon[2] 등과 같은 콩글리시 영문을 사용한다. 소프트웨어개발 업계에서조차 표 1과 같이 변수명으로 로마자 표기법, 콩글리시, 또는 이들의 조합을 사용하는 경우가 많다. 앞으로 본 논문에서는 이를 '프로그램용 이두문자'라고 칭하겠다.

(표 1) 프로그램용 이두문자의 예[7]

프로그램용 이두문자	한글표기
WITAKSANAME	위탁사명
DATAGUBUN	데이타분류
DATAGUBUNNAME	데이타분류명
TABALJUM	타발점
SUCHIIN	수취인
CHIGYULIN	취결인

* 평택대학교 컴퓨터학과 부교수

1) 대학동기_전화번호
2) 학번

소프트웨어 총 비용의 80%는 유지보수 비용이라고 한다. 그리고 처음에 그 소프트웨어를 개발한 사람이 소프트웨어의 전 생애 동안의 유지보수를 할 가능성은 거의 없다[8]. 그러므로 더욱 쉽고 올바르게 이해할 수 있도록 코드의 가독성을 높이려는 노력은 소프트웨어의 품질과 생산성 향상, 그리고 개발자의 작업 만족도 향상을 위해 매우 중요하다.

그런데 우리 소프트웨어 개발자들은 조선시대의 한자를 대신한 영문에 의해 제2의 이두문자 시대를 살고 있다. 그래서 본고에서는 이 원인을 분석하고, 한국인은 자신의 모국어인 한국어를 사용하여 한글로 변수명을 표기하는 것이 소프트웨어공학 측면뿐만 아니라 여타 측면에서도 바람직하다는 점을 역설하겠다.

2. 이론적 배경

2.1 Java의 명칭 규칙

① 명칭은 문자, 숫자, $, _로 구성되며 그 길이에는 제한이 없다.

② 명칭의 첫 문자는 문자나 $, _로 시작해야 한다.

③ 문자는 한국어, 중국어, 일본어 문자 등을 포함하며, 숫자와 특수문자를 제외한 유니코드 문자이다.

위의 Java 명칭 작성법[3]을 보면 프로그래머가 자신의 모국어를 사용하여 명칭을 만들 수 있게 해 놓았다[3]. 그리고 명칭은 암호 같은 약어사용을 지양하고 완전한 단어를 사용할 것을 권장하여, 그 길이에 제한을 두고 있지 않다. 이러한 규범들은 코드를 읽기 쉽고, 이해하기 쉽게 만들어, 코드 자체로 문서화의 효과를 얻기 위함이며[8] 그 궁극적인 목표는 소프트웨어 생산성을 높이는 것이다.

2.2 이두와 향찰

이두(吏讀)는 한자에 의한 한국어 표기법으로, 삼국시대부터 시작하여 19세기 말까지 사용되었다.

한사군 시대에 우리나라에 유입된 한자는 우리 민족의 정신을 형성하는 데 지대한 영향을 미쳤다. 우리말은 있으나 그 말을 표기할 문자가 없었던 그 당시, 한자는 우리에게 문자 생활을 영위할 수 있게 해 준 소중한 것이었다. 한자를 통해 비로소 역사를 기록할 수 있었고, 보다 손쉽게 문화를 전파하고 향유할 수 있었다[5].

(표 2) 양잠경험촬요(1415년)의 이두 예
(밑줄이 이두 부분)

한문	蠶陽物大惡水故食而不飮
이두문	蠶段陽物是乎等用良水氣乙厭却桑葉叱分喫破爲遣飮水不冬
한글표기	蠶딴陽物이온들쓰아水氣을厭却桑葉뿐喫破하고飮水안들
현대어	누에는 양물이므로 물기를 싫어해 뽕잎만 먹고 물을 마시지 않는다.

그러나 한자가 중국의 글이기 때문에 한자를 빌려서 문자생활을 영위한다는 것은 쉬운 일이 아니었다. 그래서 사람들은 어떻게 하면 좀 더 쉽게 우리말을 적을 수 있을까 하는 열망을 늘 가졌었다. 그 열망이 이루어 낸 독창적인 한자 사용법이 이두이다. 이두는 훈민정음이 반포된 조선 시대까지 이어져, 오랫동안 우리 조상들의 문자생활의 한 부분을 담당해 왔다. 이두는 표 2와 같이 한자의 음과 훈을 빌려 한국어를 적던 표기법이다[5].

한편 향찰은 이두 표기법이 최고로 발달한 형태로, 한문과는 관계없이 한자의 음과 뜻을 빌려 우리말을 전면적으로 표기하는 문자 체계이다. 그러나 이두든 향찰이든 어려운 한자를 배워야 한다는 사실에는 변함이 없었다.

2.3 유니코드에서의 한글

유니코드(Unicode)는 국제 표준으로 제정된 2바이트계 만국 공통의 국제 문자 세트(UCS : Universal Code Set)이다. 미국의 애플사, IBM사, MS사 등이 제안하여, 이를 추진하기 위해 1989년에 설립된 유니코드 협회(Unicode Consortium)에서 제정한다.

유니코드 협회는 1991년 말에 유니코드 1.0을 발표하였는데, 여기에는 KSC 5601-1987 완성형 코드로 한글 2,,350자만을 포함시켰다. 그래서 국내에서는 환영받지 못하였고, 1995년에 이르러 유니코드 2.0을 제정하면서 한글 1, 1172자 모두를 수용하였다. 이를 국내에서는 KSC 5700이라는 칭하고 국가표준으로 정하였다.

유니코드의 목적은 기존의 잡다한 문자 인코딩 방법들을 모두 유니코드로 교체하려는 것이다. 기존의 인코딩들은 그 규모나 범위 면에서 한정되어 있고, 다국어 환경에서는 호환되지 않는 결정적인 문제점이 있었다.

유니코드는 이 문제를 해결하기 위해 코드의 한 문자당 영어는 7비트, 비영어는 8비트, 한국어, 일본어, 중국어는 16비트의 값을 지녔는데, 이를 모두 16비트로 통일하였다. 그리고 전세계에서 사용하고 있는 26개 언어의 문자와 특수기호에 대해 일일이 코드값을 부여하였다.

유니코드가 수용할 수 있는 최대 문자수는 65,536자이다. 코드 할당비율을 보면 한자가 39.89%로 가장 많고, 한글 17.04%, 아스키 및 기호문자 10.39% 등의 순이다.

코드 통합에 가장 지장을 준 언어들이 한국어, 중국

3) 유니코드가 업계에 정착된 이후 다른 프로그램 언어도 이와 유사한 규정을 두고 있다.

어, 일본어와 같이 글자 수가 많은 동양권 언어였는데 유니코드 협회가 국가간 획일적으로 균등한 코드 영역 배분에 연연하지 않고, 다양한 문자 집합들을 통합하는 데 성공하면서 유니코드는 소프트웨어의 지역화L10N: Localization)와 국제화(I18N: Internationalization)에 널리 사용되게 되었다[4].

3. 프로그램용 이두문자의 배경과 문제점

3.1 한글코드의 변천사

1967년에 국내에 컴퓨터가 도입된 이래 국내 실정에 맞는 소프트웨어를 개발하려면 한글 처리가 필수였으나, 그 코드 표준이 정립되어 있지 않아 각 컴퓨터 회사마다 임의로 한글코드를 만들어 사용했었다[6].

그러다가 1974년에 완성형 코드 방식으로 KSC 5601-1974가 표준으로 제정된 이후 완성형 코드와 조합형 코드간의 기나긴 학술적 논쟁이 있어 왔다. 이후 완성형 코드로 다시 1987년에 KSC 5601-1987을 표준으로 발표하였으나 업계에서 인정받지 못하고 지속적인 논란을 불러 일으켰다.

그 당시 국내뿐 아니라 세계적으로도 문자코드의 국제 표준화는 컴퓨터계의 큰 골치 덩어리였다. 특히 소프트웨어 최강국인 미국 입장에서는 자국의 소프트웨어를 전 세계에 쉽게 판매할 수 있도록 문자코드의 국제적 표준화를 추진하는 것은 당연한 일이었다.

이러한 상황 속에서 1995년 유니코드 2.0이 출범하였으며, 비로소 운영체제, 컴파일러, DBMS 등 각종 소프트웨어에서 한글이 국제적으로 일관되게 지원될 수 있게 되었다.

3.2 제2의 이두문자 출현

일제 시대에는 일본의 한민족 말살 정책에 의해 한글과 우리말 사용이 금지 되어 섰다. 그런데 유니코드가 나오기 전까지 소프트웨어 개발 현장에서는 스스로 한글 사용을 금지할 수밖에 없었다.

미국권에서 개발된 수많은 소프트웨어에서 한글 입출력을 지원하지 않아 한글은 그야말로 천덕꾸러기 그 자체였다. 한글 파일명과 한글 변수명을 사용할 수 없다는 것은 소프트웨어 개발자에게 상당한 스트레스로 작용하였다. 세계적으로 제일 우수하다는 평가를 받고 있는 우리글, 한글이 있으되, 소프트웨어 개발에서는 프로그램용 이두문자를 쓸 수밖에 없는 상황은 우리 한글에 대한 자괴감을 주기에 충분했다. 어떤 프로그램이 제대로 동작하지 않은 이유를 각고의 노력 끝에 알아냈더니 그 원인이 결국 한글 사용이었을 때 어떤 기분이 들었을까?

이렇게 한글이 아예 지원되지 않았거나, 이런저런 문제를 야기 시켰던 시절을 경험해온 사람들은 어쩔 수 없이 프로그램용 이두문자를 어렵게 터득하여 이제 어느 정도 달인의 경지에 올라 있게 되었다. 그래서 그 사람들이 프로그래밍 교재를 만들거나 가르칠 때, 혹은 직장에서 소프트웨어 개발 팀장으로 일할 때 프로그램용 이두문자를 사용하고 권장하는 것은 당연한 일이었다.

위와 같은 사유로 지금까지 상당수의 프로그래머들이 변수명으로 영문자만 사용해야 하는 것으로 알고 있거나, 이론적으로는 가능하다는 것을 알지만 한글명칭 사용을 피하는 것이 올바른 엔지니어로서의 자세로 인식하게 되었고, 결국은 '프로그램용 이두문자 시대'가 고착화 되었다.

3.3 프로그램용 이두문자의 문제점

이두문자의 사용은 소프트웨어 생산성을 떨어뜨리고, 개발자의 작업 만족도를 떨어뜨리고, 유지보수하기 쉬운 소프트웨어 개발과는 멀어져, 소프트웨어공학의 근본 취지에 반하게 된다. 세부적으로는 다음과 같은 문제점을 가지고 있다.

① 업무현장에서 사용하는 용어와 프로그래머가 사용하는 용어 간에 괴리감이 있다.

② 이두문자형의 변수명을 고안하기 위한 시간소모가 크다.

③ 프로그래머마다 용어에 대한 표기법이 제각기 달라 의사소통에 지장을 준다.
④ 위와 같은 이유로 변수명에 대해 우리말로 해석 과정이 필요하여 가독성과 이해력을 크게 떨어뜨린다.

4. 한글 명칭 사용의 당위성

4.4 한글 기피 요소의 해소

4.1.1 현지화와 국제화의 시대 도래

2000년대 들어 소프트웨어의 성공적인 해외 진출을 위해 현지화(L10N: Localization)와 국제화(I18N: Internationalization)에 기반하여 개발되지 않으면 해외에서 외면 받을 수밖에 없다. 이것은 영어권에서 생산된 소프트웨어도 마찬가지로 세계 시장을 겨냥한 소프트웨어일수록 이 문제에 철저할 수밖에 없다.

소프트웨어 현지화란 수출 대상 국가에 특화된 언어, 문화 등을 반영해 이를 현지 환경에 적합하도록 개발하는 것을 의미한다. 소프트웨어를 현지화하기 위해서는 수출 대상 국가 언어로 번역하고 시간·숫자·주소 표시 체계 등과 같은 현지 문화를 반영해야 한다.

소프트웨어 국제화란 소프트웨어 수출 대상 국가가 변경될 때, 해당 국가의 언어, 문화 등을 쉽게 반영할 수 있도록 소프트웨어를 유연하게 개발하는 것을 말한다. 소프트웨어를 국제화하기 위해서는 다국어 지원, 소스코드와 데이터 분리, 텍스트와 그래픽 등을 분리해야 한다.

4.1.2 호환성 문제

운영체제, 컴파일러, 편집기 등 인기 있는 소프트웨어는 경쟁력을 높이기 위해 소프트웨어 현지화와 국제화 전략을 적극 구사하고 있으며 그 근간에는 유니코드가 있다. 이와 같은 세계적 상황에서도 한글로 인한 호환성 문제가 대두된다면 소프트웨어 개발 최종 단계에서 한글을 영문으로 일괄적으로 치환해주는 소프트웨어[4]를 통해 해결하는 것이 바람직하다[7].

국제적 기술 교류 문제

국제적인 기술 교류를 위해 개발자 각자가 우수한 영어 구사력을 가져야 하고, 이를 위해 프로그램 작성 때 영어를 사용하도록 노력해야 한다는 주장은 우리나라의 시인, 수필가, 시나리오 작가 등이 국제적으로 인정받기 위해서, 혹은 문화적 교류를 위해 영어로 작품을 써야 한다는 주장과 다를 바 없다. 이 얼마나 허망한 주장인가?

더구나 요즘은 정말로 가치 있는 기술들은 순식간에 한글로 번역되어 소개되고 있으며, 웹상에서 2007년에 처음으로 선보인 '구글번역기'의 성능은 날로 좋아져 현재 매우 높은 번역 품질을 보이고 있다. 아무튼 이 추세대로 나간다면 인터넷 상의 언어 제약은 머잖아 구글번역기와 같은 기계번역(machine translation) 소프트웨어에 의해 완전히 해소될 것으로 보인다.

이렇게 언어 장벽이 사라져 가는 추세 속에서 소프트웨어 개발자들은 '기회비용'[5] 개념에 따라 영어 구사력에 관심을 두는 대신 전공분야에 열성을 다하는 것이 바람직하다.

4.2 소프트웨어공학 원칙에 충실

소프트웨어공학은 소프트웨어 제품의 품질을 향상시키고, 소프트웨어 생산성과 작업 만족도를 증대시키는 것이 목적이며, 그 궁극적인 목표는 최소의 비용으로 계획된 일정보다 가능한 빠른 시일 내에 소프트웨어를 개발하는 것이다.

코드 품질의 기준으로서 가독성(readability)과 가해성(comprehensibility)[6]은 아무리 강조해도 지나치지

4) native2ascii - Native-to-ASCII Converter : Converts a file with characters in any supported character encoding to one with ASCII and/or Unicode escapes, or visa versa.
5) 한정된 자원으로 경제활동은 다른 경제활동을 할 수 있는 기회의 희생으로 이루어진다. 예컨대 자금이 한정되어 있을 때 탱크를 만드는 것은 공장을 짓는 기회의 희생이라고 볼 수 있다.

않는다. 모든 사람은 모국어로 소통할 때 가장 편안한 마음이 들고, 맥락을 이해하는 능력을 최고치로 발휘할 수 있다. 모국어는 '번역'과 같은 사고과정을 거치지 않고 자신이 의식하지 않아도 자연스럽게 자신의 의사를 표현하고 이해할 수 있는 논리체계이다.

그러므로 업계에서 사용되고 있는 용어를 그대로 소프트웨어 개발 시 사용하는 것이 절대적으로 바람직하다. 즉 우리말 단어가 존재하면 우리말 단어로, 아니면 영단어를 그대로 사용하면 되는 것이다. 간혹 이해를 돕기 위해 원어 표기가 필요하면 영문으로 주석처리를 하는 것이 좋다.

4.3 한국인의 정체성과 자긍심

우리나라는 중국이 세상의 중심이라고 여겼던 시대에는 한자를 숭배하였고, 2차 세계대전 이후 이제 미국이 세상의 중심으로 여겨지자 영어 숭배로 전환된 감이 있다.

우리나라의 경제가 커지면서 1996년 에 OECD에 가입하고 한류(韓流)가 활발하게 해외에 번지면서 우리 문화에 대한 자부심이 크게 늘어났다. 그럼에도 우리는 왜 영어가 새겨진 제품을 고품격으로 보고, 한글이 새겨진 제품을 촌스럽게 간주하는 경향이 있는가? 왜 TV 광고를 보면 끝마무리로 미국인 이 영어로 멘트를 하는가?

모국어 문화가 중요한 이유는 바로 우리의 정체성과 직결되기 때문이다. 해외에서 모국어를 사용하는 자녀들과 그렇지 않은 아이들과 비교하면 한국인이라는 자긍심에 심한 격차를 보인다. 그렇다면 소프트웨어 개발자들이 프로그램용 이두문자를 써야 하는 상황에서 한글에 대한, 한국어에 대한 자긍심이 존재하겠는가? 한국인임을 자랑스럽게 생각할 수 있겠는가?

6) 이해할 수 있는 정도. 시각적 이해 외에 내용의 이해가 포함된다. 문자는 가시성, 가독성뿐 아니라 가해성이 필요하다. 가독성은 문자나 문장을 읽을 수 있는 정도의 가해성을 포함한다.

4.4 한글의 우수성

외국 언어학자들의 한글에 대한 예찬은 거의 절대적이다. 96년도에 프랑스에서 세계 언어학자들이 한 자리에 모이는 학술회의에서 한국어를 세계 공통어로 쓰면 좋겠다는 토론도 있었다고 한다. 한겨레 논설위원 곽병찬은 다음과 같이 한글의 우수성을 얘기했다[1].

① 1997년 10월 1일, 유네스코가 세계기록유산으로 지정한 문자는?

② 1998년부터 2002년 말까지 유네스코는 말뿐인 언어 2,900여종에 가장 적합한 문자를 찾는 연구를 했는데, 여기서 최고의 평가를 받은 문자는?

③ 유네스코가 문맹퇴치 기여자에게 주는 상의 이름은 어떤 문자를 염두에 두고 지어졌나?

④ 문맹률이 세계에서 가장 낮은 나라에서 사용하는 문자는?

⑤ 일본의 오사카시는 엑스포 기념 세계민족박물관을 지어 세계의 문자를 전시했는데, 이 가운데 '가장 과학적인 문자'라는 설명이 붙어 있는 문자는?

⑥ 언어학 연구에서 세계 최고라는 영국 옥스퍼드 대학교 언어학대학이 합리성, 과학성, 독창성, 실용성 등의 기준에 따라 점수를 매긴 결과 1등을 차지한 문자는?

⑦ 컴퓨터 자판에서 모음은 오른손으로, 자음은 왼손으로 칠 수 있는 유일한 문자는?

⑧ 이동전화의 한정된 자판을 가장 능률적으로 운용할 수 있어 디지털시대의 총아로 떠오른 문자는?

5. 세종 당시 한글 사용의 반대론과 현재

이렇게 제반 여건이 소프트웨어 개발에서 한글 명칭 사용이 바람직함에도 불구하고, 이에 대해 반대하는 사람들은 존재하기 마련이다. 그래서 세종대왕이 한글 창제를 할 때 한글에 대해 반대론을 편 사람들의 주장과 그 주장이 역사를 돌이켜 볼 때 얼마나 허망한 것인지를 살펴 보겠다.

5.1 기득권 유지에 대한 불안감

세종 당시 양반들은 한글을 통해 평민이나 하인들과 의사소통을 원활히 할 수 있다는 장점보다는 문자에 대한 기득권 상실로 인해 흔들릴 수 있는 신분제에 대한 불안감이 더 컸다.

현대에 들어서도 일부 법조인들은 법전에 있는 용어들을 한글화하는 것에 반대하고 있다. 그리고 의약분업 전까지 상당수 의사들은 라틴어로 처방전을 쓰면서 자신들을 일반인들과 차별화시켰다. 다른 전문직 종사자 역시 자신들만이 알 수 있는 전문용어[7]를 거침없이 사용하면서 자신들이 특권층임을 과시하는 수단으로 사용하고 있다[2].

이것은 조선시대 양반이 훈민정음을 놔두고 300년 넘게 한자 사용을 고수하면서 지식과 권위를 독점해 온 것과 같은 맥락이며, 그 대가를 한일합방이란 치욕으로 혹독히 치루지 않았나 싶다.

5.2 중국/미국에 대한 철저한 사대주의

세종 당시 중국은 천하의 중심이고, 중국의 글자인 한자를 쓰는 것은 당연한 것이며, 한자 외의 다른 글자를 쓰는 것은 오랑캐가 되는 것이라고 생각했으며, 중국 이외의 외국은 전혀 관심의 대상이 아니었다.

현대에 들어서는 제2차 세계대전 이후 미국이 세계의 중심으로 굳건히 들어서게 되고, 6·25 전쟁을 치루면서 수 천년에 걸쳐 그토록 존중받아 오던 한자는 천덕꾸러기 신세로 전락하고, 한자의 자리를 영문자가 대신하게 되었다.

5.3 중국/미국과의 학문적 교류 장애에 대한 우려

세종 당시 한글이 이두보다도 더 비속하고, 그저 쉽기만 한 것이라 어려운 한자로 된 중국의 높은 학문과 멀어지게 만들어 우리 문화수준을 떨어지게 할 것이라고 우려하였다.

현대에 이르러서는 논문이나 인터넷을 통한 국제적인 학문적 교류가 모두 영어로 이루어지고 있기 때문에, 한글 명칭을 사용하게 되면 영어를 등한시하게 되어 국제적 기술 수준과 멀어지게 될 것이라고 주장하는 사람들이 있다.

6. 맺는말

유니코드가 나오기 전까지 그리고 소프트웨어의 현지화와 국제화가 시대적 대세가 되기 전까지 프로그램용 이두문자의 사용은 불가피한 측면이 있었다. 그러나 지금에 와서도 소프트웨어 개발 시 한글 명칭을 피해야 한다고 주장하는 것은 조선시대에 한글 사용을 반대한 사람들의 주장과 매한가지일 뿐이다.

프로그램용 이두문자의 사용은 프로그래머가 프로그램 로직에 집중해야 할 시간을 빼앗고, 엉뚱하게도 영단어 명칭을 고안하는 데 과도한 시간을 낭비하게 만든다. '언어가 사고를 지배한다'는 명제에 근거하여 엉뚱하게 작성된 이두문자형 명칭은 프로그램 작성 내내 사고과정을 어지럽히고, 특히 학생들에게 프로그래밍은 매우 난해한 것이라는 인식을 심어주게 된다. 또한 한글에 대한 부정적인 이미지를 심어줘 한국인으로서의 자긍심을 손상시킬 수 있다.

결과적으로 소프트웨어 생산성 향상을 위해, 개발자의 작업 만족도 향상을 위해, 유지보수의 용이성을 위해 프로그램용 이두문자는 우리에게서 조속히 사라져야 할 대상이다. 이를 위해 프로그래밍 교재를 쓰거나 가르치는 사람들, 그리고 개발 현장에서는 소프트웨어 개발팀장들의 적극적인 역할이 무엇보다도 중요한 때이다.

참 고 문 헌

[1] 곽병찬 기자, 유레카 칼럼, 한겨레신문, 2005. 10. 9
[2] 이철호 기자, 심장정지를 '카디악 어레스트'라고 해야 권위 있어 보이나, 동아일보, 2014.

7) 프로그램용 이두문자도 이 부류에 속할 수 있다.

[3] 자바 규약, http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html
[4] 위키백과, “유니코드”, http://ko.wikipedia.org/wiki.
[5] 위키백과, “이두”, http://ko.wikipedia.org/wiki.
[6] 전상훈, 한글 및 한국어 정보 처리 코드(한국어 정보처리의 과제), 월간 마이크로소프트, 1998년 11월호.
[7] 토론방, “한글변수 사용에 대하여”, http://www.javaservice.net/~java/bbs/read.cgi?m=resource&b=discussion&c=r_p&n=1141377328.
[8] Java Site, “Why Have Code Conventions”, http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-139411.html.

◐ 저 자 소 개 ◑

양 단 희

1989년 연세대학교 전산과학과(이학사)
1991년 연세대학교 대학원 전산과학과(이학석사)
1999년 연세대학교 대학원 컴퓨터과학과(공학박사)
1991년~1995년 현대전자 S/W 연구소
2001년~현재 정보과학회/정보처리학회/인터넷정보학회 논문지 심사위원, 인터넷정보학회 학회지 편집위원
2001년 3월~현재 평택대학교 컴퓨터학과 부교수
관심분야: 멀티미디어, 컴퓨터보안, 자연어처리, 기계학습, 정보/의미 분석