Dream + Think + Idea + come true

COMMUNICATIONS OF SOFTWARE ENGINEERING SOCIETY

12/2021 VOL.1 NO.1

소프트웨어공학 소사이어티 소식

SOFTWARE ENGINEERING SOCIETY

Contents

COMMUNICATIONS
OF SOFTWARE ENGINEERING SOCIETY

소프트웨어공학 소사이어티 소식

12/2021
VOL.1 NO.1

SOFTWARE ENGINEERING SOCIETY

• 발행인 홍장의 • 편집인 최윤자, 이주용, 지은경 • 발행일 2021.12.15

SOFTWARE ENGINEERING SOCIETY

Dream + Think + Idea + come true

인사말

■ 홍장의 교수

안녕하십니까. 소프트웨어공학 소사이어티 회장 홍장의 입니다.

소프트웨어공학 소사이어티는 국내외 소프트웨어공학 분야의 연구, 개발, 그리고 산업 적용을 목적으로 소프트웨어공학에 관심을 가진 모든 사람들에게 열려있는 학술 단체로써, 학문적, 기술적 교류는 물론 사회적 교류를 위해 다양한 노력을 기울이고 있습니다.

이러한 노력의 일환으로 소프트웨어공학 소사어티의 다양한 소식을 전해드릴 소프트웨어공학 소사이어티 소식지(Communications of the Software Engineering Society)를 새롭게 창간하게 되었습니다. 이 소식지는 소프트웨어공학 분야의 연구 동향과 구성원의 다양한 소식을 전해줄 메신저 역할을 수행함으로써, 확장하고 있는 비대면 시대에서 그 역할이 더욱 증대할 것으로 생각됩니다.

소프트웨어공학 소사이어티 소식지는 소프트웨어공학 분야의 자유 주제에 대한 기고문, 기관 탐방, 학술 행사 소개, 신진 교수 및 연구자 소개, 그리고 회원 동정 등의 내용으로 구성하였습니다. 특히 최근에 소프트웨어공학을 전공하는 새로운 연구자에 대한 소개를 통해 젊은 연구자들과의 상호 교류가 가능할 수 있도록 정보를 제공하였습니다.

새롭게 발간되는 소프트웨어공학 소사이어티 소식지가 여러분들에게 좋은 읽을거리를 제공하는 것은 물론 소사이어티와 친숙해지는 가까운 친구가 될 것을 확신합니다. 여러분의 적극적인 참여와 함께 구성원의 다양한 소식을 알려 주시면 공유할 수 있도록 노력하겠습니다.

항상 소프트웨어공학 소사이어티의 발전을 위해 도움을 주시는 회원 여러분께 감사드리며, 새롭게 출발하는 소식지 발간을 위해 수고하신 편집부회장과 편집이사들께 감사드립니다. 소프트웨어공학 소사어이티 회원 여러분의 발전과 번영이 항상 함께하기를 기원합니다.

감사합니다.

소프트웨어공학 소사이어티 회장 홍 장 의

■ 강교철(kck@postech.ac.kr)
포항공과대학교 명예교수

품질 높은 소프트웨어 개발을 위한 공학 원칙

• 소프트웨어는 아마도 인간이 만든 가장 복잡한 논리적 체계/시스템이며 소프트웨어가 우리 생활에 넓고 깊숙하게 파고들면 들수록 그 복잡도는 기하급수적으로 높아지고 있다. 따라서 소프트웨어 공학[1]의 핵심은 어떻게 이 복잡도를 다루는지에 있다고 해도 과언이 아니다.

본 기고문은 연구 논문이 아니라 본인이 현역에 있는 동안 기업에서 개발된 소프트웨어와 개발과정을 보고 느낀 점과 또 소프트웨어의 복잡도를 다루기 위하여 개발자들(software engineer)은 어떠한 자세로 개발에 임하여야 하는지를 소프트웨어공학 원칙 위주로 이야기 형식으로 풀어나간다.

본 기고문의 대상은 소프트웨어 공학에 상당한 식견이 있는 사람들보다는 소프트웨어 공학에 별다른 지식 없이 소프트웨어 개발에 종사하고 있는 상당수의 개발자를 위한 것임을 밝힌다. 그러한 개발자들이 기본적인 공학 원칙을 조금이나마 배울 수 있게 하는 것이 이 글의 목표다. 따라서 형식 또한 연구 논문 형식이 아닌 자유로운 형식을 택하였다.

1. 이 글을 쓰게 된 동기

우리가 무엇을 만들 때에 거기에는 피조물에 의하여 달성하고자 하는 목표와 그 목표를 달성하는데 도움이 되는 피조물의 속성이 있다. 예를 들어 예술품이라면 보는 사람들의 감성을 자극하고 예술가와 관람자가 서로 교감할 수 있어야 한다. 또한 클래식 음악, 월츠, 트로트 등 그 음악의 고유한 스타일이 있듯이 그 제품의 특성을 규정짓는 고유한 속성이 있을 것이다.

소프트웨어는 우리가 원하는 기능을 구현하는 것이다. 그런데 소프트웨어는 일단 개발이 끝나면 바로 유지보수(maintenance)가 시작되며 이 과정에서 새로운 기능이 추가되거나, 있는 기능이 삭제 또는 변경된다. 우리가 원하는 기능을 하드웨어(예, 계산기)가 아닌 소프트웨어(예, 계산 앱)로 구현하는 것도 이러한 변경 가능성 때문일 것이다. 따라서 우리가 만든 소프트웨어를 효율적으로 변경하는 것을 가능하게 하는 속성은 무엇인가 생각해보고 이러한 속성을 갖추도록 소프트웨어 개발에 반영하여야 하겠다.

우리가 효율적으로 소프트웨어를 유지보수 하기 위해서는 우선 소프트웨어의 구조, 모듈 간의 관계, 알고리즘 등을 쉽게 이해(understandability)할 수 있어야 하겠다. 그러기 위해서는 우선 복잡도(complexity)가 낮아야 하며 하나의 모듈을 수정했을 때에 그 파급 영향(side effects)이 다른 모듈에까지 미쳐서는 안되겠다. 변경이 될 때마다 검증(testability)하는 것은 필수적으로 일어나며 이 또한 쉽게 진행될 수 있어야 한다. 부가적으로, 개발된 소프트웨어의 전체 또는 일부 모듈이 유사한 시스템을 개발하는데 재사용(reusability)[2][3] 될 수 있으면 바람직하겠다.

이와 같이 소프트웨어는 다른 류의 제품과는 달리 계속 변경이 되며 따라서 개발하는 동안 유연한("soft") 또는 유연("soft")한 소프트웨어가 되도록 개발하고 이 유연성(softness)이 계속 유지될 수 있도록 하는 것이 소프트웨어 공학이 다른 공학과 다른 점이다.

본인이 40여년 이상 연구소, 대학, 산업체에서 직접 일하고 국내외 많은 기업에 자문을 하며 안타깝게 느낀 것은 소프트웨어가 대부분 유지보수 될 것을 고려하지 않고 개발되어 유지보수 과정에서 많은 불필요한 노력이 낭비되고 있다는 것이다. 많은 경우 개발자와 유지보수를 하는 관리자가 분리되어 있어 개발자들은 요구되는 기능만을 구현하면 되는 것으로 흔히 인식하고 있으며, 이러한 인식을 고치는 것이 매우 시급한 과제다. 소프트웨어 개발자는 유지보수가 용이한 소프트웨어를 개발할 수 있어야 “software engineer”로 불릴 수 있다.

소프트웨어의 유연성과 직접 관련이 있는 understandability, complexity, testability, reusability 와 같은 품질 속성을 갖춘 소프트웨어를 개발하는데 필요한 소프트웨어 공학 원칙을 다음 장에서 살펴 본다.

더 나아가기 전에, 본 기고문에서는 영어와 한글 단어를 자유롭게 섞어 사용하였음을 밝힌다. 컴퓨터 분야의 용어 대부분이 영어 단어가 그대로 음역 된 것이거나 실무에서도 영어 단어를 그대로 사용하는 것이 현실이기 때문에 영어와 한글 단어를 자유롭게 사용하였다. 독자의 이해를 부탁드린다.

2 소프트웨어공학 원칙

우선 원칙(principle)이란 무엇인지부터 이야기를 풀어나가야 하겠다. 원칙이란 우리가 원하는 기능과 속성을 갖춘 소프트웨어를 효율적으로 개발하고 유지 관리하는 것을 가능케 하는, 경험에 의하여 검증된 또는 합리적이고 논리적인 추론에 의하여 만들어진, engineering rule이다.

그러면 우리가 소프트웨어를 효율적으로 개발하고 유지 관리하기 위하여 필요한 소프트웨어 속성은 무엇인가? 많은 것들이 있겠지만 일반적으로 요구되는 몇 가지만 예를 들면: understandability, maintainability, modularity, testability, reusability, safety, security, correctness 등이 있겠다. 물론 real-time systems 같은 embedded system들은 stimulus에 대한 timely response가 무엇보다 중요하겠다. 위와 같은 속성을 갖춘 소프트웨어를 우리는 일반적으로 "good quality software"라 할 수 있겠다. 이러한 속성을 갖춘 소프트웨어의 개발을 가능케 하는 것이 공학 원칙이며 소프트웨어 개발, 유지 관리에 종사하는 사람들은 이를 숙지하고 항상 적용하는 것이 무엇보다 중요하다. **즉, software engineer의 사고체계에 내재화되어 있어야 한다.**

이번 장에서는 Separation of Concerns, Information Hiding (Anticipation of changes), Abstraction, Modularity 및 기타 원칙에 대해 자세히 알아보겠다. 이러한 원칙들은 각기 독립된 개념이 아니라 서로 연관되어 있음을 밝힌다.

2.1 Separation of Concerns (관심 분리)

Separation of Concerns(SoC)[3]는 국내에서 관심분리로 번역되고 있으며 공학 원칙 중에 가장 일반적인 원칙이라 할 수 있겠다. 예를 들어 Modularity도 SoC의 일종으로 볼 수 있으며 소프트웨어 개발에 요구분석, 디자인, 구현, 테스트 등으로 나누어 접근하는 것도 SoC이다. 이 장에서는 SoC가 어떻게 모듈 디자인에 적용되는지 보겠다.

1998년에 SEI에서 비정규적인 reuse workshop이 열리었다. 당시 session leader를 Will Tracz가 맡았고 긴 논의 끝에 당시 "Separation of 3Cs"라는 원칙이 만들어졌다. (당시 workshop report로 보고되었으나 지금은 reference를 찾을 수 없다.) 3C란 Context, Concept, Content로 모듈 디자인에 매우 중요한 개념이며 아래 예를 들어 설명하겠다.

예를 들어 Stack을 구현하는 모듈을 디자인한다고 하자. 우선 Stack의 기본적인 개념을 interface로 나타낸다면 Push()와 Pop() operation이다. 이는 Concept에 해당되고 모듈을 일종의 "virtual machine"으로 생각하는 것이며 사용자는 Push()와 Pop()만 알면 되는 것이다. Content는 모듈을 구현하는 data structure와 algorithm에 해당하며 외부로부터 "감추어져"(Information Hiding) 있어야 한다. Context란 이 모듈의 "사용 여건"이며 이 경우 integer, floating number, string 등 Stack에 저장되는 data type이 될 수 있겠다. 이 Context의 binding은 객체지향 언어에서 Template 같은 메커니즘으로 가능하겠다.

일반적으로 Concept는 잘 바뀌지 않으나 Context 나 Content는 쉽게 바뀔 수 있으며 이를 명확히 분리함으로 수정이 일어났을 때에 다른 모듈에 대한 영향을 최소화할 수 있다. 또한 재사용성도 높일 수 있겠다. 모듈을 디자인할 때에 항상 이 모듈의 사용 환경 (Context)은 어떻게 다를 수 있으며 구현관련 세부내용(Content)은 어떻게 바뀔 수 있는지 세밀히 분석해 봐야 하겠다. 또한 예상되는 모든 경우(Context)에 적용될 수 있는 기본적인 interface (Concept)가 모두 define 되었는지도 분석해야 하겠다.

예제가 너무 간단하여 당연하게 보일 수 있으나 실 업무에서는 잘 적용되지 않는 것이 사실이다. 그러나 유지보수성을 높이기 위하여 매우 중요한 원칙이며 잘 숙지하여 적용하였으면 한다.

2.2 Information Hiding (정보은닉)

오래전 한 workshop에서 참가자들이 가장 중요하다고 생각하는 software engineering principle이 무엇인지 투표를 한 적이 있었다. 그때 Information Hiding이 단연 1위로 뽑혔으며 심지어 이 원칙만 잘 지키면 소프트웨어 문제의 70% 이상이 해결된다는 극단적인 견해도 있었다. 이 예측의 사실 여부를 떠나 그만큼 중요하다는 데는 이론(異論)의 여지가 없겠다.

Information Hiding은 1972년 David Parnas[4]가 발표하였으며 이를 확대 적용하여 "Program families"[5]를 발표하였다. 국내에서는 정보은닉으로 번역되었으며 이를 정확히 이해하는 사람은 많지 않다. 개념을 간추리면: **모듈(예, A)을 디자인할 때 앞으로 바뀔 수 있는 A의 design decision에 사용자 모듈(dependent module)들(예, B,C 등)의 design decision이 의존하지 않도록 하라는 것이다.** 여기서 design decision은 "Information"에 해당하며 Hiding하라는 것은 타 모듈(예, B,C)을 디자인하는 사람이 A의 design decision을 알아 이에 의존하여 B, C의 design decision을 내리지 않도록 하라는 것이다. 여기서 design decision의 예를 들면 data structure, algorithm, resource allocation policy 등이 될 수 있겠다.

흔히 Information Hiding 개념을 어기는 "원흉"으로 Pointer를 꼽기도 한다. 다른 모듈로 data대신 Pointer를 넘긴다면 이미 이 모듈의 design decision을 노출한 것이 된다. 만약 Pointer 대신 직접 data를 넘기는 것으로 design decision을 바꾸거나, 아니면 그 반대의 상황이 일어났을 때, 때에 따라서는 많은 수정이 일어나기도 하는 것을 대부분 경험했을 것이다.

이 개념을 올바르게 적용하기 위해서는 "Anticipation of changes"가 우선되어야 한다. **앞으로 일어날 수 있는 design decision의 변화를 예측하고 모듈을 구현할 때에 내려지는 design decision 간에 상호의존성(dependency)이 없도록 각 design decision을 모듈에 Encapsulate 하여 디자인하라는 것이다.** 즉, 한 모듈의 design decision이 바뀔 때에 다른 모듈의 design decision 이 수정되어야 만 하는 일이 발생해서는 안 된다는 것이다. 따라서 Information Hiding을 Anticipation of changes, Modulization, Encapsulation이 포함되는 개념으로 봐도 좋다.

여기서 특히 주목해야 하는 것은 "Anticipation of changes"다. 지금 당장의 요구사항을 만족한다고 해서 좋은 소프트웨어는 아니다. 소프트웨어는 유지보수 과정에서 계속 변경이 이루어질 것이며 maintainability가 높은 소프트웨어를 만들어 testability, reusability 도 높이고 유지보수 비용도 낮출 수 있어야 하겠다. 본인은 기업에서 강의를 할 때마다 요구사항, 디자인 문서에 앞으로 어떠한 변화가 있을 것인가를 예측하여 기술하도록 권하고 있다. 비록 Agile 방법을 따

르더라도 이러한 문서는 개발자들의 사고체계를 바꾸는데 매우 중요하다고 생각한다. **"Soft"한 software, 즉 변화에 쉽게 적응시킬 수 있는 소프트웨어를 만들 수 있어야 훌륭한 software engineer다.**

2.3 Abstraction (추상화)

추상화로 번역되는 Abstraction 개념은 사실 매우 보편적인 개념이다. 따지고 보면 인간이 사고를 하고 언어를 사용할 때부터 추상화는 시작되었다고 볼 수 있다. 개체에, 자연 현상에, 상황과 상태에, 개체들 간의 관계에, 행동 등에 이름을 부여하고 (즉 개념화하고) 의사소통에 사용한 것이 추상화의 시작이라 볼 수 있겠다. 굳이 컴퓨터 분야에 사용된 예를 따진다면 database 분야에서 Conceptual modeling, Entity-relationship modeling, programming 분야에서 Function, Class, Abstract data type 등을 들 수 있겠으며 매우 보편화된 개념으로 널리 사용되어 왔다. **일반적으로 우리가 무엇을 만들 때에 피조물의 복잡성(complexity)을 다루기 위하여 추상화(Abstraction)와 정련(Refinement)의 과정을 거친다.** 건물의 설계를 하는 것은 추상화된 모델 (설계)을 만들어 분석하고 건설하기 위해서다.

일반적인 개념 임에도 불구하고 여기에서 다루고 있는 이유는 객체지향언어를 사용할 때 또는 기타 modelling을 할 때에 이 개념이 잘 적용되고 있지 않기 때문이다. "잘 적용된다"는 의미는 아래에서 설명하겠다. 많은 경우 객체지향언어를 쓰지만 객체가 하나의 기능 단위인 Function-oriented programming을 하고 있는 것이 현실이며 이는 객체의 추상화에 대한 이해 부족에 기인하는 것이 아닌가 생각한다. (물론 Function-oriented programming을 할 때도 Function에 부여하는 이름은 추상화를 거치며 아래에 설명되는 내용이 모두 적용된다.)

Abstraction은 추상화의 관점 (viewpoint)과 추상화의 수준 (level of abstraction), 추상화된 것의 정련(refinement/decomposition)을 함께 보아야 한다. 우선 추상화 한다는 개념은 상세한 것은 뒤로 미루고 (혹은 잠시 무시하고) 정해진 관점과 추상화 수준에서 중요한 정보만을 정리하여 나타내는 것이다. 이는 복잡도(complexity)를 줄이기 위해서다. 예를 들어 초상화를 그릴 때에 도화지에 얼굴의 위치를 정하고 얼굴의 윤곽을 먼저 그리고 명암 색채 등을 추가하는(refinement) 것이다. 소프트웨어의 예를 들면 만들고자 하는 혹은 이미 존재하는 시스템의 기능을 나타내고자 한다면 (예, functional viewpoint) 우선 가장 추상화 정도가 높은 수준에서 기능을 정의하고 이를 정련(refinement/decomposition)해 나갈 것이다. 어떻게 기능을 나타내느냐는 사용하는 방법론에 따라 다를 것이다. 예를 들어 functional specification techniques을 사용한다면 input-process-output 형태일 수 있으며 혹은 다른 방법론에서는 interaction scenario 혹은 use case가 될 수도 있겠다.

중요한 것은 주어진 관점과 수준에서 추상화된 정보가 그 추상화 수준에서는 complete하고 consistent해야 한다는 것이다. 예를 들어 초상화의 얼굴 윤곽을 그리는데 코를 누락하거나 혹은 이상한 위치에 배치해서는 안 될 것이다. (물론 고의적인 것이 아니라면.) 이렇게 추상화된 "정보"(예, 얼굴 윤곽)가 정련(refinement, decomposition) 과정을 거쳐 상세화될 것이다. 부언한다면 요구분석을 할 때에 구조적 기법을 쓴다면 functional viewpoint에서 modeling을 해야 하며 객체지향기법을 쓴다면 entity-relationship 관점에서 modeling이 되어야 하나 많은 경우 이것이 잘 지켜지고 있지 않다.

Abstraction의 시작은 이름을 부여하는 것(naming)이다. 그 이름이 우리의 사고 체계로 들어올 때 우리는 그 이름이 의미하는/포함하는 개체 (혹은 객체), 행위, 현상, 상태 등을 유추할 수 있다. 예를 들어 "인간"하면 현재 약 78억의 사람들이, "동물"하면 인간을 포함한 모든 살아있는 생명체들이 포함될 것이다. 당연해 보이는 것을 여기서 이야기하는 이유는 우리가 Abstraction을 할 때는 그 이름이 "project"하는 의미를 항상 생각해 봐야 한다는 것이다. 추상화된 이름에서부터 유추되는 의미가 우리가 개발하는 시스템의 범위 안에 들어오는지를 잘 살펴봐야 한다. 또한 이름이 주어진 "Abstraction"이 상세화될 때에 그 이름이 내포하는 의미와 일치하는 정보(예, attribute, method)만이 추가되어야 한다.

여기서 추상화할 때에 부여되는 이름과 그 이름이 의미하는 (project 되는) 객체, 행위, 상태, 상황 등을 잘 생각해보라는 이유는 maintainability 때문이다. 우리가 개발한 소프트웨어는 대부분의 경우 개발자가 아닌 누군가에 의하여 재사용되거나 유지보수 될 것이며 이때 쉽게 이해할 수 있도록 개발되어야 하기 때문이다. 그러나 종종 객체의 이름에서 유추하기 어려운 attribute나 operation (method) 등이 포함되어 재사용성이나 유지보수성을 떨어트리는 것을 발견할 수 있다.

추상화 정도(level of abstraction)를 정할 때에 항상 projection을 생각해야 한다. 주어진 이름에서 유추되는 "의미"가 우리가 만들고자 하는 시스템의 범위에 부족하거나 혹은 넘치지 않도록 적당한 선에서 이름을 부여하여야 하겠다.

따라서 앞에서 "잘 적용된다"는 의미는: (1) 추상화된 정보가 주어진 관점(viewpoint)과 일치하며, (2) 부여된 이름들이 내포(project)하는 의미가 서로 consistent하며 시스템의 목표에 부합하고, (3) 각 추상화 정도(level of abstraction)에 부합하는 이름과 정보만이 기술되어 있고, (4) 그 결과가 complete하고 consistent하여 분석 및 개발의 기초가 되어야 한다는 의미다. Refinement, decomposition, inheritance 등과 관련된 상세한 내용은 여기서 다루지 않겠다.

2.4 Modulization (모듈화)

Program을 모듈화한다는 것은 커다란 소프트웨어를 작은 단위(모듈)로 나누어 개발하여 각 단위를 테스트하고 이를 통합하며 테스트를 하여 전체 시스템을 개발한다는 매우 일반적인 개념이다. 그런데 여기에서 중요한 것은 시스템을 어떻게 모듈로 나누느냐 하는 것이다[4][5]. (큰 규모의 시스템은 process, component 등으로 나뉠 수 있으나 여기서는 다루지 않겠다.)

Modulization이 잘 되었는지를 판단할 수 있는 기준이 될 수 있는 모듈의 속성을 몇 가지만 살펴보면: testability, reusability, maintainability, parallel development 등이다. 이를 하나씩 살펴보도록 한다.

시스템을 모듈로 나누면 나누어진 모듈 간에는 서로의 dependency(예, call 관계)가 반드시 존재한다. Modular하게 디자인된 시스템이 testable하기 위해서는 dependency 관계가 hierarchy를 이루어야 하며 cycle이 존재해서는 안된다. 모듈 A를 테스트하기 위하여 B를 필요로 하며 B를 테스트하는데 A를 필요로 한다면 테스팅이 어려워질 것이다. Dependency hierarchy에서 제일 밑 (leaf node)에 있는 모듈부터 테스트하고 hierarchy의 위로 거슬러 올라가며 integrate하고 테스트할 수 있어야 하겠다.

모듈의 reusability, testability, maintainability 등을 평가할 때에 모듈의 Cohesion 정도와 다른 모듈과의 Coupling 정도를 보며 high cohesion, low coupling(loose coupling)을 이상적인 모듈화로 여긴다.

여기에서 high cohesion이라 하면 하나의 모듈이 하나의 기능만을 수행하며 그 내부의 요소들(attribute, method 등)은 이 기능을 구현하기 위하여 필요한 요소들만으로 구성되어 있을때에 cohesion의 정도가 높다고 말한다. 또한 Coupling이란[1] 다른 모듈과의 연결 관계를 나타내며 순수한 data만을 주고받는 coupling(Data Coupling)을 가장 이상적인 coupling으로 본다. 타 모듈의 내부 operation들 사이에 실행을 결정하게 하는 control data를 넘겨주는 것과 같은 형태의 coupling (Control Coupling)은 Information Hiding이 안된 것으로 지양할 것을 권한다. 또한 모듈 사이에서 각 모듈의 design decision 간에 dependency가 일어나는 것(Content Coupling)은 절대 피할 것을 권한다. (Information Hiding 참조)

이와 같이 high cohesion, low coupling(혹은 loose coupling) 되게 모듈화 함으로서 모듈이 수행하는 기능과 interface가 명확해져서 reusability, testability, maintainability를 높일 수 있으며 또한 모든 모듈을 동시에 구현하는 것이 가능해 진다.

이상 가장 기본적이고 중요하며 일반적으로 적용될 수 있는 원칙을 소개하였다. 다음 장에서는 객체지향프로그램에서 흔히 볼 수 있는 문제를 해결하는데 유용한 원칙을 소개하겠다.

2.5 Open-Closed 원칙

"Open for extension, but closed for modification"[7]으로 객체지향 프로그램에서 객체를 상속하여 Sub-class를 define할 때 상위 Class에 define된 attributes와 methods는 그대로 상속하나 새로운 attributes나 methods를 추가하는 방식으로 상속해야 한다는 것이다. 상위에 define된 methods를 수정해야만 하거나 또는 하위 Class에서 수정된 method 로 override할 경우 이는 상위 methods의 semantics ("Contract")을 바꾸는 것으로 특히 객체의 run-time binding이 일어날때에 테스팅을 매우 어렵게 한다.

이 원칙을 어기는 것은 흔히 볼 수 있으며 maintainability, testability를 크게 떨어트리며 특히 조심해야 할 부분이다.

2.6 Law of Demeter 원칙

"Don't talk to strangers"로 대변되는 이 원칙은 1987년 Ian Holland[8]에 의하여 제안된 원칙으로 객체 간에 low coupling과 Information Hiding을 위한 원칙이다. 객체지향개발에서 하나의 객체가 직접 interface하는 객체들 이외에 다른 객체들에 대한 정보를 알지 않도록 하라는 원칙이다.

좀 더 자세히 이야기하면, 어떤 객체 "o"의 method "m"이 있을 때에,"m"이 invoke 할 수 있는 객체들은: "o" 자신과, "m"의 parameters, "o"의 attributes, "o"가 생성한 객체, "o"의 scope 안에 있는 global variables이다. 특히 다른 method 들이 받은 객체의 method를 invoke하는 것은 지양할 것을 권한다. 예를 들어 o.m().n()과 같이 "."이 하나 이상 길게 들어가는 것은 피하라는 원칙이다. 이렇게 함으로써 understandability, maintainability를 높일 수 있을 것이다. 본인이 객체지향 프로그램을 볼 때에 invocation sequence가 길게 "."으로 연결되어 있는 statement들을 쉽게 접할 수 있었으며 이들 프로그램들을 이해하기가 매우 힘들었던 경험을 회상해 본다.

3 원칙 적용의 의미

앞에서 중요하다고 생각되는, 그래서 software engineer가 개발에 임할때에 그들의 사고방식에 내재 되었으면 하는, 공학 원칙을 중심으로 보았다. 이 장에서는 이러한 공학 원칙들이 적용되었을 때에 어떠한 효과가 있는가에 대하여 이야기해 보겠다.

우선 관심분리 원칙 (Separation of Concerns) 부터 보자. 이 원칙이 모듈 설계에 적용이 되었을 때에 3Cs의 분리는 Modulization, Information Hiding, Abstraction과 밀접하게 관련이 있다. 특히 모듈을 디자인할 때에 모듈의 interface를 virtual machine 형태로 추상화 함으로서 interface가 보다 견고하게 디자인될 것이며, 구현에 필요한 알고리즘 등 모듈의 design decision을 Encapsulate하여 변화에 용이하게 적응할 수 있게 하고 다른 모듈에 대한 영향도 최소화 시킬 수 있어 이해용이성 및 유지보수성을 높일 수 있다. 모듈의 사용환경(Context)의 변화에 따라 변할 수 있는 요소는 분리하였다가 late binding (예, Template 등) 함으로 유지보수성을 높일 수 있겠다.

정보은닉 원칙은 소프트웨어공학 연구자들이 가장 중요하게 여기는 원칙 중 하나로 변화할 수 있는 구현 요소의 예측(Anticipation of changes)과, 이 구현 요소의 Encapsulation을 통하여 변경이 일어났을 때에 다른 모듈에 주는 영향을 최소화할 수 있어 유지보수성을 크게 높일 수 있다.

추상화는 소프트웨어 개발 전 과정에 적용되는 개념이다. 본인이 그동안 여러 개발 산출물(예, requirements 모델, design 등)을 보며 느낀 것은 산출물을 개발할 때에 사용한 방법론 혹은 언어에서 요구되는 개념을 명확히 이해하지 못하고 개발하는 경우가 매우 흔하다는 것이다. 예를 들면 객체지향적인 방법을 적용하나 실제로 추상화된 객체들을 보면 function인 경우가 매우 많다. 객체지향인 언어를 썼으나 functional viewpoint로 모델링하거나 디자인하였다는 이야기이다. 그럼 무엇이 문제인가? 언어에서 요구하는 개념과 개발자가 생각하는 개념의 불일치로 사용된 언어의 특징을 잘 활용할 수 없게 된다. 예를 들어 객체들을 function 단위로 구현하면 견고한 inheritance의 구조도 만들기 어려울 뿐만 아니라 functions들이 바뀔 때에 inheritance구조 전체를 수정하거나 methods을 override 해야 하는 일이 종종 발생한다. 응용분야의 concept, entity들을 바탕으로 define하고 Open-Closed 원칙과 같은 원칙을 잘 적용한다면 understandability, adaptability, reusability, testability 등을 크게 높일 수 있다.

Modulization은 프로그램을 모듈로 어떻게 나누느냐가 주요 관건이며 정보은닉, 추상화원칙을 적용하여 모듈로 나누고 3Cs 분리 원칙, Open-Closed, Law of Demeter 원칙 등을 잘 적용하여 구현하면 모듈의 understandability, reusability, maintainability를 크게 높일 수 있다. Modulization의 좀 더 상세한 것은 사용하는 방법론에 따라 차이가 있음을 밝힌다.

4 맺으며

나는 흔히 “소프트웨어 개발은 예술과 같다”라는 말은 듣곤 한다. 나는 이러한 견해에 격하게 반대한다. 소프트웨어 분야를 고고한 예술로 격상시키기 보다는 오히려 “저 분야는 도저히 알 수 없어”라고 격하시키는 것 같이 들리기 때문이다. 소프트웨어 분야에는 그동안 축적된 경험에서 나온 훌륭한 공학 원칙들이 엄연히 존재하며 “engineering”의 한 분야이다. 소프트웨어 공학이 공학의 한 분야로 인정받기 위해서는 우리 모두 원칙에 근거한 개발을 해야 한다.

70-80년대 본인이 한창 프로그램을 개발할 당시 우리 팀멤버들은 누가 시키지 않아도 계속해서 자기가 맡은 부분의 새로운 버전을 내어놓기 위하여 많은 노력을 했다. 왜냐하면 내가 만든 소프트웨어는 나의 기술력의 결정체이며 나의 자존심이고 나의 얼굴이기 때문이었다. 아마도 50%이상의 노력은 Refactoring 하는데 들어갔을 것이다. 개발된 소프트웨어는 개발자가 아닌 누군가에 의해서 변경되고 유지보수 되며 재사용되기 때문에 Refactoring을 게을리해서는 안 되겠다.

소프트웨어 개발자들이 이미 검증된 공학원칙들을 잘 숙지하고 개발에 적용하도록 부단히 노력하여 품질 높은 소프트웨어를 개발할 수 있길 바라며, 개발자가 단순히 “Coder”로 인식되는 것이 아니라 “Engineer”로 당당히 인정받는 날이 빨리 오길 고대해본다.

References

1. Pressman, R.S., “Software Engineering A Practitioner’s Approach”, 7th Ed., McGraw Hill International Edition (2010)
2. Frakes, W. B., Kang, K., "Software Reuse Research: Status and Future," IEEE Transactions on Software Engineering (TSE), Vol. 31, No. 7, pp. 529-536 (2005)
3. Dijkstra, Edsger W, “On the role of scientific thought”, Selected Writings on Computing: A Personal Perspective, NY, NY,USA:Springer-Verlag, pp. 60-66 (1982)
4. Parnas, D. L., “On the criteria to be used in decomposing systems into modules”, CACM, 15(12), pp. 1053-1058 (1972)
5. Parnas, D. L., “On the design and development of program families,” IEEE Transactions od Software Engineering, Vol. SE-2(1), pp. 1-9 (1976)
6. Kang, K.C., Lee, J., and Donohoe, P., "Feature-Oriented Product Line Engineering," IEEE Software, Vol. 19, No. 4, pp. 58-65 (2002)
7. Meyer, B., “Object-Oriented Software Construction”, Prentice Hall (1988)
8. Lieberherr, K.J.; Holland, I.M.. "Assuring good style for object-oriented programs". IEEE Software. 6 (5): pp. 38–48 (1989)

INTERVIEW

• 김윤호 박사 (Dr. Yunho Kim)
조교수
소프트웨어 공학 연구실
한양대학교 컴퓨터소프트웨어학부
yunhokim@hanyang.ac.kr
https://se.hanyang.ac.kr

신진연구자 소개 I

주요 약력

2007.02 한국과학기술원 전자전산학부 전산학전공 (학사)

2017.02 한국과학기술원 전산학부 (박사, 지도교수: 김문주)

2017.02-2018.02 한국과학기술원 정보전자연구소 연수연구원

2018.03-2020.08 한국과학기술원 전산학부카이스트 연구조교수

2020.09. 현재 한양대학교 컴퓨터소프트웨어학부 조교수

주요 연구분야

- Automated test generation
- Mutation testing
- Fault localization
- Industrial application of advanced testing and debugging technologies

대표 논문

[1] Yunho Kim and Shin Hong, Learning-based Mutant Reduction Using Fine-grained Mutation Operators, Journal of Software Testing Verification and Reliability (STVR), August 2021

[2] Yunho Kim and Shin Hong, DEMINER: Test Generation for High Test Coverage through Mutant Exploration, Journal of Software Testing Verification and Reliability (STVR), Volume 31, Issue 1-2, 2021

[3] Yunho Kim, Seokhyun Mun, Shin Yoo and Moonzoo Kim, Precise Learn-to-Rank Fault Localization using Dynamic and Static Features of Target Programs, ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 28 Issue 4, October 2019

[4] Yunho Kim, Shin Hong, and Moonzoo Kim, Target-Driven Compositional Concolic Testing with Function Summary Refinement for Effective Bug Detection, The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), Tallinn, Estonia, August 26-30 2019

[5] Yunho Kim, Yunja Choi, and Moonzoo Kim, Precise Concolic Unit Testing of C Programs with Extended Units and Symbolic Alarm Filtering, Intl. Conf. on Software Engineering (ICSE), Gothenburg, Sweden, May 27-June 3, 2018

안녕하세요 한양대학교 컴퓨터소프트웨어학부에서 조교수로 재직중인 김윤호입니다.

먼저 새로 창간되는 소프트웨어 공학 소사이어티 소식지에서 제 소개를 할 수 있어 큰 영광입니다. 처음 원고 초청을 받았을 때 지금 하고 있는 최신 연구 내용을 실어야 할까, 내가 지금 하고 있는 연구 분야의 동향에 대한 전반적인 소개글이 좋을까 고민이 많았습니다. 하지만 본 지면의 성격이 전문적인 연구 분야 소개보다는 소프트웨어 공학 소사이어티의 신진연구인력으로 합류하게 된 제가 어떤 사람인지, 그리고 제가 소속된 한양대학교 컴퓨터소프트웨어학부가 어떤 곳인지 소개하는 것이 좋을 것 같아 조금은 편한 분위기에서 제 소개를 하려고 합니다.

Q. 간단하게 자기 소개 부탁드릴게요

A. 안녕하세요. 한양대학교 컴퓨터소프트웨어학부에서 소프트웨어 공학, 특히 소프트웨어 테스팅과 디버깅의 자동화 연구를 하고 있는 김윤호입니다. 작년 9월에 막 임용되었으니 한양대학교에 재직한지 만 1년이 조금 넘었네요. 한양대학교에 임용되기 전에는 2017년 2월에 한국과학기술원 김문주 교수님의 지도로 박사학위를 받았고 한국과학기술원 정보전자연구소 연수연구원과 전산학부의 연구 조교수로 재직하면서 소프트웨어 테스팅 및 디버깅 연구를 수행하였습니다.

Q. 현재 어떤 연구를 수행중인가요?

A. 제가 관심 있게 진행하고 있는 연구 분야는 크게 소프트웨어 테스팅 자동화랑 디버깅 자동화입니다. 소프트웨어 테스팅 자동화 연구는 기호 실행(symbolic execution)과 퍼징(fuzzing) 등의 테스트 입력 값 생성 기술을 사용해서 사람이 고생하지 않고 소프트웨어 버그를 자동으로 찾는 것이 목표입니다. 테스팅 자동화 연구를 할 때 가장 어려운 점은 테스트 대상 소프트웨어의 크기와 복잡도가 급격하게 증가하고 있다는 점입니다. 그래서 기존의 테스트 자동화 연구로는 점점 커지는 소프트웨어를 분석하는데 한계가 많이 있었습니다. 이 문제를 해결하기 위해 저는 큰 소프트웨어를 유닛 단위로 쪼개고, 각각의 유닛을 먼저 분석한 다음, 그 분석 결과를 다시 합성하여 전체 소프트웨어의 테스트를 만드는데 사용했습니다. 다시 말해 전통적인 컴퓨터공학의 문제 해결 방식인 분할 정복 방식을 테스팅 자동화에 도입했다고 보시면 될 것 같습니다.

또 다른 연구 분야는 소프트웨어 디버깅을 자동화하기 위한 오류 위치 추정 기법 연구입니다. 테스트를 통해 소프트웨어 버그가 있다는 사실을 알았다면 이제 버그를 고쳐야 하는데 문제는 소프트웨어가 크고 복잡해서 버그의 원인이 어디 있는지 찾기가 어렵다는 점입니다. 물론 디버깅 문제는 컴퓨터공학의 태초부터 있던 문제라 다양한 해결책들이 제시가 되었는데, 문제는 일반적으로 좋은 결과를 내는 해결책이 없고 디버깅 하고자 하는 소프트웨어의 코드 특성에 따라 각 해결책들의 성능이 크게 차이가 난다는 점에 있습니다. 그래서 저는 기계학습 기술을 사용해서 이 문제를 해결했습니다. 각각의 오류 위치 추정 기법 결과와 코드 특성을 학습하여 디버깅 모델을 만들고 새로운 코드가 주어졌을 때 생성한 디버깅 모델을 적용하여 최적의 오류 위치 추정 기법 결과를 생성할 수 있었습니다. 특히 디버깅 자동화 연구는 제가 소속된 소프트웨어 재난연구센터(센터장: 경북대학교 최윤자 교수님)에서 소프트웨어 재난을 일으킨 오류의 원인을 신속하게 식별하여 빠르게 제거하는데 기여하는 중요 연구 분야입니다.

Q. 지금 재직중인 한양대학교 컴퓨터소프트웨어학부는 어떤 곳인가요?

A. 한양대학교 컴퓨터소프트웨어학부는 한양대학교의 서울캠퍼스에 위치한 컴퓨팅 분야를 가르치고 연구하는 학부입니다. 현재 30명의 전임교수와 600여명의 학부생, 그리고 300여명의 석/박사과정 대학원생으로 구성되어 있고요. 컴퓨터소프트웨어학부의 역사는 한양대학교 역사에 비해 굉장히 짧은 편입니다.

컴퓨터소프트웨어학부의 첫 시작은 2000년에 전자전기공학부로부터 시작하고 그 사이에 몇 차례의 학부 개편 과정을 거쳐 현재의 컴퓨터소프트웨어학부가 생겨났습니다. 저는 21년의 컴퓨터소프트웨어학부 역사에서 최초의, 그리고 아직까지 유일한 소프트웨어공학 분야의 전임교원입니다.

정면에 보이는 건물이 ITBT관, 혹은 정보통신관으로 불리는 건물이고 우측에 있는 건물이 산학기술관입니다. 제 연구실을 포함해 대부분의 컴퓨터소프트웨어학부 연구실은 주로 이 두 건물에 있습니다.

Q. 한양대학교 컴퓨터소프트웨어학부에 소프트웨어 공학 분야 전임교원이 없었나요?

A. 네, 약간 의외의 사실인데 컴퓨터소프트웨어학부에는 제가 부임하기 전까지 소프트웨어공학 분야를 전문적으로 연구하고 교육하는 전임 교원이 없었습니다. 소프트웨어공학 과목 교육은 시스템 분야 교수님들이 담당하시거나 산학협력 교수님들이 담당을 하셨고요. 학부 역사 초기에는 아무래도 전자전기공학부에서 독립한 만큼 소프트웨어 분야보다는 기존 전자공학 분야와 많이 연관된 시스템이나 하드웨어에 관련된 분야의 교수님들이 많이 계셨습니다. 점차 소프트웨어 분야의 교수님들을 충원하기 시작했는데 소프트웨어공학 분야 전임 교원 초빙이 어려웠다는 얘기를 전해들었습니다.

Q. 학부 전체에 혼자 소프트웨어 공학을 담당하고 있는 건 어떤 장, 단점이 있을까요?

A. 아무래도 가장 큰 장점은 다른 사람들의 눈치를 보지 않고 자유롭게 소프트웨어공학 커리큘럼을 만들고 운영하면서 학생들에게 현대 소프트웨어 공학이 어떻게 발전했는지 보여줄 수 있다는 점입니다. 기존 소프트웨어공학 수업은 사실 학생들에게 인기 있는 수업은 아니었는데요. 학생들의 얘기를 정리해보면 뭔가 필요하고 의미있는 내용들인 거 같긴 한데 본인들이 취업할 기업에서는 사용하지 않을 것 같은 내용들이라 큰 관심이 가지 않는다는 것이었습니다. 그래서 소프트웨어공학 수업을 통해 기존의 소프트웨어 개발 프로세스와 문서화에 대한 내용은 줄이고 2021년 현재 소프트웨어 기업들에서 적용하고 있는 방법론을 가르치려고 많이 노력하고 있습니다. 특히 TDD, CI/CD 등의 개발 활동을 실제로 실습하면서 그 장점을 몸으로 익힐 수 있게 하는데 많은 노력을 기울이고 있습니다. 학부에서도 제가 진행중인 소프트웨어공학 교육 개선에 관심과 기대를 보이고 있어 어깨가 무겁습니다.

단점이라면 좀 외롭다는 점 같습니다. 재직한지 이제 막 1년을 지나면서 교육이나 연구에서 다양한 어려움을 겪게 되는데 비슷한 분야의 선배 교수님들께 조언을 구하고 싶을 때가 있습니다. 물론 학부 다른 교수님들이나 소프트웨어공학 소사이어티의 선배 교수님들께 많은 조언을 구하면서 문제를 해결해나가고 있지만 같은 학부의 같은 분야 교수님이 계시면 더 구체적인 조언을 구할 수 있지 않을까 하는 생각을 합니다.

처음에는 같은 분야를 연구하는 교수님이 없어 학부 내에서 공동연구가 어렵지 않을까 생각하기도 했는데요. 오히려 이 점은 단점보다는 장점이 더 큰 것 같습니다. 자연스럽게 다른 학교의 교수님들과 공동 연구를 하게 되는 계기도 되고요. 같은 학부 내에서는 오히려 기계학습이나 보안 등의 소프트웨어공학이 아닌 타 분야를 연구하는 교수님들과 같이 공동연구를 함으로써 연구 분야를 더 넓힐 좋은 기회라고 생각합니다.

Q. 마지막으로 하고 싶은 말은?

A. 제가 막 신진연구인력으로 활동을 시작하던 시기에 마침 코로나 바이러스가 창궐하면서 소프트웨어 공학 소사이어티의 커뮤니티 활동이 많이 줄어들었습니다. 현재 백신 접종률이 높아지면서 점차 일상으로의 복귀가 시작되고 있는데 소프트웨어공학 소사이어티의 커뮤니티 활동도 이전처럼 돌아와 예전같은 모습을 볼 수 있으면 좋겠습니다. 모두들 건강한 모습으로 다시 뵐 수 있길 희망합니다

INTERVIEW

• 신동환 박사 (Dr. Donghwan Shin)
Research Scientist
University of Luxembourg
donghwan.shin@uni.lu

신진연구자 소개 II

주요 약력

2010.08 카이스트 전산학과 (학사)

2012.08 카이스트 전산학과 (석사, 지도교수: 배두환)

2018.02 카이스트 전산학과 (박사, 지도교수: 배두환)

2018.03-2018.08 카이스트 (박사후연구원, 지도교수: 배두환)

2018.09-현재 University of Luxembourg (박사후연구원, 지도교수: Lionel Briand)

주요 연구분야

- Software mutation testing
- Testing for Machine Learning (ML)-enabled Cyber-Physical Systems (CPS)
- Log analysis

대표 논문

[1] Fitash Ul Haq, Donghwan Shin, Lionel C. Briand, Thomas Stifter, and Jun Wang. "Automatic test suite generation for key-points detection DNNs using many-objective search (experience paper)." In Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 91-102. 2021.

[2] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel Briand, “Can Offline Testing of Deep Neural Networks Replace Their Online Testing?” Empirical Software Engineering 26, 90 (2021). https://doi.org/10.1007/s10664-021-09982-4

[3] Messaoudi, Salma, Donghwan Shin, Annibale Panichella, Domenico Bianculli, and Lionel Briand. "Log-based Slicing for System-level Test Cases." In 2021 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA). 2021.

[4] Mike Papadakis, Donghwan Shin, Shin Yoo, and Doo-Hwan Bae. "Are mutation scores correlated with real fault detection? a large scale empirical study on the relationship between mutants and real faults." In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), pp. 537-548. IEEE, 2018.

[5] Donghwan Shin, Shin Yoo, and Doo-Hwan Bae. "A theoretical and empirical study of diversity-aware mutation adequacy criterion." IEEE Transactions on Software Engineering 44, no. 10 (2017): 914-931.

소개글

소프트웨어공학 소사이어티 여러분, 안녕하십니까. 저는 현재 University of Luxembourg에서 Research Scientist로 일하고 있는 신동환입니다. 새로운 학회 소식지에 포함될 신진연구자 소개글에 대한 투고를 부탁받았을 때, 특히 기존의 논문 형태의 틀에 얽매이지 않고 보다 자유로운 형식으로 서로의 소식을 공유하기 위한 목적이라는 편집부 이주용 교수님의 말씀을 듣고, 어떤 방식으로 저에 대한 소개를 효과적으로 할 수 있을까 많이 생각했습니다. 여러가지 고민끝에, 단순히 저의 최근 연구 내용들 뿐만아니라 이곳 룩셈부르크의 생활 그리고 유럽에서 맞는 COVID-19 상황 등에 대해서도 소식을 나누기 위해서, 제가 주변에서 자주 들었던 질문들과 그 외에 별도로 소개하고 싶은 내용들을 조합해서 가상의 인터뷰 형식으로 저를 소개하도록 하겠습니다. 펜과 종이를 준비해서 밑줄을 그어가며 읽는 논문 같은 느낌보다는 따뜻한 커피 한잔과 함께 가볍게 읽는 토막글 느낌으로 준비했으니 편하게 즐겨주시길 바랍니다.

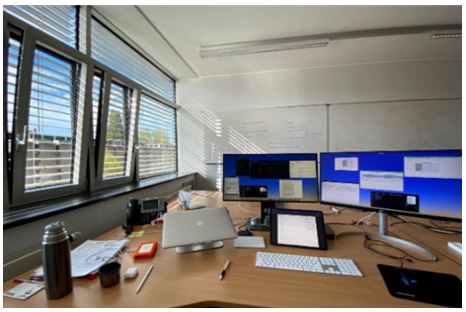
Q. 룩셈부르크는 어떤 나라인가요?

룩셈브르크의 유명한 내려다본 풍경입니다. 강을 기준으로 왼쪽은 독일, 오른쪽은 룩셈부르트입니다.

A.룩셈부르크는 유럽의 독일, 프랑스, 벨기에 사이에 있는, 경기도 크기의 1/4 정도 되는 작은 나라입니다. 룩셈부르크 시내를 기준으로 동, 서, 남쪽 방향으로 30분정도 차로 달리면 각각 독일, 벨기에, 프랑스와 마주한 국경을 넘게 될 정도입니다. 이렇게 크기는 작지만 지리적인 이점을 잘 활용하고 전세계 금융기업들을 적극적으로 유치하여 금융업을 성공적으로 성장시킨 덕분에 현재 1인당 국내총생산 (GDP) 기준으로 세계 1위에 위치하고 있습니다. 최근에는 특히 우주 개발 산업에 많은 투자를 하고 있으며, 세계 1위 인공위성 운영 및 서비스 기업인 SES를 비롯하여 많은 민간우주기업들이 자리하고 있습니다. 국가 공용어로 룩셈부르크어 이외에도 독일어와 프랑스어를 사용하는데, 실제로 모든 공문서는 3가지 언어를 병기하도록 되어있

고, 일상생활에서는 예를 들어 카페나 레스토랑에 가면 대부분 프랑스어로 된 메뉴판을 보실 수 있습니다. 이곳에서 3년이나 살았으니 저도 프랑스어를 조금이라도 익혔을 법도 한데, 의외로 많은 사람들이 회화 수준의 영어를 기본적으로 하는 덕분에 저는 아직까지 인사 수준의 프랑스어만으로 잘 지내고 있습니다.

Q. 어떤 계기로 룩셈부르크에 가게 되었나요?



연구실 제 자리입니다. 안타깝게도 날씨가 이렇게 좋은 날이 흔하지는 않습니다.

A. 여느 박사과정 학생과 마찬가지로 저도 박사과정을 마칠 즈음이 되어 박사후연구원 자리에 관심을 갖기 시작했습니다. 온라인에서 찾은 여러 구인 공고들 중에서도 특히 University of Luxembourg에서 올라온 것에 관심을 갖게 되었는데, 그 전부터 이미 해당 대학의 연구자와 공동연구를 한 경험이 있어서 조금 더 친숙하게 다가오기도 했고, 특히 해당 구인광고를 올렸던 Software Verification and Validation (SVV) 그룹의 Lionel Briand 교수는 제가 석사 때 Mutation Testing 연구를 시작하면서부터 수없이 읽었던 TSE 2006 논문 "Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria"의 저자 중 한 명이어서 더욱 특별하게 와 닿았습니다.

그 전에는 한번도 해외에서 거주해본 경험이 없어서 해외에서 박사후연구원을 하겠다는 결정을 하기까지 고민을 많이 했지만, 카이스트에서 학부과정부터 박사과정까지 10년 넘도록 지내면서 좋았던 만큼 새로운 곳에서 또 다른 것들을 배우는 것이 중요하다는 생각으로 과감하게 룩셈부르크행을 결정하게 되었습니다.

Q. 룩셈부르크 대학에서 어떤 연구를 하고 있나요?

A. 제가 있는 SnT 연구센터의 SVV 그룹은 주로 소프트웨어 검증 및 테스트와 관련된 다양한 연구를 진행하고 있습니다. 특히 이곳은 다양한 산학연계 프로젝트를 기반으로 연구를 하는 것으로 유명한데, 기본적으로 모든 박사과정 학생들이 각자의 산학연계 프로젝트에 포함되어 있으며, 박사과정 동안에 계속해서 해당 산업 파트너와 긴밀한 소통을 하며 실제 산업계에서 발생하는 문제들을 해결하기 위한 연구를 수행하게 됩니다. 저는 박사후연구원으로써 여러 프로젝트에 참여하며 해당 프로젝트의 박사과정 학생들을 지도하는 역할을 맡고 있는데, 특히 (1) 세계 1위 인공위성 운영 및 서비스 업체인 SES와 함께 효과적인 인공위성 관제 시스템 테스트를 위한 소프트웨어 실행 로그 분석 및 활용에 대한 연구, (2) 자동차 센서 시스템 분야의 글로벌 리더인 IEE와 함께 자율주행 자동차에 필요한 인공지능 기반 사이버 물리 시스템에 대한 효과적인 테스트 기법 개발, 그리고 (3) 미션 크리티컬 시스템 인프라 개발 분야의 선두업체인 HITEC과 함께 차세대 Edge Computing 플랫폼에 탑재될 보안 모듈 개발 및 테스트 연구를 진행하고 있습니다. 혹시 더 자세한 연구 및 프로젝트 내용이 궁금하시면 제 홈페이지(https://donghwan-shin.github.io)에서 더 많은 정보를 찾아보실 수 있습니다.

Q. 그곳의 COVID-19상황은 어떤가요?

A. 당연하게도 COVID-19로 인해서 힘겨운 시간을 보내왔습니다. 특히 프랑스와 독일과 가까운 이곳에서는 작년 여름 COVID-19로 인한 사상자가 급증하기 시작할 무렵에 날씨가 좋은 여름날에도 불구하고 도시 전체에 암울하고 어두운 분위기가 깔린 것만 같았습니다. 그래도 정부가 적극적으로 개입하여 실내 마스크 착용을 빠르게 의무화하고 가능한 모든 업종에 대하여 재택근무를 지시하는 등 조치를 취한 덕분에 룩셈부르크 내에서는 아주 심각한 상황을 피하고 주변국에 비해서 빠르게 회복할 수 있었던 것 같습니다. 현재는 실내에 10인 이상이 모이는 경우 혹은 2미터 거리 두기가 불가능한 경우를 제외하면 마스크를 착용하지 않아도 되는 수준까지 돌아왔고, 저도 지난 10월부터 매주 3일은 출근하고 2일만 재택근무를 하고 있습니다.

Q. University of Luxembourg, 혹은 더 넓게 유럽에서의 박사후연구원 생활에 대한 장단점이 있다면?

A. 제가 생각하는 가장 큰 장점은 보다 다양한 사람들과 함께 연구하는 경험을 할 수 있다는 것입니다. 지난 3년간 SVV 그룹 안에 있는 다양한 연구원들 뿐만 아니라, University of Ottawa, Imperial College London, Delft University of Technology, RISE Research Institutes of Sweden 등 여러 외부 기관의 연구자들과 공동 연구를 할 수 있었습니다. 그 과정에서 그들이 생각하는 방식, 연구하는 태도, 논문을 쓰는 방식 등 여러가지를 경험할 수 있었고, 그것들이 제가 조금 더 발전하는데 큰 역할을 했다고 생각합니다. 한국에서 지내던 것과 비교해서 또 하나의 두드러진 장점은 바로 차를 타고 쉽게 국경을 넘을 수 있다는 것입니다. 처음으로 운전을 하고 룩셈부르크에서 벨기에로 갔던 날, 바닥에 실제로 그어져 있지도 않은 그 무형의 국경을 넘는 순간의 오묘한 감정이 아직도 생생합니다. 단순히 국경을 쉽게 넘을 수 있다는 사실보다도, 유럽 연합에 속한 사람들은 그렇게 쉽게 더 넓은 곳을 여행하고, 더 많이 경험하고, 더 다양한 문화와 언어를 배우며, 더 넓은 무대에서 자신에게 맞는 직업과 거주 환경을 찾을 수 있다는 사실이 부럽기도 합니다.

그러나 장점만 있을 수는 없겠지요. 대표적인 단점으로는 우선 병원을 방문해서 진료를 받기가 힘들다는 것입니다. 단순히 언어적인 문제도 있지만 이곳 특유의 느긋한 문화 덕분에 병원 예약을 한다면 최소 1개월 뒤에 의사를 만나볼 수 있다는 점에서 아플 때 빠르게 진료를 받을 수 있는 한국과는 많이 차이가 있습니다. 이러한 느긋한 문화는 유럽의 특징인 것 같기도 한데, 예를 들어 처음 이곳에 이사를 왔을 때 집에 인터넷 설치를 하는데 2주가 걸린 일은 아직도 제가 지인들에게 종종 이야기하는 에피소드 입니다. 그래도 그만큼 일하는 사람들의 입장에서는 과도한 업무처리를 하지 않아도 된다는 점에서 이것은 장점이 될 수도 있겠습니다.

Q. 끝으로 소프트웨어 공학 소사이어티 소식지를 읽는 독자분들에게 하고싶은 말은?

A. 제가 좋아하는 한 예능 프로그램이 있는데, 연예인이 아닌 일반인 게스트들이 나와서 그들의 이야기를 들려주곤 합니다. 한번은 어떤 게스트가 80세가 넘으신 어르신께 삶을 어떻게 살아야 하느냐고 질문했더니 그 어르신께서 이렇게 말씀하셨다고 합니다: "많이 먹고, 많이 돌아다니고, 걱정하지 말고 살아라." 누구나 그렇듯 이런저런 걱정을 안고 살아가는 저에게 이 말씀이 큰 위안이 되었습니다. 어쩌면 우리는 연구자라는 직업 특성상 남보다 더 예리하게 관찰하고 더 많이 생각하는 것이 습관일 수 있지만, 그럼에도 불구하고 걱정은 조금 더 내려놓고 그만큼 더 편안한 마음으로 항상 건강하시고 행복하시길 바랍니다.

국내외 학술행사 소개 I

C CONFERENCES

우당탕탕 Student Volunteer 참여기

KAIST SE 연구실
박사과정 현상원

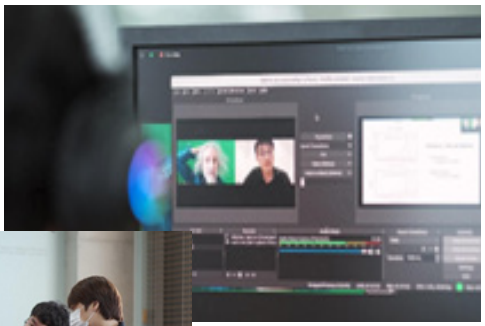
ICSE 2020 학술대회 탐방기

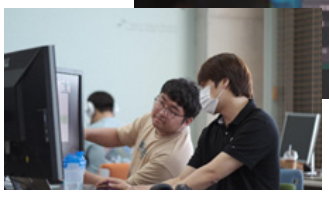
더위의 시작을 알리는 초여름 비가 그칠 무렵부터, KAIST SE 연구실은 때아닌 분주함으로 가득하였다. 2020년은 소프트웨어 공학 분야에서 가장 명망있는 학회인 ICSE (International Conference on Software Engineering)가 대한민국의 서울에서 열리는 영광스러운 연도였다. 더욱이, SE 연구실의 지도 교수님인 배두환 교수님께서 ICSE의 Co-Chair를 맡으셨기 때문에 남다른 이벤트로 느껴질 수밖에 없었다. 총 10명의 석/박사 학생들이 SV (Student Volunteer)로 참여하였으며, 약 2달의 기간동안 ICSE의 원활한 온라인 개최를 위해 다양한 준비를 진행하였다.

본래 학회에서 SV의 업무라면 세션 체어분들을 돕거나 학회 참여자들에게 여러가지 안내를 하는 것이라 기대할 수 있지만, 2020년은 코로나 바이러스에 의해 많은 학회들이 온라인 학회 개최를 피해갈 수 없었다. ICSE 또한 온라인으로 개최됨에 따라 우리는 당초 기대와는 달랐던 막중한(?) 업무를 맡게 되었는데, 바로 온라인 세션의 기술적 관리였다. 외부 회사에서 ICSE의 온라인 개최를 위한 서비스를 제공하였지만, 해당 서비스를 사용하는 과정에서 발생하는 다양한 문제들에 대한 대응과 세션 체어분들에 대한 사전 교육 등을 SV가 전담하게 되었다.

SV는 크게 두 가지 역할로 나누어졌는데, Zoom을 관리하는 Zoomer와 Zoom에서 진행되는 발표를 OBS (Open Broadcaster Software)를 통해 유튜브로 송출하는 OBSer이었다. OBSer는 말그대로 OBS가 적절하게 Zoom 영상을 송출하는 지를 관리하는 역할을 수행하였으며, Zoomer는 실제 발표 세션을 위한 세션 체어분들의 사전 교육, 체크 리스트 생성 및 수행, 발표 세션에서 발생하는 문제 대응 등의 역할을 맡았다. 나는 Asia Time Zone의 백제 세션을 관리하는 Zoomer이었으며, 약 3일간 해당 세션에 참여하여 Zoomer의 역할을 수행하였다. 다양한 사건들이 있었지만, 가장 힘들었던 건 세션 체어분들 중 몇몇분들이 연락이 되지 않았던 점들이었다. 실제로 몇몇 체어분들은 시작 며칠전에 사전 교육에 참여하시거나 일정 문제 때문에 다른 체어분들로 바뀌셨다. 발표 세션 진행중에도 몇몇 웃픈 사건들이 있었는데, 쉬는 시간을 까먹으신 세션 체어분이 그냥 진행하시다가 강제 중지당하기도 하였고, Zoom의 자동 로그인 기능 때문에 다른 Time Zone의 Zoomer가 ICSE Zoom 계정이 로그인된 노트북을 켜자마자 진행중이던 Zoom 세션이 통째로 종료되기도 하였다.

그럼에도 최고의 학회를 특등석에서 직관할 수 있었던 점은 빼놓을 수 없는 장점이었으며, SV의 관리를 맡으신 교수님들과 다른 교수님들께서도 SV들을 위해 맛있는 저녁과 간식들을 제공해주시는 등 많은 배려를 해주셨다. 정말 색다른 경험을 했었다는 생각이 들며, 학회의 원활한 진행에 직접적인 기여를 했다는 사실이 뿌듯하게 느껴졌다. 그리고 훌륭한 논문들의 발표를 들으며 많은 공부를 할 수 있었고, 다음에는 SV나 참가자가 아니라 발표자로 ICSE에 참여하고 싶다는 작은 소망을 가지게 되었다.





소프트웨어와 안전 summit 행사 소개

■ 소프트웨어재난연구센터
서은영

소프트웨어와 사회안전협회는 소프트웨어 안전 산업 및 기술 육성과 발전을 위한 전문성과 지속성을 가진 구심점 역할을 수행하기 위해 2020년에 설립된 기업 중심의 단체로써, 소프트웨어 안전분야 정책개발, 기술교류 및 전파를 위하여 소프트웨어공학 소사이어티와 2020년 12월 MOU를 체결하였으며 소사이어티의 연구자들이 협회 활동에 전문가로써 참여하고 있다.

"안전한 사회, 안전한 시민을 위한 적극적 사고 예방"을 위해 소프트웨어전문가와 안전전문가의 콜라보로 컨퍼런스가 개최되었다. 2021년 10월 14일 zoom과 youtube를 통한 온라인 스트리밍으로 진행된 '소프트웨어와 사회안전 Summit 2021'은 사단법인 소프트웨어와 사회안전협회와 소프트웨어정책연구소가 주관한 행사이다.

포스트 코로나 시대로 접어들면서 안전에 대한 욕구가 그 어느 때보다도 급속하게 증가하고 있다. 4차 산업 혁명이 도래하면서 우리의 삶은 '편의성'과 '효율성'이라는 목적으로 많은 변화와 시도들이 진행되고 있으며 그 중심에 소프트웨어의 변화가 있다. 이로 인해 우리 사회 곳곳에서 우리의 안전을 위협하는 사고와 문제들이 빈번하게 발생하고 있다. 이제 우리는 자동차, 교통, 보안, 치안, 금융, 의료 등 우리의 모든 삶의 분야에서 '안전'이라는 측면에 대해 시대에 걸맞은 변화와 발전이 이루어질수 있도록 적극적인 협의가 필요하다.

▲ Summit 2021

https://news.mt.co.kr/mtview.php?no=2021101409250063933

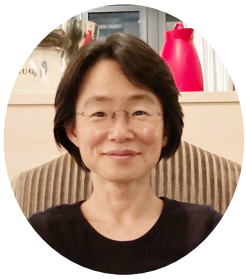
이번 SW와 사회안전 컨퍼런스에서는 사회안전 분야의 전문가들을 모시고 각 분야의 적극적 사고 예방을 위해 우리가 나아가야 할 방안을 모색하였다. 화재안전, 교통/보행자 안전, 영유아 및 노약자 안전, 장애인 안전에 대해 도메인 전문가와 SW전문가의 발표가 있었다. 또한 적극적 시민 안전을 위한 정책과 SW기술 융합의 방향성에 대한 패널토론도 이어졌다.

1. 4차산업혁명시대의 적극적 사고예방:SW전문가의 관점
   민상윤 협회장(소프트웨어와 사회안전협회)
2. 화재사고 예방 노력과 한계, 그리고 대안
   윤성규 발행인(세이프투데이), 홍성복 전무(위니텍) 외 SW전문가
3. 4차 산업혁명시대, 보행교통 안전을 위한 소프트웨어의 방향
   조민제 연구원(치안정책연구소), 이용태 단장(ETRI) 외 SW전문가
4. 영유아 돌연사 방지에 대한 실태 및 현황
   김가람 연구소장(니어베베), 이승주 대표(SPID) 외 SW전문가
5. 시각장애인 안전사고의 현 실태
   홍서준 연구원(한국시각장애인연합회), 나관상 교수(경기대) 외 SW전문가
6. 적극적 시민안전을 위한 정책과 SW기술 융합의 방향성
   차성덕 교수(고려대), 홍장의 회장(SW 공학소사이어티), 권기현 교수(경기대), 민상윤 협회장(소프트웨어와 사회안전협회), 박태형 박사(소프트웨어정책연구소)

▲ 영유아안전 발표자료 참조

국내외 학술행사 소개 III

ISSTA 2022 소개



■ 류석영 대회장
한국과학기술원

안녕하세요, KAIST 전산학부에서 매 학기 프로그래밍 언어 과목을 강의하고, 프로그래밍 언어 연구 그룹 (PLRG@KAIST)을 이끌며, 전산학부장을 맡고 있는 류석영입니다. 저희 연구 그룹에서는 프로그래밍 언어 분야의 엄밀한 기법을 활용하여 소프트웨어 공학과 정보보안 분야의 실제적인 문제를 해결하는 연구를 수행하고 있습니다. 특히, JavaScript 프로그래밍 언어로 작성한 다양한 종류의 소프트웨어에 존재하는 결함을 검출하는 데 세계 최고의 성능을 자랑하는 SAFE 분석기를 개발하였습니다. 최근에는 자연어인 영어로 작성한 JavaScript 언어 명세서를 분석하여 JavaScript 인터프리터, 테스트 코드, JavaScript 명세서의 타입 분석기 등을 자동으로 생성하는 연구를 수행하여, ASE 2020, ASE 2021, ICSE 2021에서 논문을 발표하였고, 그 중 ICSE 2021에서는 ACM SIGSOFT Distinguished Paper Award를 수상했습니다. 추가로, 블록체인 용 언어인 Solidity 코드와 다양한 프로그래밍 언어를 동시에 사용하는 프로그램의 결함 및 취약성을 검출하는 연구로, ICSE 2019, ASE 2020, ICSE 2020, ICSE 2021에서 논문을 발표하였고, 그 중 ASE 2020에서는 ACM SIGSOFT Distinguished Paper Award를 수상했습니다.

이렇게 귀한 자리를 빌어, 제가 General Chair로 함께 하는 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA) 2022 국제학술대회를 소개드리게 되어 매우 기쁘게 생각합니다. ISSTA는 한국정보과학회에서 격년으로 개편하는 우수학술대회 리스트 중 최우수학술대회에 해당합니다. ISSTA 2022는 2022년 7월 18일(월)부터 7월 22일(금)까지 5일 동안 대전의 Daejeon Convention Center (DCC)에서 개최됩니다. ISSTA 2022의 준비 및 진행 상황에 대한 자세한 정보는 계속해서 학회 홈페이지인 https://conf.researchr.org/home/issta-2022에 공지할 예정입니다. ACM의 후원으로 해마다 개최되는 ISSTA 학회는 다양한 소프트웨어 업체의 큰 관심과 재정적인 후원을 받아 운영하고 있습니다. 관심있는 국내 소프트웨어 업계의 재정 후원을 부탁드립니다. ISSTA 2022의 재정 후원에 참여하셔서 국내 소프트웨어 업체를 전 세계적으로 알리는 기회가 되기를 기대합니다. 학회 홈페이지에서 확인할 수 있는 ISSTA 2022의 조직위원회는 15명의 위원 중 6명의 위원이 여성으로, 여성 조직위원이 위원회의 40%를 차지하고 있습니다:

https://conf.researchr.org/committee/issta-2022/issta-2022-organizing-comittee

ISSTA 2022의 General Chair는 KAIST 전산학부의 제가 맡고 있고, 프로그램 위원장은 그리스 아테네 대학의 Yannis Smaragdakis 교수님께서 담당하셔서 매우 큰 힘이 되고 있습니다. Yannis Smaragdakis 교수님은 Datalog을 활용하여 다양한 프로그램을 효율적으로 분석하는 연구의 세계적인 대가이면서도, 매우 꼼꼼하고 성실하고 엄밀한 성격으로 ISSTA 2022의 프로그램 위원장으로서 탁월한 능력을 발휘하시리라 기대가 큽니다. ISSTA 2022는 코로나19 상황으로 인해 2년 가까이 비대면 행사로만 진행해온 국제학술대회를 대면 행사로 진행한다는 데에 매우 큰 의의를 가지고 있습니다. 기후 변화로 인한 환경 문제가 점차 심각해지는 상황에서 굳이 대면 행사를 진행해야 할 지 의견이 분분하지만, 지난 2년 동안의 비대면 국제학술대회 개최 상황을 고려할 때 ISSTA 2022의 대면 행사 개최는 매우 시의적절한 결정이라고 생각합니다. 코로나19 상황은 언제 어떻게 바뀔 지 아무도 예상할 수 없겠지만, 모두의 건강과 안전을 최우선으로 고려하여 진행할 예정입니다.

ISSTA 2022를 한국에서 개최하는 것은 여러 가지로 큰 의미가 있습니다. 소프트웨어 공학 분야 최우수 국제학술대회의 대표격인 ICSE를 2020년에 최초로 한국에 유치하였지만, 안타깝게도 코로나19 상황으로 인해 비대면으로 진행할 수밖에 없었습니다. 이제 위드 코로나 시대를 맞이하며, ISSTA 2022를 한국에서 대면으로 개최하는 것에 대해 매우 기대가 큽니다. 이런 의미에서, KAIST 전산학부 유신 교수님께서 Facebook에 올리신 개인적인 소회를 공유합니다:

매우 감동적이지요? 유신 교수님께서는 ISSTA 2022 Workshop Co-Chair로 함께 해주고 계십니다.

한국에서 소프트웨어 공학을 연구하시는 교수님과 학생, 하루가 다르게 발생하는 새로운 문제를 해결하시느라 불철주야 노력하시는 소프트웨어 업계 분들 모두 ISSTA 2022에 초대합니다. 각자의 자리에서 외롭게 고군분투하던 문제를 국제적인 학술대회에 모여 함께 나누고 해결방안을 찾을 수 있기를 기대합니다. 여러분 모두 진심으로 환영합니다. 2022년 7월 대전 DCC에서 개최하는 ISSTA 2022에서 뵙기를 기대하겠습니다. 감사합니다!

국내외 학술행사 소개 IV

2022년 국내외 학술대회 일정

■ 이주용 (울산과학기술원)

먼저 2022년도 KCSE에 대한 얘기부터 해야겠네요. 제가 우연히 이번에 소사이어티 편집위원과 KCSE 학술위원장을 겸임하게 되었습니다. 2022년에는 "신뢰할 수 있는 인공지능을 위한 소프트웨어공학 기술"이라는 주제로 2022년 1월 19일(수)부터 21일(금)까지 3일간 열릴 예정입니다. 어떻게 소프트웨어의 신뢰성을 담보할 것인가는 소프트웨어 공학에서 전통적으로 중요하게 연구되어 온 문제입니다. 나날이 널리 퍼지고 있는 인공지능 소프트웨어는 신뢰성 문제에 대한 중요성을 더욱 부각시키고 있습니다. 신뢰성에 대한 담보 없이는 인공지능의 활용도가 제한적일 수밖에 없기 때문입니다. KCSE 2022에 참석하셔서 이 흥미롭고 심각한 문제를 어떻게 해결할 수 있을까에 대해 많은 논의가 이루어지기를 바랍니다.

KCSE 2022 외에 2022년에 열리는 학술대회에 대한 정보를 싣습니다. 많은 도움이 되기를 바랍니다.

학술대회	대회 일자	논문 마감	URL
KCSE (한국 소프트웨어공학 학술대회)	1.19 ~ 1.21	2021.12.6	http://sigsoft.or.kr/kcse2022/
CSEET (Conference on Software Engineering Education and Training)	1.4~1.7	마감 완료	https://conferences.computer.org/cseet/
MODELS (International Conference on Model-Driven Engineering and Software Development)	2.6~2.8	2021.11.29	https://modelsward.scitevents.org/
ICWS (International Conference on Web Services)	3.28~3.29	2022.2.28	https://waset.org/web-services-conference-in-march-2022-in-paris
FASE (International Conference on Fundamental Approaches to Software Engineering)	4.2~4.7	마감 완료	https://etaps.org/2022/fase
ICST (International Conference on Software Testing, Verification and Validation)	4.4~4.13	마감 완료	https://conf.researchr.org/series/icst
ICSE (International Conference on Software Engineering)	5.21 ~5.29	마감 완료	https://conf.researchr.org/home/icse-2022
MSR (Mining Software Repositories Conference)	5.23~5.24	2022.1.20	https://conf.researchr.org/home/msr-2022
ICSOC (International Conference on Service Oriented Computing)	5.26~5.27	2022.4.23	https://waset.org/service-oriented-computing-conference-in-may-2022-in-barcelona
ISSTA (International Symposium on Software Testing and Analysis)	7.18 ~7.22	2022.1.28	https://conf.researchr.org/home/issta-2022
RE (International Requirements Engineering conference)	8.15~8.19	2022.2.24	https://conf.researchr.org/home/RE-2022
ASE (International Conference On Automated Software Engineering)	9.26~10.1	2022.5.26	https://conf.researchr.org/home/ase-2022
ICSME (International Conference on Software Maintenance and Evolution)	10.3~10.7	2022.4.1	https://cyprusconferences.org/icsme2022/
ESEC/FSE (Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering)	11.14 ~11.18	2022.3.17	https://conf.researchr.org/home/fse-2022

About the Center

• 최윤자 (경북대학교)

• 홍 신 (한동대학교)

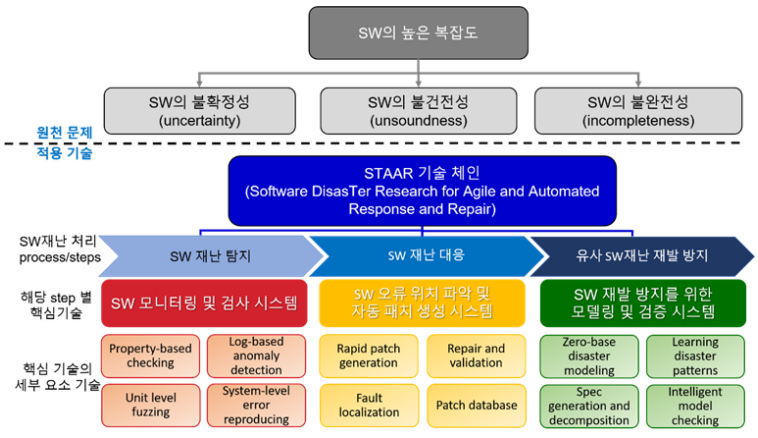
소프트웨어재난 연구센터

1. 센터소개

지난 10월 27일 소프트웨어재난연구센터가 경북대학교 글로벌플라자 12층에 개소하였습니다. 소프트웨어재난연구센터는 과학기술정보통신부와 한국연구재단이 주관하는 2021년 선도연구센터 사업의 공학분야에 선정되어 앞으로 7년간 과학기술정보통신부의 지원을 받게 된 소프트웨어 분야 선도연구센터(ERC) 입니다. 소프트웨어재난(災難)은 의도하지 않은 소프트웨어 동작으로 인해 광범위한 인명, 재산, 또는 사회적 피해가 발생하는 상황으로, 대다수의 사회 기반이 소프트웨어 시스템으로 운영되는 오늘날 사회의 중요한 위협으로 인식되고 있습니다. 1968년 Margaret Hamilton 박사가 소프트웨어공학이라는 용어를 처음 사용하기 시작한 이래 50여년간, 전 세계 소프트웨어 커뮤니티에서는 안전하고 신뢰할 수 있는 소프트웨어의 개발을 위해 할 수 있는 모든 노력을 다 해 왔습니다. 그럼에도 불구하고, 소프트웨어로 인한 재난은 오히려 증가하고 그 규모와 피해도 방대해지고 있습니다. 자동차의 급발진 사고, 자율주행차의 인명사고, 항공기 추락사고, 빈번한 소프트웨어 보안사고, 불과 한 달 전에 발생한 KT 의 통신장애 사고 등, 소프트웨어재난의 위협은 바로 우리 곁에 도사리고 있습니다.

이러한 소프트웨어재난 문제의 해결을 위해 소프트웨어재난 연구센터는 재난의 발생을 사전에 100% 예방하려는 기존의 접근방식에서 벗어나, 재난의 발생 가능성을 인정하고 재난상황에 신속하게 대응하여 문제의 확산을 막는 방법과 기술의 개발에 역점을 두고 있습니다. 화재가 발생했을 때의 소방관의 역할과 같은 것이라고 생각할 수 있습니다. 이를 위하여, 시스템 수준의 재난상황을 신속하게 탐지하는 자동분석 기술 (런타임 시스템 로그분석 기술, SW시스템 자동테스팅 기술 등), 탐지된 재난상황을 소프트웨어 코드 수준의 원인으로 자동분석하는 국지화 기술 (오류 재현 기술, SW오류위치 자동추정 기술 등), 코드 수준의 오류를 긴급 대응하여 재난의 확산을 방지하는 기술 (SW 핫패치 생성 기술), 코드 수준 오류를 안전한 코드로 자동 수정하는 기술 (SW자동 패치 생성 기술, 패치 검증 기술 등), 그리고 식별된 오류가 더 이상 유사한 재난을 재발하지 않음을 보이는 엄밀검증 기술(재난상황 모델링 기술, 지능형 엄밀검증 기술, 재난데이터베이스 구축기술 등) 들을 연구개발하여 SW재난의 신속한 탐지 및 대응과 재발 방지를 위한 SW재난관리 체계인 STAAR(Software DisasTer Research for Agile and Automated Response and Repair) 프레임워크(그림 1)로 완성하고자 합니다. 소프트웨어재난연구센터는 이러한 원천기술들의 개발부터 실증연구까지를 목표로 하여 앞으로 7년간의 대장정을 시작하였습니다.

STAAR
SOFTWEAR DISASTER RESEARCH CENTER
AGILE & AUTOMATED RESPONSE & REPAIR



▲ 그림1. 소프트웨어재난관리 프레임워크 STAAR 개념도

▲ 그림 2. 2021년 10월 27일 소프트웨어재난연구센터 개소식

그림 3. 소프트웨어재난연구센터 참여교수

2. 참여교수 소개

소프트웨어재난 연구센터는 소프트웨어재난 탐지 연구그룹, 소프트웨어재난 대응 연구그룹, 소프트웨어재난 재발방지 연구그룹의 세 개의 연구그룹으로 나누어져 STAAR 프레임워크의 개발을 추진하고 있으며, 전국 7개대학 13분의 해당 분야 최고수준의 기술력을 갖춘 교수님들과 70여명의 석박사 과정 연구원들이 참여하고 있습니다.

2.1. 소프트웨어재난 탐지 연구그룹

소프트웨어재난 탐지기술 연구그룹에서는 시스템 실패부터 메모리 상태까지 다양한 수준에서 소프트웨어 오동작을 관측/분석할 수 있게 지원하는 자동화 기법을 연구 개발하여 연계하는 방법을 만들 목표를 가지고 있습니다. 이 연구그룹에서는 소프트웨어 분석 및 자동 테스팅 기술의 전문성을 갖춘 KAIST 김문주 교수와 류석영 교수, 경북대 탁병철 교수, 한동대 홍신 교수가 참여하며, 런타임 시스템 로그 분석, 프로그램 분석, 테스팅 분석 및 재현 기술의 개발을 위한 연구 개발과 공동연구를 이끌어갈 계획입니다. 또한, 소프트웨어 검증 자동화 전문 기업인 브이플러스랩(V+Lab), 슈어소프트테크와 함께 산학 협력을 진행할 것입니다.

2.2. 소프트웨어재난 대응 연구그룹

소프트웨어 시스템의 오류로 인한 재난 발생 시 신속한 복구를 지원하고 피해 확산을 최소화하기 위해서는 오류가 발생한 소프트웨어를 실행시간(on-the-fly)에 정정하는 자동화된 디버깅 기술이 필수적입니다. 소프트웨어재난 대응 기술 연구그룹은 소프트웨어 자동수정 분야에 우수한 연구를 이어가고 있는 고려대 오학주 교수, UNIST 이주용 교수, 경북대 권영우 교수, 김동선 교수, 한양대 김윤호 교수가 참여하여 소프트웨어 오류의 신속 식별, 신속 복구, 신속 검증, 신속 코드 수정을 지원하는 기술을 개발하고 연계함으로써 전통적인 소프트웨어 디버깅 관행의 한계점을 해결하는 연구에 도전하며, 코드마인드와 위니텍과 산학 협력을 수행할 예정입니다.

2.3. 소프트웨어재난 재발방지 연구그룹

소프트웨어재난의 엄밀한 분석은 재발방지를 위해 필수적인 과정이지만, 일반적인 SW 오류를 분석하기 위해 개발된 전통적인 모델 검증 기술은 사회 인프라 등에 사용되는 실제적인 SW 시스템을 엄밀하게 분석하는데 많은 한계점을 가지고 있습니다. 소프트웨어재난 재발방지 기술 연구그룹은 식별되고 수정된 재난오류를 중심으로 소프트웨어 시스템의 모델 수준의 검증을 수행하여 향후 동일한 원인에 의해 발생하는 소프트웨어재난의 재발을 방지하는 기술을 연구합니다. 이 연구그룹에는 소프트웨어 모델링 및 검증에 전문성을 갖추고 있는 경북대 최윤자 교수, POSTECH 배경민 교수, 한양대 이우석 교수와 KAIST 허기홍 교수가 참여하여, 모델자동생성, 명세추출 및 세분화, 지능형 모델검증 기술 등의 연구를 수행하며, 포멀웍스, 무지개연구소와 연구 교류를 추진할 예정입니다.

3. 역사적 의의

소프트웨어재난 연구센터는 (인공지능을 제외한) 순수 소프트웨어 분야에서는 ERC 30년의 역사에서 매우 드물게 세번째로 선정된 소프트웨어분야 선도연구센터입니다. 또한, 참여교수들 중 70%에 달하는 아홉분의 교수님들이 지난 2008년에 서울대학교 이광근 교수님께서 연구책임자로 ERC 에 선정되었던 무결점소프트웨어연구센터에 신임교수 또는 석박사과정 연구원으로 참여했었던 이력이 있습니다. 순수 소프트웨어 분야에서 두번째 선도 연구센터였던 무결점소프트웨어연구센터를 통해 양성된 인력들이 10여년 후 세번째 선도연구센터의 주축이 되어 국내외 소프트웨어 분석/검증 연구분야의 활성화를 주도해 나갈 것이라는 점에서 매우 큰 의의가 있습니다. 저희 센터는 연구목표의 달성 뿐 아니라, 국내외 최고수준의 인력양성을 통하여 소프트웨어 분석/검증 기술의 저변확대에 기여하고자 하며, 네번째, 다섯번째 소프트웨어분야 ERC 가 계속해서 탄생할 수 있도록 마중물의 역할을 하겠습니다.

4. 맺음말

소프트웨어재난 연구센터는 소프트웨어재난 탐지, 신속대응, 재발방지 기술을 연계한 프레임워크 개발에 도전함으로써, 소프트웨어재난이라는 '질병'에 대응하는 정확하고 빠른 치료제 기술의 개발을 향해 출발하였습니다. 세계적으로도 선례를 찾아볼 수 없는 매우 도전적인 과제를 시작하면서 저희 참여연구원들은 실패에 대한 두려움을 감히 용기로 바꾸어 보고자 합니다. 소프트웨어공학 소사이어티 여러분들의 관심과 응원 부탁드립니다.

[연구원 모집]

소프트웨어재난 연구센터에서는 함께하실 석사후/박사후 연구원을 상시 모집하고 있습니다. 자유양식의 이력서를 센터의 서은영 (eyseo1015@gmail.com) 선생님께 제출하시면 연락드리겠습니다.

[홈페이지]

https://staar.knu.ac.kr

V VIEWPOINTS

소사이이어티 광장 I

소사이어티 알림

- 소프트웨어공학 소사이어티에서는 매년 소프트웨어공학 우수논문상을 추천하여 시상하고 있습니다. 내년부터는 최우수 학술대회에 논문 발표로 참가하는 학생에게 장려금(약 100만원 수준)을 지원할 예정입니다.
- 소프트웨어공학 소사이어티 소개 동영상 : https://www.youtube.com/watch?v=HWGsy-PyIe0
- 소프트웨어공학 소사이어티 페이스북 : https://www.facebook.com/groups/668196744037453

축하합니다!

• 신규 임용

- 차수영 박사(고려대학교, 지도교수: 오학주) 2021년 9월 성균관대학교 소프트웨어융합대학 조교수 임용
- 황성재 박사(KAIST, 지도교수: 류석영) 2021년 9월 성균관대학교 소프트웨어융합대학 조교수 임용

• 박사학위수여

- 손정주 박사(KAIST, 지도교수: 유신) 2021년 8월 박사학위 취득.
  (논문제목: 결함 위치 식별과 결함 예측간의 보완적 상호작용)
  졸업 후 University of Luxembourg 박사후연구원 시작
- 이낙원 박사 (KAIST, 지도교수: 백종문/공동 지도교수: 김문주) 2022년 2월 박사학위 취득.
  (논문제목: A Novel Target Directed Model Checking for Fast Abstract Reachability Analysis)
  졸업 후 한양대학교에서 박사후 연구원 시작 예정

• 박사과정 국제학술지 논문게제

- 경상대학교 조희태 박사과정
  Cho, Heetae, Seonah Lee, and Sungwon Kang, "Classifying issue reports according to feature descriptions in a user manual based on a deep learning model," Information and Software Technology 142 (2022): 106743.
  https://www.sciencedirect.com/science/article/pii/S0950584921001890
- KAIST 이낙원 박사과정
  N.Lee, Y.Kim, M.Kim, D.Ryu and J.Baik, "Directed Model Checking for Fast Abstract Reachability Analysis", IEEE Access, Volume 9, page 158738-158750, Nov 25, 2021.
  https://ieeexplore.ieee.org/abstract/document/9627119

• 구인

- UNIST(울산과학기술원) 컴퓨터공학과 우수교원후보 풀(Pool) 상시 모집
- 경북대학교 컴퓨터학부 우수교원후보 풀(Pool) 상시 모집: 자유양식의 CV 를 scse@knu.ac.kr 로 제출

소사이이어티 광장 II

KAIST 김문주 교수가 창업한 V+Lab 판교 테크노밸리로 확장 이전

1. V+Lab 소개

V+Lab은 4차 산업혁명의 핵심인 SW의 품질을 향상하고 SW 테스팅/검증에 소요되는 기간 및 비용을 크게 절감하는 SW 자동 테스팅 도구 및 서비스를 제공하고자, KAIST 전산학부 김문주 교수가 2019.10월 창업한 Deep Tech 스타트업 기업이다 (https://vpluslab.kr/).

그림 1. V+Lab 김문주 대표

V+Lab은 AI 기반 Concolic 테스팅 기술을 사용해 SW 오류를 자동으로 빠르게 발견하는 SW 자동 테스트 도구 CROWN 2.0 개발 및 연간 사용 라이센스를 판매하며, 적은 비용으로 빠르게 SW 신뢰성을 향상하는 테스팅 서비스 및 교육을 제공하고 있다.

CROWN 2.0의 핵심인 Concolic 테스팅 기술은 소스 코드 분석과 동적 테스팅, AI 기반의 SMT 솔버를 동시에 이용하는 자동 테스팅 기법이다 (소스 코드 분석 및 동적 테스팅에서 얻은 정보를 사용하여, 특정 코드 라인을 실행하기 위한 입력값의 조건을 구한 뒤, SMT solver로 조건을 만족하는 입력을 구하는 작업 수행). CROWN 2.0은 소스코드 분석 및 단위 테스팅 자동화에 필요한 드라이버와 스텁 코드를 모두 자동으로 생성하여 테스트 생성 과정을 전자동화하는 토털솔루션이다.

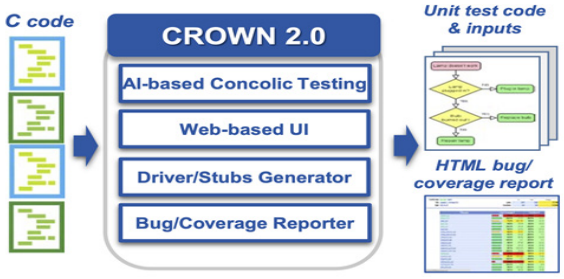


그림 2. CROWN 2.0 구성도

CROWN 2.0은 자동차, 보안, 국방 등 안전필수 시스템을 대상으로 높은 테스팅 효과를 달성하는 세계 최초의 SW 자동 테스팅 상용 도구로써, 김문주 대표가 세계적으로 연구를 선도하고 있는 AI Concolic 테스팅 기술을 핵심엔진으로 사용한다. 김문주 대표는 KAIST 김윤호 연구교수와 (현. 한양대학교 조교수) CROWN 2.0 이전 연구 프로토타입을 현대모비스의 20만줄 규모의 차량용 SW에 적용하여, 테스팅 인력을 절반 이상 감소시키는 등 (연합뉴스 18.07.22.: 현대모비스, AI기반 소프트웨어 검증시스템 도입 "효율 2배로"), 삼성, LG, 현대차, 만도, LIGnex1 등 안전필수 시스템 기업에 적용하여 수백여건의 심각한 crash 오류를 빠르고 정확하게 검출하였다.

V+Lab이 실무에 적용 가능한 자동화 솔루션인 CROWN 2.0을 만들 수 있었던 원동력은 10년 이상 Concolic 테스팅 연구로 다져진 기술력이다. 김문주 교수는 지난 15여년간 소프트웨어 공학 부문 최고 학술대회인 ICSE, FSE, ASE 등에 꾸준히 SW 자동 테스팅 연구 논문을 발표해 왔고, ICST 2018을 비롯해 국내외 학술대회에서 다수의 우수논문상을 받았다. 또한, 소프트웨어 공학 분야 최고의 국제학회인 ICSE 2020에서 산학협력 연구를 발표하는 Software Engineering In Practice (SEIP) 트랙의 프로그램 좌장을 맡아, 실용적인 연구에 대한 기여를 국제적으로 인정받았다.

2. V+Lab 사업 현황 및 비전

현재 V+Lab의 주 타겟은 국내외 자동차, 보안, 국방 등 안전필수 시스템 SW 시장으로, 해당 시장은 지속적으로 크게 성장하고 있다 (예. 세계 자동차 SW 자동 테스팅 시장 2020년 8.8조원, 2030년 13.9조원 추정 (Survey by Yano Research 2020/09)).

해당 시장은 소프트웨어 테스팅 제품이 B2B로 거래되기 때문에, 백년지대계의 장기적 안목으로 함께 성장할 고객의 신뢰를 얻을 수 있는 고품질 제품 및 서비스제공에 최선의 노력을 기울인 결과, 벤처기업 인증 (혁신성장 유형)을 획득하였고, 창업 2년 만에 현대자동차, ETRI, 국가보안기술연구소, 포항공대 등 대표적인 산/학/연 기관에 CROWN 2.0을 납품하였다.

일반적으로 SW 개발 비용 중 절반 이상이 테스팅에 소요되기 때문에, 전세계적으로 이를 줄일 수 있는 SW 테스팅 자동화 도구에 대한 수요는 매우 크다. V+Lab은 세계시장 공략 첫 단계로 2024년 일본 자동차 SW 테스팅 시장 진출을 목표로 하고 있다 (일본 JAIST의 Aoki 교수, Denso의 Chiba 박사 등 자동차 SW 테스팅/검증 연구를 적극적으로 수행한 해외 연구자들과의 인적 네트워크를 활용하여 현지 진출 준비).

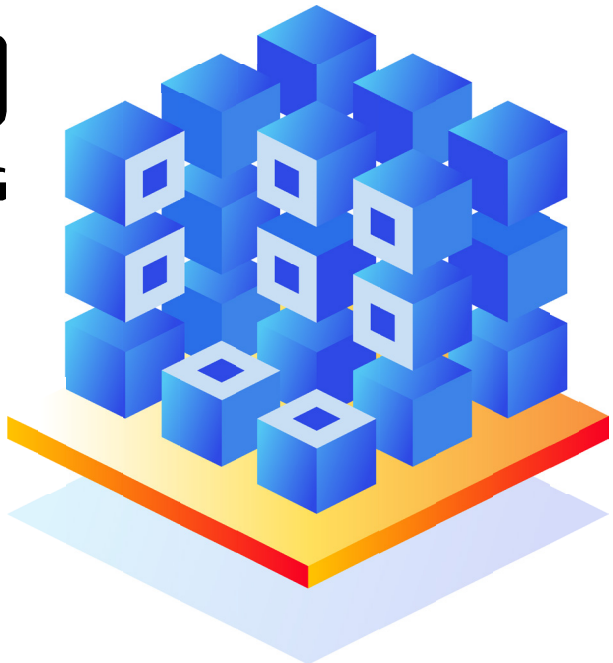
또한, 안전필수 시스템 도메인 뿐 아니라, 머신러닝 기반 AI SW 등 보다 넓은 도메인을 대상으로 적용할 수 있도록, SW 자동 테스팅 기술을 폭넓게 발전시키며, 발견한 오류의 위치를 정확하게 자동으로 추정한 후, 해당 오류를 수정하는 patch를 자동으로 생성하는 도구들을 개발할 예정이다. 그리고, 유닛 테스팅 결과를 synthesis를 통해 통합 테스팅으로 진화시키는 종합 SW 테스팅 및 검증 플랫폼을 연구/개발할 계획이다.

상기한 비전을 실현하기 위해, 많은 소프트웨어공학 소사이어티 회원분들

의 축하 속에, 2021.8월 대전에서 판교 테크노밸리로 회사를 확장 이전하였다 (주소: 13493 경기도 성남시 분당구 판교역로 230 삼환하이펙스 B동 202호. 031-698-3134). 앞으로 판교 테크노밸리의 대규모 인프라 및 네트워을 활용하여 (예. 판교 소재 LIG Nex1과 컨소시움을 형성하여 대규모 국방부 과제 수주 및 한컴 인텔리전스와 SW 신뢰성 제고 사업 협력), V+Lab이 SW 자동 테스팅 분야의 글로벌 선도기업으로서 한단계 더 성장할 것을 기대한다.

그림 3. 판교 테크노밸리의 V+Lab 사무실 사진

V+Lab
INNOVATING SW TESTING



: 기고문 및 소식 모집

소프트웨어공학 소사이어티 소식지는 여러 연구자분들의 생각과 소식을 나누는 광장입니다. 다음과 같은 구성으로 소식지를 구성하고자 하오니, 여러분들의 적극적인 참여를 바랍니다. 투고글의 형식은 자유형식이며, 분량은 A4 기준 2~4페이지 입니다.

- 기고문 : 소프트웨어공학 및 소사이어티에 대한 생각 (자유주제)
- 신진연구자 소개: 만 40세 이하 또는 박사학위 취득 후 7년 이내의 연구자 소개
- 국내외 학술행사 소개: 주요 학술행사 소개, 학술행사 참여 후기 등
- 기관소개: 소프트웨어공학연구 관련기관 소개
- 소사이어티 광장: 소사이어티의 새로운 소식 나눔

SOFTWARE
ENGINEERING
SOCIETY

제출방법:

- 이메일 제출 (ksepup@gmail.com)

문의처:

- 최윤자 교수 (경북대학교, 053-950-7549, yuchoi76@knu.ac.kr)
- 지은경 교수 (한국과학기술원, 042-350-7810, ekjee@se.kaist.ac.kr)
- 이주용 교수 (울산과학기술원, 052-217-2123, jooyong@unist.ac.kr)

: 소사이어티 조직도

SOFTWARE ENGINEERING SOCIETY

회장 홍장의
충북대학교

총무 서영석
영남대학교

총무 한종대
상명대학교

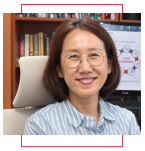
감사 김정아
가톨릭관동대학교

감사 김재웅
TTA

부회장(기획) 이찬근
중앙대학교

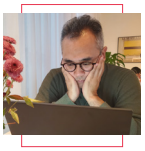
부회장(학술) 고인영
KAIST

부회장(교육) 이정원
아주대학교



부회장(편집) 최윤자
경북대학교

부회장(대외협력) 백종문
KAIST



부회장(조직 및 지원) 유준범
건국대학교

분과위원장
(블록체인) 박용범
단국대학교

기획이사 김 순 태
전북대학교

기획이사 남 재 창
한동대학교

학술이사 유 신
KAIST

학술이사 배 경 민
포항공과대학교

교육이사 박 수 진
서강대학교

교육이사 서 주 영
아주대학교

편집이사 지 은 경
KAIST

편집이사 이 주 용
UNIST

대외협력이사 이 근
삼성전자

대외협력이사 최 희 원
LG전자

조직및지원이사 민 상 윤
솔루션링크

조직및지원이사 윤 경 아
SK텔레콤

발행정보

발행일 2021년 12월 15일
발행인 홍장의
발행처 사단법인 한국정보과학회 소프트웨어공학소사이어티
연락처 충북 청주시 서원구 충대로 1번지 충북대학교 전자정보대학 3관 320호 소프트웨어학과
홍장의 (전화 : 043-261-2261, 팩스 : 043-273-2265, 홈페이지 : http://www.sigsoft.or.kr)

SOFTWARE
ENGINEERING
SOCIETY

사단법인 한국정보과학회 소프트웨어공학소사이어티

주소: 충북 청주시 서원구 충대로 1번지 충북대학교 전자정보대학 3관 320호 소프트웨어학과
홍장의 (전화 : 043-261-2261, 팩스 : 043-273-2265)
홈페이지 : http://www.sigsoft.or.kr