소프트웨어 공학 과제 1: No Silver Bullet

20201617 이상연, 20200185 박정주, 20201593 송은수

20200095 임정연, 20201651 최승규, 20211547 신지원

1. 내용 요약

논문 제목에 포함된 단어인 'Silver Bullet'은 늑대인간 이야기 속 은총알을 뜻하며, '마법 같은 해결책' 정도로 의역할 수 있습니다. 필자는 크게 네 가지 주제에 대해 다루며 '**소프트웨어 공학에서 은총알이 존재할 수 없는 이유**'에 대해 논합니다.

첫 번째 주제로, 소프트웨어 공학의 **본질적 어려움**에 대해 이야기합니다. 먼저, 소프트웨어는 큰 복잡성을 가집니다. 클래스, 반복문 등을 통해 반복을 제거하므로 중복되는 요소 없이 복잡합니다. 또, 소프트웨어는 필연적으로 호환성을 고려해야 합니다. 여러 사람에 의해 완성되고 때에 따라 인터페이스에 맞춰져야 합니다. 거기에 더해 소프트웨어는 항상 변경 가능성이 열려 있습니다. 큰 변화가 불가한 건물, 자동차 등에 비해 항상 변경의 여지가 있습니다. 다른 요소에 비해 많은 변경이 가능해 외부 요인에 의해 항상 변경의 압박을 받습니다. 비가시성 또한 본질적으로 주어지는 어려움입니다. 소프트웨어는 물리적으로 존재하지 않는 것을 나타내므로 시각화하기 어렵고, 이 점이 설계 과정을 까다롭게 합니다.

두 번째로는 **과거의 해결책**들을 언급하며, 그 해결책들이 본질적 어려움이 아닌 부가적 문제들을 해결하는데 그쳤다는 사실을 설명합니다. 고수준 언어, 시간 공유, 통합 개발 환경 등이 그것이며, 개발 과정에서 편리함은 제공하지만 복잡성과 같은 본질적인 어려움을 해결할 수 없음을 지적합니다.

셋째로, **해결책으로 기대되는 것들**과 예상되는 한계점에 대해 말합니다. 더 발전된 고수준 언어, 객체지향 등은 설계 단계를 간소화할 수 있지만 이로 얻는 보상이 한정적이며 역시 본질적 어려움은 제거할 수 없습니다. 인공지능을 활용한 전문가 시스템, 자동 프로그래밍 등이 해결책이 될 수 있지만 실현에는 큰 어려움이 따릅니다. 그래픽 프로그래밍, 프로그래밍 검증, 워크스테이션 등은 작업 도중의 부하는 줄일 수 있으나 여전히 비가시성, 복잡함 등의 문제가 남을 수밖에 없습니다.

끝으로, **그럼에도 불구하고 개선하기 위해 나아가야 할 방향**을 제시합니다. 개념을 구상하는 것이 작업의 대부분을 차지하므로, 구현을 쉽게 하는 방법만으로는 생산성 향상을 기대하기 힘듭니다. 따라서 다양한 접근 방식을 통해 개념 구상에 드는 노력을 줄여 조금씩 개선을 도모할 필요가 있습니다. 빠르게 프로토타입을 개발하고 수정하거나, 탑다운 방식으로 조금씩 상세히 프로그래밍하는 방법이 있습니다. 새로 개발하기보다 기존의 프로그램을 구매해 사용하는 편이 나을 수도 있고, 훌륭한 개발자를 기르는 방안도 있습니다

2. 토의

<Essential Difficulties>

이상연: 안녕하세요, 논문에서 다루는 내용에 대해 글의 순서대로 의견을 나눠볼까 합니다. 저는 진행을 맡으면서 토의에도 조금씩 참여할 예정입니다. 먼저 '소프트웨어 공학이 가지는 본질적 어려움'에 관해 글을 읽고 든 생각을 자유롭게 나눠주세요.

최승규: 저는 복잡성에 관한 문단에서 소프트웨어에는 중복되는 요소가 없다는 말이 생각해본 적 없던 부분이라 기억에 남아요. 함수나 변수를 선언하는 방법 등으로 반복을 최소화하는 과정을 이미 거치며 프로그래밍하고 있었는데 이를 자각하지 못하고 있던 것 같아서 놀랐어요.
변경 가능성에 관한 이야기도 크게 와닿아요. 이것 때문에 변경 가능성에 대한 대책을 세울 수 있도록 하는 소프트웨어 공학이 중요하다고 생각해요.

박정주: 필자는 비가시성에 대해 설명하면서 소프트웨어의 본질은 추상적인 개념이며, 소프트웨어 구상 시 어려운 부분은 이러한 개념적 구성의 명세, 설계, 테스트라고 말했어요. 표현을 위해 코드를 짜는 것은 사소한 일이라고 주장했는데, 여기에 동의해요. 실제로 구현보다는 기획과 프로젝트 설계에 훨씬 많은 시간을 들이는 것 같더라고요. 어떤 기능이 필수적인지, 그 기능을 유저에게 어떻게 보여줄 것인지 결정하는 것이 실제로 코드를 짜는 것보다 훨씬 중요하다고 생각합니다.

신지원: 비가시성에 대해 이야기하면서 소프트웨어 구조를 표현한 그래프에서 계층이 나타날 때까지 링크를 잘라야 한다고 주장했는데, 저는 여기에 의문이 들었어요. 모든 소프트웨어는 입력과 출력이 있고, 이를 보다 체계적으로 만들기 위해 설계자가 의도한 대로 구조화되어야 한다고 생각해요. 그렇다면 프로젝트의 계층 구조를 만들기 위해 임의로 링크를 자르는 것이 올바른 조치가 맞는 걸까요? 이 주장은 설계자가 역으로 계층 구조에 얽매이도록 한다고 생각해요.

이상연: 개발을 편리하게 해줄 방법이 많아졌지만 그럼에도 아직 소프트웨어 개발은 어렵고, 공부해야 할 것들이 많기 때문에 전반적으로 필자의 말들은 오늘날에도 타당한 주장인 것 같습니다.
비가시성에 대한 필자의 대처에는 의문을 제시하시기도 했지만, 조원분들도 대체로 글에서 주장하는 소프트웨어 공학의 본질적 어려움에 대해 공감하셨네요. 다음으로는 이에 대한 '과거의 해결책들과, 해결책이 될 수 있는 후보들'에 대해 이야기해보겠습니다.

<Past Breakthrough Solved Accidental Difficulties, Hopes for the Silver>

임정연: 우발적 어려움을 해결한 방법으로 고수준 언어만을 강조해서 처음엔 의아했어요. Low-level도 매우 중요하고, 더 전문성이 들어가는 분야라고 생각해서 조금 의문이 들었어요. 그런데 Changeability가 소프트웨어의 큰 장점이라고 이야기한 앞부분을 다시 보니, 소프트웨어의 발전을 위해 Changeability를 활용할 수 있겠다는 생각이 들었어요. 고수준 언어를 적극적으로 활용하면 input이 늘어나고, 그에 따른 발전이 있으니 강조했다고 생각해요.

박정주: 오늘날 GPT는 expert system은 아니지만, 프로그래밍뿐만 아니라 광범위한 도메인에서 expert system과 같은 역할을 하고 있어요. 당시에는 머신러닝, 딥러닝은 물론이고 이렇게 거대한 LLM이 만들어

질 것이라고 상상하지 못했을 텐데, 필자가 굉장한 인사이트를 가지고 있었다는 생각을 해요.

최승규: 저는 필자가 AI 의 발전 가능성을 너무 낮게 본 것 같아요. 컴퓨팅 파워의 증가로 인공지능은 당시와는 비교할 수 없이 똑똑해요. GPT, Github Copliot 등이 필자가 불가능할 것이라 봤던 Expert System 에 가깝다고 봐요. 어쩌면 이게 커다란 혁신이 될 수 있다고 생각해요.

신지원: 저는 앞서 꺼내신 의견과 반대로 필자가 AI 의 한계를 잘 진단했다고 생각해요. AI 가 개념의 구현은 쉽게 해준 것이 사실이지만, 개념의 구상 자체에는 큰 영향을 주지 못하는 것 같아요. 기획 의도, 법률적 규제, 물리적 상황 등에 맞물려 매우 다양한 디자인 패턴이 소프트웨어 개발 과정에서 나타나는데, 창의성이 필요한 이런 일들을 AI 가 해결해줄 수는 없다고 보여요.

송은수: 요즘 인공지능은 무엇을 하려고 하는지를 말하면 구체적으로 필요한 걸 알아서 제시해줘요. 또한 창의성이라는 것도 결국 기존의 지식에 기반하는 것인데, 이미 데이터화 된 아이디어와 자료가 많기 때문에 AI 또한 창의력을 발휘할 수 있지 않을까 싶어요. AI 가 은총알이 될 수 있지 않을까요?

이상연: 고수준 언어의 효용에 대해서 이야기해 주셨고, 인공지능에 관한 필자의 예상에는 의견이 갈렸네요. 마지막으로 '개념적 본질에 관한 해결책'에 대한 문단을 가지고 이야기해 보겠습니다.

<Promising Attacks on the Conceptual Essence>

송은수: 글의 초반부를 읽으며 단계 별로 나누어서 접근하는 전략으로 개념에 관한 본질적 어려움을 해결할 수 있지 않을까 하는 생각을 했어요. 예를 들면 코딩 따로, 라우팅 따로, 배포 따로 접근하는 식으로요. 빠르게 프로토타입을 개발하고 수정하는 방법이나, 탑-다운 방식으로 하나씩 접근하는 방식 등 필자가 제시한 해결책이 제가 생각한 해결책과 유사해서 신기했어요.
논문 후반부에선 좋은 디자인의 중요성만 너무 강조한 것 같아요. 디자인을 잘하려면 소프트웨어의 본질적인 하드웨어 메모리나, 알고리즘을 잘 알아야 할 텐데 이를 더 강조해야 했다고 생각해요.

임정연: 소프트웨어를 빌드하는 것보다 구매하는 편이 싸다는 말이 인상 깊었어요. 특히 읽으면서 클라우드 분야의 예시가 생각났어요. 불과 몇 십년 전만 해도 자체 서버를 가지는 회사들이 많았던 반면, 요즘에는 클라우드 플랫폼을 이용하는 경우가 많아요. 대표적인 예시로 넷플릭스처럼 큰 회사도 AWS 클라우드를 이용하고 있어요. 앞으로 소프트웨어가 발전하면 발전할수록, 이런 추세가 강화될 것이라고 생각해요. 이러한 견해를 40년 전에 발표한 필자의 통찰력이 놀랍기도 해요.

이상연: 필자는 소프트웨어 개발의 복잡성을 한 번에 해소할 수 있는 은총알은 결국 존재하지 않다는 의견을 한 번 더 강조했어요. 이 주장은 오늘날까지도 유효한 주장이라고 생각해요. AI를 비롯한 수많은 기술이 어려움을 줄이는 데 큰 도움을 주는 건 사실이지만 완전히 해결해주지는 않으니까요.
한편 이렇게 많은 기술들이 등장하고 사라지고 급격하게 변화하는 시대일수록 기본적이고 기초적인 분야에 대한 지식이 중요하다고 생각해요. 이러한 기초가 확실하게 뒷받침될 때 좋은 소프트웨어를 개발할 수 있을 거예요.
이번 토의를 통해 소프트웨어 개발의 복잡성에 대한 이해를 높이고, 기술의 발전에 따라 지속적인 도전과 학습이 필요하다는 것을 확인했습니다. 앞으로도 꾸준히 학습에 정진하여 더 나은 소프트웨어 개발자로 함께 성장할 수 있길 바라요. 의견 나눠 주셔서 감사합니다.