소프트웨어공학과 초등교육

박성순°, 한선관, 이철환
경인교육대학교 초등컴퓨터교육과
jamespl@dreamwiz.com, han@gin.ac.kr, chlee56@gin.ac.kr

Software Engineering and Elementary Education

seong-soon Park°, Sun-Gwan Han, Cul-Hwan Lee
Gyeongin National University of Education, Dept. of Computer Education

요 약

컴퓨터 시스템에 있어 소프트웨어가 차지하는 비중과 중요성이 점차 부각되는 시점에서 '소프트웨어 위기'라 불리는 다양한 문제점들이 대두되고 있다. 21세기 미래정보사회를 이끌어 갈 어린이들이 빠르게는 유아기부터 다양한 종류의 소프트웨어를 사용하고 있으며 이런 흐름이 하나의 생활양식과 문화로 인식되어 가고 있다. 이에 본 연구에서는 컴퓨터 교육 내용학의 한 분야인 소프트웨어 공학을 초등학생에게 지도할 수 있는 방안을 모색하여 소프트웨어의 개념과 다양한 소프트웨어의 장단점을 알고 자신에게 필요한 소프트웨어를 선택하며 나아가 미래 소프트웨어 개발의 꿈을 키울 수 있도록 초등학생의 수준에 적합한 학습모형을 제시하였다.

1. 서론

1.1 연구의 필요성 및 목적

컴퓨터와 소프트웨어가 사회조직에서 중요성이 증가하고 소프트웨어의 전문성이 증가되면서 산업체에서는 소프트웨어의 처리 절차 개선에 대한 관심이 커졌고, 크고 복잡한 컴퓨터 시스템에서의 양질의 공학 실습을 하는 것이 전략적인 중요성으로 부각되었다. 이러한 추세에 따라 1968 NATO SE 협회가 열린 이래로 여러 국가에서 소프트웨어 공학 훈련에 대한 관심이 꾸준히 증가되어 왔다[1]. 1970년을 기점으로 시작된 우리나라 전자계산교육은 그동안 많은 발전을 하여 현재 4년제 대학의 컴퓨터 전공학과가 100여개 설치 운영되고 있다[2]. 그러나 우리 나라의 응용 소프트웨어의 개발은 프로그래머 개개인의 능력에 의존하고 있는 것이 사실이며 컴퓨터를 전공한 졸업자들조차 실습능력의 부족, 적절성의 부족, 소프트웨어 개발에 포함되는 관리적 요인에 대한 이해가 부족하다[3]. 이러한 원인은 보다 전문성과 경험을 갖춘 소프트웨어 직업인으로서의 훈련과정의 부실에서 찾을 수 있는데 이를 위한 해결 방안의 하나는 컴퓨터를 처음 접하는 초기 교육단계에서부터 소프트웨어에 대한 올바른 이해와 사용법 등을 체계적으로 습득할 수 있도록 교육하는 것이라 본다.

우리나라의 컴퓨터교육은 전자계산기를 다루기 위한 '전산교육'으로 시작하였으나, 1993년 이후 컴퓨터교육이 교육개혁의 중심이 되면서 '교육정보화', 제 7차 교육과정에서는 '정보통신기술 활용교육'으로 표현되기 시작하였다[4].

1983년 이후 정부의 '국가 전산망 사업'과 교육개혁 심의회의의 '초·중등학교 컴퓨터교육을 위한 기초 연구' 결과를 기반으로 직업교육중심에서 보통교육으로 컴퓨터 교육을 수용할 수 있게 되었다. 이 연구는 초·중등학교의 교수·학습방법의 개선, 과학·기술교육의 발전과 미래 정보사회를 대비하기 위한 학교 컴퓨터 교육의 필요성을 정당화하였다. 특히 제 5차, 6차에서 컴퓨터 교육은 더욱 적극적으로 수용되었으며, 컴퓨터 교육의 기회 확대, 컴퓨터를 이용하는 학습 방법이 점차적으로 초등학교에서도 다루도록 확대되었다. 이처럼 예전에는 대학에서의 교육이 컴퓨터 교육의 시작이 되어왔지만, 최근에는 초·중·고에서도 컴퓨터 교육이 이루어짐에 따라 학생들의 예비

지식의 범위가 넓어지고 학생들은 컴퓨터에 관한 기초지식과 기술을 지니고 있어 손쉽게 자신에게 필요한 소프트웨어를 선택하고 구입하여 사용하는 것에 익숙한 반면 이러한 소프트웨어가 만들어지는 과정에 대한 이해가 부족하여 불법복제 등의 비정상적인 사용으로 인한 다양한 피해와 사회적 문제들을 가져오고 있다.

따라서 이 논문의 목적은 현재 대학교육과정에서 실시하고 있는 소프트웨어 공학 교육과정 중 초등학생들에게 적합한 내용을 선정하여 다양한 소프트웨어의 장단점을 알고 자신에게 필요한 소프트웨어를 선택하며 나아가 미래 소프트웨어 개발의 꿈을 키울 수 있도록 교수-학습 모델을 제시하고자 한다.

1.2 연구내용 및 방법

이상에서 제시한 연구의 목적을 달성하기 위하여 본 연구에서는 다음과 같은 내용와 방법을 수행하였다.

첫째, 소프트웨어 공학과 컴퓨터학과의 관계를 살펴봄으로써 소프트웨어 공학의 정의와 그 개념을 이해하도록 하였다. 그리고 소프트웨어 공학 교육이 초등학생들에게 요구되는 이유를 언급하였다.

둘째, 소프트웨어 공학 교육을 위해 제시된 코스의 유형과 프로젝트의 유형에 관한 선행연구 고찰을 통해 초등학생에게 적용 가능한 코스와 프로젝트의 유형을 살펴보았다.

셋째, 선정된 코스와 프로젝트 유형을 가지고 초등학생에게 적합한 수준의 소프트웨어 공학 교육 교수-학습 모형을 제시하였다.

2. 이론적 배경

이 장에서는 소프트웨어 공학의 교육이 초등학생들에게 필요함을 이해하기 위하여 컴퓨터학과 소프트웨어 공학의 차이를 살펴보도록 한다. 먼저 소프트웨어 공학의 정의와 컴퓨터학의 정의를 이해하고, 그 관계를 정의한다. 그리고 초등학생을 대상으로 한 소프트웨어 공학 교육의 필요성을 살펴보았다.

2.1 소프트웨어 공학의 정의

'소프트웨어 공학'이란 용어가 널리 사용되기 시작한 것은 1968년 NATO 협회의 위원인 F.L.Bauer가 다음과 같은 정의를 내린 이후부터이다[5].

"실제 장비에서 신뢰성 있게 운용될 수 있는 경제적인 소프트웨어를 얻기 위한 안전한 공학적 원리의 확립과 사용"

IEEE 용어풀이에서는 소프트웨어 공학을 다음과 같이 정의하였다[6].

"소프트웨어의 개발, 운영, 관리와 폐기에 대한 체계적인 접근"

SEI에서 내린 소프트웨어 공학의 정의를 살펴보면 공학으로서의 의미를 잘 이해할 수 있다[7].

① 공학이란 인류를 위한 실제적인 문제에 대하여 효율적인(cost-effective) 문제해결을 창조하고 구축하는데 있어서 과학적 지식을 체계적으로 응용하는 것이다.

②소프트웨어 공학은 소프트웨어 문제에 대한 효율적인 해답을 얻기 위해서 컴퓨터학의 원리와 수학의 법칙을 응용한 공학의 형식이다.

③소프트웨어에 있어서 '창조와 구축'은 전체적인 소프트웨어의 생명주기를 위해서 유지관리를 포함해야 한다.

④ '비효율적(cost-effective)'이란 돈의 비용뿐만 아니라 시간, 일정, 인간자원까지 포함하는 것이다. 투자되는 자원에 대해 좋은 가치를 얻는 것도 포함된다. 좋은 가치란 적절한 방법에 의해 측정되어 인정된 품질을 일컫는다.

⑤소프트웨어 공학은 컴퓨터학과 수학으로부터만의 응용이 아니다.

⑥현재 '소프트웨어 공학'이라는 용어는 사람들에게 이해가 잘 안 되어 있고 단순 코딩부터 시스템 설계에 대한 관리분야에 이르기까지 의미의 충돌을 가지고 있다. 우리는 소프트웨어 공학이라는 용어가 우리가 수용하지

못하는 의무를 함축하고 있다는 것을 인식한다.

⑦'소프트웨어 공학'이라는 용어는 하나의 서술이라기 보다는 위에서 언급한 정의들을 추구하는 것을 의미한다.

위의 정의들을 종합하여 보면 소프트웨어 공학은 설계 기술, 양질의 관리 수행, 컴퓨터 학과 수학적 형식의 공학 원칙에 대한 이해와 응용을 필요로 한다. 즉 소프트웨어 개발생산성을 향상시키고 개발된 소프트웨어 품질을 보증하여 사용자에게 만족감을 부여하는 방법을 연구하기 위해 전사학적, 경영학적, 경영과학적인 토대 위에서 체계적인 기술과 방법론을 모색하는 종합적인 학문이다[8].

이러한 소프트웨어 공학의 정의에 따르면 소프트웨어 엔지니어는 다른 컴퓨터 관련 직업과 분명한 차이점이 있다. NSF(National Science Foundation)에서는 컴퓨터에 관련된 직업을 다음과 같이 10가지 범주로 분류하였다 :컴퓨터학자, 컴퓨터 하드웨어 엔지니어, 컴퓨터 소프트웨어 엔지니어, 통신전문가, 시스템 프로그래머, 시스템 분석가, 프로그래머, 컴퓨터운영전문가, 기술지원전문가, 컴퓨터지도자.

이들 중 컴퓨터학자, 컴퓨터 프로그래머, 소프트웨어 공학자 각각의 역할을 보면 컴퓨터학자는 컴퓨터 하드웨어나 소프트웨어의 분야에서 이론가, 연구가, 설계자로서 종사하는 지도자 위치의 역할을 담당하는 이를 말한다. 컴퓨터 프로그래머는 전문대 이상의 학위를 가지고, 컴퓨터에 의해 허용된 명령을 사용하여 쓰고, 테스트하고 이들 명령을 응용하는 전문가이다. 이런 직업의 범주에서 소프트웨어 엔지니어는 공학이나 컴퓨터학의 학위를 가지고, 소프트웨어 시스템 전반에 걸친 설계의 지식을 공급할 수 있고, 운영의 특징을 설정하고, 품질표준과 절차테스트, 그리고 사용자의 요구를 분석 정의할 수 있는 고도의 훈련가로 정의한다[13]. 그러므로 소프트웨어 엔지니어는 엄밀한 의미에서 소프트웨어 프로그래머와는 구별되어야 한다. 소프트웨어 엔지니어는 복잡하고 큰 규모의 소프트웨어 시스템의 유도, 정의, 설계, 검증, 수행, 테스트, 문서화, 유지의 기능을 갖추어야 한다. 그리고 소프트웨어 개발에 사용되는 도구와 유지보수에 적절한 재사용 요소(예를 들면 재사용 가능한 모듈)에 대한 사용 능력을 갖추어야 한다. 뿐만 아니라 소프트웨어 공학은 개발에 드는 적절한 비용과 개발기간의 산출, 소프트웨어 시스템의 생명주기를 포함하여 이에 따르는 책임을 져야 한다. 더불어 소프트웨어 엔지니어는 소프트웨어 공학의 도구와 방법들이 여전히 개발 단계에 있고 급격하게 변화함에 따라, 새로 등장하는 방법들을 배우고 사용하는데 기초가 될 이론에 대한 이해력을 지니도록 노력해야 한다.

2.2 소프트웨어 공학의 컴퓨터학과의 관계

앞의 정의에서 알 수 있듯이 컴퓨터학자와 소프트에서 엔지니어는 그 담당하는 역할이 다르다. 그러나 컴퓨터학과 소프트웨어 공학은 가지고 있는 특성이 유사하다. 그리고 컴퓨터학은 소프트웨어 공학 교육의 주요 요소 중 하나이다. 그러나 두 학문의 영역이나 원칙이 어느 한쪽에 종속되어 있지는 않다. 컴퓨터학과 소프트웨어 공학의 차이를 전통적인 과학과 공학의 관계를 들어 비교하면 "과학자는 배우기 위하여 만들고 엔지니어는 만들기 위하여 배운다"고 할 수 있다. 일반적으로 인식하고 있듯이 소프트웨어 공학이 컴퓨터학의 일부 요소는 아니라는 점이다.

소트프웨어 공학은 컴퓨터학에 비하여 경험이 더 강조된다. 그러므로 일반적으로 컴퓨터학은 컴퓨팅 이론을 개발하고 형식화하는 학문으로 보고, 소프트웨어 공학은 컴퓨터학을 기본으로 한 경험적이고 실제적인 학문이라고 본다. 따라서 그 교육에 있어서도 컴퓨터학 이론을 기본으로 하되 공학의 원리와 실제의 경험에 비중을 둔 교육과정이 요구된다[9].

2.3 선행 연구 고찰

태유정(1998)은 연구에서 분석한 소프트웨어 공학 교육과정 모델을 살펴봄으로써 초등

학생에게 어떤 형태의 교육이 적용될 수 있는지를 제시하였다.

교육이 이루어지는 구성과 방법에 따라 다양한 수업의 형태가 있다. 소프트웨어 공학의 교육을 위하여 선택될 수 있는 코스의 유형은 Artifact 모델, 주제접근 모델, 소규모 프로젝트 모델, 대규모 프로젝트 모델, Only 프로젝트 모델이 있는데 코스의 유형별 장·단점은 아래와 같다[9].

<표 1>코스의 유형별 장·단점

유형	과정내용	장점	단점
Artifact	강의,질문	단기간에 많은 양의 교육이 가능. 주제에 깊게 집중.	경험이 없음.
주제접근	강의,주간발표 (토론포함)	주제에 깊게 집중	전 문 과 정에서 적절
소규모 프로젝트	강의, 소규모 프로젝트	소프트웨어 개념을 응용하는 경험을 제공 관리가 용이	큰 규모의 경험이 부족
대규모 프로젝트	강의,대규모 프로젝트	실제와 유사한 경험.	관리 어려움.
프로젝트로만	프로젝트	독립적인 프로젝트 추진 가능.	소 프 트 웨어 공학의 주제 학습이 어렵다.

코스의 유형에 따라 프로젝트가 포함될 수 있다. 소프트웨어 공학의 가장 일반적인 모델에서 프로젝트 과정은 필수적이다. 그러므로 프로젝트에는 어떤 유형들이 있으며, 프로젝트를 과정으로 선택할 때 고려해야 할 사항은 무엇인지를 살펴본다.

프로젝트를 진행하는데 있어서 무엇보다도 중요한 점은 관리이다. 교사는 프로젝트를 수행하기에 앞서 학급 크기, 개발환경의 선택, 강의 계획서의 작성 등을 준비해야 하는데 이들 요소들은 과정의 성공적인 수행을 위해 매우 중요한 요인들이다. 이처럼 교사에게 주어지는 관리의 비중과 프로젝트를 수행하는 규모나 방법에 따라 Toy Project, Mix-and-Match Components, 외부 고객을 위한 프로젝트, 개별적 프로젝트로 나눈다.

<표 2>프로젝트의 유형별 장·단점

유형	주제	장점	단점
Toy프로젝트	모든 팀이 동일한 주제를 개발. 교사로부터 할당.	관리가 용이.	소프트웨어공학의 주요 문제에 대한 습득이 부족
Mix-and-Match	팀별로 구성요소를 개발→통합	대규모 프로젝트와 유사한 경험을 제공. 관리용이, 단위프로그램의 재사용가능	정확한 규약을 정하는데 어려움.
외부고객을 위한 프로젝트	제품의 구성요소를 개발.	고객과의 관계를 학습한다. 실제사용으로 인한 동기부여가 가능.	관리가 어려움
개별적 프로젝트	팀별로 개별주제를 할당	팀간의 독립적인 학습이 가능. 관리가 용이.	관리 어려움.

3. 소프트웨어 공학과 초등 교육

3.1 소프트웨어 공학 교육의 필요성

1968년 10월 NATO의 과학위원회 주최에 의한 소프트웨어 기술자, 전문가들의 회의에서 '소프트웨어 위기'라고 하는 단어가 처음 제창되었다. 이는 소프트웨어에 포함된 잠재적인 에러에 대한 보수, 개발이 끝난 소프트웨어에 대한 보수, 소프트웨어 기술자의 부족, 신규로 개발이 요구되는 소프트웨어를 어떻게 만들 것인가, 소프트웨어 오류가 사회적 문제를 일으키는 것에 어떻게 대응할 것인가에 대한 문제의식에서 시작되었다.

즉, 초창기의 컴퓨터 분야의 주요 관심사는 하드웨어의 획득과 유지 보수였으며 소프트웨어의 중요성은 그다지 중요하게 대두되지 않았으나 컴퓨터 하드웨어의 발달과 더불어 소프트웨어가 거대화, 복잡화에 되었으며 이에 따른 개발비용의 증대로 인하여 막대한 개발비용이 문제가 되었다. 즉, 하드웨어에 들어가는 비용은 절감되는 반면 소프트웨어에 들어가는 비용은 증가하게 되었다. 이것은 비로소 하드웨어와 소프트웨어의 가격 비용을 역전시키는 결과를 가져 왔다. 그리고, 소프트웨어의 보수에 들어가는 공정의 증대와 이에 따른 부수 작업이 많아지면서 개발 요원들도 그 쪽

으로 투입되어야했고 보수 작업에 대한 비용이 증가하게 되어 새로운 소프트웨어에 대한 개발이 지연되게 되었다. 그 결과 개발 요청은 있지만 개발을 할 수 없는 상태가 된다. 이것을 개발적체 라고 한다.

이와 같은 상황은 개발에 투입 될 인력을 보수 작업에 투입하게 되어 개발에 필요한 소프트웨어 기술자 부족으로 소프트웨어 수요에 비해 공급능력을 저하시키게 되었고 이러한 현상은 소프트웨어 공학의 교육 요구를 증대시키게되었다.

또한 Freeman은 다음의 영역을 소프트웨어 엔지니어의 능력으로 완수할 수 있도록 소프트웨어 공학 교육의 요구가 증가한다고 보았다[10].

①복잡하고 큰 규모의 시스템이 증가.
②개발 서비스에 대한 소비자의 요구가 증가함에 따라 이에 부응하는 소프트웨어 패키지 산업이 생김.
③자동화, 특히 인공지능에 대한 인식이 증가하면서 소프트웨어 엔지니어가 중요해짐.
④개발 과정에 관심을 가지게 됨.
⑤설계활동에 대한 주의 부족.
⑥많은 산업체에서의 소프트웨어 엔지니어를 위한 업무 훈련의 실패.

즉, 새로운 양질의 소프트웨어를 요구하는 소비자의 수요와 이미 개발된 소프트웨어를 관리하면서 대두되는 문제로 인하여 단순히 소프트웨어 개발자가 아닌 소프트웨어 엔지니어의 양성이 요구되어진 것이다.

3.2 초등학교 소프트웨어 공학 교육의 필요성

우리나라 대학교육과정에서 이러한 소프트웨어 엔지니어 양성의 중요성을 인식하고 전문적인 교육을 시행하고 있으나 보다 체계적인 교육을 위하여 초등학교 단계에서부터 소프트웨어 공학적인 교육내용이 필요하다고 본다. 이는 소프트웨어 엔지니어를 양성한다는 직업적인 목표보다는 소프트웨어 공학에 대한 바른 이해를 통해 일상생활에서 사용하고 있는 소프트웨어의 중요성을 알고 올바른 사용법을 배우며 더 나아가 미래사회의 유능한 소프트웨어 엔지니어의 꿈을 키워갈 수 있도록 할 수 있기 때문이다.

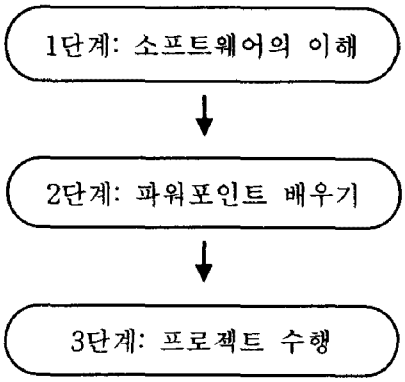
4.초등 소프트웨어 공학교육 학습모형

앞에서 살펴본 바와 같이 소프트웨어 공학 교육을 위한 다양한 코스와 이러한 코스를 효과적으로 교육하기 위한 프로젝트는 따로 분리되어 교육되기보다는 상호 보완적이고 유기적인 관계로 구조화되어야만 가장 효과적이라 본다. 이번 장에서는 초등단계에 적합한 소프트웨어 공학 교육의 코스와 프로젝트 과정을 제시하고 초등학생에게 적합한 교수-학습 모형을 설계하고자 한다.

4.1 초등 소프트웨어 공학 교육의 코스

초등학생들에게 가장 쉽고 재미있는 소프트웨어 공학 교육이 이루어지기 위해서는 소프트웨어 공학의 복잡한 이론에 대한 강의식 접근보다는 초등학생들이 할 수 있는 간단한 소규모의 프로젝트를 진행하면서 간접적으로 소프트웨어의 공학적인 개념을 교육하고, 관리가 용이하며 일반적으로 교사가 미리 선정한 주제를 가지고 소그룹의 팀이 협동하며 진행하는 방법이 적합하다.

이러한 소규모의 프로젝트를 진행하기 위해서 먼저 선행되어야 할 학습은 소프트웨어에 대한 기본적인 개념과 소프트웨어의 종류, 다양한 소프트웨어의 장단점을 비교하는 것과 이러한 소프트웨어가 우리에게 가져다 주는 이로움과 해로움을 알아야 한다. 또한 소프트웨어를 만들기 위한 기본적인 프로그래밍 능력이 필요한데 초등학생이 구현하기 쉬운 파워포인트를 학습하면 쉽고 재미있게 프로그래밍을 체험할 수 있을 것이다. 위와 같은 일련의 초등 소프트웨어 공학 교육의 코스를 도식화하면 다음과 같다.



<그림1> 초등 소프트웨어 공학 교육의 코스

4.2 초등 소프트웨어 공학 교육 프로젝트모델

초등 소프트웨어 공학 교육을 위한 프로젝트 모델은 가장 널리 이용되는 일명 폭포수 모형(water fall model)이라는 단계적 모형(phased life-cycle model)을 적용했다.

<표 3>단계적 모형의 절차

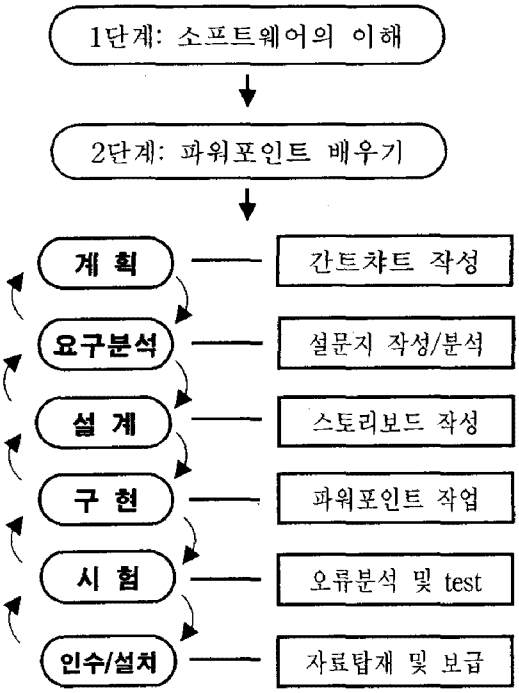
단계	내 용
계획	문제 분석, 시스템의 성격 파악을 통한 비용과 기간 예측
요구분석	기능, 용이성, 이식성 등의 목표 시스템의 품질 파악
설계	분석된 결과를 프로그램으로 구성하기 위한 설계 단계
구현	프로그램밍 언어를 사용하여 설계를 실체화시키는 단계
시험	구현된 결과물이 제대로 개발되었는지를 테스트하는 단계
인수/설치	최종 결과물을 발주자에게 넘기는 단계

이 모형은 1950년대에 항공 방위 소프트웨어 개발 경험으로 소개된 모형으로 응용분야가 단순하거나 잘 알려진 경우에 적합하다. 각 단계는 다음과 같다.

<그림2> 단계적 모형의 흐름

4.3 초등 소프트웨어 공학 교수-학습 모형

위와 같은 단계적 모형을 적용한 초등 소프트웨어 공학 교육을 위한 교수-학습 모형은 앞에서 제시한 <그림1>의 1, 2단계에 <그림2>의 프로젝트 수행 모델을 결합한 형태로 다음과 같다.



<그림3> 초등 소프트웨어 공학 교수-학습 모형

계획 단계에서의 활동으로는 간트챠트를 작성해 볼 수 있는데 이를 통해 전반적인 프로젝트의 흐름과 진행도를 점검하고 진행하는 방법을 배운다. 요구분석 단계에서는 계획단계

에서 결정된 프로젝트에 대한 타당성과 요구사항을 수렴하는 설문조사활동의 단계로 이를 통하여 개발될 소프트웨어의 완성도를 높일 수 있다. 설계 단계에서는 만들어질 프로그램의 화면구성과 흐름 등을 스토리보딩 작업으로 구조화하는 단계이다. 구현단계에서는 파워포인트를 활용하여 실제적으로 개발하고자 하는 소프트웨어를 만들어 가는 단계이다. 초등학생들에게 어려운 언어를 통한 코딩보다는 쉬운 저작툴을 이용하여 쉽고 간단하게 원하는 프로그램을 만들어 보게 한다. 예를 들어 마이크로소프트사의 엑셀을 이용하여 가계부를 만들어 보거나 파워포인트를 활용하여 지역문화유적을 소개하는 프로그램을 만들어 볼 수 있을 것이다.

시험(test)단계에서는 프로그램의 오류를 찾아 수정하고 발전시키는 단계로 모둠별, 학급별로 시연회를 열고 개발된 프로그램의 장점과 단점을 분석하고 좀 더 보완하고 개선할 점 등을 찾아볼 수 있다. 이러한 시험단계를 거쳐 개선된 프로그램을 많은 사람들이 사용할 수 있도록 학급, 학교홈페이지에 탑제하거나 cd 자료로 보급하는 단계를 통해 일반화할 수 있을 것이다.

아울러 학년 단계에 맞는 교수학습활동의 체계적인 지도를 위해 학습목표, 학년별교육내용, 평가방안을 다음과 같이 제시하였다.

1)학습목표

앞에 모형을 교육하기 위한 교육과정을 통하여 학생들에게 요구되는 학습목표를 다음과 같이 제시하였다.

첫째, 소프트웨어의 개념과 종류를 알고 다양한 소프트웨어를 올바르게 사용할 수 있는 능력을 기르도록 한다.

둘째, 다양한 종류의 소프트웨어를 사용해 보고 이를 실생활에 활용할 수 있는 응용력을 갖추도록 한다.

셋째, 소프트웨어 개발 프로젝트에 참여하고 이를 통해 소프트웨어 공학의 이해와 미래사회가 요구하는 전문인이 되고자 하는 의지를 심어준다.

2) 학년별 교육내용

초등소프트웨어 공학교육 내용은 4, 5, 6학년을 대상으로 하여 4학년 과정에서는 1단계로 소프트웨어의 기본적인 이해를 목표로 소프트웨어의 개념, 소프트웨어의 종류와 활용방법을 배우고 소프트웨어가 우리에게 주는 이로움과 해로움 찾아보고 올바른 방법으로 자신에게 필요한 소프트웨어를 선택하고 활용할 수 있도록 하는 단계이다. 특히 이 단계를 통해 불법복제와 바이러스, 해킹등의 정보통신윤리교육을 병행하여 지도할 수 있으며 2단계 5학년 과정에서는 파워포인트의 기능과 엑셀등을 습득하는 단계이다. 1학기는 엑셀을 2학기는 파워포인트의 기능을 익힐 수 있다. 마지막 3단계 6학년 과정에서는 앞에서 배운 지식과 기능을 활용하여 소규모 프로젝트를 진행하는 단계로 실제 소프트웨어 개발단계를 진행하도록 구성하였다. 예를 들어 1학기는 엑셀을 활용하여 가계부나 금전출납부를 만들어 보고 2학기에는 파워포인트를 활용하여 우리고장 문화유적등을 안내하는 응용 소프트웨어 프로그램을 만드는 과정이다.

<표 4>교육과정 구조

단계	학년	내 용
1단계	4학년	-소프트웨어의 개념 -소프트웨어의 종류와 활용 -소프트웨어의 명암 -올바른 소프트웨어의 선택 -불법복제의 피해
2단계	5학년	-엑셀기능 익히기 (금전출납부 만들기) -파워포인트의 기능 습득
3단계	6학년	-파워포인트를 활용한 학습용 소프트웨어 개발 (모둠 구성 및 역할분담) (예)경기도의 문화유산, 내 고장 관광명소 소개등... -개발된 자료 홈페이지에 탑재하기 및 cd로 제작하기

3) 평가

각 단계별로 다양한 평가 방법을 적용할 수 있는데 1단계에서는 소프트웨어의 이론적 접

근에 대한 지적평가를 실시할 수 있으며, 2단계에서는 개별적인 기능숙달 정도를 평가할 수 있다. 또한 3단계에서는 팀별로 맡은 역할의 수행 정도, 협력도 등의 태도와 개발된 자료의 완성도를 측정할 수 있다.

5. 결론 및 향후과제

21세기 지식정보화 사회는 수많은 정보 중에서 자신에게 필요한 정보를 얼마나 정확하고 올바른 방법으로 빠르게 습득할 수 있는가의 여부가 중요시되며 소프트웨어는 이러한 다양한 정보를 효율적으로 가공하고 재생할 수 있도록 돕는 역할을 하고 있다.

소프트웨어를 사용하는 사용자의 기술적, 환경적, 기능적 요구가 변화하며 이러한 변화를 수용하기 위하여 지속적인 유지와 보수가 필요하게 되었고 소프트웨어 시스템의 적용범위도 넓어져 사회적 역할이 증대되었다. 이는 소프트웨어가 사회적인 문제로까지 발전할 가능성이 높아진다는 것이다.

이러한 사회적 문제를 해결하기 위해 많은 대학에서 소프트웨어 공학교육을 실시하고 전문가를 양성하고 있으나 보다 체계적인 교육이 이루어지기 위해서는 초등교육 단계에서부터 이러한 소프트웨어 공학교육을 실시하여 올바른 소프트웨어의 선택법과 사용법을 습득하는 단계를 포함한 소프트웨어 공학의 기초교육을 실시할 필요성이 있는 것이다.

이처럼 본 연구를 통해 개발된 초등 소프트웨어 공학교육 학습모델을 적용할 경우 얻을 수 있는 효과는 다음과 같다.

첫째, 소프트웨어의 개념을 알고 자신에게 필요한 소프트웨어를 바르게 선택하고 사용할 수 있다.

둘째, 소프트웨어가 가져다 주는 이로움과 해로움을 알고 더욱 발전된 소프트웨어를 개발하고자 하는 의지를 키울 수 있다.

셋째, 초등수준의 간단한 프로젝트를 진행하며 소프트웨어 공학적 원리를 체험할 수 있다.

넷째, 프로젝트를 진행하며 서로 돕고 협력하여 사회성이 향상될 것이다.

향후 연구과제로 본 연구를 더욱 발전시키고 그 효과성을 알아보기 위해 학습모형의 각 단계별 교수학습 과정을 개발해야 할 것이며 또한 학습자를 대상으로 개발된 교수학습과정을 적용하고 일반집단과의 비교를 통해 그 효과성을 검증해야 할 것이다.

6. 참고문헌

[1] Norman E.Gibbs, Richard E. Fairley(1987). Software Engineering Education The Educational Needs of the Software Community. Springer-Verlag.

[2] 왕창종, "바람직한 컴퓨터교육", 정보과학회지, 제 14권, 제 1호, pp.72-76, 1996.

[3] Rechard Fairley(1985). Software Engineering Concept. McGRAW-HILL.

[4] 이옥화 외(2000). 『컴퓨터교육의 이해』. 서울:영진닷컴, p.35.

[5] Bauer, F. L(1972). Software Engineering. Information Processing.

[6] IEEE(1983). Standard Glossary of Software Engineering Terminology. ANSI/IEEE Std.

[7] Gary Ford(1990). 1990 SEI Report on Undergraduate Software Engineering Education. Software Engineering Institute, Carnegie Mellon University.

[8] 이주헌(1993). 『실용 소프트웨어 공학론』. 서울 : 법영사, p. 21-101, p.494-527.

[9] 태유정, "대학에서의 소프트웨어 공학 교육과정 설계", 이화여자대학교 석사논문, pp.13-22, 1997.

[10] Gary Ford (1991) SEI Report on Graduate Software Engineering Education. Software Engineering Institute, Carnegie Mellon University.