7

소프트웨어공학 교육을 위한 프로세스 모형화

충북대학교 ▌홍 장 의*
한국과학기술원 ▌배 두 환*

1. 서 론

소프트웨어공학 기술이 단순히 사용자 서비스를 충족시킬 수 있는 소프트웨어를 개발할 수 있느냐의 문제를 떠나, 이제는 품질의 향상을 통한 비용 절감과 비즈니스 전략을 수립하기 위한 필수 요소 기술로 자리 잡고 있다. 따라서 소프트웨어를 기반으로 하는 모든 사회적 활동에는 소프트웨어 공학 기술의 적용이 필요하게 되었고, 이들의 체계적인 적용이 사회적 경쟁력 및 이익 창출의 결과로 연결되고 있다.

소프트웨어공학이라 함은 소프트웨어 수명주기의 전과정에서 적용하는 체계적, 정량적, 그리고 원리화된 공학적 접근 방법들의 총체적인 집합 또는 이들을 다루는 학문을 의미하며[1,2], 소프트웨어 엔지니어가 수행하는 모든 활동들을 지칭한다. 소프트웨어공학은 고전적인 의미에서 사용자가 원하는 소프트웨어를 개발하기 위한 분석, 설계, 코딩, 설치 및 운영 등의 분야에서 핵심적인 역할을 수행해 왔지만 이제는 서비스 통합 및 전사적 소프트웨어 전략 수립이라는 큰 영역으로까지 그 의미가 확대되고 있다.

이와 같은 사회적, 기술적 측면에서 소프트웨어공학을 바라볼 때, 올바른 소프트웨어 엔지니어를 양성하는 일은 매우 중요한 일이 아닐 수 없다. 소프트웨어공학의 인재 양성을 위한 교육 차원에서 보면, 단순히 공학적인 기본 이론을 학습하는 과정이외에도 이러한 기술을 실무에 어떻게 적용해야 하는가에 대한 응용 기술의 습득이 무엇보다도 중요하다. 기존에는 소프트웨어를 개발해 본 모든 사람들이 소프트웨어공학 활동을 수행하는 것이며, 따라서 누구든지 소프트웨어를 개발해 보았다면 소프트웨어 엔지니어가 될 수 있다고 여겨왔다. 그러나 과거의 다양한 소프트웨어 개발 프로젝트의 실패 사례들에서도 보여준 것처럼 전문적인 소프트웨어 공학 기술의 적용 없이는 우리가 기대하는 높은 품질의 소프트웨어 개발이 불가능하며, 또한 소프트웨어를 비즈니스의 기회로 삼을 수 없게 된다.

따라서 소프트웨어공학은 소프트웨어 기반의 지식사회에서 무엇보다 중요한 기반 학문으로 자리 잡고 있으며, 이에 대한 체계적인 교육을 통해 우수한 소프트웨어 엔지니어를 양성하는 것이 무엇보다 중요한 시점이 되었다. 최근 3년전 IEEE와 ACM에서는 소프트웨어공학 교육에 대한 교과과정 편성 및 교육 자료에 대한 가이드라인[3]을 제시하였다. 또한 IEEE에서는 SWEBOK[4]을 통해 소프트웨어공학 분야의 전문적인 지식 영역이 무엇인가를 정의하였다. 이러한 연구들은 소프트웨어공학에 대한 교육의 접근 방법에 대한 구체적인 정보를 제공해주고 있으며, 많은 학자들이 이들의 유용성을 확인하고 있다.

본 고에서는 이와 같은 연구들에서 제시한 교육의 지식 영역을 정의하기보다는 전문적인 소프트웨어 엔지니어를 양성하기 위해 소프트웨어공학 분야의 교육이 어떠한 접근 방법으로 이루어져야 하는 가에 대하여 살펴보았다. 이는 체계적인 교육이 이루어질 수 있도록 하기 위한 교육 프로세스의 모형화를 통하여 제시되었다.

본 고의 구성은 다음과 같다. 2장에서는 배경지식으로 기존에 연구되어 왔던 소프트웨어공학 교육의 가이드라인들을 살펴보고, 3장에서는 소프트웨어공학의 세부 교육 목표에 대한 정의와 함께 이러한 목표를 달성하기 위해 요구되는 교육 품질 요소들을 제시한다. 4장에서는 3장에서 제시한 교육 목표에 견주어 어떻게 교육이 전개되어야 하는가를 표현하는 학습 모형을 제시하고, 5장에서는 모형을 지원하기 위한 교육 도구들에 대하여 설명한다. 6장에서는 소프트웨어공학 교육에 대한 성과 분석 사례를 제시하고, 그리고 마지막으로 7장에서 결론을 맺는다.

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학IT연구센터 지원사업의 연구결과로 수행되었음(IITA-2007-(C1090-0701-0032)).
* 종신회원

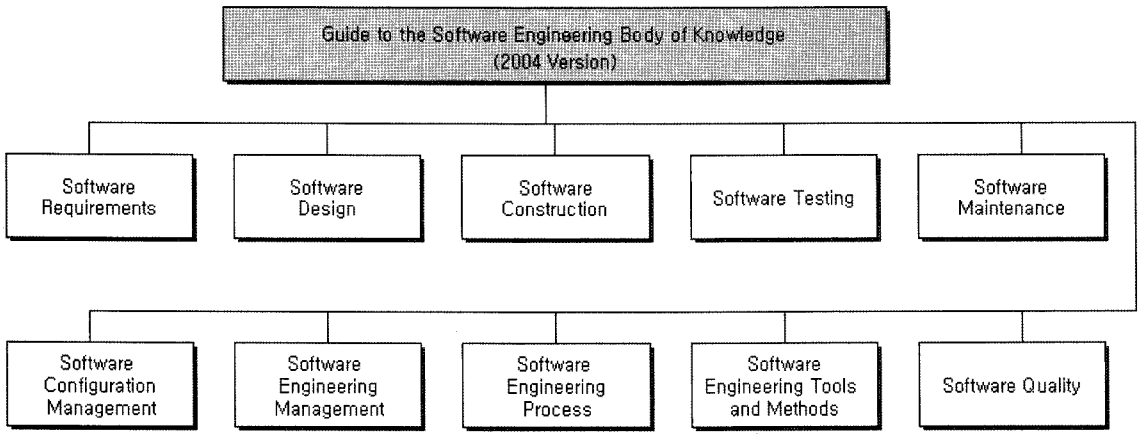


그림 1 SWEBOK(ver. 2004)의 10개 지식영역[4]

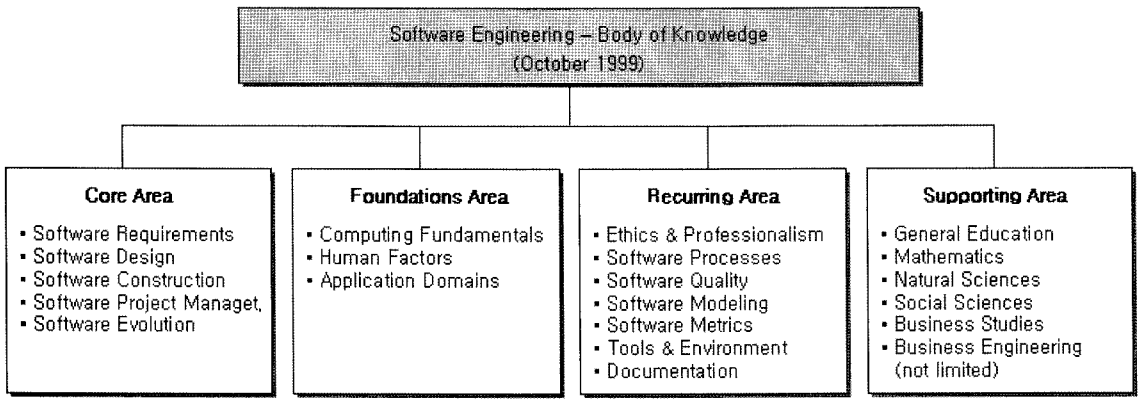


그림 2 CMU/SEI의 SE-BOK의 4개 지식영역[5]

2. 배경 지식

소프트웨어공학은 1960년대 소프트웨어 위기(software crisis)가 대두되면서 그 중요성이 강조되기 시작하였으며, 최근 5년 동안에는 소프트웨어공학의 올바른 교육이 소프트웨어 위기에 대한 대처방안으로 여겨지고 있다.

2.1 SWEBOK

SWEBOK(SoftWare Engineering Body of Knowledge)은 1990년대 중반부터 IEEE과 ACM에서 결성한 컴퓨터분야의 교육위원회가 소프트웨어공학 분야에서 필요한 지식의 실체를 정의하고, 이들 간의 관계성을 분석한 연구 결과로 작성되었다[4]. 2001년에 Trial version으로 공개한 SWEBOK은 현재 2004년 버전을 제공하고 있으며, 소프트웨어공학 분야의 핵심적인 지식 및 기술에 대하여 10개의 지식영역(KA, Knowledge Area)으로 구분하고, 각 지식영역에 대하여 세부 주제들을 정의하고 있다. SWEBOK의 특징은 광범위하게 고려될 수 있는 소프트웨어공학의 학문적 주제와 일반 컴퓨터공학 관련 주제 간의 명확한 구분을 위하여 전산학(computer science), 수학(mathematics), 프로젝트 관리(project management) 등을 소프트웨어공학 영역과 별도로 구분하고 있다는 점이다. 그림 1은 SWEBOK이 제시하고 있는 10개의 지식영역을 나타난 것이다. 이러한 10개의 지식영역이외에 관련성있는 8개의 학문영역 - 컴퓨터공학, 컴퓨터과학, 관리[1], 수학, 프로젝트관리[2], 품질관리, 소프트웨어 인간공학(ergonomics), 그리고 시스템공학 - 을 제시하고 있다.

2.2 CMU/SEI SE-BOK

CMU/SEI는 1999년에 소프트웨어공학 분야의 체계적인 교육과정 개발을 위한 가이드라인으로 소프트웨어공학 지식의 실체(SE-BOK, Software Engineering-Body of Knowledge)를 정의하고 이들을 교육하기 위한 교과 과정을 제시하였다[5]. 이 가이드라인에서는 소프트웨어공학의 지식 주체를 크게 4개의 지식영역으로 구분하고, 각 영역별로 지식요소(Knowledge Component)

1) 관리(management): 재무관리, 판매관리, 운영관리, 인적관리 등의 비즈니스 경영분야

2) 프로젝트관리(project management): 소프트웨어 프로젝트 관리와는 구별되는 개념으로 비용관리, 프로젝트 통합관리, 일정관리, 리스크관리 등의 영역을 의미하며, PMBOK(IEEE Std 1490-2003)에 정의된 주제분야

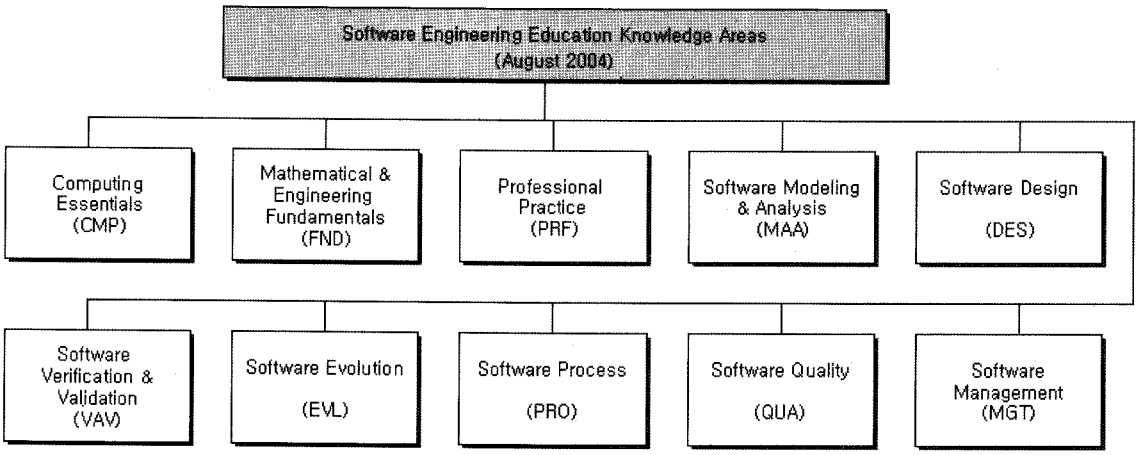


그림 3 SE2004의 10개 지식영역[3]

를 정의하였다. 또한 ABET(Accreditation Board for Engineering Technology)에서 요구하는 공학교육의 인증 수준을 충족할 수 있도록 소프트웨어공학 교육에 대한 학부 과정의 교과 이수모형을 제시하고 있다. CMU/SEI가 정의한 지식영역은 그림 2에서 보여주는 것과 같이 핵심영역(core area), 기초영역(foundation area),

표 1 SE2004의 지식영역(KA)과 지식단위(KU)[3]

KA/KU	Title	hrs	KA/KU	Title	hrs
CMP	Computing Essentials	172	VAV	Software V & V	42
CMP.cf	Computer Science foundations	140	VAV.fnd	V&V terminology and foundations	5
CMP.ct	Construction technologies	20	VAV.rev	Reviews	6
CMP.tl	Construction tools	4	VAV.tst	Testing	21
CMP.fm	Formal construction methods	8	VAV.hct	Human computer UI testing and evaluation	6
			VAV.par	Problem analysis and reporting	4
FND	Mathematical & Engineering Fundamentals	89	EVL	Software Evolution	10
FND.mf	Mathematical foundations	56	EVO.pro	Evolution processes	6
FND.ef	Engineering foundations for software	23	EVO.ac	Evolution activities	4
FND.ec	Engineering economics for software	10			
PRF	Professional Practice	35	PRO	Software Process	13
PRF.psy	Group dynamics / psychology	5	PRO.con	Process concepts	3
PRF.com	Communications skills (specific to SE)	10	PRO.imp	Process implementation	10
PRF.pr	Professionalism	20			
MAA	Software Modeling & Analysis	53	QUA	Software Quality	16
MAA.md	Modeling foundations	19	QUA.cc	Software quality concepts and culture	2
MAA.tm	Types of models	12	QUA.std	Software quality standards	2
MAA.af	Analysis fundamentals	6	QUA.pro	Software quality processes	4
MAA.rfd	Requirements fundamentals	3	QUA.pca	Process assurance	4
MAA.er	Eliciting requirements	4	QUA.pda	Product assurance	4
MAA.rsd	Requirements specification & Doc.	6			
MAA.rv	Requirements validation	3			
DES	Software Design	45	MGT	Software Management	19
DES.con	Design concepts	3	MGT.con	Management concepts	2
DES.str	Design strategies	6	MGT.pp	Project planning	6
DES.ar	Architectural design	9	MGT.per	Project personnel and organization	2
DES.hci	Human computer interface design	12	MGT.ctl	Project control	4
DES.dd	Detailed design	12	MGT.cm	Software configuration management	5
DES.ste	Design support tools and evaluation	3			

순환영역(recurring area), 그리고 지원영역(supporting area)으로 구분하고 있으며 각 영역에 대하여 지식요소들은 정의하였다.

2.3 Software Engineering 2004

Software Engineering 2004(SE2004)는 소프트웨어 공학의 대학(학부) 교육에 있어서 교과과정(curriculum) 편성을 위한 가이드라인을 제공하는데 목적이 있다[3]. IEEE와 ACM에 의해 공동 개발된 SE2004 문서는 소프트웨어공학을 공부하는 학습자가 알아야 하는 지식과 이들 지식을 습득하기 위해 가르쳐야 하는 교과과정을 제시하고 있으며, 이러한 지식의 도출을 위하여 SWEBOK에서 정의된 지식 영역을 참고하고 있다. SE2004 문서에서는 소프트웨어공학과 관련된 지식을 계층적으로 지식영역(Knowledge area), 지식단위(Knowledge Unit), 그리고 주제(topic)로 구분하였으며, 그림 3에서와 같이 10개의 지식영역을 정의하였다.

그림 3의 10개 지식영역에 대한 지식단위 내역은 표 1에 나타내었다. 표 1에서는 지식영역별로 식별된 지식 단위와 각 지식단위별로 이루어져야 하는 교육의 최소 권장 시간을 제안하고 있다.

2.4 기타 연구들

소프트웨어공학에 대한 바람직한 교육을 위한 많은 연구들이 진행되고 있으며, 또한 이러한 연구 결과를 바탕으로 다양한 교육 프로그램들이 진행되고 있다. 1987년에 시작된 CSEE&T(Conference on Software Engineering Education and Training) 학회는 2007년 현재 20회를 맞이하면서 소프트웨어공학 분야의 교육 방법과 내용에 대한 다양한 연구 논문을 발표하고 있다[6,7]. 또한 Boehm의 다양한 이론과 방법들이 소프트웨어공학의 교육 및 훈련 과정에 미친 영향을 분석하는 연구들[8,9]도 진행된 바 있다.

국내에서도 소프트웨어공학에 대한 매우 다양한 교육이 진행되고 있는데, 이론과 응용 능력을 배양하는 대학의 소프트웨어공학 교육을 비롯하여 소프트웨어 관련 협회 및 산업체의 전문화된 교육 프로그램들이 운영되고 있다. 특히 한국과학기술원에서는 산업체 요원들에 대한 소프트웨어공학의 심화 교육을 위해 소프트웨어 전문가과정을 2004학년도부터 학제전공으로 운영해오고 있으며[10], ICU도 CMU/SEI와의 협약을 통해 소프트웨어공학 전문가 양성 프로그램을 운영하고 있다[11].

3. 교육 목표 및 교육 품질 요소

3.1 소프트웨어공학 교육 목표

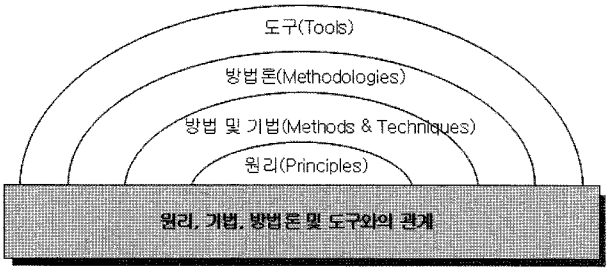


그림 4 소프트웨어 공학의 지식 유형간의 관계[2]

소프트웨어공학 교육의 목표는 "창의적이고 전문적인 소프트웨어 엔지니어의 양성"일 것이며, 이 목표를 달성하기 위해서는 보다 구체적인 세부 교육 목표가 정의되어야 할 필요가 있다. 본 고에서는 소프트웨어공학 교육의 세부 목표를 정의하기 위하여 Ghezzi가 제시한 소프트웨어공학의 지식 유형간의 관계[2]에 근간을 두었다. 그림 4에서 보는 것처럼 소프트웨어공학의 지식유형은 원리, 방법 및 기법, 방법론 그리고 도구로 이루어져 있다.

소프트웨어공학은 성공적인 소프트웨어 개발을 위한 공학적 이슈를 다루는 학문이며, 이러한 이슈들은 제품(product)과 절차(process)의 두 가지 측면을 모두 고려해야 한다[1,12]. 올바른 절차에 따라 양질의 소프트웨어 제품을 개발하기 위해서는 적합한 방법 및 기법이 소프트웨어 개발 과정에서 적용되어야 하며, 이들은 문제 해결을 위한 접근방법으로 간주되는 방법론 차원에서 통합되어야 한다. 이와 같은 개념에 근거하여 그림 4를 기반으로 정의할 수 있는 소프트웨어공학 교육의 세부 목표는 각각의 지식 유형에 따라 다음과 같다.

- 목표 1 : 소프트웨어공학의 기본 개념 및 원리를 이해한다.
- 목표 2 : 소프트웨어 개발을 위한 방법 및 기법을 숙지한다.
- 목표 3 : 문제를 해결하기 위한 방법론을 익히고 적용한다.
- 목표 4 : 소프트웨어공학 관련 다양한 도구들을 활용한다.

정의된 세부 교육 목표를 달성하기 위해서는 각 목표별로 어떠한 내용들을 어떻게 교육할 것인가에 대한 프로세스가 정립되어야 하며, 그러한 교육 프로세스 상에서 달성되어야 할 교육 품질 요소들이 고려되어야 한다.

3.2 교육 품질 요소

소프트웨어공학 교육의 목표 달성을 위한 교육 프

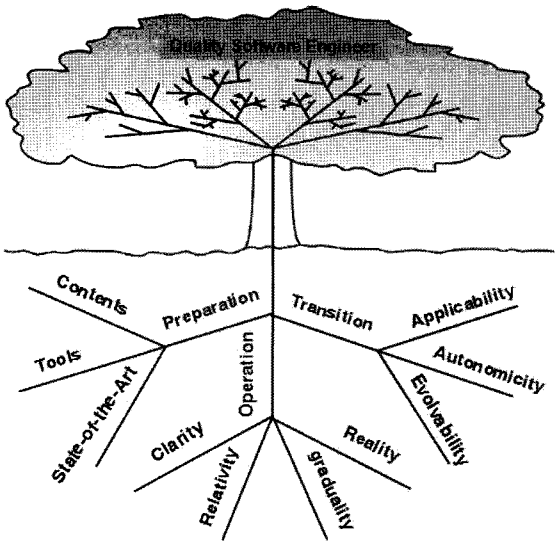


그림 5 소프트웨어공학 교육의 품질 요소 트리

로세스 측면의 접근 방법에 대하여 논의하기에 앞서, 교육 과정에서 달성해야 하는 주요한 품질 요소들을 정의할 수 있다. 이러한 품질요소는 교육의 세부목표 달성을 위해 공통적으로 성취해야 하는 요소들로써, 소프트웨어공학의 지식영역에 대한 교육의 효과 및 달성도를 나타내는 척도가 될 수 있다. 소프트웨어공학 교육의 품질 모델을 MaCall의 Quality Factor Tree Model[13]에 의해 표현할 때, 교육 품질 요소(factor)들은 교육 준비(preparation) 요소, 운영(operation) 요소, 그리고 전환(transition) 요소로 구분할 수 있다. 그림 5는 MaCall의 Factor Model Tree에 의해 표현된 교육 품질 요소들을 정의한 것이다.

그림 5에서 정의하고 있는 품질 요소들에 대하여 구체적으로 설명하면 다음과 같다.

3.2.1 교육 준비(Preparation) 요소

이 분야의 품질요소는 소프트웨어공학 교육을 위한 준비요소가 교육의 품질요소를 결정할 수 있다는 관점에서 제시한다.

(1) 컨텐츠(Contents): 교육 컨텐츠는 소프트웨어공학 교육을 위한 주제를 무엇을 할 것인가에 대한 품질 요소를 의미한다. 이러한 컨텐츠는 교육 시작전에 강의 계획서에 기술된다. 강의 계획서에 포함된 교육 컨텐츠는 학습자들의 수준과 교육 환경에 따라 정의되어야 하며, 이들의 교육 및 학습이 얼마나 효과적으로 이루어질 수 있는가가 교육 품질을 결정하게 된다. 특히 주어진 일정에 따라 교육을 수행해야 하기 때문에 합리적인 컨텐츠의 범위 및 내용 결정이 중요하다.

(2) 도구(Tools): 소프트웨어공학에서 다루고자 하는 대부분의 주제들은 CASE 도구에 의해 구현 가능하다. 따라서 교육을 통해 소프트웨어공학의 주제들을 보다 효과적이고 능률적으로 전달하기 위해서는 어떠한 도구를 어떤 주제에 사용할 것인지가 결정되어야 한다. 또한 CASE 도구이외의 양식지(form)와 적용 표준도 정의해야 한다. 도구를 통한 교육은 학습의 효과 및 학습 생산성을 향상시키는 품질 요소이다.

(3) 최신성(State-of-the-Art): 기술의 발전에 따라 소프트웨어공학의 주제들도 많이 변화하고 있다. 변화하는 주제들을 소프트웨어공학의 각 지식영역에 반영하여 최신의 내용을 기반으로 학습이 이루어지도록 해야 한다. 그러나 소프트웨어공학 교육의 딜레마는 바탕이 되는 소프트웨어공학의 고전적인 기본 지식과 최신의 공학적 이론들을 제한된 일정 내에 어떻게 전달하는 데 있다. 따라서 학습자의 수준과 사회적 요구에 따른 적정한 범위 및 수준을 결정하여 교육할 때, 소프트웨어공학 교육의 의미가 있으며, 교육의 품질도 높일 수 있다.

3.2.2 교육 운영(Operation) 요소

소프트웨어공학에 대한 교육의 수행 과정에서 교육의 품질을 향상시키기 위한 요소들이 무엇인가를 정의한다.

(4) 명확성(Clarity): 소프트웨어공학에서 정의하고 있는 다양한 용어 및 개념에 대한 명확한 이해가 선행되어야 한다. 예를 들어 "품질이란 무엇인가 ?"라는 질문에 학습자의 명확한 대답이 가능해야 한다. 기본 개념에 대한 정확한 정의를 알고 있다는 것은 실체 파악이 명확하게 이루어졌다는 차원에서 매우 중요한 교육 품질의 결정 요소이다.

(5) 관련성(Relativity): 개념에 대한 정확한 숙지를 기반으로 개념간의 상호 연관성을 명확히 이해해야 한다. 예를 들면, 모듈에 대한 응집력과 결합력은 상호 어떠한 관계가 있는지를 이해할 수 있어야 한다. 또한 개념간의 관련성이 서로 다른 측면을 나타내는 수평 관계인지, 추상화와 상세화를 나타내는 수직 관계인지에 대하여 숙지해야 한다. 이러한 개념간의 관련성에 대한 이해가 없다면 교육의 진행이 어렵고, 또한 학습자로부터 기대할 수 있는 교육성과도 달성할 수 없다.

(6) 점진성(Graduality): 소프트웨어공학 교육의 대상이 되는 모든 세부 주제들은 점진적이며 단계적인 학습이 이루어져야 한다. 소프트웨어공학의 바탕이 되는 원리는 무엇이며, 이것이 왜 중요한지, 그리고 이들이 어떻게 소프트웨어 개발 과정에서 적용되는지를 이

해하는 것이 중요하다. 점진적이고 단계적인 교육이 진행될 때, 소프트웨어공학 분야의 심화된 주제에 대하여 학습이 가능하게 된다.

(7) 실증성(Reality): 소프트웨어공학을 처음 접하는 학습자들이 느끼는 사항중의 하나는 소프트웨어공학이 대체적으로 추상적인 개념을 다루고 있다는 것이다. 따라서 소프트웨어공학 분야의 지식들이 실무에서는 어떻게 쓰이고 있는지 실증적인 예시가 필요하다. 구체적으로 산업체 현장에서 어떻게 적용되고 있는지에 대한 실례를 들어 설명하는 것은 학습 효과 및 이해도를 높이는 요소라 할 수 있다.

3.2.3 교육 전환(Transition) 요소

이 품질 요소는 소프트웨어공학에 대한 원리, 방법 및 기법, 방법론 등의 학습을 수행한 후, 이들을 실무에 적용하기 위해 요구되는 능력에 대한 것이다.

(8) 적용성(Applicability) : 소프트웨어공학의 다양한 방법 및 기법들을 소프트웨어 개발 과정에서 적용할 수 있도록 하는 교육이 이루어져야 한다. 이론 및 현상에 대한 교육이 비록 충실하게 이루어졌다 할지라도 현장에서의 업무에 적용할 수 없다면 교육의 의미는 상실된다. 따라서 소프트웨어공학의 주제들이 산업체 현장에서 잘 적용될 수 있도록 하는 것이 교육 성과와 직결된다.

(9) 자율성(Autonomicity): 소프트웨어공학의 세부 주제들에 대한 이론적 배경을 근간으로 하는 실습 과정이 반드시 수반되어야 하며, 실습을 수행하는 과정에서 학습자의 자율적인 참여하에 스스로 주어진 문제를 풀어갈 수 있도록 가이드해야 한다. 사전에 학습한 방법 및 기법들을 스스로 실습과정에 적용해봄으로써, 학습자 개인의 기술적 능력뿐만 아니라 프로젝트 팀원으로써의 임무 수행 능력을 향상시킬 수 있다.

(10) 진화성(Evolvability): 소프트웨어공학에 대한 교육을 이수한 후에, 새롭게 출현하는 공학적 이론을 습득하고 실무에 적용하기 위해서, 또는 소프트웨어 개발에서의 현실적인 문제들을 해결하기 위해서는 바탕이 되는 지식을 기반으로 새로운 문제를 풀기 위한 해법을 찾을 수 있어야 한다. 그러한 해법들은 기반 지식의 진화를 통해 이루어질 수 있으며, 교육을 통한 창의적 사고방식을 배양함으로써 그 목표를 달성할 수 있다.

4. 소프트웨어공학 교육의 학습 모형화

학습 모형은 소프트웨어공학의 다양한 지식유형 및

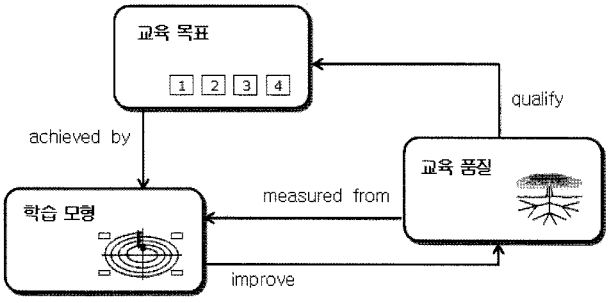


그림 6 교육목표, 교육 품질 및 학습 모형간의 관계

지식수준을 고려하는 교육 진행 절차를 모형화한 것으로써, 3장에서 정의한 교육 목표를 달성하고 교육 품질을 향상시키기 위한 목적으로 제시되었다. 교육 목표, 교육 품질, 그리고 학습 모형간의 관계를 그림 6에 나타내었다. 그림 6에 표현한 것처럼 교육 목표는 학습 모형을 통해 달성되며, 교육 품질은 학습 목표의 달성도를 나타낸다. 그리고 학습모형을 구성하는 요소들로부터 교육품질이 측정된다.

그림 6에 나타난 학습 모형에 대한 정의는 그림 4에 나타난 지식 유형, 즉 원리, 방법 및 기법, 방법론, 그리고 도구에 대한 단계적 학습 과정을 모형화한 일차원 모형과 기초학습, 기반학습, 심화학습, 그리고 종합학습의 지식수준에 근거한 이차원 모형으로 정의하였다.

4.1 일차원 학습 모형

소프트웨어공학의 다양한 이론 및 체계를 학습하기 위해서는 기본적으로 원리 및 개념의 이해가 선행되어야 한다. 최근에 출간되는 많은 소프트웨어공학 교재를 살펴보면 대체적으로 교육의 내용을 소프트웨어의 수명주기단계에 따라 제시하고 있다[12]. 소프트웨어 수명주기 단계를 기준으로 이루어지는 소프트웨어공학 교육은 개발 절차의 이해와 절차에 따른 태스크들을 인지하고 숙련하는 데 목적이 있으며, 일반적인 소프트웨어 개발을 보다 절차적으로 수행하고자 하는데 주안점을 둔다. 하지만 소프트웨어공학은 좋은 품질의 소프트웨어를 개발하는 것을 목표로 하고 있으며, 이를 위해서는 기반이 되는 원리 및 개념에 대한 충분한 선행 학습이 이루어져야 한다. 이 목표를 달성함으로써 방법론 기반의 개발 절차를 보다 효과적으로 적용할 수 있을 것이다.

따라서 소프트웨어공학 교육의 일차원 학습 모형화는 소프트웨어 수명주기 모델이 아닌 주제 유형에 따른 학습이 이루어지도록 해야 한다. 예를 들면, 기본적으로 소프트웨어공학의 원리, 품질 및 품질 요소, 수명주기 모델 등에 대한 이론 및 개념에 대한 선행

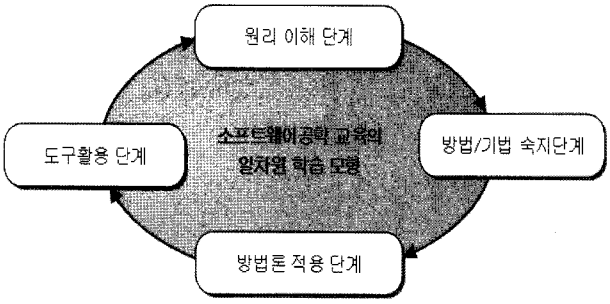


그림 7 소프트웨어공학 교육의 일차원 학습 모형

학습을 수행하고, 이를 기반으로 형상관리, 품질관리 등의 방법 및 기법에 대하여 학습해야 할 것이다. 그림 7은 이러한 일차원 학습 모형의 진행 단계를 도식화 한 것이다.

4.2 이차원 학습 모형

소프트웨어공학 교육에서의 이차원 학습 모형은 일차원 학습 모형을 포함하는 상위 단계의 학습 모형으로 정의하였다. 넓은 의미의 소프트웨어공학에 대한 체계적인 학습을 수행하기 위하여 기초학습, 기반학습, 심화학습, 그리고 종합학습의 단계로 학습 모형을 설계하였으며 이들은 다음과 같은 특성을 갖는다.

• 기초학습단계 : 기초 학습단계는 정해진 구문론과 의미론을 갖는 프로그래밍 언어를 사용하여 소프트웨어를 개발하는 학습단계이다. C 또는 C++ 언어를 이용한 프로그래밍 과정을 학습함으로써, 소프트웨어의 구성요소가 무엇인지를 배우고, 체계적인 공학적 접근 방법의 필요성을 인식한다.

• 기반학습단계 : 이 단계에서는 소프트웨어 개발의 체계적인 접근을 위한 공학적 이론 및 방법들을 숙지한다. 모듈화, 소프트웨어 품질, 프로세스 모델 등의 학습을 통해 체계적인 소프트웨어 개발 방법을 숙지한다.

• 심화학습단계 : 심화 학습단계는 기반학습단계의 이론을 근간으로 소프트웨어 개발의 체계적인 방법론을 적용하는 과정이다. 소프트웨어 개발을 위한 분석 및 설계를 비롯한 문서화, 형상 관리, 프로젝트 관리 등의 기법을 적용하는 실습과정을 수행한다.

• 종합학습 단계 : 이 단계에서는 하위의 세 가지 학습 단계를 종합하는 단계로써, 주어진 문제를 프로젝트 팀 단위로 풀어가는 학습 과정을 경험한다. 요구사항을 기반으로 대안 분석 및 비용분석을 수행하고, 객체지향 방법론 등의 적용을 통한 분석 및 설계, 그리고 이들에 대한 체계적인 문서화를 팀 프로젝트 수행을 통해 경험한다.

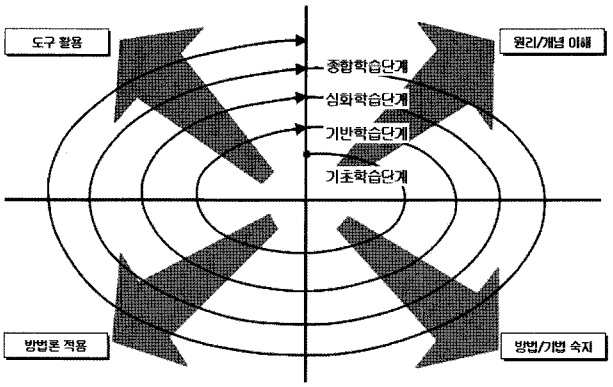


그림 8 소프트웨어공학 교육의 학습 모형

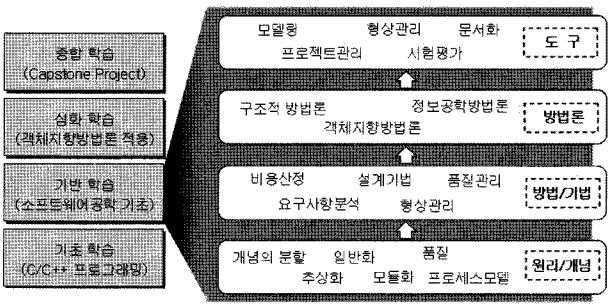


그림 9 소프트웨어공학 학습모형의 기반학습단계 예시

이와 같은 이차원 학습모형과 일차원 학습 모형의 통합된 교육 모형은 그림 8과 같다.

그림 8에서 제시된 소프트웨어공학 교육의 학습 모형에 대한 기반 학습 단계의 지식 유형에 대한 예시는 그림 9와 같다. 그림 9에 나타난 것처럼 소프트웨어공학 기초 교과 영역에서의 모듈화, 추상화 및 일반화, 품질요소 등에 대한 원리 학습을 시작으로 요구사항 분석기법, 설계 기법, 형상관리 기법 등의 숙지하며, 이들을 통합한 개발 방법론의 프로세스를 학습하게 된다.

5. 소프트웨어공학 교육의 지원 도구

앞서 4장에서 제시한 소프트웨어공학 교육의 학습 모형을 기반으로 교육을 진행하기 위한 도구들을 살펴보고, 이들의 활용 방안에 대하여 설명한다. 본 절에서 설명하는 교육지원 도구들은 소프트웨어공학 관련 강좌를 통해 활용할 수 있는 도구들의 예시이다.

5.1 학생 및 강의 평가

학습 과정에 대한 평가는 매우 중요하다. 학습자의 교과 내용에 대한 이해를 점검하거나 추후 보완해야 할 학습의 지식 유형이 무엇인지를 알 수 있기 때문이다. 평가는 두 가지 측면에서 진행되는데 첫 번째가 학생 평가이다. 학생 평가는 강좌의 시작 시점에서

수행하는 선수 테스트와 중간, 기말 평가를 수행한다. 선수 테스트는 소프트웨어를 개발하면서 느꼈던 문제점이나 어려웠던 점에 대하여 질문하고, 이를 통해 소프트웨어공학이 왜 필요한지를 생각해 보게 하는 수단으로 활용된다. 중간, 기말 평가에서는 학습의 지식 유형별로 개념 정의, 개념 응용, 기법 적용 등의 단계적인 학습의 수준을 평가하고, 학습 성취도가 낮은 부분을 분석하여 점검한다. 강의 평가는 학생들이 수강한 강좌의 교육 내용과 방법을 평가하는 것으로써, 어떠한 강의 방식을 원하는지, 어떻게 수업을 진행하는 것이 흥미 있는지를 알 수 있게 하는 수단으로 활용된다.

5.2 팀 프로젝트

강좌 진행과 함께, 소프트웨어공학의 기술을 적용하기 위한 팀 프로젝트를 수행한다. 소프트웨어공학과 직접적으로 관련된 교과목은 크게 소프트웨어공학 기초[2,12]와 객체지향 시스템 설계[14,15] 교과목이다. 소프트웨어공학 기초 교과목에서는 팀 프로젝트를 통해 공학적 이론과 방법들을 적용하는 과제를 수행한다. 팀별로 대상 시스템을 정하고, 비용 산정 기법, 프로젝트 계획, 사용자 인터뷰 방법, 테스트 사례 작성법, 문서화 방법, 형상(버전) 관리방법 등을 적용하는 과제를 수행한다. 객체지향시스템 설계 교과목에서는 소프트웨어 개발 수명주기 단계를 기반으로 응용 시스템에 대한 분석 및 설계 과정을 진행한다. 이러한 과정에서 작성되는 모든 결과들은 문서화를 통해 유지관리 하도록 지도한다.

5.3 CASE Tool

소프트웨어공학 기초 교과 및 객체지향시스템 설계 교과를 운영하는 과정에서 필요한 또 한 가지 교육 도구는 CASE Tool이다. 기능점수 산정도구, 작업분할도(WBS) 및 칸트 챠트 작성도구, 객체지향 모델링 도구 등이 효과적인 교육을 위해 활용된다. 이러한 도구들은 비교적 손쉽게 사용할 수 있는 도구들로 인터넷을 통해 확보하거나 기존의 도구와 통합된 형태로 제공하는 것들이다. 이러한 도구 이외에 다양한 양식지, 예를 들면 사용자 인터뷰 양식, Use Case 정의 양식, 테스트 사례 작성 양식 등을 사용하여 이론적 학습 내용을 사례를 통해 적용하는 것이 필수적으로 필요하다.

5.4 프로젝트 검토회

팀 프로젝트 수행에 대한 결과의 검토회는 매우 중요하다. 작성한 산출물의 품질이 어떠한 가에 대한 것은 프로젝트에 참여한 모든 학습자들이 궁금해 하는 부분이다. 프로젝트 수행 결과를 문서화하여 제출하는 것에 국한하지 않고, 이들에 대한 점검 및 점검 결과의 피드백이 필요하다. 프로젝트 검토회는 다양한 방법에 의해 진행될 수 있는데, 그 중 한 가지가 동료 검토(peer review) 방법[16]이다. 프로젝트 수행 결과를 다른 팀이 검토하고, 검토 결과를 발표하도록 하는 것이다. 이렇게 함으로써, 자신들이 작성한 결과와 다른 팀이 작성한 결과를 비교 분석하고, 평가할 수 있는 기회를 제공할 수 있다.

5.5 적절한 교육 컨텐츠

주어진 기간 내에 방대한 소프트웨어공학의 주제를 다루는 것은 어렵고 힘든 일이 아닐 수 없다. 특히 대학 교육은 제한된 시간 내에 해당 교과목의 모든 내용들을 교육해야 하는 한계를 갖는다. 교수법을 연구하는 많은 교육 공학자들의 주장처럼 100%의 내용을 가르치기 보다는 60%의 내용을 가르치고 나머지는 자율적인 학습을 통해 배워갈 수 있도록 해야 한다는 것이 매우 중요하다. 교수자의 입장에서는 조금이라도 더 학습자에게 교과 내용을 전달하고자 하나, 이는 쉽게 잊어버리거나 스쳐지나가는 일종의 이야기 거리에 불과하다. 소프트웨어공학 기초 교과목에서 구조적 방법론 또는 객체지향 방법론과 같은 소프트웨어 개발 방법론 전체를 다루는 것은 매우 위험스러운 생각이다. 따라서 소프트웨어 개발 방법론에 대한 부분은 별도의 교과목을 통해 교육할 수 있도록 교육 컨텐츠가 구성되어야 한다.

5.6 산업체 특강

소프트웨어공학 관련 교과목들은 비교적 이론적인 부분보다는 실무에 적용하기 위한 많은 방법 및 기법들을 담고 있다. 따라서 단순히 교재를 중심으로 하는 강의와 프로젝트 기반의 실습을 통해서 교육을 진행하는 것에는 한계가 있다. 다시 말해서 교수의 풍부한 경험과 지식을 바탕으로 교육 내용에 대한 실무 적용 방법 및 사례들을 제시 하지만, 보다 현장감 있는 목소리를 들을 수 있는 기회가 제공되어야 한다는 것이다. 교과의 내용을 마무리하는 시점에서 산업체의 해당 분야 전문가를 초청하여 특강을 진행함으로써, 학습자들이 갖는 궁금증을 풀어주고 현장의 상황을 이해할 수 있게 한다. 산업체 특강의 효과는 어떠한 주제를 가지고 진행하는가에 따라 크게 달라지는데, 예를 들면 "산업체에서 요구하는 IT 분야의 능력" 등과 같은 일반화된 주제가 아니라 "금융 소프트웨어 개발에서의 형상관리 기법의 적용" 등과 같은 보다 구체적이고 학습 내용과 직접 관련된 주제를 선택해야 한다.

6. 소프트웨어공학 교육의 성과 사례 분석

본 장에서는 소프트웨어공학 강좌를 수강한 학생들에 대한 학업 성취도를 평가한 결과를 사례로 제시하고, 그 결과의 의미를 살펴보고자 한다.

6.1 학업성취도 평가 사례

사례로 제시하는 교과는 기반단계에 해당되는 소프트웨어공학 기초 교과목에 대한 종합 평가 결과에 대한 것이다. 평가는 크게 5개의 주제 영역에 대한 평가로 이루어졌으며, 이는 개념정의 영역, 분석기법영역, 설계기법영역, 테스팅 영역, 그리고 개발프로세스 영역이다. 전체 42명의 학생을 대상으로 평가한 주제영역의 학업 성취도를 각 10점 만점으로 정규화하여 나타낸 그래프는 그림 10과 같다. 참고적으로 평가의 대상인 학습자는 소프트웨어 개발 방법론에 대한 심화학습이 이루어지지 않은 상태이다.

그림 10에 나타난 결과를 보면 개념이해, 설계기법, 테스팅 기법에 대해서는 상대적으로 높은 평가 결과를 얻었다. 설계기법에서는 주어진 코드로부터 모듈의 응집력을 평가하는 문제였으며, 테스팅 기법은 요구사항으로부터 블랙박스(block-box) 테스팅을 위한 동치분할(equivalence partitioning)을 수행하는 문제였다. 이러한 주제영역은 교육과정에서 진행된 실습 수행의 효과로 해석된다. 그러나 간단히 주어진 요구사항을 기반으로 분석 모델을 작성하는 분석기법영역의 학업성취도는 제일 낮았으며, 개발 절차에 따른 산출물을 정의하는 개발 프로세스 영역에서도 상대적으로 낮은 성취도를 보였다.

결론적으로 본 평가사례로부터 개념에 대한 이해를 높이면 방법이나 기법의 지식유형에 대한 학습성과가 높아짐을 알 수 있으며, 실습과정을 통해서 교육내용에 대한 적용성과 자율성이 높아짐을 알 수 있다.

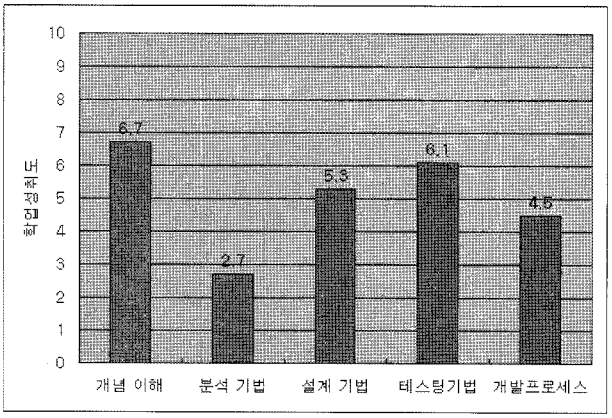


그림 10 주제영역별 학업성취도 평가 사례

참고적으로 평가 대상 학생들에 대한 전체적인 학업성취도에서 평균은 49.8이었으며 표준편차는 17.4이다.

6.2 프로젝트 검토회 사례

본 절에서 제시하는 사례는 객체지향시스템 설계 교과를 수강한 학생들이 팀 프로젝트를 수행하고, 그 과정에서 작성된 산출물을 동료 검토 방법에 의해 평가한 결과이다. 프로젝트는 총 10팀으로 편성되어 진행하였으며, 팀별로 4~5명의 팀원으로 구성되었다. 프로젝트를 통해 작성된 산출물은 시스템 정의서, 프로젝트 계획서, 요구사항 정의서, 요구사항 분석서, 소프트웨어 설계서의 5종이며, 이중에서 요구사항 정의서를 베이스라인으로 하여 요구사항 분석서에 대한 동료검토를 수행하였으며 그 결과는 표 2와 같다.

표 2에 나타난 데이터로부터 10팀에 의해 각각 수행된 요구사항분석서의 동료검토 결과는 문서량 평균 26.9페이지에 대하여 평균 미팅시간은 9시간 31분이 소요되었으며, 발견된 결함의 수는 평균적으로 59개에 달했다. 결함의 유형코드는 모든 개발 단계에서 적용될 수 있는 총21개로 주어졌으며, 이중에서 기타 결함(OT: Other)을 제외하고 결함코드 IC(Incorrect)와 OM(Omission)에 해당하는 결함이 가장 많이 발견되었다. 동료검토를 통해 분석된 결함유형별 분포는 그림 11과 같다.

그림 11의 사례 결과에서 OT는 검토자가 결함유형을 다르게 지정한 부분을 포함하고 있으며, 이는 검토자의 결함유형에 대한 숙지 미숙으로 비롯되었다. CL(Clarification)과 CS(Consistency)는 모호하게 기술되었거나 일관성이 없는 결함이며, TY(Typo)는 오타 결함에 해당된다. 발견된 결함의 유형으로부터 프로젝트

표 2 프로젝트 산출물 동료검토 결과 사례

팀명	팀원 (명)	문서량 (페이지)	미팅소요시간 (hh:mm)	총결함수	중결함	경결함
1	4	23	9:20	45	7	38
2	4	27	20:40	82	18	64
3	4	22	9:10	71	37	34
4	5	37	5:30	43	5	38
5	5	25	8:00	83	53	30
6	4	19	10:05	28	13	15
7	5	40	10:50	141	62	79
8	4	12	8:00	42	22	20
9	4	33	8:30	35	17	18
10	4	31	5:40	20	14	6

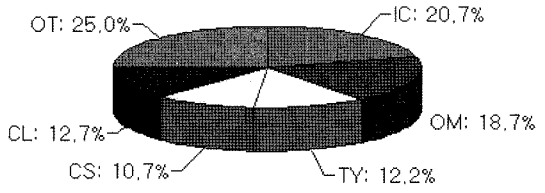


그림 11 발견된 결함유형의 분포 사례

수행과정에서 적용된 객체지향 개발 방법론에 대한 개념과 표현 방법에 대한 학습이 충분히 이루어졌음을 확인할 수 있다. 그러나 이러한 사례 제시의 더 큰 의미는 이러한 학습 도구를 교육 과정에 적용함으로써 학습자의 흥미 유발이나 자율적인 참여도가 확실히 향상된다는 것이다.

7. 결 론

본 고에서는 소프트웨어공학 분야의 체계적인 교육을 위한 프로세스 모형을 제시하였으며, 이를 위하여 먼저 교육 목표를 정의하고, 정의된 목표 달성을 위한 교육 절차를 제시하였다. 교육의 절차는 지식 유형을 기반으로 하는 일차원 학습 모형과 교육 수준을 기반으로 하는 이차원 학습 모형으로 구분하여 제시하였다.

이러한 학습 모형에서는 소프트웨어공학에서 다루는 원리와 개념에 대한 명확한 이해를 선행으로 다양한 주제영역의 방법 및 기법을 숙지하고, 소프트웨어개발 방법론을 통해 이들을 적용하도록 가이드하고 있다. 또한 기초, 기반, 심화 및 종합 학습 단계를 통해 소프트웨어공학 분야의 체계적이고 안정적인 교육과 학습이 이루어질 수 있도록 제시하였다. 제시된 학습 모형에 따라 소프트웨어공학 교육을 수행할 때, 수반되어야 하는 교육 지원 도구는 무엇인가에 대하여 간략히 설명하였다.

본 고에서 제시한 학습 모형 및 관련 도구들이 비록 일반적일 수는 있으나, 이에 대한 교육 현장에서의 적용은 무엇보다 가치 있는 것일 수 있다. 최근 4, 5년 사이에 소프트웨어공학 분야에 대한 다양한 프로그램들이 대두되었다. 대학에서의 교육뿐만 아니라 소프트웨어 관련 단체 및 기업체에서 많은 교육 프로그램을 운영하고 있다. 그러나 이러한 교육 프로그램들이 단편적인 전문 지식을 전달하고, 소프트웨어 개발의 수단만을 제공하게 된다면 바람직한 소프트웨어공학 교육이라고 볼 수 없을 것이다. 각 교육과정이 소프트웨어공학에서 어떠한 세부 목표를 달성하기 위해 진행되고 있으며, 또한 다른 지식 영역들과의 연계성은 무엇인지를 명확히 파악할 수 있는 체계적인 교육이 이루어져야 할 것이다.

참고문헌

[1] ANSI/IEEE, IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 610.12-1990, Feb., 1991.

[2] C. Ghezzi, M. Jazayeri, and D. Mandrioli, Fundamentals of Software Engineering, 2nd ed., Prentice Hall, 2003.

[3] JTFCC, Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, IEEE & ACM, August, 2004.

[4] A. Abran and J. Moore, SWEBOK: Guide to Software Engineering Body of Knowledge, 2004 Version, IEEE Computer Society, 2004.

[5] D. Bagert, et. al., "Guideline for Software Engineering Education Version 1.0," Technical Report CMU/SEI-99-TR-032, October, 1999.

[6] D. Parnas, "Resolving Dilemmas in Software Engineering Education," CSEET'07, Ireland, July 2007.

[7] A. J. Cowling, "Stages in Teaching Software Design," CSEET'07, Ireland, July 2007.

[8] R. Valerdi and R. Madachy, "Impact and contributions of MBASE on software engineering graduate courses," Journal of Systems and Software vol. 80, 2007, pp.1185-1190

[9] D. Frailey, "Experience teaching Barry Boehm's techniques in industrial and academic settings," Journal of Systems and Software vol. 80, 2007, pp.1217-1221

[10] KAIST Software Expert Program, http://software.kaist.ac.kr.

[11] ICU Dual Degree Program, http://www.icu.ac.kr.

[12] I. Sommerville, Software Engineering, 8th ed., Addison Wesley, 2007.

[13] J. McCall and G. Walters, Factors in Software Quality, NTIS, Springfield, VA, 1997.

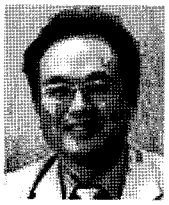
[14] M. Blaha and J. Rumbaugh, Object-Oriented Modeling and Design with UML, 2nd ed., Prentice Hall, 2005.

[15] A. Dennis, B. Wixom, and D. Tegarden, System Design and Design with UML Version 2.0: An Object-Oriented Approach, Wiley, 2005.

[16] K.E. Weigers, Peer Reviews in Software: A Practical Guide, Addison-Wesley, 2001.

홍 장 의

1988 충북대학교 전산학과(이학사)
1990 중앙대학교 전산학과(석사)
2001 한국과학기술원 전산학과(박사)
2002 국방과학연구소 선임연구원
2004 (주)솔루션링크 기술연구소장
2004~현재 충북대학교 전기전자컴퓨터 공학부 조교수
관심분야 : 임베디드 소프트웨어 공학, 소프트웨어 개발 방법론, 소프트웨어 품질, 소프트웨어 프로세스 개선
E-mail : jehong@chungbuk.ac.kr



배 두 환

1980 서울대학교 조선공학(학사)
1987 미국 Univ. of Wisconsin - Milwaukee 전산학(석사)
1992 미국 Univ. of Florida 전산학(박사)
1995~현재 한국과학기술원 전산학과 교수
2002~현재 소프트웨어프로세스개선 센터 소장
관심분야 : 소프트웨어 프로세스, 객체지향 프로그래밍, 컴포넌트기반 프로그래밍, 임베디드 소프트웨어 설계, 관점지향 프로그래밍
E-mail : bae@se.kaist.ac.kr