特輯 技術解說

국내외 소프트웨어 엔지니어링 기술 동향 및 향후 추세

金 薰, 任 在 傑
現代電子産業(株)

최근 정보 관련 산업의 국제화 시대에 발맞추어 그 근간이 되는 소프트웨어 공학의 발전이 중요시 되는 바 본고에서는 현재 국내 소프트웨어 공학 분야의 문제점과 해결 방안을 제시하고, 국외에서 최근 연구되고 있는 새로운 분야의 국내 소프트웨어 산업에의 연결 방안 및 국내 실정에 맞는 개발 방향을 제시 하고자 한다. 특히 최근 미국 등 선진 각국에서 연구 제시되고 있는 객체지향 소프트웨어 개발 방식(object oriented S. W. development method), 소프트웨어 재사용(S. W. reusability), 시스템 가능성 검증용인 프로토타이핑(prototyping)기술 등에 대해 설명하고 이러한 기술들을 국내 S. W. 개발업체 및 산업체에서 효과적으로 이용할 수 있는 방안에 대해 의견을 제시하고자 한다.

I. 서 론

정보산업 분야에 있어 미국내의 소프트웨어 관련 분야의 매출액이 하드웨어 관련 분야를 앞지를 날이 멀지 않은 시점이다. 국내 기업체 및 정보 관련 학계에서도 소프트웨어의 중요성에 대한 인식은 하고 있으나, 이에 대한 대책 마련에는 소홀히 하는 경향이 있으며, 정부 차원의 장기적인 안목에서의 정책이 적극적으로 추진되지 못하고 있는 실정이다. 소프트웨어 공학(software engineering)이란 이러한 소프트웨어 개발을 추진함에 있어 보다 효율적이고 부가 가치가 높은 방향으로 나아가는 길을 여러 가지 측면에서 검토하고 그 방법을 제시하는 학문이다. 최근 미국에서 거론되고 있는 소프트웨어 위기(S. W. crisis)론을 보면 값이 계속 하락하고 있는 하드웨어에 비해 임금인상등의 요인으로 개발비용이 급증하는 소프트웨어 문제를 이대로 방치해 둘 수 없다는 주장에 근거를 둔다. 산업화 정보화가 진전될수록 수요가 격증하는 소프트웨어 분야의 적정한 공급 및 가격 안정을 위해서는 개발 효율을 높이는 방법이 필수적이며, 소프트웨어 공학 분야의 연구 진행이 이를 푸는 열쇠가 된다 하겠다. 미국과 일본에서 적극 검토 연구되고 있는 소프트웨어 공장(S. W. factory) 등은 이의 좋은 예라고 할 수 있겠다.

최근 국내외의 소프트웨어 공학 분야의 연구 추세를 보면 소프트웨어 재사용(S. W. reusability), 소프트웨어 생명 주기(S. W. life cycle)의 각 단계별 효율 증대, 소프트웨어 생산성을 높이기 위한 소프트웨어 공장(S. W. factory), 특수 목적을 위해 필요한 각종 프로그램 언어에 대한 연구, 개발될 시스템의 사전 점검을 위한 프로토타입(S. W. prototype)에 대한 연구, 프로그램 효율을 측정하는 소프트웨어 계측(S. W. metrics), 소프트웨어 자동 생산 시스템(CASE tool)등에 관한 연구 등 무척 그 분야가 다양화 되어 있다.

그러나 이러한 연구는 주로 구미 각국에서 적극적으로 추진되고 있으며, 국내에서는 소프트웨어 공학에 대한 정확한 인식의 부족으로 그 연구진행 및 개발에의 응용이 무척 느린 실정이다.

이제 국내에서도 시간과 인원만 투입하면 시스템이 만들어진다라는 기존의 관념을 버리고 새로운 기법의 연구 결과를 응용하여 시스템 개발 효율을 보다 높여가야 할 시점에 있으며, 나아가 국내 현실에 맞는 소프트웨어 공학을 발전시켜 이 분야에 보다 현실적으로 공헌될 수 있는 연구 결과가 많이 나오고, 이를 적극 수용하여 국내 소프트웨어 업계의 국제 경쟁력을 높여 나가야 할 것이다.

최근 들어 대기업의 전산실 외에도 소프트웨어 개발

만을 목적으로 하는 많은 중소기업체들이 설립되어 극심한 경쟁속에서도 새로운 시스템을 개발하여 시장에 선보이며, 사명감을 갖고 이 분야의 업무에 전념하고 있음은 감명 깊은 일이다. 그러나 안타까운 일은 개발 담당자의 경력이 적고, 또 오래된 개발 방식을 사용하며, software life cycle등의 기본적인 기법 적용에 충실하지 않고, 시스템 구축에만 급급한 실정인 바, 따라서 software의 국제 경쟁력이 부족하고, 복잡한 대형 시스템(여러 팀이 나누어 시스템을 개발하고 추후에 연결하여 테스트를 실시하는 중장기 시스템) 개발은 별로 추진하지 않고 있으며 국내 소프트웨어 기술 수준이 낮다는 이유로 외국업체에 외주를 주는 실정이다. 그러나 한편 생각해 보면, 소프트웨어 공학 기법을 따라 소프트웨어 라이프 사이클에 기반을 두어 충실히 계획을 수립하고, 보다 효율적으로 개발관리를 시도하면 국내에서도 중대형 소프트웨어를 구축할 수 있는 능력이 있다고 본다.

본 고에서는 이러한 국내 현실에 기반을 두고 외국의 최근 연구 분야를 소개하며, 정보 과학 분야의 발전과 더불어 국내 소프트웨어 개발 산업이 지속적으로 성장하여 조기에 정착 할 수 있도록 적극적인 방안을 제시하고자 한다.

Ⅱ. 국내 소프트웨어 산업의 소프트웨어공학 기술 적용 실태

전산 기술이 국내에 도입 정착되기 시작한 지 약 20년, 정보 산업 분야의 발전은 급속도로 진전 되었으나, 아직도 개발 수준은 그리 높지 못하고 있는 실정인 바, 이에 대한 제반 현황들을 제시하고 검토해 보도록 하자.

① 아직도 대기업 전산실 중심의 소형 시스템이나 응용 소프트웨어 구축이 주가 되고 있음.

② 소프트웨어 공학 기술 적용은 life cycle 가운데 사양서 작성, 코오딩, 테스팅 등에 주력하고 기타 주요 단계 생략.

③ 전산화 제도 및 가공(CAD/CAM) 분야 등 고급 기술 분야 국산화 개발 속도가 느리며 이의 기반 기술 부족.

④ 대형 시스템 개발은 한국전자통신연구소, 한국과학기술연구소, 시스템공학연구소 등 정부 출연 연구소 주관으로 추진중임.

⑤ 특수 대형 시스템 개발비 과다로 인해 대기업에서도 개발을 기피하고 고가의 외국산 패키지(package)들을 도입함으로써 소프트웨어 분야의 국내 기술축적이 느림.

⑥ 국내 개발 가능 시스템도 업계의 개발 기피로 인해 그 적기를 놓침으로써 추후 소요되는 새로운 버전(version)의 소프트웨어를 모두 외국에서 도입해야 함.

⑦ 중소 소프트웨어 업체에서 어렵게 개발된 시스템도 그 기능이 낮다는 이유로 국내 사용자들이 구입을 기피, 비슷한 기능의 고가의 외국산 시스템을 이용함으로써 소프트웨어 산업 발전을 저해.

⑧ 국내 소프트웨어 개발 종사자나 학계 인사들이 외국 선진 전문가나 학계와의 교류 부족으로 새로운 기술 정보 입수 지연.

⑨ 데이타베이스, 인공지능 분야의 국산화 시스템 구축을 추진하고 있으나, 기반 기술이 많이 부족한 실정임.

⑩ 정부주도 행정 전산망 구축 및 표준화 추진계획에 기대.

상기 문제점 및 현상들을 보면서 추후 국내 정보산업 분야의 원만한 발전을 위한 제언을 하기로 한다.

① 정부 주도하에 소프트웨어 분야의 활성화를 위해 보다 획기적인 정책지원이 필요함.

② 소프트웨어 개발 전문가 및 학계 인사가 자주 접촉하며 기술 협력 및 공동 개발 등을 추진함.

③ 외국 선진 전문가와 적극적인 인적교류 및 정보교환을 시도한다.

④ 소프트웨어 분야에 대한 적극적인 연구 추진 및 개발에의 적용으로 기술력 증진.

⑤ 시스템 개발시 기본적인 소프트웨어 공학 이론에 따라 life cycle 적용에 충실을 기함.

⑥ 국내 개발 소프트웨어의 적극적인 구매 및 활용으로 개발 의욕을 높임.

⑦ 주요 기술분야(CAD, computer graphics 등) 및 DB, AI 등 대형 시스템 개발은 정부 주도하에 적극 추진.

⑧ 소프트웨어 개발 분야의 표준화를 정부 주도하에 조기 실현, 생산성 향상.

⑨ 응용 소프트웨어 분야 외에도 시스템 소프트웨어 등 고부가가치의 시스템을 개발토록 적극 추진함.

⑩ 개발 계획 수립시 국내에서 기존에 개발되지 아니한 새로운 분야에 대해서는 국내 실정에 따른 새로운 아이디어로 외국산과의 대응 능력 배양.

⑪ 외국에서는 개발되지 않았으나 국내 실정에 견주어서 특수한 분야에 적용할 수 있는 경쟁력 있는 개발 프로젝트 추진.

⑫ 기반기술에 해당되지만 국내 수요가 적은 특수 분야의 소프트웨어는 정부에서 우선 구입, 업계의 개발 의욕 증진.

이 외에 학계 및 소프트웨어 공학 분야 전문가를 중심으로 위원회를 구성 재사용율(reusability) 증가 방안, 프로토타입(prototype) 개발 방안, 등 새로운 기술에 대한 정립 및 산업계에서의 이용 방안등을 모색하고, 이를 전수함으로써 실질적으로 새로운 기법을 사용한 생산성 향상이 가능한 시스템 개발을 적극 유도해 나가는 일이 중요하다고 본다.

Ⅲ. 국내 소프트웨어 기술 현황

국내 소프트웨어 산업계와는 달리 미국을 위시한 선진 각국의 업계에 대해서는 다른 각도에서 검토해 보아야 하겠다. 그들은 오랜 역사의 소프트웨어 개발 경력이 있으며, 저변에 깔린 기술을 바탕으로 많은 수의 소프트웨어 관련 기술자 및 학자들이 이를 이끌어 주고 있다. 따라서 소프트웨어 산업계가 자생적으로 발전하고 있는 지금은 정부 주도하의 산업 육성에 의존하기 보다, 자연스럽게 시장 경제의 규칙에 따라 업체별로 치열한 경쟁 속에서 스스로 자구책을 찾으며 발전해 나가고 있는 현실이다. 미국의 예를 들면, 각 분야별로 자국내의 광대한 시장을 배경으로 소프트웨어 분야가 독보적인 발전을 거듭하고 있으며, 각 분야 세계 최대의 소프트웨어 업체임을 서로간에 경쟁적으로 내세우고 있는 실정이다. 캘리포니아주의 산호세 부근의 실리콘 벨리에 가 보면 많은 수의 소프트웨어 엔지니어가 회사마다 가득히 있어 각각의 노우하우(know-how)를 가지고 전 세계를 향해 소프트웨어 제품 소개에 열을 올리고 있다. 오랜 전통이 말해주듯 소프트웨어 엔지니어의 업무습관 및 자세 또한 우리와 다르다. 그들은 소프트웨어 개발 전체 과정을 통해 축적된 경험을 쫓아 일사불란하게 개발업무를 수행하고 있으며, 소프트웨어 라이프 사이클(life cycle)에 따라 철저히 진행 단계를 추진하면서, 그때 그때 참고내용(documentation)을 철저히 기록 유지하고, 서로간의 협조 체제가 잘 이루어져 있다. 우리가 그들과 경쟁을 하고 생산성을 높이기 위해서는, 그들이 가지고 있는 기술을 전수받는 외에 우리가 자체적으로 소프트웨어 공학을 연구 발전시켜 이를 개발업체에 전파 할 뿐 아니라, 외국에서 연구 되어지고 실용화하여 사용되고 있는 최신 기술들을 우리 것으로 소화시켜 기술 국산화 정착에 힘써야 하겠다.

최근 외국 학계에서 활발히 연구되고 있는 분야를 보면 소프트웨어 재사용(reusability) 분야, 소프트웨어 자동생성기술(CASE) 분야, 객체지향 소프트웨어 개발(object oriented software development) 기술, 프로토타입(prototype) 개발 운용기술, 소프트웨어 성능 측정(software metrics), 분산 시스템용 소프트웨어(distributed software) 운용 기술 등 그 분야가 무척 다양화 되고 있다. 본 고에서는 몇가지 기술들에 대해 간단히 설명하고 우리가 이를 소화 응용하는 관점에서 논해 보고자 한다.

1. 소프트웨어 재사용 (software reusability)

소프트웨어 재사용의 목적은 소프트웨어 시스템의 개발 및 유지 비용 절감과 상품화 기간 단축에 있다.[2]

프로그래머라면 누구나 반복되는 작업을 줄이는 문제에 대해 그 해결책을 생각하면서, 해결방안으로 프로그램 코딩시간을 줄이고, 그 개발된 소단위 프로그램(component)을 재사용하는 방법을 검토해 보았을 것이다.

현재 이는 몇가지 방법으로 연구 모색되고 있는데, 재사용 모듈(module)을 라이브러리(library)화 해놓고 이를 저장해 적절히 불러오는 방식이 주로 연구되고 있으며, 그 목록을 수록해 놓은 캐달로그(catalog)를 비치해 참조케 하는 방식도 추진하고 있다.[3]

최근 각광을 받고 있는 객체지향 방식(object oriented design method)은 각각의 객체(object)들인 프로그램 모듈(module)들이 상위 단계(level)인 크래스(class)의 정보를 그대로 이용할 수 있어 재사용율을 높인다.

재사용 소프트웨어 라이브러리는 재사용 효과가 무척 크며, 모든 내용물(item)들은 표준화 과정을 거쳐야 하며, 디자인(design) 컴퍼넌트부터 테스트 케이스(test case) 데이타 및 재사용 가능 코딩 모듈(module)이 포함된다. 미국의 MCL(Myers Corners Lab.)에서 개발하여 사용을 하고 있는 "MCL Reuse Library"의 경우를 보면, 이를 이용한 프로젝트 소스 코드(source code)의 약 15% 가량을 reuse libraries를 사용했다 한다.[4] 또한 재사용이 가능한 컴퍼넌트를 많이 확보하기 위해서는 프로그램 구현(implementation) 단계에서 보다는 사양서(specification) 작성 단계에서부터 추진해야 되겠다. 즉 사양서 작성시엔 보다 융통성이 많아 가능한 library 확보가 쉬운 반면에 구현단계에서는 한정적으로 되어 적절한 컴퍼넌트를 찾기가 쉽지 않은 까닭이다.[6]

재사용 시스템이 갖추어야 할 특성을 살펴보면 첫째, 컴퍼넌트(component)들의 적절한 분류(classification)가 필요하다. 라이브러리로부터 원하는 컴퍼넌트를 찾을 때 뿐 아니라 적절히 라이브러리를 구축하는 데도 필요하기 때문이다. 둘째, 라이브러리의 관리로써 데이타베이스 구축 및 신규 라이브러리의 등록 변경 삭제 기능 등을 원활히 수행해야 하며 필요한 컴퍼넌트를 빨리 정확하게 찾는 기능 또한 중요하다. 따라서 적절한 데이타베이스 구축기술 확보 및 이에 대한 끊임없는 연구가 필요하다. B-tree, B+-tree 등을 이용할 수 있고, 그 데이타는 frame 형식을 갖출 수도 있다. 셋째, 편리한 사용자 인터페이스(user interface)가 갖추어져야 한다.[5] 사용자가 편리하게 느끼고 친숙해 질 수 있는 시스템으로 멀티 윈도우(multi-window) 시스템이 좋으며 디자인 컴퍼넌트의 경우 도형으로 직접 보여 주어야 될 것이다. 재사용 시스템은 인공지능 기법을 이용하여 라이브러리를 지식 베이스(knowledge base)화 하여 규칙(rule)을 이용하여 필요 컴퍼넌트를 선택할 수도 있을 것이다.

2. 객체지향 소프트웨어 개발 방식(object oriented software developement method)

소프트웨어 모듈(mobule)들을 각각 하나의 객체(object)로 보고 이들 객체끼리 상호연결 및 정보교환이 가능하며, 객체내에 구체적인 데이타들을 저장하는 객체지향 방식을 이용하면 보다 친숙한 시스템을 구축할 수 있다. 즉 기존의 구조적(structured) 소프트웨어 개발 방식(software development method)은 컴퓨터의 업무처리에 맞추어 기계적인 구조로 되어있어 자연스러움이 없는데 반해, 객체지향 방식은 인간처럼 사물을 연상하고(하나의 객체로서) 그 사물간의 관계 및 행위 등을 표시하여 서로 간에 정보를 주고 받는 방식을 택하므로써 보다 인간적이고 이해하기 쉬운 장점이 있다. 이 방식은 객체지향 언어인 SMALLTALK, C++, EIFFEL, OBJECTIVE C, ACTOR 등 전용언어, ADA처럼 그 일부 기능을 갖고 있는 언어 등 여러 가지를 사용할 수 있다.

일반적으로 객체지향 방식으로 시스템을 구축하려면 먼저 객체들을 파악해야 하며, 각각의 객체간의 행위(action)를 표시하고 그 객체안에 구체적인 객체 자체의 임무(method)를 표시해 주어야 한다. 예를들어 한 시스템의 임무를 하나의 문장으로 표시한다면 그 문장내에 있는 모든 주어들은 각각 객체(object)가 되며 동사들은 각각 행위(action)가 되어 객체내에 기술되게 된다. 이제 각 객체들을 분석하여 비슷한 성격을 가지고 있고 또한 자체 임무가 같은 객체끼리 묶어 크래스(calss)를 구성하게 된다. 이때 크래스는 그 하위 단계인 객체들의 공통 정보를 타입 형태로 보유하고 있어 필요시 이를 제공하게 된다. 이를 상속(inheritance)이라 하며 이로써 공통 부분의 표시 영역이 줄어들게 되고 동시에 이용됨으로써 재사용율을 높인다.

다른 특징으로서는 캡슐화(encapsulation)가 있는데 이는 실체(instance)로서 나타내는 객체 내부의 구체적인 구조는 외부에는 은폐되어 있으며, 단지 그 기능만을 공개하고 있는 것이다. ADA, C++의 경우를 보면 구체적인 구조 등은 사용(private)구역에 표시해 두고, 공개 가능한 내용은 공용(public)구역에 표시하므로써 외부에 은폐시키고 이를 쉽게 불러 쓸 수 있게 조치하므로써 효용성이 높아지게 되는 것이다.

객체지향 방식의 소프트웨어 개발의 경우 여러 분야에 응용이 가능하다. CASE(computer aided software engineering) 시스템 개발시 모듈화 되어 저장된 컴퍼넌트(component) 라이브러리들을 객체화 하여 적절히 운용함으로써 쉽게 시스템화가 가능할 것으로 본다. 또한 객체지향 데이타베이스는 최근 많은 연구 대상이 되며, 또 사용되고 있고, 최신의 CAD(computer aided design) 관련 소프트웨어 중 그 데이타들의 운용을 객체지향 기법으로 하고 있는 것도 있다. 따라서 이 분야의 연구가 필수적이며, 앞으로 객체지향 언어들로 주요 프로그램 언어들이 바뀔 날도 멀지 않았다고 본다. 기존 "C"언어로 구축된 시스템이 "C++"로 변경되어질 수도 있을 것이다.

3. 프로토타이핑(prototyping)

아무리 아이디어가 좋아도 시스템의 실제 구축시 그 성공여부를 예상하기란 쉽지 않다. 사용자의 요구사항(requirement)을 분석할 때 동시에 테스트용 시스템(prototype)을 간단히 구축해서 그 시스템의 성공여부를 예상할 수 있다. 이는 짧은 시간내에 이루어져야 하며 따라서 rapid prototyping기법이라 한다. 이에 대한 연구는 최근 몇년간 많이 이루어져 왔으며, 주로 prototype 구축의 신속성과, 간단한 구축여부에 그 초점이 맞추어져 왔다. 따라서 소프트웨어 컴퍼넌트(component)의 재사용을 통해 속도를 높이고, 이들 라이브러리에 저장해 두고 필요한 모듈을 활용토록 함에 의견을 같이하고 있으며, 이는 인공지능 기법을 이용하고, 사용자 인터페이스를 이용하여 보다 쉽게 구축이 가능할 것으로 본다. 여기 프로토타이핑 적용시 필요한 단계를 기술해 본다.[1]

① 소프트웨어 요구서(software request)를 검토 평가하면서, 개발 대상인 소프트웨어가 프로토타이핑을 추진하는 대상으로서 만족할만 한지 결정한다.

② 결정된 프로젝트를 대상으로 하여 분석담당자(analist)는 요구사항(requirements)의 간소화된 표현(abbreviated representation)을 기술한다.

③ 요구사항이 검토된 후 프로토타입(prototype)을 위해 간소화된 디자인 사양서를 작성한다.

④ 프로토타입 소프트웨어가 만들어지면 이를 테스트하고 다듬는다.

⑤ 프로토타입의 일차 테스트가 끝나면 시스템의 고객에게 전달되어 시험 가동을 실시후변경 요구사항을 제시토록 한다.

⑥ 반복하여 4, 5단계를 시행하며 모든 요구사항이 만족할 만큼 정형화되도록 한다.

상기 작업을 쉽게 추진하기 위해서는 이를 빨리 구축하는 rapid prototyping 기법을 쓰는 바 재사용이 가능한 소프트웨어 컴퍼넌트(component)를 이용하게 된다. 이 컴퍼넌트는 구체적인 정보가 없는 데이타 구조형태(data structure)이거나 소프트웨어 구조적 컴퍼넌트(architectural component)로 이루어진다.

4. 소프트웨어 성능계측 (software quality metrics)

소프트웨의 성능을 올바로 평가하는 방법은 쉽지 않다. 따라서 그 방법으로 단지 간접적 측정을 많이 이용한다. 수 변수(variable)들 간의 관계를 정확히 표시하는 방법을 쓰게 된다. 여기 대표적인 두가지 방식을 소개한다.

1) Halstead's software science

이 방법은 프로그램 구현(implementation)이 끝난 뒤 각종 부호 및 변수의 숫자를 세어 계산하는 방식이다. 그 방식을 설명하면,

n_1–프로그램상의 서로 다른 부호 종류수(number of distinct operators)

n_2–프로그램상의 서로 다른 변수 종류수(number of distinct operands)

N_1–프로그램상의 부호의 총 갯수(total number of operator occurrences)

N_2–프로그램상의 변수의 총 갯수(total number of operand occurrences)

상기 측정치로부터 프로그램의 길이(overall program length)를 계산하면

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

프로그램 볼륨(program volume)은

$$V = N \log_2 (n_1 + n_2)$$

위 Halstead의 방식은 실험적인 평가를 시도하는데 유용하게 쓰여진다.

그림 1에 한 sorting 프로그램의 예를 들었다. $n_1=10$, $n_2=7$, $N_1=28$, $N_2=22$로 FORTRAN 언어를 사용했는데

〈Interchange sort program〉

```
SUBROUTINE SORT(X, N)
DIMENSION X(N)
IF (N.LT.2) RETURN
DO 20 I=2, N
    DO 10 J=1,I
    IF (X(I).GE.X(J)) GO TO 10
        SAVE=X(I)
        X(I)=X(J)
        X(J)=SAVE
10 CONTINUE
20 CONTINUE
   RETURN
   END
```

〈Operators of the interchange sort program〉

Operator	Count
1 End of statement	7
2 Array subscript	6
3 =	5
4 IF ( )	2
5 DO	2
6 ,	2
7 End of program	1
8 .LT.	1
9 .GE.	1
$n_1=10$ GO TO 10	1
	$28=N_1$

〈Operands of the interchange sort program〉

Operand	Count
1 X	6
2 I	5
3 J	4
4 N	2
5 2	2
6 SAVE	2
$n_2=7$ 1	1
	$22=N_2$

그림 1.

volume 값이 204가 나왔고, 이를 assembler 언어를 사용해 보면 328이 나온다. 이로부터 어셈블러 언어가 coding시 더 많은 노력이 필요하다는 결론을 얻을 수 있다.[11]

2) McCabe's complexity measure

이 방식은 프로그램의 control flow 표현에 바탕을 두었으며, 프로그램의 업무처리 과정을 그래프로 그렸을 때 이루어지는 구획(region)의 수를 세어 이를 판단하여 복잡도를 평가하는 방식이다. 그림 2를 보면 a 작업(task)엔 b, c, d 중 하나가 뒤따르게 되며 b 작업 뒤엔 항상 e가 뒤따르고, e에서는 b로 가는 루프(loop)를 형성하든가 a 또는 f로 가게 되겠다. 이때 이 그래프가 이루는 지역구분은 R1, R2, …, R5로 5개가 된다. 따라서 complexity metrics는 V(G)=5가 된다. 즉 처리 진행방식이 복잡하거나 루프가 많을 경우 그 복잡도는 올라가 McCabe의 metrics 값도 높아진다. 여기서 McCabe는 V(G) 계산치가 10이내 정도가 하나의 모듈(modue)크기로 적당하다고 주장하고 있다.[10] 우리도 새로운 시스템 개발시 알고리즘을 만들 때는 이러한 metrics를 염두에 두고 추진함이 좋을 것이다.

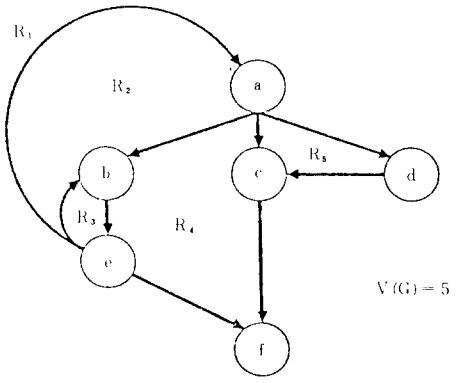


그림 2.

Ⅳ. 국내 주요 소프트웨어 개발 현황

최근 국내에서는 정부출연 연구소, 대기업, 학계 등을 중심으로 대규모 소프트웨어 개발의 추진이 활발히 진행되고 있다. ETRI를 중심으로 국가기간 전산망 구축을 위한 각종 표준화 작업 및 관련 소프트웨어 개발이 진행되고 있고, 행정전산망 주전산기 Ⅰ, Ⅱ용 소프트웨어 개발은 시스템 소프트웨어, 병렬 소프트웨어, DBMS, 응용 소프트웨어 등이 포함된 대형 프로젝트이며, ETRI 주관의 인공지능 컴퓨터 개발은 차세대 컴퓨터로서 멀티미디어 기능을 가진 입출력 시스템 관리, 멀티미디어 정보처리용 시스템 소프트웨어, 통신망 인터페이스 개발 등을 포함한 프로젝트이다. 이에 관련되어 지식 멀티미디어 DB개발은 필히 수행되어야 할 중요한 과제로서 올해부터 체신부 자금 지원하에 추진중이다. 데이타베이스 분야는 우리나라가 많이 뒤떨어진 분야로 시급한 개발 및 데이타베이스 구축이 필요한 바 우선 각종 통계 DB로서 국내외 기술동향, 인력관련 통계 등이 포함된다. 이외에 통신용 emulation S/W, DB 디렉토리 등의 개발도 계획 추진중이며, 각종 상용 데이타베이스 구축도 활발히 진행중이다. 컴퓨터 제조회사를 중심으로 추진되는 PC급 컴퓨터 운영체계(O.S.)의 국산화 개발도 추진되고 있고, 완료된 사례도 발표되었으며, 각종 PC용 소프트웨어는 국산화 추진뿐 아니라 국내에서 자체개발해 사용 판매되는 것도 상당수 있다. 최근 개발되고 있는 각종 전자제품에 필히 포함되는 소프트웨어도 그 규모가 점차 확대되고 고급화 추세에 있는 바, 이의 개발도 전자업체에서 신규제품 개발과 동시에 추진되고 있다. 최근 각광을 받고 있는 퍼지(fuzzy)시스템을 이용한 세탁기, 냉장고, 에어컨 등이 이에 해당된다. 통신분야 소프트웨어로서 EDI 변환 및 통신 처리 S/W 패키지 개발도 추진중에 있으며, 일부 업체에서는 자체개발 EDI를 사내에 설치 운용하기도 한다. 한글의 복잡성을 전산화 처리키 위해 한글 인식 소프트웨어 및 글자형(font) 개발 시스템이 선보이고 있고 font 라이브러리 개발 과제도 추진중이다. 근래 사용이 확대되고 있는 웍스테이션(workstation)용 소프트웨어 개발도 활발하다. 또한 전문 기술용인 CAD(computer aided design)와 CAE(computer aided engineering)용 소프트웨어에 대한 개발도 진행되고 있으며, ECAD용으로 서두로직회사에서 자체 개발에 성공한 MyCAD는 무척 의미있는 시스템이다. MyCAD용 S/W도 KIST에서 개발에 성공 상용화되어 현재 동남아 수개국에 수출되고 있기도 하다. 다중처리 시스템 실시간 운영체제 개발도 체신부 기술개발자금으로 추진중이다. 현대전자에서 최근 개발에 성공한 FTAM(file transfer, access, and management) 시스템은 이기종간 통신할 수 있는 프로토콜을 보여주며, 유닉스버전 서버및 클라이언트와 DOS 버전의 클라이언트에서 구현한 것이 특징이다. 서울대 및 KIST가 주관하여 개발을 추진중인 자연언어(natural language)

의 인식을 위한 시스템 개발은 한국어, 영어 등을 인식 자동번역이 가능하도록 추진중에 있으며, user interface를 이용 간단한 대화는 인식이 가능한 수준에 이르고 있다. 멀티미디어 시스템은 마이크로 프로세서, 저장장치, 고속 통신망 등의 발전으로 의료, 교육, 산업, 광고, 스포츠 등 다양한 분야에서 그 사용 가능성이 높아지고 있으며 시장규모도 확대되고 있다.[7] ETRI를 중심으로 자연언어, digital video 및 audio, image 등을 이용한 입력 매체와 더불어 user interface 구축에 관한 연구는 무척 고무적인 프로젝트이다.[8]

V. 결론 및 향후 추세

이상으로 우리는 국내 소프트웨어 기술 및 개발 현황, 국외의 최근 연구 동향에 대해 살펴 보았다. 추후 소프트웨어 분야의 연구 개발에 대한 투자는 더욱 확대되어야 할 것이며, 가능하면 정부차원의 투자가 더욱 기대되고 있다. 국내 소프트웨어의 발전을 위해 소프트웨어 공학과를 신설한다는 최근의 정부 발표는 매우 고무적인 일이다. 이 분야의 발전이 곧 우리나라 정보산업의 발전을 선도한다는 인식아래 관련 전문가 및 학계인사 모두가 성의껏 대처해 나가야 할 것이다. 그러기 위해서는 첫째 소프트웨어 공학에 대한 일반의 인식을 바꾸는데 주력하여 단지 프로그램 작성 정도의 기술이 아니라는 것을 심어주어야 할 것이다. 둘째 기본적인 소프트웨어 공학 기술을 확산시켜 모든 시스템 개발에 적용토록 하고, 라이프 사이클(life cycle)에 따른 개발 과정의 충실한 전개를 유도해 나가야 하겠다. 셋째 정부차원의 소프트웨어 지적 보호정책의 확립과, 사용자 및 사용업체의 협조 분위기가 확산 되어야 하겠다. 또한 국내 개발 소프트웨어를 구매하여 그 사용율 증가에 협조해야 할 것이다. 넷째, 소프트웨어 공학 관련 신기술을 국내 환경에 맞추어 정착 발전시켜 나가도록 산·학·연 전문가들이 협조체제를 구축 신기술의 국내 조기 정착을 시도해야 할 것이다. 다섯째 외국 선진 기술자들 및 학자들과의 잦은 교류를 통해 새로운 기술의 입수에 적극적인 자세를 보여야 하겠다. 이제는 우리 소프트웨어 엔지니어의 수준도 그 자질면에서는 외국 기술자와 비교해 손색이 없다 하겠다. 개발 환경을 잘 갖추어 주면 좋은 아이디어가 떠오를 때 새로운 시스템 개발이 쉽게 진행될 수 있을 것이다. 개발 대상 소프트웨어 중 단기적으로 보아 국내 전산화 추진에 필요한 관리, 경영 등의 소프트웨어 개발 이외에 데이타베이스, CAD/CAM 등 기술 관련 소프트웨어 가운데 아직 개발되지는 않았으나 새로운 아이디어를 적용하여 어느정도 승산이 있는 부분에 적극 도전, 시스템화 함으로써 외국과의 경쟁에서 이길 수 있는 우리의 영역 확보가 필요하다. 항상 외국 기술에 뒤 따라 가는 방법으로는 급변하는 요즈음 사회에서는 승산이 적다. 그러나 장기적인 안목으로 보아 꼭 필요한 요소 기술에 대해서는 기업체에서 보다는 정부주관으로 정부출연 연구소나, 정부의 연구비 지원으로 대학교 등에서 집중 연구 개발을 시도함이 타당하다. CAD/CAM 소프트웨어의 국산화 개발, 지능형 컴퓨터 개발 등이 이에 해당되며, 일부는 이미 정부 주도로 추진하고 있기도 하다. 지금 시점에서 보다 적극적으로 산·학·연이 협조하여 소프트웨어 기술 발전에 전력 추진한다면 우리의 앞날은 밝다 하겠다. 우수한 과학재능을 가진 많은 젊은 과학도의 잠재력에 큰 기대를 가져도 좋을 것이다. 더구나 정부 주관으로 경기도 용인에 소프트웨어 산업단지를 조성 한다는 소식은 매우 고무적이다. 이젠 외국의 소프트웨어를 도입하기에 급급한 상황에서 벗어나 우리 본연의 자리를 잡아갈 날이 멀지 않았음을 보여줄 때라고 본다. 또 곧 이러한 날이 오리라 확신하는 바이다.

參 考 文 獻

[1] Roger S. Pressman, *Software Engineering*, McGraw-Hill Book Co., 1987.

[2] Roger P. Beck, S. R. Desai, R. P. Radigan, "Software Reuse : A Competitive Advantage", International Confer. on Communication '91, Dever, vol. 3, p. 1505, June 1991.

[3] J. O. Boese, E. J. Reid, "Software Reuse : A Challenge", ICC '91, Denver, vol. 3, p. 1497, June 1991.

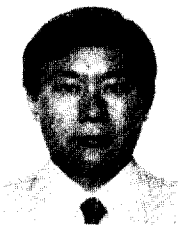
[4] Jesus R. Tirso, "Establishing A Software Reuse Support Structure", ICC '91, Denver, vol. 3. p. 1503, June 1991.

[5] 이경환, "객체지향 방법과 소프트웨어 재사용", 정보과학회회지, 제8권 제5호, p. 6, 1990년 10월.

[6] Robert H. Dunn, *Software Quality, Concept and Plans*, Prentice Hall, p. 70, 1990.

[7] 김형준, 안병준, 강진호, "멀티미디어 표준화 동향", 한국정보과학회지 p. 38, 1991년 6월.
[8] Seungjun Oh, Doohyun Kim, Ho Yang, "The Integrated Multimedia Interface for Intelligent Computer", International Workshop on Intelligent Computer 1990, Daejon Korea, Nov. 1990.

筆 者 紹 介



金 薰

1951年 8月 8日生
1978年 2月 서울대학교 물리학과 (이학사)
1985年 8月 미국 Old Dominion University, Norfolk, VA, 전산학과(석사)
1990年 8月 미국 University of Illinois at Chicago, 전기전산학과(박사과정 수료) 소프트웨어 공학 전공
1978年 1月～1983年 8月 현대엔진공업주식회사 전산연구부, 과장
1990年 9月～현재 현대전자산업(주) 산업전자연구소, 책임연구원
주관심분야는: CAD/CAM Software 개발, Object Oriented Software Development Method, Expert 시스템을 이용한 Software 개발

任 在 傑

1952年 7月 24日生
1981年 2月 동국대학교 전산과 (학사)
1987年 3月 미국 일리노이주립대 시카고캠퍼스 전기전산과 (석사)
1990年 12月 미국 일리노이주립대 시카고캠퍼스 전기전산과 (박사)
1974年 3月～1979年 3月 초등교사
1980年 12月～1981年 7月 경제기획원 조사통계국
1991年 2月～현재 현대전자산업(주)
주관심분야 : Applied Graph Theory, Knowledge Engineering, Software Engineering